Version of 2 January 2012

Course Information

A project-centered class covering various topics in programming language implementation.

Course structure for Spring 2012

Graduate students: register for CRN 43089. Undergraduate students: the course is also listed as CS 491; register for CRN 42480.

The topics this semester are intermediate representations, code transformations, and machine code generation in the compilation of strict functional programming languages. In particular, we will study representations of control flow, the continuation-passing style and the CPS transformation and its relatives, the A-normal form and the static single assignment form. We will examine how higher-order functions can be efficiently implemented. In addition to their use in compilation, the transformations we will study can be viewed as programming techniques, and we will look at examples of their use.

Another area of interest is the definition and implementation of programming languages via abstract machines and virtual machines (bytecode compilation). Again, continuations play a big role in this development.

Format: Lectures; research paper presentations.

Prerequisites in detail

No specific courses are prerequisites for this one.

Students should be familiar with several high-level programming languages, so that they can appreciate the purpose and the tasks of a compiler.

Students should be experienced programmers able to develop large programming projects implemented quickly. The ability to keep up with deadlines is important.

There are some specific topics that will be assumed to form the background of each participant:

- functional programming in general, and Scheme, ML, or Haskell in particular
- understanding recursive data types, recursive functions to compute over them, and structural induction to prove things about them
- familiarity with computer organization and architecture, operating systems, machine language and assembly language programming, and the C programming language

Lectures

Tuesdays and Thursdays 9:30-10:45 in Dane Smith Hall 126

Instructor

Darko Stefanovic, office FEC 345C, phone 2776561, email darko — office hours by appointment

Teaching assistant

None

Grading

The grade will be determined as follows:

- Programming projects 70%
- Oral presentations 30%

You are expected to attend class regularly, read the assigned reading before class, give occasional oral presentations of research papers, and participate in class discussion.

Programming assignment hand-in policy

Programming assignments are to be submitted on-line. Detailed instructions will be provided with each assignment.

Topics

- Introduction to compilation
- The structure of compilers
- Front-end design
- Back-end design
- Common issues in the compilation of functional languages
- Compilation of strict functional languages
- Intermediate representations for strict functional languages
- Front-end design for strict functional languages
- Modules, functors, and their implementation
- Abstract machines for lambda calculi
- Representing and analyzing control flow

- Higher-order functions and their implementation
- Parametric polymorphism and its implementation
- Back-end design for strict functional languages