Course Information

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

Course structure for Fall 2018

Graduate students: register for CRN 63257. Undergraduate students: the course is also listed as CS 491; register for CRN 63272.

Course covers intermediate representations, code transformations, and machine code generation in the compilation of functional programming languages. We will examine how higher-order functions can be efficiently implemented. Another area of interest is the definition, implementation, and validation of programming languages via abstract machines.

Format: Readings; code review; discussion; occasional lectures; discussion; research paper presentations.

Prerequisites

CS341 and any one of CS357, CS558, CS556, or equivalent experience, mainly programming in a functional language; willingness to work in a team.

Assumed background:

- 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
- some familiarity with computer organization and architecture, operating systems, machine language and assembly language programming, and the C programming language

Lectures

Tuesdays and Thursdays 2-3:15 in FEC3100.

Instructor

Darko Stefanovic, office FEC2020, phone 2776561, email darko — office hours Mondays 11:00-12:00, Tuesday 3:15-4:00, Thursdays 3:15-4:00

Teaching assistant

None
Grading
You are expected to attend class regularly, read the assigned reading before class, give occasional oral presentations on research papers and team projects, and participate in class discussion. Your grade will be determined as follows:

- Programming projects, including reports: 50%
- Oral presentations: 30%
- Discussion and participation: 20%

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

Textbooks
All reading materials will be provided electronically, free of charge.

Sample topics (selection varies from year to year)

- Introduction to compilation
- The structure of compilers
- Front-end design
- Back-end design
- Common issues in the compilation of functional languages
- Abstract machines for lambda calculi
- Representing and analyzing control flow
- Call-by-value, call-by-name, and call-by-need
- Higher-order functions and their implementation
- Parametric polymorphism and its implementation
- Validation of programs and compilers

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.