Course Information

General

This course counts for 3 credits. Undergraduate students may also register for the corresponding CS 491. Students outside CS should consult the instructor beforehand.

Course objectives

At the completion of this course students will be able to:

1. Apply compiler algorithms to symbolic input data processing in various application domains.
3. Design and program a compiler for a functional programming language.

Course structure for Spring 2024

The course covers intermediate representations, program transformations, and code generation in the compilation of functional programming languages. At the end of the course, students will be able to implement a core higher-order functional language via abstract machines.

Format: Readings; code review; discussion; lectures; presentations of classical papers in the literature.

Prerequisites

CS341 or equivalent experience with computer organization and architecture; any one of CS357, CS558, CS556, or equivalent experience, mainly programming in a functional language; willingness to work in a team. Recommended: CS454/554 or equivalent introductory compiler course.

Instructor

Darko Stefanovic, office FEC2020, phone 2776561, email darko — office hours TBA

Lectures

TBA
Teaching assistant
TBA

Grading
You are expected to attend class regularly, read the assigned reading before class, periodically give 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.

List of topics
- Introduction to compilation
- The structure of compilers
- Front-end design
- Back-end design
- Common issues in the compilation of functional languages
- Types and type checking
- 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
• Program transformations
• Program analyses and optimizations
• Validation of programs and compilers