Preliminary version of 31 October 2025

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.
2. Measure and evaluate program performance.
3. Design and program a compiler for a functional programming language.

Course structure for Spring 2026

Graduate students: register for CRN 83359. Undergraduate students: the course is also listed as CS 491, CRN 82413.

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 have implemented 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, but not required: CS454/554 or equivalent introductory compiler course.

Lectures

Mondays/Wednesdays/Fridays, 1:00–1:50.

Instructor

Darko Stefanovic, office FEC2020, phone 2776561, email `darko` — office hours Mondays and Fridays 2:00–3:00.

Teaching assistant

none

Grading

You are expected to attend class regularly, read the assigned reading before class, participate in class discussion, and periodically give oral presentations on team projects. 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. Late programming assignment submissions will be penalized $2n^2\%$, where n is the number of days late.

Textbooks

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

List of topics

- Introduction to compilation
- The structure of compilers
- Front-end design for functional languages
- Back-end design for functional languages
- Common issues in the compilation of functional languages
- Types and type checking

- Parametric polymorphism and its implementation
- Abstract machines for lambda calculi
- Call-by-value, call-by-name, and call-by-need
- Higher-order functions and their implementation
- Representing and analyzing control flow
- Program transformations
- Program analyses and optimizations
- Validation of programs and compilers

UNM statement of compliance with ADA

Every instructor should include an official statement in their course syllabus. The suggested syllabus statement should include the following text:

"In accordance with University Policy 2310 and the Americans with Disabilities Act (ADA), academic accommodations may be made for any student who notifies the instructor of the need for an accommodation. It is imperative that you take the initiative to bring such needs to the instructor's attention, as I am not legally permitted to inquire. Students who may require assistance in emergency evacuations should contact the instructor as to the most appropriate procedures to follow. Contact Accessibility Resource Center at 277-3506 for additional information.
If you need an accommodation based on how course requirement[s] interact with the impact of a disability, you should contact me to arrange an appointment as soon as possible. At the appointment we can discuss the course format and requirements, anticipate the need for adjustments and explore potential accommodations. I rely on the Disability Services Office for assistance in developing strategies and verifying accommodation needs. If you have not previously contacted them I encourage you to do so."