CSE 583: Advanced Compilers
Fall 2025 Syllabus

Class and Instructor

- Lecture: Mon/Wed, 10:30am-12:20pm, 1500 EECS
- Prof. Scott Mahlke
- Email: mahlke@umich.edu
- Lecture will also be available live on Zoom 10:30am-12:20pm MW and recorded

Audio/Video Recordings

Course lectures will be audio/video recorded and made available to all students in this course. As part of your participation in this course, you may be recorded. If you do not wish to be recorded, please contact the instructor (Scott Mahlke, mahlke@umich.edu) the first week of class to discuss alternative arrangements. To prevent revealing your identity on recordings, please mute your video during lecture. Also, questions can be submitted via Zoom chat if you do not wish to reveal your voice.

Students may not record or distribute any class activity without written permission from the instructor, except as necessary as part of approved accommodations for students with disabilities. Any approved recordings may only be used for the student's own private use.

Course Description

An in-depth study of compiler backend design for high-performance architectures. Basic topics include control-flow and dataflow analysis, optimization, instruction scheduling, modulo scheduling, and register allocation. Advanced topics include memory dependence analysis, automatic vectorization/thread extraction, streaming applications, predicated and speculative execution, dynamic compilation, and security. The focus is backend compilation, thus familiarity with computer architecture and compilers is recommended.

Reference Books

1. *Advanced Compiler Design & Implementation*, Muchnick, Morgan Kaufmann, 1997.
2. *Compilers: Principles, Techniques, and Tools (2nd edition)*, Aho, Lam, Sethi, Ullman, Pearson Addison-Wesley, 2007. (1st edition is also fine)

Prerequisites

Strong C++ programming skills (EECS 281), good background in computer architecture (EECS 370 at minimum), some familiarity with compilers (EECS 483 or prior compiler class is desirable but not needed).

Grade

Midterm exam - 25%
Project - 45%
Homeworks – 15%
Paper presentation – 10%
Class participation – 5%

Midterm exam - There will be one in-class (1.5 hour) exam at about the 2/3 point of the class. The tentative date is on the course schedule. The exam will be open book/notes and has a hybrid format with in-person and remote options.

Project - The projects will consist of designing and implementing an advanced compiler technique within the LLVM compiler infrastructure (or other compiler system in certain cases). A report describing the project should be submitted along with a brief presentation and/or demonstration of the resulting implementation. Typical projects consist of 3-5 students; 1 or 2 person projects are discouraged due to the extra work incurred by those teams. There will be a project proposal and project update for each group scheduled during the semester.

Homeworks – 2 programming assignments will be done in the early portion of the semester. Each homework will consist of implementing something within the LLVM compiler system and showing its operation on several test programs. Each student must do their own work and turn in their own assignment.

Paper presentation – During the research topic portion of the class, each project group will present a research paper to the class related to their project. Each group is responsible for selecting an appropriate conference paper and giving a 15-minute presentation to the class. Scheduling will be done via signup on a shared calendar.

Class participation - Students are encouraged to take an active role in this class by asking questions or providing comments.

Rough Topic list

- Control flow analysis and optimization
  - Basics: control flow graphs, dominators, loop detection
  - Regions: traces, superblocks
  - Predicated execution: control dependence analysis, hyperblocks
  - Code layout, alignment
- Dataflow analysis and optimization
  - Basics: liveness, reaching defs
  - Static single assignment form
  - Classical and ILP optimization
  - Analysis applications (security, reliability)
- Code generation
  - Basics: dependences, latencies, ASAP/ALAP times
  - Instruction scheduling, superblock scheduling, control speculation
  - Modulo scheduling, rotating registers
  - Register allocation
- Compilation for multicore
  - Parallelization of loops: Vector, DOALL, DOACROSS, DSWP
  - Intro. to dynamic (JIT) compilation
- Research topics (presentations by the class)