CS 4390/5390 STC: Advanced Compilers
Fall 2025 Syllabus
Lecture: CRNs 11828/12917; MW 1:30-2:50pm; MAIN 211
Instructor: Shirley Moore
Course webpage: http://svmoore.pbworks.com/

Compiler technology is an important topic today in light of the rapid expansion of new programming models and processor architectures, including domain-specific languages and architectures. This trend is creating a huge demand for new compilers and for the compiler engineers who can design and implement them. This course will cover universal compiler topics such as intermediate representations, data flow, and code optimization, as well as more research-oriented topics such as parallelization, just-in-time compilation, garbage collection, and deep learning compilers. This course is not about parsing and syntax, but rather it concerns the semantics of programs and how programs can be converted into executable code that implements those semantics on a given processor architecture, while meeting correctness, performance, and security requirements.

- **Instructor**
  - Shirley Moore, svmoore@utep.edu
    - Office: CCSB 3.0608
    - Office phone: 915-747-5054
    - Office hours: Mondays 12:00-1:00pm, Thursdays 3:00-4:00pm, others by appointment (in person or virtual)
    - Webpage: http://www.cs.utep.edu/svmoore

- **Prerequisites**
  - CS 3432 or equivalent

- **Texts and Readings**
  - Recommended: *LLVM Code Generation: A deep dive into compiler backend development*, by Quentin Colombet, Packt Publishing, May 2025. ISBN-13: 978-1837637782
  - Other readings and resources will be posted on the course webpage.

- **Learning Outcomes**
  After successfully completing the course, students will be able to
  - Explain the concept and purpose of an intermediate representation (IR) of a program.
  - Construct a control flow graph (CFG) from an intermediate representation, identifying basic blocks, dominators, loops, and branches.
  - Apply classical dataflow analysis algorithms (e.g., reaching definitions, liveness analysis, available expressions) and implement their fixed-point computation using iterative and worklist methods.
  - Design and analyze SSA (Static Single Assignment) form, including conversion to and from SSA, and explain its impact on optimizations.
  - Describe register allocation algorithms, including graph coloring and linear scan, and evaluate their time and space complexity.
  - Explain instruction scheduling strategies, such as list scheduling or software pipelining, and discuss their effect on performance.

- Analyze the correctness and convergence properties of common optimization techniques, such as dead code elimination, common subexpression elimination, and constant propagation.
- Use LLVM tools and APIs to analyze and transform an IR**,** including writing custom LLVM passes.
- Build and debug LLVM passes that modify, optimize, or instrument code.
- Implement machine-independent optimizations (e.g., constant folding, loop unrolling, dead code elimination) using LLVM's infrastructure.
- Design and implement machine-dependent code generation components**,** such as instruction selection, register allocation, and calling conventions for a target architecture.
- Use LLVM's built-in analysis infrastructure (e.g., data flow, alias, dominator trees) to inform optimizations.
- Verify correctness of code transformations and use tools such as opt, llc, and llvm-lit for testing and regression validation.
- Evaluate the impact of backend transformations on runtime performance using profiling and benchmarking tools.

The above outcomes will be assessed through both analytical exercises (e.g., proving correctness of a dataflow algorithm or performing hand analysis on a CFG) and implementation assignments (e.g., implementing an optimizer that uses liveness and SSA).

Stretch outcomes (if time or for projects)

- Design and implement a simple JIT compiler using LLVM ORC (On-Request Compilation) that generates and executes code at runtime for a subset of a high-level language.
- Analyze and implement compiler optimizations relevant to deep learning workloads (e.g., tensor algebra simplification, operator fusion, and memory layout transformations) and evaluate their performance using LLVM-based or MLIR-based backends.

- **Computers**
  - You will need access to a computer on which you can write and run Python programs and on which you can compile and use LLVM.
  - Students doing projects may be provided with access to advanced architectures such as recent GPUs or domain-specific architectures.
- **Attendance**
  - Attendance is mandatory at all lecture sessions unless special circumstances are arranged ahead of time with the instructor, or as soon afterwards as possible in the event of an emergency. Discussions and hands-on exercises will be part of lecture sessions, and it will not be possible to make up those activities.
- **Grading Breakdown**

90-100% = A 80-89% = B 70-79% = C 60-69% = D 59% and Below = F

- Programming assignments: 300 points
- Midterm exam: 150 points
- Final exam: 200 points
- Participation: 150 points
- Term project: 200 points

Term Project: The term project is required for graduate students and is optional for undergraduate students. The project will be an individual project on a topic of your choice. The instructor will prepare a list of suggested topics, but you will be free to choose a different topic that interests you subject to instructor approval.

Participation: Students will be required to participate in weekly discussions. Students will also be given smaller assignments that will build towards the larger programming assignments. Each of these activities will be given point values that add up to the total 150-point participation grade. These points cannot be made up, so students are expected to stay active in the course by participating each week.

Research paper: As part of the participation grade, doctoral students will be required to read and present a research paper selected from a list prepared by the instructor or of their own choosing if approved by the instructor.

Disabilities and Accommodations

- If you have a disability and accommodations, please contact The Center for Accommodations and Support Services (CASS) at 747-5148, or by email to **cass@utep.edu**. For additional information, please visit the CASS website at **www.sa.utep.edu/cass**.

Academic Honesty

- Students are expected to conduct themselves in a professional and courteous manner, as prescribed by the Standards of Conduct: https://www.utep.edu/hoop/section-2/student-conduct-and-discipline.html
- Submitted work should be unmistakably your own. You may not transcribe or copy a solution taken from another person, book, or other source (e.g., a web page). Copying other's work will not be tolerated. Professors are required to report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.
- Permitted collaboration: Students may discuss requirements, background information, test sets, solution strategies, and the output of their programs. However, implementations and documentation must be their own creative work. Students are required to document advice received from others and all resources utilized in the preparation of their assignments.
- If academic dishonesty is suspected: The Dean of Students office will be contacted for adjudication. A temporary "incomplete" grade will be issued if their investigation extends beyond the grading period.

Guidance on Artificial Intelligence

The use of generative AI tools such as Chat GPT is permitted in this course, provided such use is properly acknowledged and documented. Specific more detailed guidelines concerning permissible uses of AI may be included with individual assignments. Please be aware that you must fully understand and be able to explain any solutions that you submit. Solutions that solve a different problem or that do not follow the assignment guidelines will not be accepted or graded.

Students must cite any borrowed content sources to comply with all applicable citation guidelines, copyright law, and avoid plagiarism. Instances that violate these guidelines will be referred to the Office of Student Conduct and Conflict Resolution.

Plagiarism Detecting Software

Some of your course work may be submitted to the SafeAssign or MOSS plagiarism detecting software. These tools are used to review submissions for originality and will help you learn how to properly attribute sources.

Copyright Statement for Course Materials

All materials used in this course are protected by copyright law. The course materials are only for the use of students currently enrolled in this course and only for the purpose of this course. They may not be further disseminated.

Illness Precautions

Please stay home if you have symptoms of a communicable illness. If you are feeling unwell, please let me know as soon as possible, so that we can work on appropriate accommodations.

UTEP Resources

UTEP provides a variety of student services and support:

Technology Resources

- Help Desk: Students experiencing technological challenges (email, Blackboard, software, etc.) can submit a ticket to the UTEP Helpdesk for assistance. Contact the Helpdesk via phone, email, chat, website, or in person if on campus.

Academic Resources

- UTEP Library: Access a wide range of resources including online, full-text access to thousands of journals and eBooks plus reference service and librarian assistance for enrolled students.
- University Writing Center (UWC): Submit papers here for assistance with writing style and formatting, ask a tutor for help and explore other writing resources.
- RefWorks: A bibliographic citation tool; check out the RefWorks tutorial and Fact Sheet and Quick-Start Guide.

Individual Resources

- Military Student Success Center: Assists personnel in any branch of service to reach their educational goals.

Course Schedule

The course lecture schedule will be posted at http://svmoore.pbworks.com/