ECE 595.1

Course Run

ECE595.1: Introduction to Compilers – Compiler Basics

Instructor

- Milind Kulkarni, *Associate Professor of Electrical and Computer Engineering, Purdue University*

Audience

First- or second-year graduate students without prior compiler experience. Students are expected to have programming experience, especially with data structures and recursion.

Course Description

This course is an introductory course on compilers. We will cover the full path that a compiler takes in translating high-level source code (e.g., in a language like C) to assembly code that can be run on a machine. We will cover the processes of translating source code into a compiler's intermediate representation, then generating code from that intermediate representation. Students will also build a basic compiler that translates C code into RISC-V assembly.

Course Learning Outcomes

After completing this course, you will be able to:

- Explain regular expressions and lexing procedures
- Explain context-free grammars and LL(1) parsing
- Explain semantic actions and AST construction
- Explain how to translate ASTs into executable code
- Build a basic compiler that can translate a program into executable code

Required Text and Materials

There is no required text for this class. However, students may find the book *Engineering a Compiler*, by Cooper and Torczon (2nd edition, Morgan Kauffman, ISBN: 978-0120884780) helpful.

Gradescope
Gradescope will be used in this course as a method for grading your submitted problem sets

GitHub
Programming assignments will be managed through GitHub classroom

Prerequisites

Comfort with programming, especially data structures and recursion. Experience with object oriented programming will also be beneficial.

Students must have taken ECE 595.1 — Introduction to Compilers – Compiler Basics

Grading

This course will be graded based on the following criteria:

Assessment Type	Description	% of Final Grade
Problem sets	There will be 2 problem sets that will test your knowledge of theoretical aspects of this course	20%
Programming assignments	There will be 3 programming assignments that will walk students through building a compiler that can translate C code into RISC-V assembly	60%
Exam	There will be a final exam summarizing the content of the course	20%

Course Content and Activities

Week	Module	Lessons	Activities & Assignments
1	1.1 Welcome and Introduction	Getting Started/Overview	Lecture Videos Handouts

1	1.2 What is a compiler? 1.3 How does a compiler work? 1.4 Regular expressions	What is a compiler Types of compilers Phases of a compiler Structure of a compiler What is a regular expression? Regular expression syntax	Lecture Videos Handouts Quiz: Identify types and purposes of a compiler Lecture Videos Handouts Quiz: Identify the phases of a compiler Lecture Videos Handouts Quiz: What are regular expressions
2	2.1 Finite automata 2.2 Context free grammars	Finite automata Non-deterministic finite automata Building an NFA Pros and cons of NFAs From NFAs to DFAs Regex engines Parsing overview Context free grammars	Lecture Videos Handouts Quiz: Automata construction Problem Set: Regular expressions and NFAs Lecture Videos Handouts Quiz: Context free grammar
3	3.1 Parsing 3.2 Parsing	Parsing basics Recursive descent parsing First and follow sets First sets example Follow sets example Putting it all together When does LL(1) fail? ANTLR basics	Lecture Videos Handouts Quiz: First and follow sets Problem Set: CFGs and First and follow sets Lecture Videos Handouts Programming assignment: Extend lexer and parser
4	4.1 Symbol table 4.2 Abstract Syntax Trees: basics and expressions	Semantic actions Basic semantic actions in ANTLR Symbol tables What is an abstract syntax tree? Abstract syntax trees for expressions Generating code from ASTs Code generation example	Lecture Videos Handouts Lecture Videos Handouts Programming assignment: Generate code for expressions and assignments
5	5.1 Abstract syntax trees: control structures 5.2 Code generation: complex control structures	ASTs for control structures Generating code for control structures Generating code for loops	Lecture Videos Handouts Lecture Videos Handouts

5		Switch statements	Programming assignment: generate code for if statements and loops

Estimated Effort

- 10-12 hours/week
- 5 weeks total

Languages

Content: English | Videos: English | Transcripts: English

Course Difficulty

- Introductory

Accessibility Support

Purdue University strives to make learning experiences as accessible as possible. If you anticipate or experience physical or academic barriers based on disability, you are welcome to let an instructor know so that we can discuss options. You are also encouraged to contact the Disability Resource Center at: mailto:drc@purdue.edu or by phone: 765-494-1247.

Visit edX's Website Accessibility Policy for information about accessibility on edX.

Course Help

To get help with course content, click the Discussion tab and post a question in "Course Q&A". By commenting in the pinned discussion post, the course team will be able to respond to your question more quickly.

Technical Help

For general questions about using the edX platform, please refer to these resources:

- Technical Documentation: https://docs.edx.org
- Learner Help Center: https://support.edx.org/hc/en-us
- To get help with a technical problem, visit the *Help* link to contact edX Support.

Discussion Guidelines

Please follow the Discussion Guidelines when contributing to discussions in this course. Here are a few of the key points you should remember:

- Do not use offensive language. Present ideas appropriately.
- Be cautious in using Internet language. For example, do not capitalize all letters since this suggests shouting.
- Avoid using vernacular and/or slang language. This could possibly lead to misinterpretation.
- Keep an “open-mind” and be willing to express even your minority opinion.
- Make substantive posts or comments. Avoid comments that do not contribute to the discussion, like "thanks" or "good post."
- Do not hesitate to ask for feedback.
- Be concise and to the point. Give other students the opportunity to join in the discussion.
- Think and edit before you push the “Send” button.

Academic Integrity

Academic integrity is one of the highest values that Purdue University holds. Individuals are encouraged to alert university officials to potential breaches of this value by either emailing or by calling 765-494-8778. While information may be submitted anonymously, the more information that is submitted provides the greatest opportunity for the university to investigate the concern.

The Purdue Honor Pledge

"As a boilermaker pursuing academic excellence, I pledge to be honest and true in all that I do. Accountable together - we are Purdue"

Nondiscrimination Statement

Purdue University is committed to maintaining a community which recognizes and values the inherent worth and dignity of every person; fosters tolerance, sensitivity, understanding, and mutual respect among its members; and encourages each individual to strive to reach his or her own potential. In pursuit of its goal of academic excellence, the University seeks to develop and nurture diversity. The University believes that diversity among its many members strengthens the institution, stimulates creativity, promotes the exchange of ideas, and enriches campus life. Link to Purdue's nondiscrimination policy statement.