# ECE 595.2

## Course Run

ECE595.2: Introduction to Compilers – Code Generation

## Instructor

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

## Audience

First- or second-year graduate students who have taken ECE 595.1. Students are expected to have programming experience, especially with data structures and recursion.

## Course Description

This course covers advanced compiler topics: generating code for functions, performing type checking to avoid bugs, performing basic compiler optimizations, and performing register allocation. We will cover the theoretical basis of many of these optimizations as well as how they are implemented in compilers. Students will extend the basic compiler constructed in ECE 595.1 to add these advanced features to their compiler that translates C code into RISC-V assembly.

## Course Learning Outcomes

After completing this course, you will be able to:

- Explain how functions work in programming languages and how to generate code for them
- Explain the purpose of type checking and how to implement it
- Explain basic local optimizations: common sub-expression elimination and dead code elimination
- Explain local and global register allocation
- Build a compiler that supports functions, type checking, and register allocation

## 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 2 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<br>Overview | Lecture Videos<br>Handouts |
| | 1.2 Recap | What have we learned? | Lecture Videos<br>Handouts |
| | 1.3 Functions | Function basics<br>How do programs manage functions<br>Function calling conventions<br>Adding functions to symbol tables | Lecture Videos<br>Handouts<br>Quiz: Function behavior |

| | | | |
|---|---|---|---|
| **2** | 2.1 Compiling functions<br><br>2.2 Type checking | Stack organization<br>Code generation for functions<br>Caller saves vs callee saves<br><br>What are types?<br>Dynamic type checking | Lecture Videos<br>Handouts<br>Quiz: Function code generation<br><br>Lecture Videos<br>Handouts |
| **3** | 3.1 Type checking in compilers<br><br>3.2 Optimization: peephole | Static types<br>Static type checking<br>Implementing type checking<br><br>Optimization overview<br>Intermediate Representations<br>Peephole optimizations<br>Local optimizations | Lecture Videos<br>Handouts<br>Quiz: Type checking rules<br>Problem set: Type checking<br><br>Lecture Videos<br>Handouts<br>Programming assignment: Add functions and type checking to compiler |
| **4** | 4.1 Optimization: common subexpression elimination<br><br>4.2 Optimization: dead code elimination | Basic Blocks<br>Control Flow Graphs<br>Common sub-expression elimination<br>CSE example<br>Dealing with aliasing<br><br>Live and dead code<br>Liveness analysis<br>Dead code elimination | Lecture Videos<br>Handouts<br>Quiz: CSE<br><br>Lecture Videos<br>Handouts<br>Quiz: Liveness<br>Problem set: Local optimizations |
| **5** | 5.1 Local register allocation<br><br>5.2 Global register allocation | Register allocation basics<br>Local register allocation<br>Example of register allocation<br>Register allocation details<br><br>Graph coloring register allocation<br>Handling spills | Lecture Videos<br>Handouts<br><br>Lecture Videos<br>Handouts<br>Programming assignment: perform local register allocation |

## 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.