Welcome and Overview

CSE443 Compilers

Carl Alphonce

Department of Computer Science and Engineering

University at Buffalo

Welcome to CSE443

Name:	Carl Alphonce
Email:	alphonce@buffalo.edu
Office:	Davis 343
Office Hours:	Tuesdays 9:00 AM - 10:45 AM or by appt

The syllabus is on the course website, `https://www.cse.buffalo.edu/courses/cse443/`.

Course components

lectures	curated tour through topics
readings	nitty-gritty details of topics
recitations	meet with your PM
team meetings	meet with me

Assessment

project (50%) design and build a compiler (team-based)
final exam (20%) essay-style, with choice (individual)
teamwork (20%) four sprints, assessed by PM
presentation (10%) oral demo (team-based)

Support

open office hours for CSE220 and CSE443

recitations teamwork guidance and accountability

team meetings team-based office hours (techincal questions)

piazza general support for course content

Academic Integrity: Expectations

project	everyone must contribute *equitably*
final exam	everyone must answer questions individually, without assistance

Use of AI assistants is neither appropriate nor permitted in this course.

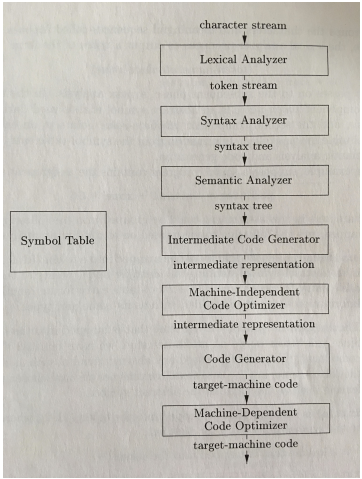
Overall Toolchain

preprocessor $\rightarrow$ *compiler* $\rightarrow$ *assembler* $\rightarrow$ *linker* $\rightarrow$ *loader*

The Compiler

high-level program (source)

low-level program (assembly)



(figure 1.6, page 5 of text)

Compiler: Lexical Analysis

- *character stream* $\rightarrow$ *token stream*

Compiler: Syntactic Analysis

- *token stream* $\rightarrow$ *parse tree*

Compiler: Semantic Analysis

- verify conditions on parse tree

Compiler: Code Generation

- *parse tree* $\rightarrow$ *code*
  - *parse tree* $\rightarrow$ *intermediate code* (internal representation)
  - *parse tree* $\rightarrow$ *target machine* (assembly code)

Ideal endpoint for project is to produce correct x86-64 assembly code.

Compiler: Optimization

- machine-independent *intermediate code* $\rightarrow$ *intermediate code*
- machine-dependent *intermediate code* $\rightarrow$ *assembly code*

We will only touch briefly on this. Project implementation *may* include some machine-independent optimizations.

Philosophy: Learn by Doing

There is a lot of really cool theory underpinning what a compiler does.

In this class:

- majority of learning happens via project implementation
- theory / implementation connections will be discussed
- high-level / low-level connections will be evident

We will sketch some proofs, but you are not expected to reproduce them on your own.

Homework

- form teams and identify potential meeting times
  - follow instructions in Piazza
- before next class, read all sections of chapter 1 in textbook

License

To retrieve a copy of this material, or related materials, see `https://www.cse.buffalo.edu/courses/cse443`.