COSE312: Compilers

Lecture 0 — Course Overview

Hakjoo Oh
2023 Spring

Basic Information

Instructor: Hakjoo Oh

- **Position:** Associate professor in Computer Science, Korea University
- **Expertise:** Programming Languages and Software Engineering
- **Office:** 616c, Science Library
- **Email:** `hakjoo_oh@korea.ac.kr`
- **Office Hours:** by appointment

TA:

- Jisuk Byun (`jisukbyun@korea.ac.kr`)
- Wonseok Oh (`marinelay@korea.ac.kr`)

Course Website:

- Course materials:
  `https://github.com/kupl-courses/COSE312-2023spring`
- Blackboard: submission of assignments, Q&A

Prerequisites

- COSE 212 Programming Languages
- Experience in functional programming (e.g., OCaml, Scala)
- Theory of computation, Discrete maths, Data structures, Algorithms, Architecture, etc

What is Compiler?

A compiler is a software system that translates programs written in a high-level programming language into a low-level machine language.

Why bother to take a compiler course?

- Compilers are one of the most important software systems.
- To deeply understand computer science in general.
  - computation theory (automata, grammars), algorithms (greedy/dynamic programming), fixed point theory (data-flow analysis), software engineering, etc.
- A good application of theory to practical problems.
- Writing a compiler is a substantial programming experience.

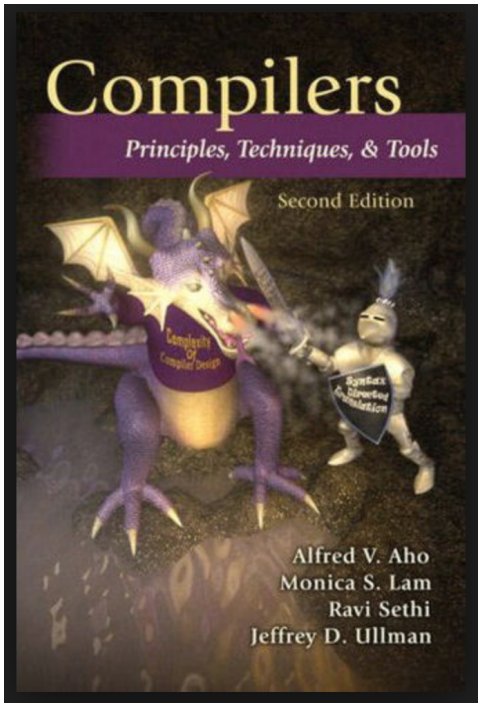
Course Overview (tentative)

You will learn principles and techniques for compiler construction.

- **Lexical analysis:** lexical tokens, regular expressions, finite automata, lexical analyzer generators
- **Syntax analysis:** context-free grammars, top-down parsing, bottom-up parsing, parser generators
- **Semantic analysis:** optimization, verification, data-flow analysis, static analysis
- **Translation:** syntax-directed translation, three address code, control flow graph, basic blocks
- **Code generation (optional):** register allocation and assignments, instruction selection, machine code generation

References

- Self-contained slides will be provided.
- Compilers: Principles, Techniques, and Tools (Second Edition) by Aho, Lam, Sethi, and Ullman. MIT Press.



Grading

- Programming assignments: 90%
- Attendance and participation: 10%

Assignment policy:

- No late submissions will be accepted.
- All assignments must be your own work.
  - Copying gets you 0 for the HW score. We use software for detecting code clones.

Schedule (tentative)

- Lecture (8 weeks):
  - Week 1: Lexical analysis
  - Week 2: Syntax analysis
  - Week 3: Translation
  - Week 4: OCaml Lab (by TAs)
  - Week 5–8: Analysis and optimization
- Assignments (12 weeks):
  - Week 1: OCaml exercises (0%)
  - Week 3: Parser (10%)
  - Week 5: Translator (30%)
  - Week 7: Analyzer (30%)
  - Week 10: Optimizer (30%)