COSE312: Compilers

Lecture 0 — Course Overview

Hakjoo Oh
2025 Spring

Basic Information

Instructor: Hakjoo Oh

- **Position:** 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:

- Joonghoon Lee (`joonghoonlee@korea.ac.kr`)
- Jiho Shinn (`tlswlgh0801@gmail.com`)

Course Website:

- `https://prl.korea.ac.kr/courses/cose312/2025/`
- Blackboard

Prerequisites

- COSE 212 Programming Languages
- Extensive experience in programming
- Theory of computation, Discrete maths, Algorithms, 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.

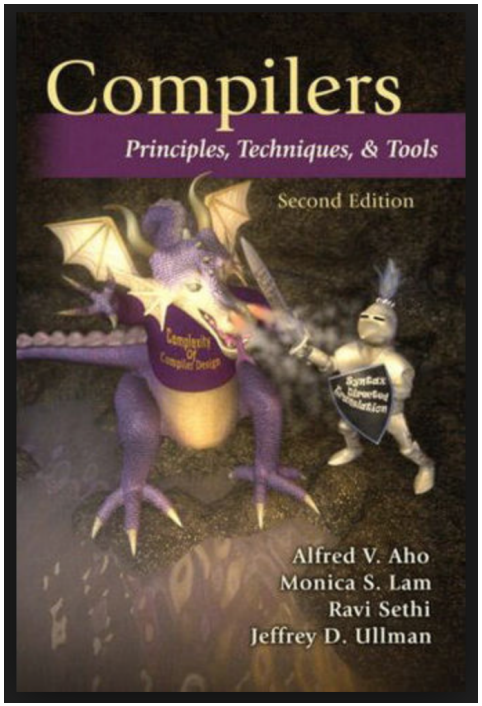
Course Overview (tentative)

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 (tentative)

- Programming Assignments (in OCaml) – 50%
  1. OCaml exercises – 10%
  2. Lexing and parsing – 10%
  3. Translator – 10%
  4. Semantic analyzer – 10%
  5. Optimizer – 10%
- Mid-term exam: (in class, 75 minutes) – 20%
- Final exam: (in class, 75 minutes) – 20%
- Attendance – 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)

Weeks	Topics
Week 1	Introduction
Week 2	Lexical Analysis
Week 3	Lexical Analysis
Week 4	Syntax Analysis
Week 5	Syntax Analysis
Week 6	Syntax Analysis
Week 7	Mid-term Exam
Week 8	Translation
Week 9	Intermediate Representations
Week 10	Semantic Analysis
Week 11	Semantic Analysis
Week 12	Semantic Analysis
Week 13	Code Optimization
Week 14	Code Generation
Week 15	Final exam