COSE312: Compilers

Lecture 0 — Course Overview

Hakjoo Oh
2026 Spring

Basic Information

Instructor: Hakjoo Oh

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

TA:

- Jeongseop Lim (`jeongseop_lim@korea.ac.kr`)
- Junyong Heo (`junyong_heo@korea.ac.kr`)

Course Materials:

- `https://prl.korea.ac.kr/courses/cose312/2026/`
- `https://lms.korea.ac.kr`

Prerequisites

- COSE 212 Programming Languages
- Extensive experience in programming
- Familiarity with functional programming
  - If not, you can learn OCaml on your own
    - `https://ocaml.org/docs`
    - `https://prl.korea.ac.kr/courses/cose312/2025/ocaml-tutorial.pdf`
- Theory of computation, Discrete maths, Algorithms, etc

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.

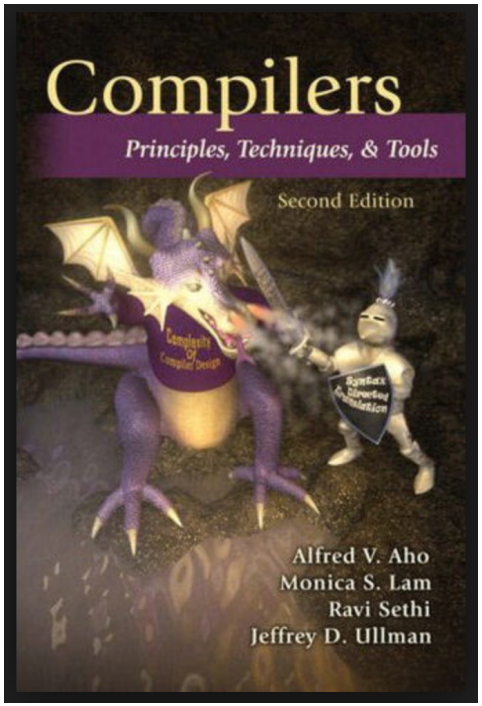
Course Overview (tentative)

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) – 30%
  1. OCaml exercises – 0%
  2. Lexing and parsing – 5%
  3. Translator – 5%
  4. Semantic analyzer – 10%
  5. Optimizer – 10%
- Mid-term exam: (in class) – 30%
- Final exam: (in class) – 30%
- Attendance – 10%

Assignment policy:

- No late submissions will be accepted.
- All homework assignments must be your own work. AI tools (e.g., code assistants or generative models) may be used only as a supplementary aid for understanding concepts or exploring ideas. However, the final submitted code and solutions must be written, understood, and fully owned by you.
- You may discuss the assignments with your peers, but copying or sharing code (including AI-generated code without substantial personal modification and understanding) is strictly prohibited. Any violation will result in a score of 0 for the entire homework.

Schedule (tentative)

- Lectures: 8 weeks (March and April)

Weeks	Topics
Week 1 (3/10, 3/12)	Introduction, Lexical Analysis
Week 2 (3/17, 3/19)	Top-down Parsing
Week 3 (3/24, 3/26)	Bottom-up Parsing
Week 4 (**3/31**, 4/2)	Mid-Term Exam, Operational Semantics,
Week 5 (4/7, 4/9)	Translation, Denotational Semantics
Week 6 (4/14, 4/16)	Semantic Analysis
Week 7 (4/21, 4/23)	Code Optimization & Generation
Week 8 (**4/28**)	Final Exam

- Assignments: 5 assignments (March, April, May)
  1. OCaml exercises – due: 3/15
  2. Lexing and parsing – due: 4/2
  3. Translator – due: 4/12
  4. Semantic analyzer – due: 4/25
  5. Optimizer – due 5/24