

COSE312: Compilers

Lecture 0 — Course Overview

Hakjoo Oh
2017 Spring

Basic Information

Instructor: Hakjoo Oh

- **Position:** Assistant professor in Computer Science and Engineering, Korea University
- **Expertise:** Programming Languages
- **Office:** 616c, Science Library
- **Email:** hakjoo_oh@korea.ac.kr
- **Office Hours:** 1:00pm–3:00pm Mondays and Wednesdays (by appointment)

TA:

- Sunbeom So (sunbeom_so@gmail.com)
- Myungho Lee (audxo14@gmail.com)

Course Website:

- <http://pr1.korea.ac.kr/~hakjoo/courses/cose312/2017/>
- Course materials will be available here.

What is Compiler?

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.

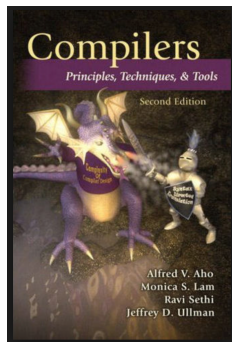
Course Overview

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.



Prerequisites

- Extensive programming experiences
- Major CS courses: discrete maths, data structures, algorithms, programming languages, theory of computation, architecture, etc

Grading

- Programming assignments – 30%
- Midterm exam – 30%
- Final exam – 30%
- Attendance and participation – 10%

Assignment policy:

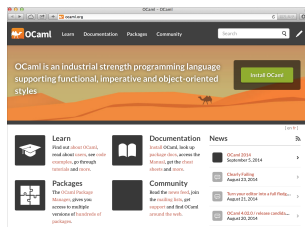
- No late submissions will be accepted.
- All assignments must be your own work.
 - ▶ Copying gets you 0 for the entire HW score. We use software for detecting code clones.

Programming Assignments in ML

ML is a family of programming languages including SML, OCaml, F#, etc.

- Support higher-order, strict, mostly pure, and typed, with algebraic data types.
- Inspired the design of many modern programming languages.
- Suitable for implementing language processors.
- A good deal of syntax.

We will use OCaml:



Schedule (tentative)

Weeks	Topics
1	Introduction
2	Lexical Analysis
3	Lexical Analysis
4	Syntax Analysis
5	Syntax Analysis
6	Syntax Analysis
7	Syntax Analysis
8	Intermediate Representation
9	Mid-term exam
10	Intermediate Representation
11	Code Optimization
12	Code Optimization
13	Program Analysis
14	Program Analysis
15	Code Generation (optional)
16	Final exam

Today: Icebreaking

Introduce yourself (< 1min):

- Free format. Say anything.
- Nothing to talk about? major, grade, interests, hobbies, specialty, goal, motivation for this course, what you expect from this course, etc