Introduction to Compiler Design

Course Introduction

Professor Yi-Ping You (游逸平)
Department of Computer Science
http://www.cs.nctu.edu.tw/~ypyou/

Administrative Stuff

- Course information
  - Introduction to Compiler Design
  - Credit: 3
  - Time: Tuesdays 9:00-12:00
  - Place: EC016
- Course website
  - https://compiler-f23.github.io/
    - The URL is also provided on my Web page
    - A github account is required!
      - Sign up if you don’t have a github account

Identification Registration

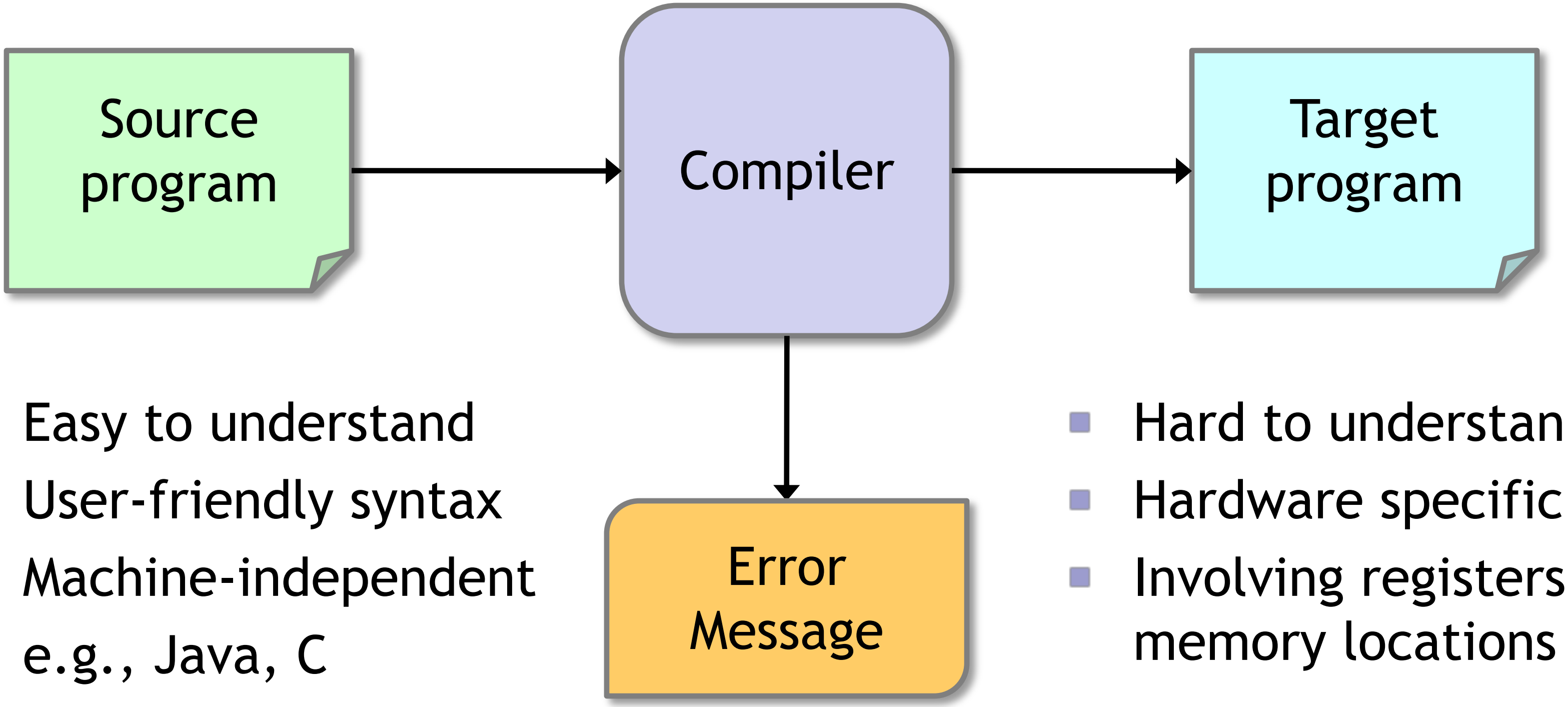
- Google Forms
  - https://ppt.cc/fJ2M3x

Prerequisites

- Programming Languages, Data Structures, and Assembly Language and System Programming
- You are assumed to have experience with programming, especially in C language

What is a Compiler?

- A language translator



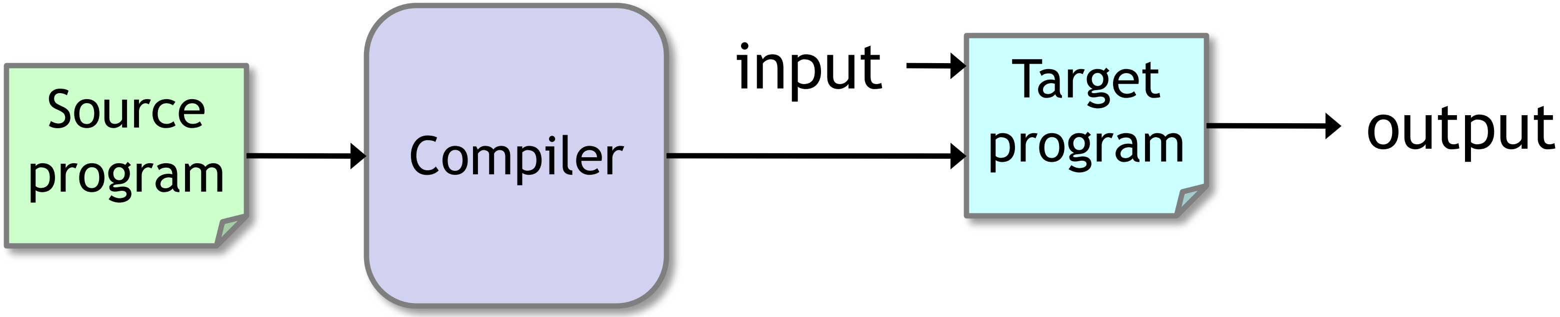
- Easy to understand
- User-friendly syntax
- Machine-independent
- e.g., Java, C

- Hard to understand
- Hardware specific
- Involving registers & memory locations

Terminology

- Compiler:
  - a program that translates an *executable* program in one language into an *executable* program in another language
  - we expect the program produced by the compiler to be better, in some way, than the original
- Interpreter:
  - a program that reads an *executable* program and produces the results of running that program
  - usually, this involves executing the source program in some fashion
- Our course is mainly about compilers but many of the same issues arise in interpreters

Compiler v.s. Interpreter



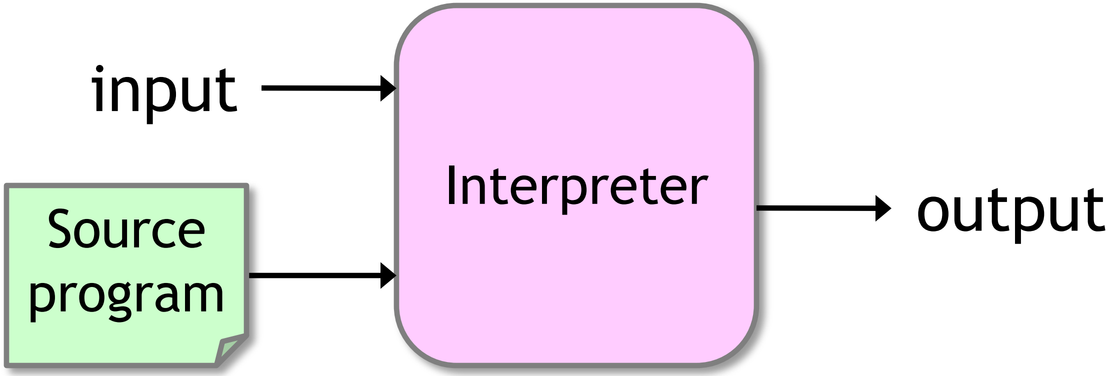
Source program

Compiler

input

Target program

output



input

Source program

Interpreter

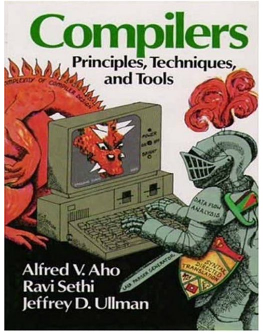
output

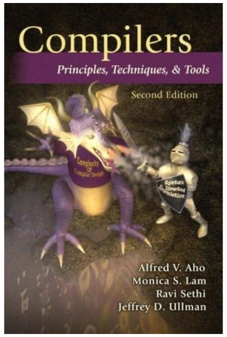
Aims of This Course

- To learn structure of compilers.
- To learn basic techniques used in compiler construction such as lexical analysis, top-down and bottom-up parsing, context-sensitive analysis, and code generation.
- To learn basic data structures used in compiler construction such as abstract syntax trees, symbol tables, three-address code, and stack machines.
- To learn software tools used in compiler construction such as lexical analyzer generators (Lex), and parser generators (Yacc).
- To construct a compiler for a simple language using the above techniques and tools.

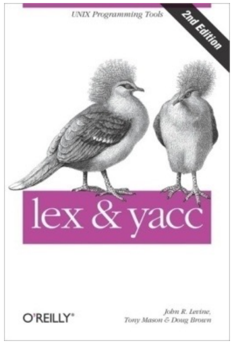
Textbook

- *Compilers: Principles, Techniques, and Tools* (2nd edition), by Aho, Lam, Sethi, and Ullman, Addison Wesley, 2007.
  - A.k.a. dragon book





- Reference books
  - *Lex & Yacc*, by Doug Brown, John Levine, and Tony Mason, O'Reilly Media, 1995.



Lecture Topics (1/2)

- Introduction
  - Course introduction
  - Overview of compilers, phases of a compiler
- Lexical analysis, scanning
  - Regular expressions, lex/flex, start on FSA
  - How lex works, NFAs, DFAs
- Syntax analysis, parsing
  - Context free grammars, ambiguity
  - Yacc/Bison introduction, top-down parsers
  - Top-down parsers, intro to bottom-up parsers
  - Bottom-up parsers
  - LR parsing

Lecture Topics (2/2)

- Context-Sensitive Analysis
  - Syntax-directed definitions, intro to semantic analysis
  - Type checking
- Runtime environments
  - Symbol tables
  - Stack/heap management
- Code generation
  - Generating code for RISC-V assembly language
- Code analysis and optimizations
  - Control-flow analysis and optimizations
  - Data-flow analysis and optimizations

Course Schedule

Week	Subject	Readings	Projects
1	Course Introduction / Overview of Compilers	Ch 1	
2	Regular Languages / Lexical Analysis	Ch 3	
3	Lexical Analysis / Lex	Ch 3	P1 out, 9/26
4	Context-Free Language / Top-Down Parsing		
5	National Day	Ch 4	P1 due, 10/10
6	Top-Down Parsing / Bottom-Up Parsing	Ch 4	
7	Bottom-Up Parsing / Yacc - The Parser Generator	Ch 4, 5	P2 out, 10/24
8	Midterm Exam		
9	Syntax-Directed Translation	Ch 5	
10	Syntax-Directed Translation	Ch 5	P2 due, 11/12 P3 out, 11/14
11	Intermediate-Code Generation	Ch 6	
12	Intermediate-Code Generation	Ch 6	
13	Run-Time Environment	Ch 7	P3 due, 12/3 P4 out, 12/6
14	Run-Time Environment / Code Generation	Ch 7, 8	
15	Code Generation / Control-Flow Analysis and Optimizations	Ch 8, 9	
16	Final Exam		P4 due, 12/24 P5 out, 12/26 P5 due, 1/12

Grading

- Grades will be assigned based on
  - Midterm exam (25%)
  - Final exam (25%)
  - Course project (50%)
    - In 5 parts (10% each)
    - Slackers beware!
      - The penalty for late homework is **15% per day** (weekends count as 1 day). Late homework will not be accepted after the solutions have been posted.
    - NO PLAGIARISM!
      - Homework assignments must be individual work
- These weights are subject to minor variation

Office Hours and TAs

- Contacting me
  - ypyou@cs.nctu.edu.tw
  - Office –EC532
  - Office hours – by appointment
  - Can also catch me right after class
- TAs (Compiler-f23@sslab.cs.nctu.edu.tw)
  - 賴奕廷 - ytlai@sslab.cs.nctu.edu.tw
  - 張娟鳴 - cmchang@sslab.cs.nctu.edu.tw
  - 方嘉賢 - jxfang@sslab.cs.nctu.edu.tw
  - 馮信華 - hhfeng@sslab.cs.nctu.edu.tw
  - Primary contact for project/coding questions