

COMPILER CONSTRUCTION

Course Overview

Chia-Heng Tu
Dept. of Computer Science and Information Engineering
National Cheng Kung University
Spring 2020

NCKU
National Cheng Kung University



Our Team

- Instructor: Chia-Heng Tu
  - chiaheng@mail.ncku.edu.tw
  - Office @ Room 65B03
  - Office hours: by appointment
  - Tel: 06-2757575 ext. 62527

- Angels: 黃柏瑄、黃浩然、林宸玄
  - Office @ Room 65704
    (Advanced Systems Research Lab)
  - Tel: 06-2757575 ext. 62530 #2704
  - Email: asrlab@csie.ncku.edu.tw
    Email subject starts with ``[Compiler2020]''
  - Please check **Moodle** frequently for news update



Class Arrangement

- A 3-hour class is separated into three time slots:
  1. 9:10 ~ 10:30 (1st half)
  2. 10:30 ~ 10:50 (Let's take a nap/rest)
  3. 10:50 ~ 12:00 (2nd half)



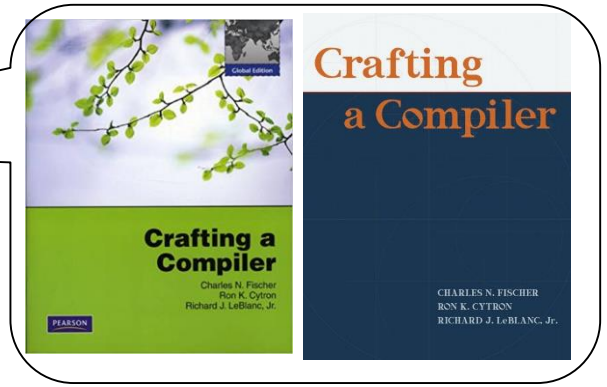
Requirements

- Pre-requisite:
  - Programming in C
  - Computer architecture
  - Computing theory

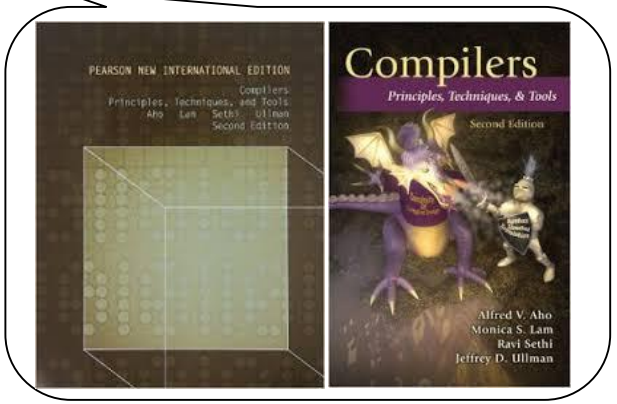
- Efforts:
  - Attend classes (or check the recorded videos offline)
  - Read the slides/textbook(s)
  - Do/Demo the programming HWs
  - Take the quizzes & midterm/final examinations



Textbooks and References



- **Crafting a Compiler*, Pearson, 2010
  - By Fischer, Cytron, and LeBlanc
    (ISBN: 0138017859, 9780138017859)
    (ISBN: 0136067050, 9780136067054)
  - Thank Prof. Jason Jen-Yen CHEN for his course slides
- ***Compilers: Principles, Techniques, and Tools***, Addison Wesley, 2007 (2nd edition) (a.k.a. Dragon Book)
  - By Aho, Lam, Sethi, and Ullman



- *Lex & Yacc*, , O'Reilly Media, 1995
  - By Doug Brown, John Levine, and Tony Mason
- *The JavaTM Virtual Machine Specification*, Addison-Wesley 1999 (2nd edition)
  - By Tim Lindholm and Frank Yellin
- *Jasmin*, an assembler for Java bytecode



Grading

- In-class Quiz: 20%
- Midterm: 20%
- Final: 20%
- Programming Assignments: 40%

These weights are subject to minor variation



In-class Quiz, 20%

- 2 quizzes before Midterm
- 2 quizzes before Final
- It will be announced on the **Moodle** one week before



Midterm and Final, 40%

- It is important that you learn something from the class
- Score is a means to evaluate what you have learned

- ***Zero* tolerance for cheating on the exams**



Programming Assignments, 40%

- Walk through the process of **building a compiler**
  - Translate source code to machine code
    - e.g., C language to Java assembly language
  - **Three assignments** in total
    - Homework #1: 10%
    - Homework #2: 15%
    - Homework #3: 15%
    - Grade: each assignment has **basic** requirements (100%) and may has **optional** achievements (extra points)
  - Submit the code/project to **NCKU Moodle** based on the instructions



Programming Assignments, 40% (Cont'd)

Honor code

- Homework must be **individual work**
  - While you are allowed (and encouraged) to work together in understanding the concepts of the course, sharing of algorithms or code is NOT ALLOWED
  - **Software plagiarism detection tools** will be used to check the similarity of the code you uploaded
    - We will have a chat over a cup of coffee together if your code is ***similar*** with the other(s)



Programming Assignments, 40% (Cont'd)

- Penalty for late upload
  - **30% *discount***
  - **within seven days of the given deadline**

- **Exact deadlines will be announced along with the assignments**



Tentative Time Table (Version I)

3/6	1.	Course Introduction
3/13	2.	Overview & A Simple Compiler
3/20	3.	A Simple Compiler & Theory and Practice of Scanning
3/27	4.	Theory and Practice of Scanning & Grammars and Parsing
4/3	5.	Spring break! No Class!!!
4/10	6.	Lex (**HW #1**) & Quiz 1
4/17	7.	Grammars and Parsing & Top-Down Parsing I
4/24	8.	Yacc (**HW #2**) & Quiz 2
5/1	9.	Midterm
5/8	10.	Top-Down Parsing II & Bottom-Up Parsing I
5/15	11.	Yacc & Jasmin (**HW #3**) & Quiz 3
5/22	12.	Bottom-Up Parsing II
5/29	13.	Intermediate Representations & Runtime Support
6/5	14.	Code Analyses and Optimizations & Quiz 4
6/12	**15.**	**Project demo (A simple compiler)**
6/19	16.	Final

↑ Your senior project demonstration event is taken place on 6/12.

← Compiler homework demo is taken place in the AFTERNOON on 6/12

← Will be updated later



Tentative Time Table (Version II)

3/6	1.	Course Introduction
3/13	2.	Overview & A Simple Compiler
3/20	3.	A Simple Compiler & Theory and Practice of Scanning
3/27	4.	Theory and Practice of Scanning & Grammars and Parsing
4/3	5.	Spring break! No Class!!!
4/10	6.	Lex (**HW #1**) & Quiz 1
4/17	7.	Grammars and Parsing & Top-Down Parsing I
4/24	8.	Yacc (**HW #2**) & Quiz 2
5/1	9.	Midterm
5/8	10.	Top-Down Parsing II & Bottom-Up Parsing I
5/15	11.	Yacc & Jasmin (**HW #3**) & Quiz 3
5/22	12.	Bottom-Up Parsing II
5/29	13.	Intermediate Representations & Runtime Support
6/5	14.	Code Analyses and Optimizations & Quiz 4
6/12	15.	Final
6/19	**16.**	**Project demo (A simple compiler)**

↑ Your senior project demonstration event is taken place on 6/30.

← Compiler homework demo is taken place in the MORNING on 6/12



Why Study Compilation?

- Compilers are important system software components
  - They are intimately interconnected with **architecture, systems, programming methodology, language design, etc.**
- Compilers include many applications of theory to practice
  - Scanning, parsing, static analysis, instruction selection
- Many applications have input formats that look like languages
  - MATLAB, Mathematica
- Writing a compiler exposes practical algorithmic & engineering issues
  - Approximating hard problems; efficiency & scalability



CS Topics Related to Compilers Construction

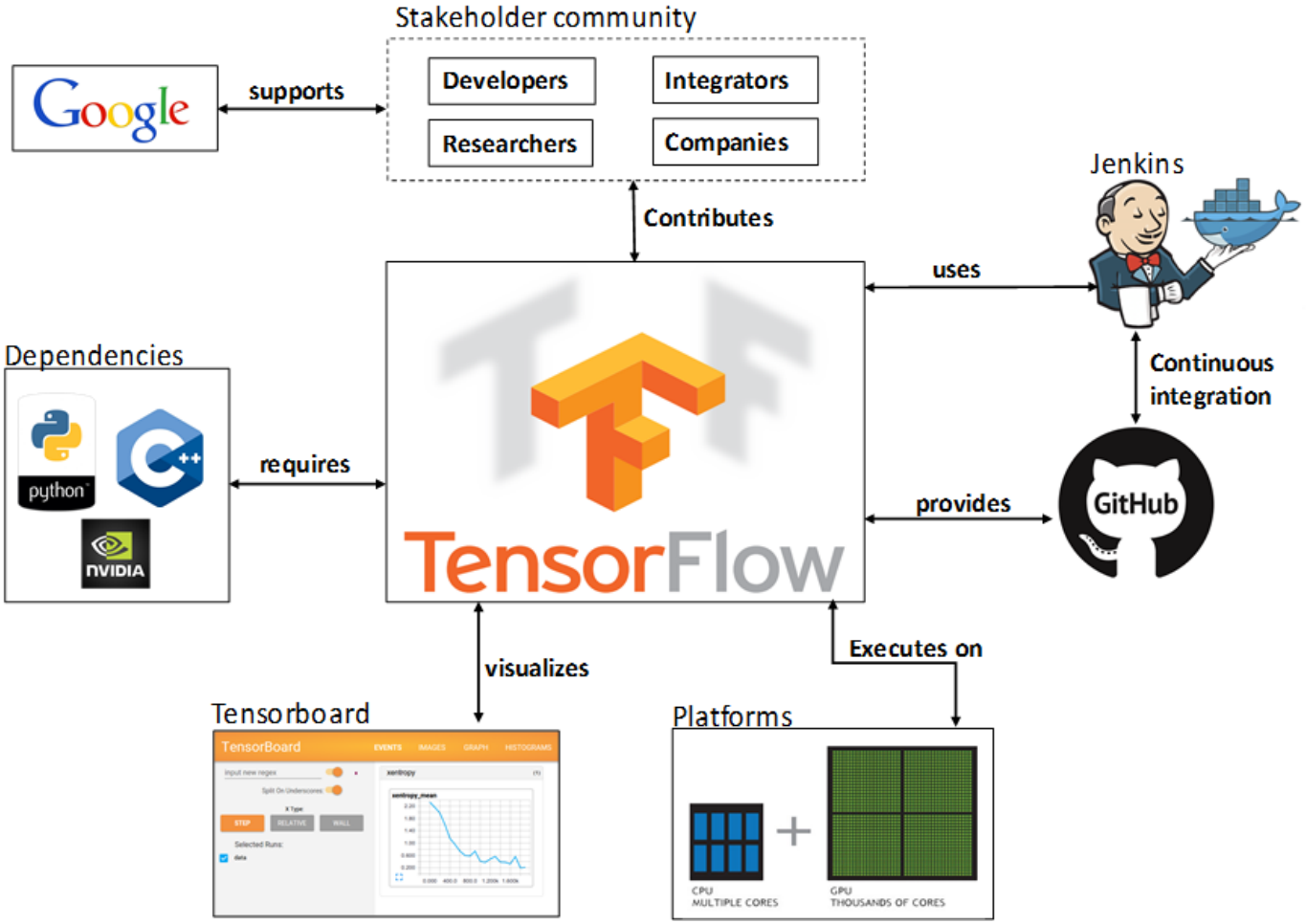
- Theory
  - Finite State Automata, Grammars and Parsing, data-flow
- Algorithms
  - Graph manipulation, dynamic programming
- Data structures
  - Symbol tables, abstract syntax trees
- Systems
  - Allocation and naming, multi-pass systems, compiler construction
- Computer Architecture
  - Memory hierarchy, instruction selection, interlocks and latencies, parallelism
- Security
  - Detection of and Protection against vulnerabilities
- Software Engineering
  - Software development environments, debugging
- Artificial Intelligence
  - Heuristic based search for best optimizations



Challenging and Interesting Problems

- Compiler Construction poses Challenging and Interesting Problems:
  - Compilers must do a lot but also run fast
  - Compilers have primary responsibility for run-time performance
  - Compilers are responsible for making it acceptable to use the full power of the programming language
  - Computer architects perpetually create new challenges for the compiler by building more complex machines
  - Compilers must hide that complexity from the programmer
  - Success requires mastery of complex interactions

Surrounding Elements & TensorFlow Library





QUESTIONS?