



COMPILER CONSTRUCTION

Course Overview

Chia-Heng Tu
Dept. of Computer Science and Information
Engineering
National Cheng Kung University
Spring 2017



The A Team

- Instructor: Chia-Heng Tu (涂嘉恒)
 - chiaheng@mail.ncku.edu.tw
 - Office @ Room 65B03
 - Office hours: by appointment
 - Tel: 06-2757575 ext. 62527
- TA: 鄭沐軒、何育萱、王紹華
 - Office @ Room 65708
(Advanced Systems Research Lab)
 - Tel: 06-2757575 ext. 62520 #2709
 - Email: asrlab@csie.ncku.edu.tw
Email subject starts with ``[Compiler2017]``
 - 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)



Class Arrangement (Cont'd)

- We will cover more than what are in the book(s)
 - Which could prepare you for the future
- If possible, at the beginning of each class, I will:
 - share the latest **Tech News** with you, or
 - introduce example applications of the *compiler technology*
- **Comments/feedbacks of this course are welcome**



More about me ...

- Assistant Professor, CSIE, NCKU (Summer 2016)
- Postdoctoral Researcher, MediaTek-NTU Research Center, NTU (2015 – 2016)
- R&D Manager, Institute for Information Industry, 2012 – 2015
(For Research and Development Substitute Services)
- Teaching Assistant, NTU & NTOU, 2003-2011
- Engineer Intern, Qualcomm CDMA Technologies Taiwan, Summer 2007
- Ph.D. from GINM, NTU (2005-2012)
 - **Performance and Power Profiling with Emulated Systems**
 - Supervised by *Prof. Shih-Hao Hung*
- B.S. & M.E. from CS, NTOU (1999-2005)
 - **Hierarchical Shape Analysis**
 - Supervised by *Prof. Yuan-Shin Hwang*





My Research Interests

- Heterogeneous Parallel Computing: Application-driven performance enhancement with accelerators
 - Optimizations of parallel **software**
 1. Applications running in user- and kernel-space
 2. Caffe and TensorFlow for machine learning
 3. OpenMP, OpenCL, and CUDA for computation offloading
 - Design and optimizations of parallel **computer architecture**
 - E.g., Multicore CPU, GPU, FPGA, and DSP
 - Compiler plays an important role for efficient execution



Requirements

- Pre-requisite:
 - Programming in C
 - Computer architecture
 - Computing theory
- Efforts:
 - Attend classes
 - Read the slides/textbook(s)
 - Do/Demo the programming HWs
 - Take the quizzes & midterm/final examinations



Textbooks and References

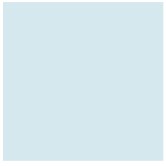
- **Crafting a Compiler*, Pearson, 2010
 - By Fischer, Cytron, and LeBlanc
 - Thank Prof. Jason Jen-Yen CHEN for his [course slides](#)
- *Compilers: Principles, Techniques, and Tools*, Addison Wesley, 2007 (2nd edition) (a.k.a. Dragon Book)
 - By Aho, Lam, Sethi, and Ullman
- *Lex & Yacc*, , O'Reilly Media, 1995
 - By Doug Brown, John Levine, and Tony Mason
- [*The Java™ 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: 25%
- Final: 25%
- Programming Assignments: 30%

These weights are subject to minor variation



In-class Quiz, 20%

- 2~3 quizzes before Midterm
- 2~3 quizzes before Final
- It will be announced on the **Moodle** one week before



Programming Homework, 30%

- Walk through the process of **building a compiler**
 - Translate source code to machine code; e.g., C to Java Bytecode
 - **Three assignments (30%, 30%, 40%)** in total
 - Grade: each assignment has **basic** requirements (100%) and **optional** achievements (extra points)
 - Submit the code/project to **NCKU Moodle** based on the instructions



Programming Homework, 30% (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
 - I will buy you a coffee if your code is **similar** with the other(s)



Programming Homework, 30% (Cont'd)

- The penalty for late upload is **20% discount**
 - if an assignment is one week later than its deadline
 - up to **30% discount**

Example scenarios:

- if you were unable to upload the assignment **#1** one week after the deadline of **#1**, you will get at most 80 for your assignment **#1**
- if you were unable to upload the assignment **#1** before the demonstration of the assignment **#3**, you will get at most 70 for your assignment **#1**
- **Exact dates will be announced along with the assignments**



Tentative Time Table

2/24	1.	Introduction and Overview
3/3	2.	A Simple Compiler
3/10	3.	Theory and Practice of Scanning
3/17	4.	Lex (HW #1)
3/24	5.	Grammars and Parsing
3/31	6.	Top-Down Parsing I
4/7	7.	Top-Down Parsing II
4/14	8.	Midterm
4/21	9.	Bottom-Up Parsing I
4/28	10.	Yacc (HW #2)
5/5	11.	Bottom-Up Parsing II
5/12	12.	Syntax-Directed Translation
5/19	13.	Intermediate Representations
5/26	14.	Code Generation for a Virtual Machine (HW #3)
6/2	15.	Runtime Support
6/9	16.	Target Code Generation
6/16	17.	Final
6/23	18.	Project demo (A simple compiler)

← Could be changed

← Check Moodle



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



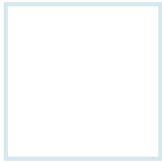
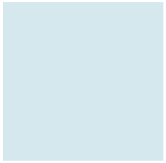
CS Topics Related to Compilers Construction

- 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



QUESTIONS?