Compiler Technology of Programming Languages

syllabus

Farn Wang 王凡

farn@ntu.edu.tw

Instructor and TA

Instructor: 王 凡 教授

- Office Hours: Tuesday 12-13, BL 616
- farn@ntu.edu.tw
- http://cc.ee.ntu.edu.tw/~farn/courses/Compiler

TA：陳煒凱

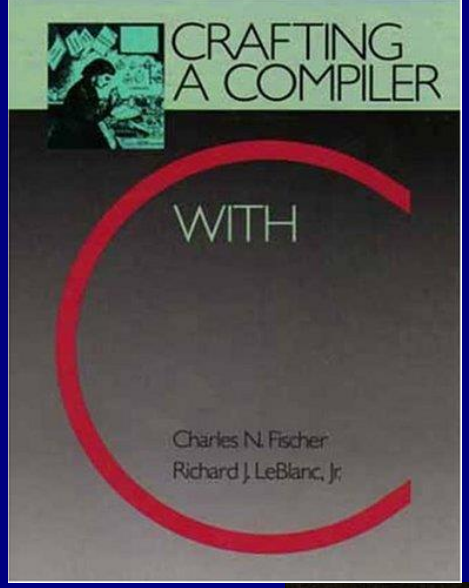
(d09921029@g.ntu.edu.tw)

Adding in the course

(2022/09/06)

- Open to all students in the school course add/delete system.
- Students do not need the signature or any procedure from the teacher to add in this course.

Textbooks



C.N. Fischer, R.K. Cytron,
and R.J. LeBlanc, Jr.

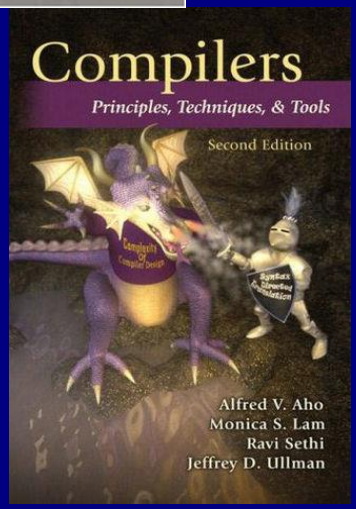
Crafting a Compiler with C

Pearson

開發圖書公司

23585 新北市中和區中山路二段327巷1號6樓,

(02) 8242-3988



- Reference book:

A.V. Aho, M. Lam, R. Sethi, and J.D. Ullman

Compilers: Principles, Techniques and Tools

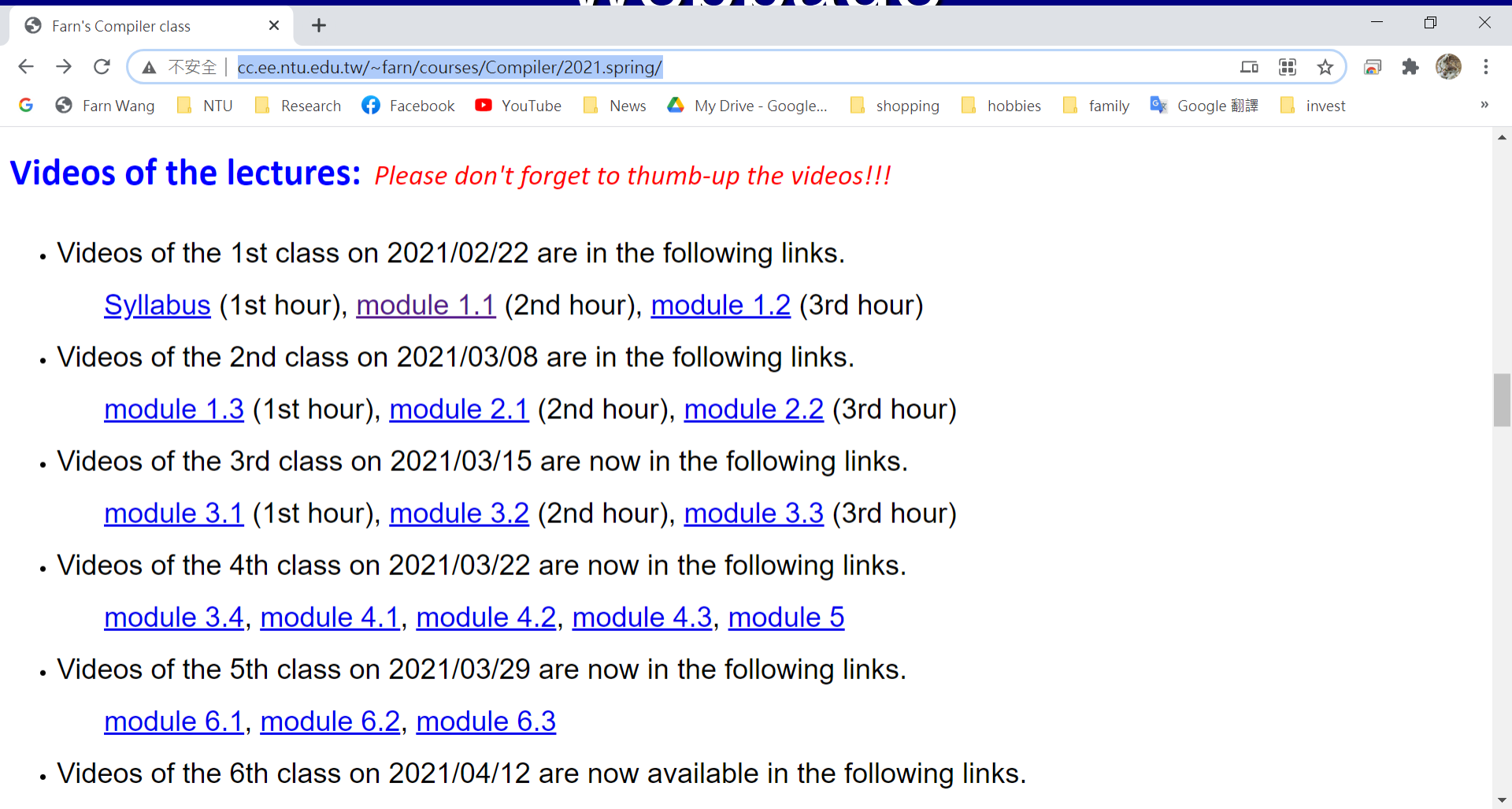
the ***Dragon*** book, 2nd edition, 2007 (1st edition in 1986)

Resources

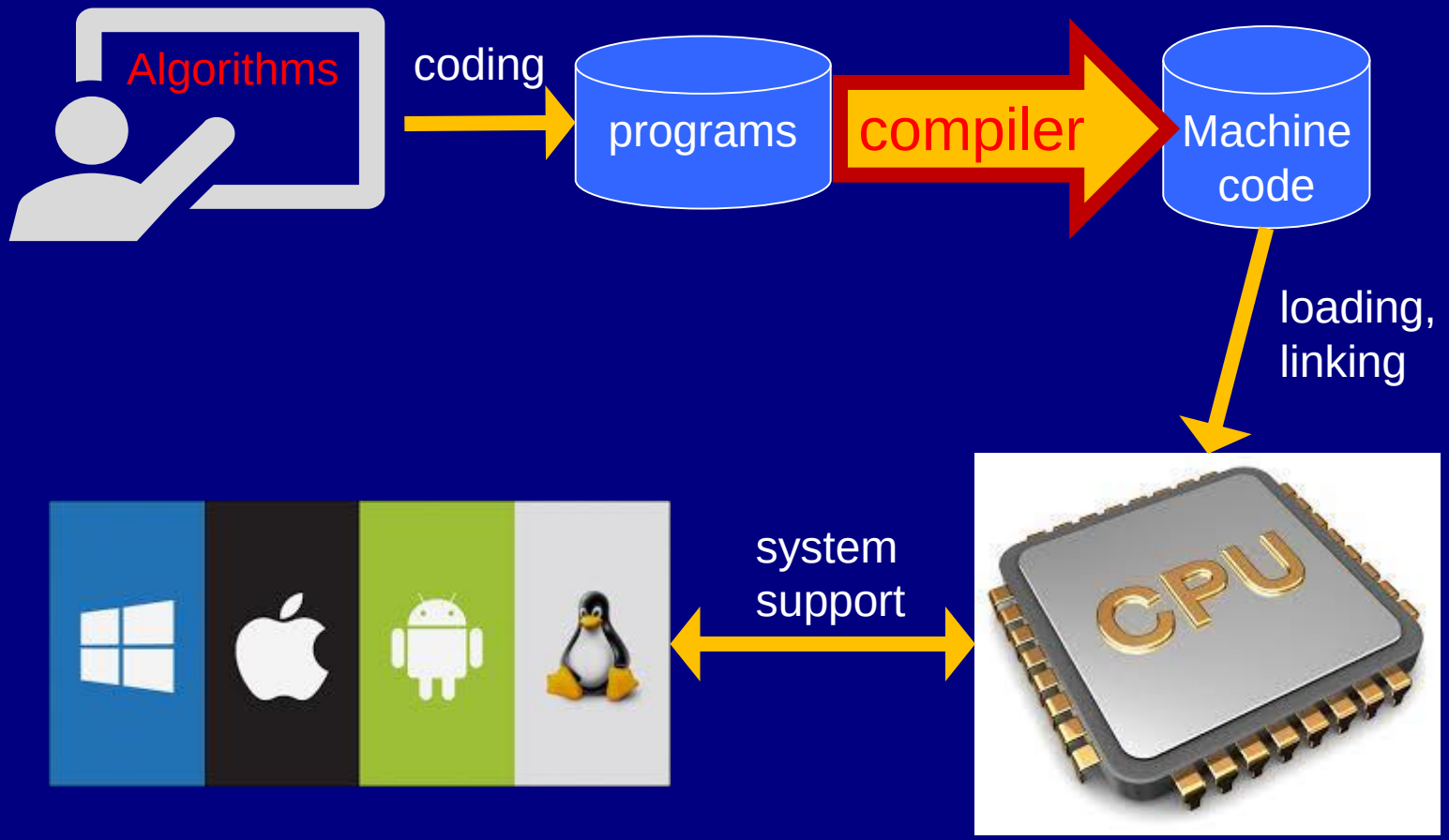
- Teacher's homepage
  http://cc.ee.ntu.edu.tw/~farn
  http://cc.ee.ntu.edu.tw/~farn/courses/Compiler/index.htm
- NTU COOL
- CEIBA
- Course videos uploaded before the each week's class.

Thanks to Prof. Wei Chung Hsu（徐慰中）for sharing the course matierials.

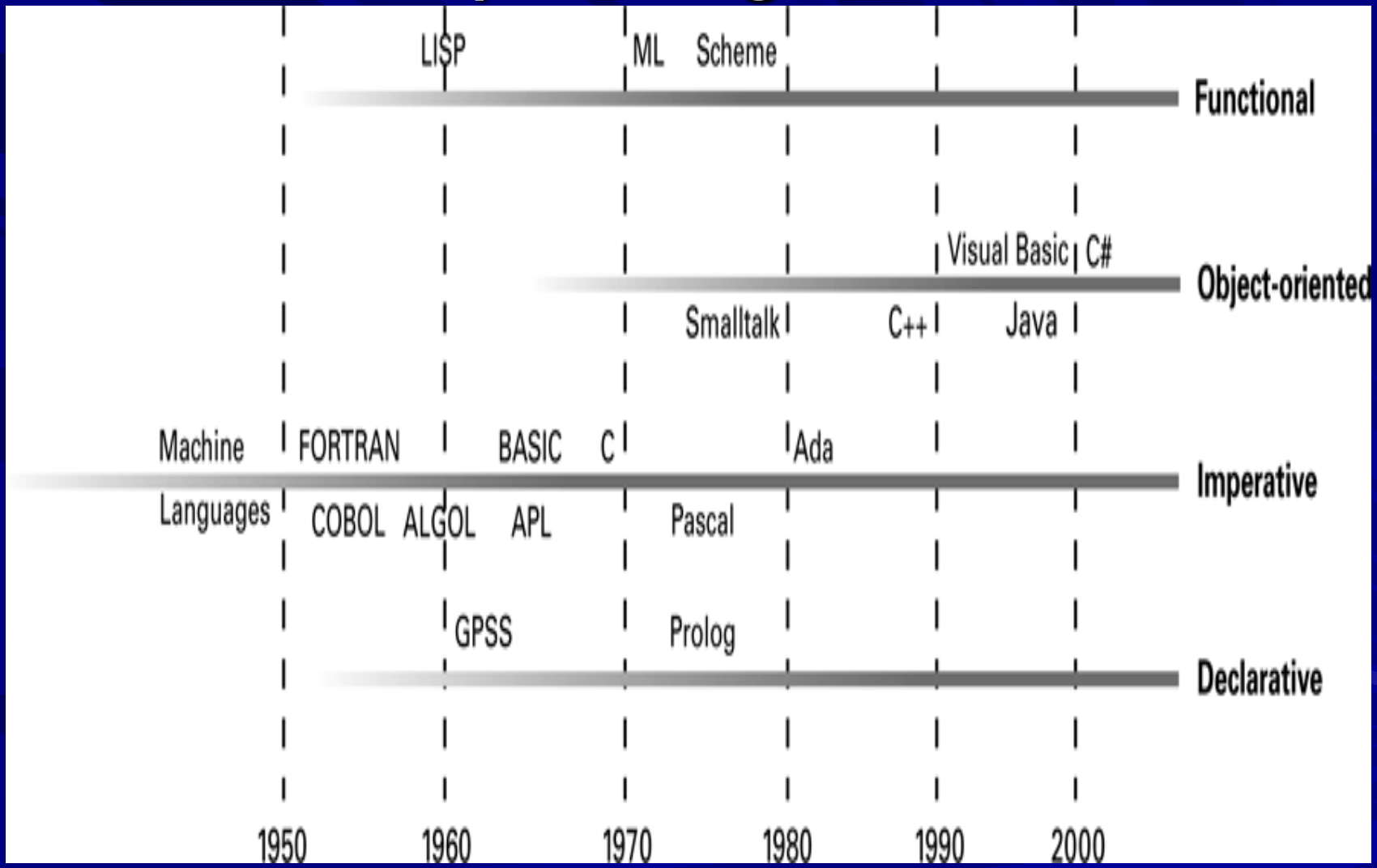
Video links in the course webpage



What are compilers ?



The evolution of programming paradigms



Why compilers

- Machine code is tedious!
  - Instructions, registers
- CPU is very complex for app programmers to run efficiently.
  - Caches, hardware accelerators, GPU, OOO
- Software crisis

Software crisis

the driving force of technical evolution

- Software becomes complicated.
- Modern software products need integrated design consideration in
  - CPU architecture,
  - compilation,
  - algorithms, and
  - user-experience

  to achieve competitive performance.

Software Crisis

Year	Operating System	SLOC (Million)
1993	Windows NT 3.1	4-5
1994	Windows NT 3.5	7-8
1996	Windows NT 4.0	11-12
2000	Windows 2000	>29
2001	Windows XP	40
2003	Windows Server 2003	50

Vincent Maraia

Build Master, The:Microsoft's Software Configuration Management Best Practices

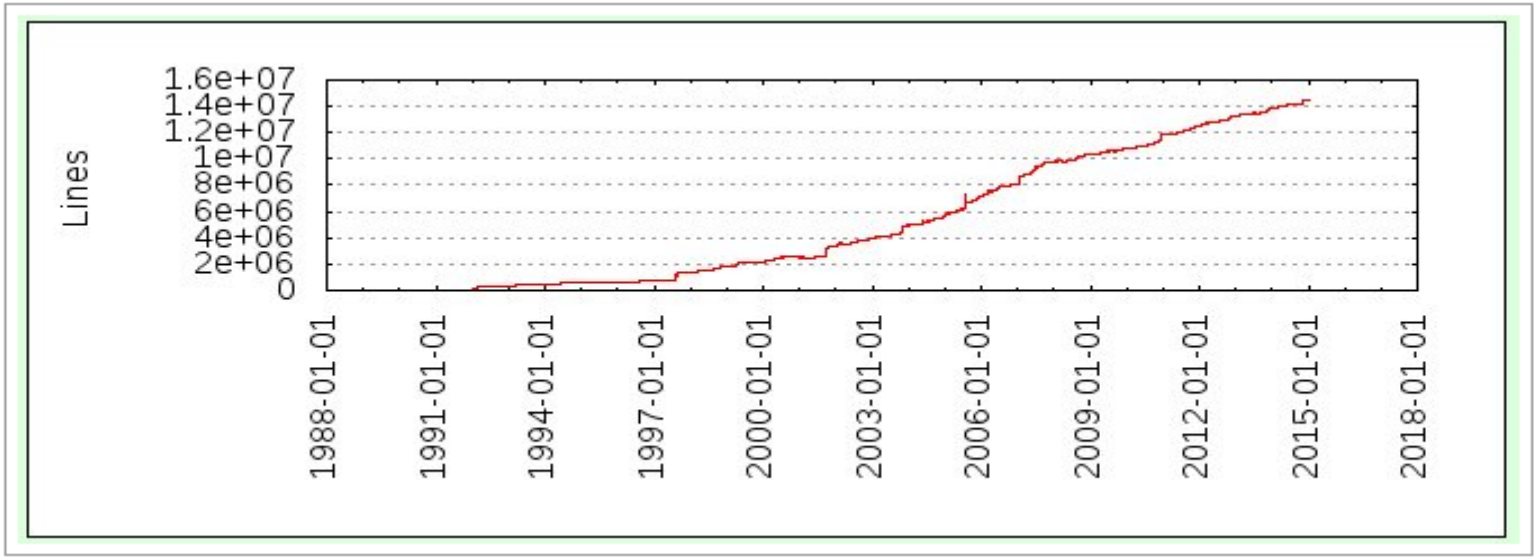
Addison-Wesley

October 2005

ISBN13: 9780321332059, ISBN10: 0-321-33205-9

Software *Crisis*

GCC (4.9) Soars Past 14.5 M Lines Of Code



Linux kernel (4.1) is over 22M Lines Of Code

Software Crisis

Operating System	SLOC (Million)
Debian 2.2	55-59
Debian 3.0	104
Debian 3.1	215
Debian 4.0	283
OpenSolaris	9.7
FreeBSD	8.8
Mac OS X 10.4	86
Linux kernel 2.6.0	5.2
Linux kernel 2.6.29	11.0
Linux kernel 2.6.32	12.6

source lines of code

http://en.wikipedia.org/wiki/Source_lines_of_code

Software Crisis

Productivity of SEers did not scale!

Software	Estimates (LOC/P-month)
Real-time embedded systems	40-160
Systems programs	150-400 LOC/P-month
Commercial applications	200-800 LOC/P-month

* including all necessary activities in software development.

Ian Sommerville

Software cost estimation, chapter 29

Software Engineering, 5th edition, Addison-Wesley

modified by Spiros Mancoridis 1998

Software Crisis

- Software quality is the main deciding factor to the competitiveness of software products.
- Software QA relies on software testing.
- Software testing, direct and indirect, is the main cost in software development!
  - 50% of development cost.
  - 50% of human resources.
  - The cost rises faster than ever.

Software Crisis

Statistically speaking, testing occupies

- 20 percent of the overall development time for a single-component application,
- 20 to 30 percent for a two-component application
- 30 to 35 percent for an application with GUI.
- 35 to 50 percent for a distributed application with GUI.

Software Crisis
- solutions

I double every 18 month!

Oh, oh!
I produce <1000 LOC a month!

AI

Automated programming

Theory

- Program proof!
- Automated verification!
- Algorithm templates!

OS

- System services!
- Abstract API!

SE

- OO
- CI
- Testing
- Disciplene

PL

- Abstraction
- Optimization
- User-friendliness

Software Crisis

Solutions from programming languages

Compilers

- Abstraction
  - Subroutines
  - Libraries
  - OO
- Optimization
  - Runtime, compile-time, OOO, caches
- User-friendliness

Why do you need this course ?

- Learning what it takes in making good apps.
- Understanding the interaction among CPU, OS, and compilers.
- Dealing with CPU, OS, and app people.
- Learning language/document processing.
- Your first-time nontrivial complete system.

It is simply interesting!

Contents

- Fundamentals of compilers.
  - lexical analysis
  - parsing
  - semantics analysis
  - code generation
  - code optimization
- survey assignments on cutting edge research topics.
- Experiments with compilers via projects.

Grading Policy 1/4

- 4 Compiler implementation projects: 40%
- Midterm presentation: 20%
- Final Exam: 40% (in-class)
  - Closed book and internet

Grading Policy 2/4

- 4 Compiler Implementation Projects: 40%
  - ACDC (10%),
    - deadline 2022/10/11
  - Lexical analysis and parsing (10%)
    - Deadline 2022/11/1
  - Semantics analysis (10%)
    - Deadline 2022/11/29
  - Code generation and optimization (10%)
    - Deadline 2022/12/13
  - Team of one! In C or C++!
  - Details: visit the class websites

Grading Policy 3/4

- Midterm presentation: 20%
  - One main paper from the main conference regular papers of ACM SOSP 2018-19, ACM ISCA 2019-20, IEEE ICPP 2019-20, ACM POPL 2019-20, or ACM PODC 2019-20.
    - Affiliated workshop papers are not allowed.
    - Tool and short papers are not allowed.
    - Main paper may not duplicate.
  - 2 papers related to the main paper for comparison.
  - Submit and make a 10-min presentation in PP.
  - Team of one!

Grading Policy 4/4

- Final Exam: 40% (in-class)
- 80% of the points in Final exam are variation from the exercises in the textbook.
- Closed book
- No discussion in person and in network.

Course Schedule

1.	9/6	a) Course Organization, b) Chapter 1: Introduction
2.	9/13	a) Chapter 2: A Simple Compiler
3.	9/20	a) Chapter 3: Scanner; b) Project 1 announcement: a simple compiler
4.	9/27	a) Chapter 4: Grammar and Parsing; b) Deadline of registration of midterm presentation papers!
5.	10/4	a) Chapter 5: Top-Down Parsing;
6.	10/11	a) Project 1 deadline; b) Project 2 announcement: C-Compiler: A Scanner and a Parser
7.	10/18	a) Chapter 6: Bottom-up Parsing
8.	10/25	a) Chapter 7: Syntax Directed Translation; b) Midterm presentation 1. c) Deadline of the midterm presentation report in PPSX!

Course Schedule

9.	11/1	a) Chapter 12: Runtime Support; b) project 2 deadline; c) project 3 announcement: A Type Checker.
10.	11/8	a) Chapter 8: Declaration Processing and Symbol Table, b) Chapter 9: Semantic Analysis
11.	11/15	School celebration
12.	11/22	a) Code Generation for ARMv8; b) Midterm presentation 2.
13.	11/29	a) Code Generation for Data Accesses and Simple Register Allocation; b) project 3 deadline; c) project 4 announcement: Code Generation and optimization
14.	12/6	a) Code Generation for Control Structures b) Code Generation for Arrays, Procedure Calls
15.	12/13	a) Optimization: Parallelization, CSE, Register Allocation, Code Scheduling, b) Vectorization, Memory Hierarchy related optimization, …; c) Project 4 submission deadline!
16.	12/20	Final exam