

# Compiler Technology of Programming Languages

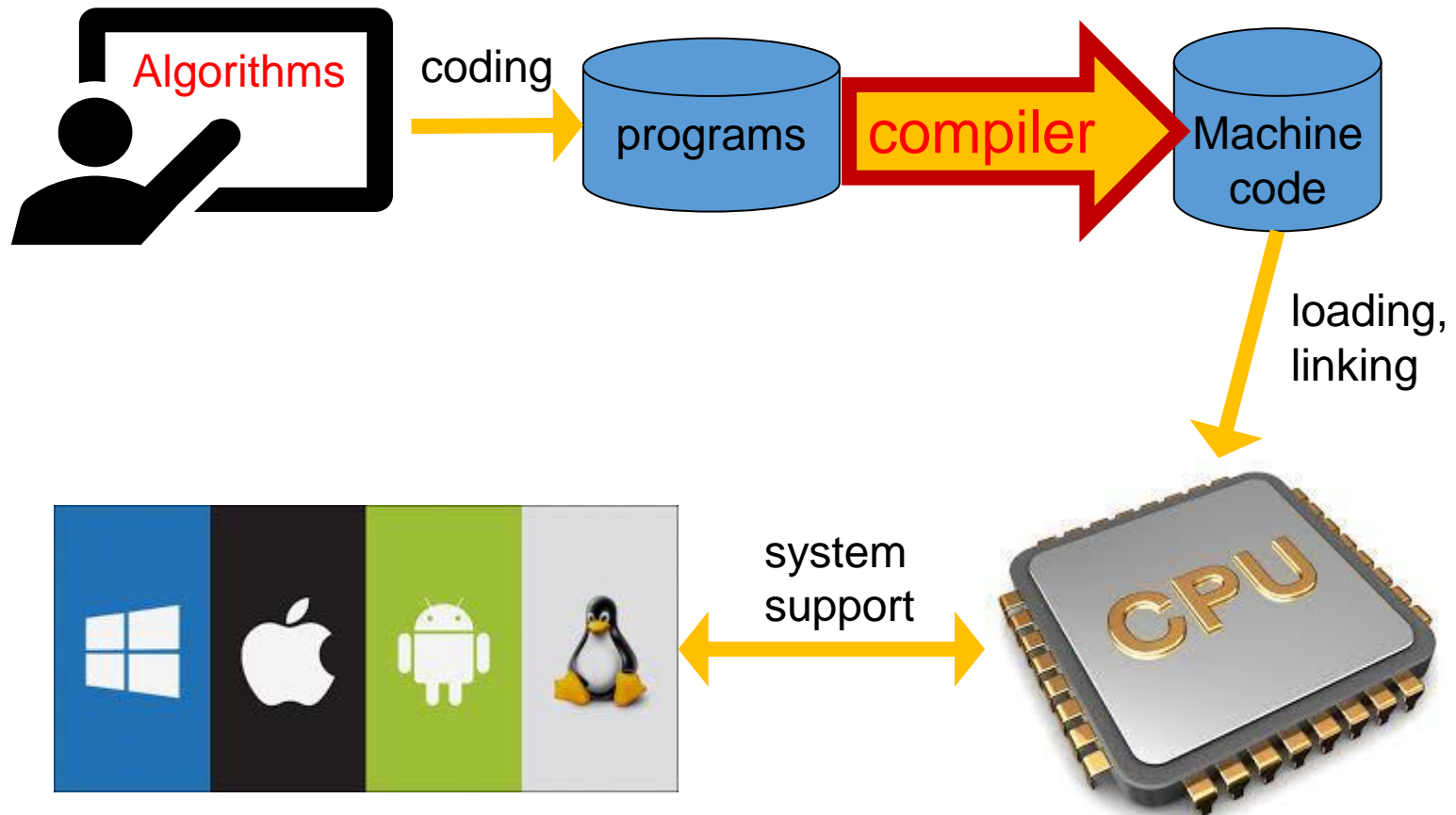
## syllabus

Farn Wang 王凡  
[farn@ntu.edu.tw](mailto:farn@ntu.edu.tw)

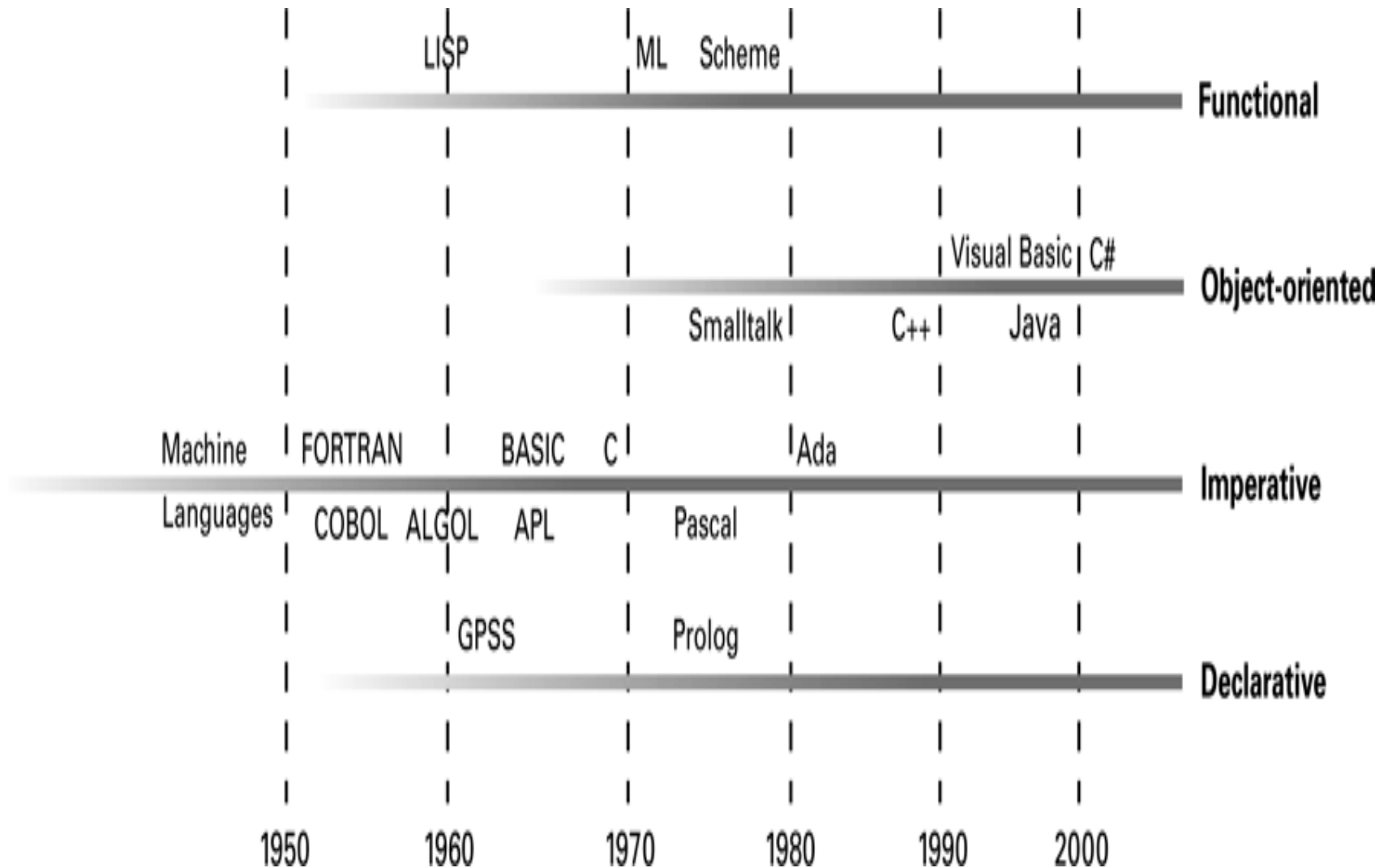
<https://farnking.github.io/Farn-Wang-WWW/courses/Compiler/>

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

# What are Compilers ?



# 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

# Why Learning Compilers?

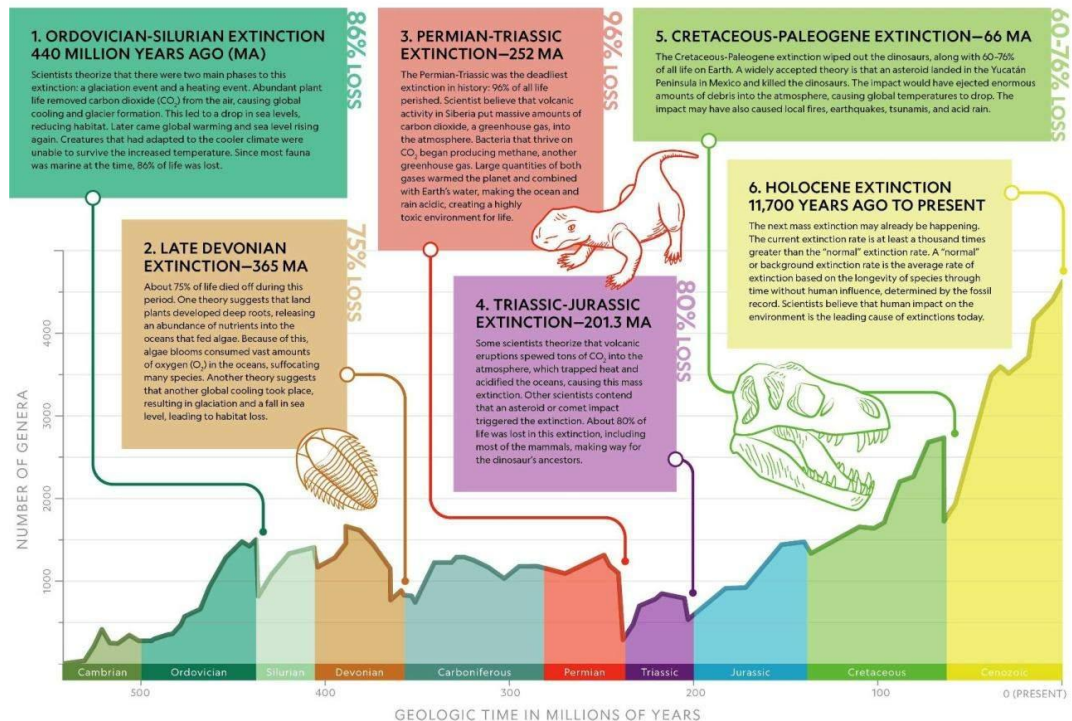
LLM

- Dramatic changes in programming paradigm!
- Massive extinction of average programmers

## MASS EXTINCTIONS

A mass extinction is a sharp spike in the rate of extinction of species caused by a catastrophic event or rapid environmental change. Scientists have been able to identify five mass extinctions in Earth's history, each of which led to a loss of more than 75 percent of animal species.

NATIONAL GEOGRAPHIC



# Why Learning Compilers in the LLM Era?

職缺下滑率最高前十名：

<https://news.cnyes.com/news/id/6113572>

1. 網頁開發工程師 下滑 72%
2. .Net 開發工程師 下滑 68%
3. Java 開發工程師 下滑 68%
4. 前端開發工程師 下滑 67%
5. iOS 開發工程師 下滑 62%
6. 使用者體驗設計師 下滑 61%
7. 程式分析師 下滑 58%
8. 品質保證工程師 下滑 57%
9. 資深 Java 開發工程師 下滑 55%
10. 軟體測試工程師 下滑 53%

職缺成長率最高前十名：

1. SAP 專案主管 成長 356%
2. AI / 機器學習工程師 成長 334%
3. Oracle 人力資源管理系統經理 成長 263%
4. 資料中心技術員 成長 144%
5. SAP 顧問 成長 98%
6. Oracle 顧問 成長 67%
7. 資深軟體工程師 成長 60%
8. 機器學習工程師 成長 59%
9. 技術主管 成長 51%
10. 平台工程師 成長 43%

***Get ready for the drastic paradigm shift!  
Prepare to survive with several deep techniques.***

# 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-experienceto 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*

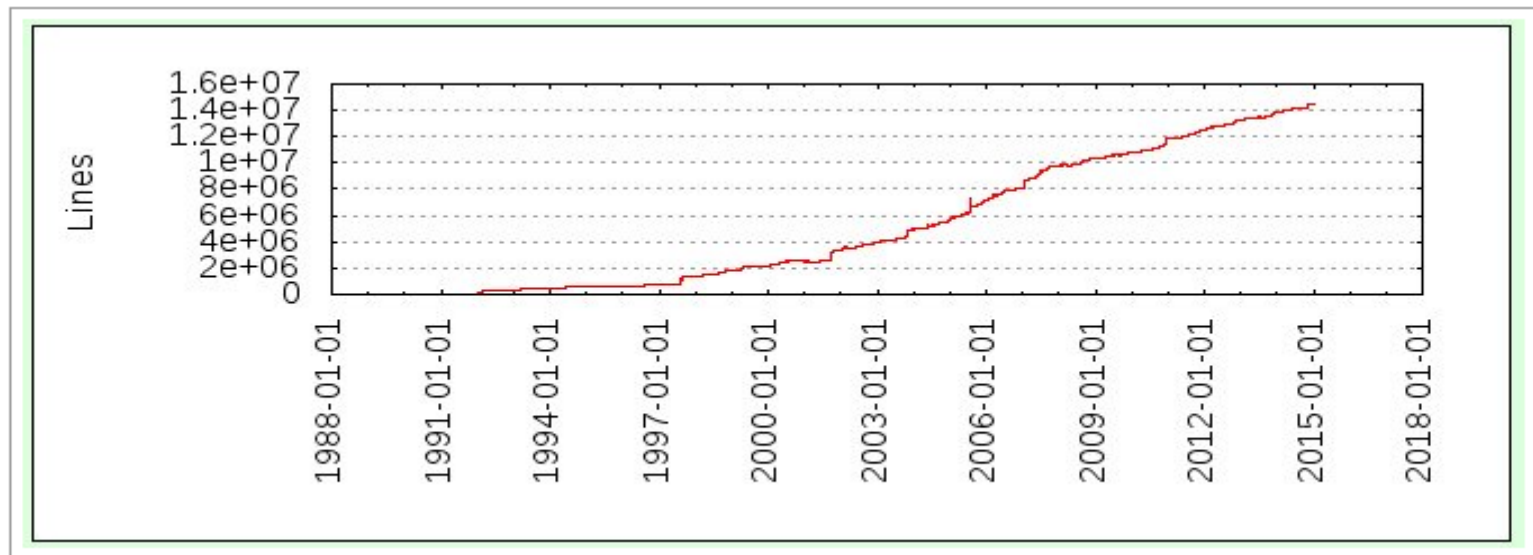
*ISBN13: 9780321332059,*

*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
<a href="#">OpenSolaris</a>	9.7
<a href="#">FreeBSD</a>	8.8
<a href="#">Mac OS X</a> 10.4	86
<a href="#">Linux kernel</a> 2.6.0	5.2
<a href="#">Linux kernel</a> 2.6.29	11.0
<a href="#">Linux kernel</a> 2.6.32	12.6

source lines  
of code

*[http://en.wikipedia.org/wiki/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, 5<sup>th</sup> edition, Addison-Wesley*

*modified by Spiros Mancoridis 1998*

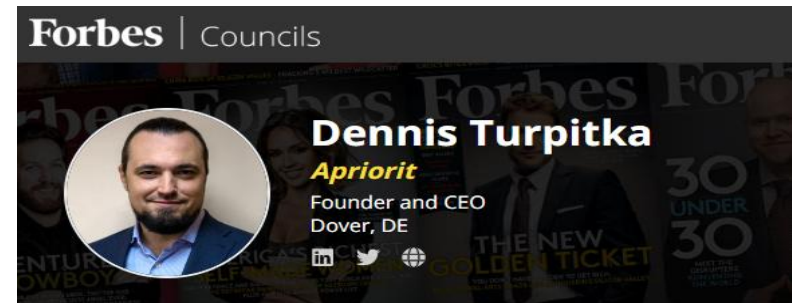
# Software Crisis

- 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.



Founder and CEO of Apriorit, a software development company that provides services to Fortune 500 tech giants. Entrepreneur, 17+ years in cybersecurity.

## Skills

General Management , Business Strategy, Information Security

APRIORIT    

Apriorit provides software product engineering services to the technology development life cycle. We are proud of our work with more than 400 clients.

# Software Crisis

## - solutions

I double every 18 month!

AI

Automated programming

Theory

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

OS

- System services!
- Abstract API!

SE

- OO
- CI
- Testing
- Discipline

PL

- Abstraction
- Optimization
- User-friendliness

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

# Software Crisis

Solutions from programming languages

## *Compilers*

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

# What to learn in 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!**



# Textbooks

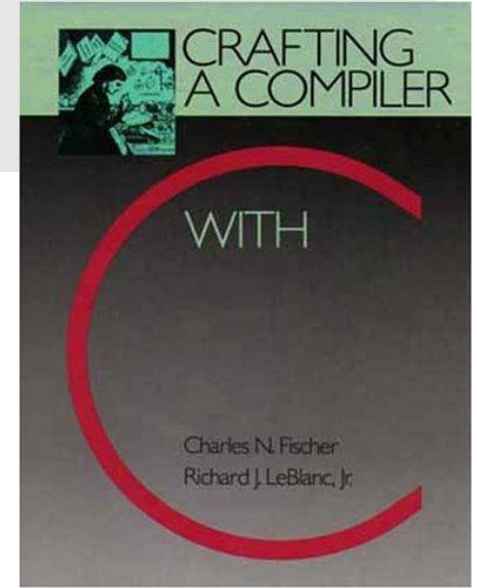
C.N. Fischer, R.K. Cytron, and R.J. LeBlanc, Jr.

## *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

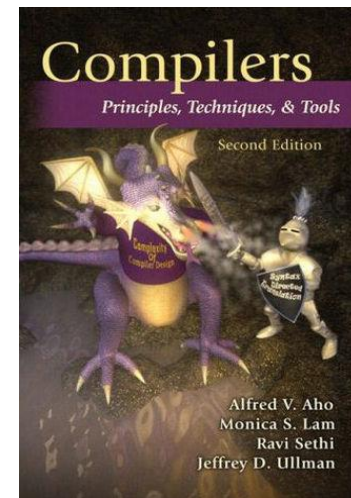


## Reference book:

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

## *Compilers: Principles, Techniques and Tools*

the **Dragon** book, 2<sup>nd</sup> edition, 2007 (1<sup>st</sup> edition in 1986)



# 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 2025/10/17
  - Lexical analysis and parsing (10%)
    - Deadline 2025/11/14
  - Semantics analysis (10%)
    - Deadline 2025/12/5
  - Code generation and optimization (10%)
    - Deadline 2025/12/26
  - 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 2024-25, ACM ISCA 2023-24, IEEE ICPP 2023-24, ACM POPL 2023-24, or ACM PODC 2023-24.
    - 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.	2025/9/5	a) Curriculum Planning, b) Chapter 1: Introduction
2.	9/12	a) Chapter 2: A Simple Compiler
3.	9/19	a) Chapter 3: Scanner; b) Project 1 announcement: a simple compiler
4.	9/26	a) Chapter 4: Grammar and Parsing; b) Deadline of registration of midterm presentation papers!
5.	10/3	a) Chapter 5: Top-Down Parsing;
6.	10/10	National Holiday
7.	10/17	a) Project 1 deadline; b) Project 2 announcement: C-Compiler: A Scanner and a Parser
8.	10/24	National Holiday
9.	10/31	a) Chapter 6: Bottom-up Parsing

# Course Schedule

10.	11/7	a) Chapter 7: Syntax Directed Translation; b) Deadline of midterm reports!
11.	11/14	a) Chapter 8: Declaration Processing and Symbol Table, b) <a href="#">project 2 deadline</a> ; c) <a href="#">project 3 announcement: A Type Checker</a> .
12.	11/21	a) Chapter 9: Semantic Analysis
13.	11/28	a) Machine Code Generation b) Code Generation for Data Accesses and Simple Register Allocation;
14.	12/5	a) Code Generation for Control Structures b) Code Generation for Arrays, Procedure Calls c) project 3 deadline; d) project 4 announcement: Code Generation and optimization
15.	12/12	a) Optimization: Parallelization, CSE, Register Allocation, Code Scheduling, b) Vectorization, Memory Hierarchy related optimization, ...;
16.	12/19	<a href="#">Final exam</a>
17.	12/26	<a href="#">Project 4 submission deadline!</a>



# Instructor and TA

**Instructor: 王 凡 教授**

[farn@ntu.edu.tw](mailto:farn@ntu.edu.tw)

<https://farnking.github.io/Farn-Wang-WWW/>

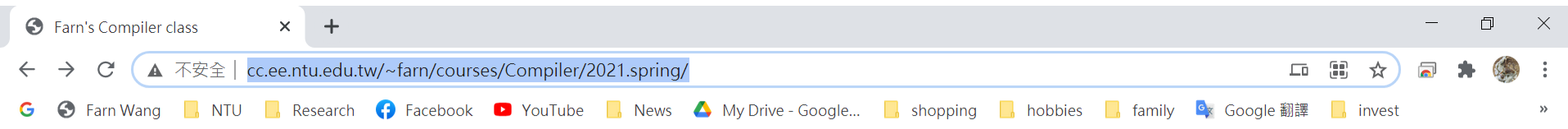
**Teaching Assistant : 楊冠彥**

[f11921091@ntu.edu.tw](mailto:f11921091@ntu.edu.tw)

Office Hours: **Friday 12-13, BL 112**



# Video links in the course webpage



## Videos of the lectures: *Please don't forget to thumb-up the videos!!!*

- Videos of the 1st class on 2021/02/22 are in the following links.  
[Syllabus](#) (1st hour), [module 1.1](#) (2nd hour), [module 1.2](#) (3rd hour)
- Videos of the 2nd class on 2021/03/08 are in the following links.  
[module 1.3](#) (1st hour), [module 2.1](#) (2nd hour), [module 2.2](#) (3rd hour)
- Videos of the 3rd class on 2021/03/15 are now in the following links.  
[module 3.1](#) (1st hour), [module 3.2](#) (2nd hour), [module 3.3](#) (3rd hour)
- Videos of the 4th class on 2021/03/22 are now in the following links.  
[module 3.4](#), [module 4.1](#), [module 4.2](#), [module 4.3](#), [module 5](#)
- Videos of the 5th class on 2021/03/29 are now in the following links.  
[module 6.1](#), [module 6.2](#), [module 6.3](#)
- Videos of the 6th class on 2021/04/12 are now available in the following links.

# Resources

## Teacher's homepage

- <https://farnking.github.io/Farn-Wang-WWW/>
- <https://farnking.github.io/Farn-Wang-WWW/courses/Compiler/>
- NTU COOL
- CEIBA
- Course videos uploaded before the each week's class.