2309 CS106 & CS442 & SE442 Course Outline

Subject Code	:	CS106; CS442; SE442
Subject Title	:	Principles of Compilers (CS106); Compiler Construction (CS442 & SE442)
Course Type	:	Elective
Level	:	1
Credits	:	3
Teaching Activity	:	Lecture 45 hours
Prior Knowledge*	:	C programming
Class Schedule	:	

Class	Week	Time	Classroom	Date
CS106(D1) CS442(D1) SE442(D1)	WED	19:00 – 21:50	C309	01/Sep/2022 - 17/Dec/2022

Instructor	:	Zhiyao Liang
Contact Number	:	8897-2940
E-mail Address	:	zyliang@must.edu.mo
Office	:	Room A324
Office Hour	:	Mon. 13:00 -15:00 Tue. 9:00 - 11:00 Wed. 13:30 -17:30 Thu. 9:00 - 11:00

COURSE DESCRIPTION

This course is designed to introduce the basic concepts and techniques of compilers. It covers some essential aspects of constructing a compiler: specification of programming languages, lexical analysis, syntactic analysis, semantic analysis, run-time environment, and code generation. It also briefly discusses some of the advanced topics. A course project covers different stages of constructing a compiler, including a scanner, parser, semantic analyzer, and code generator. The source language and target language can be freshly designed. The project is non-trivial while feasible. Fundamentals of the theory of computation, including regular expression, grammar, and finite state machines, will be taught and practiced.

TEXTBOOKS

No recommended textbook. The learning materials will be provided to students during the classes.

Reference book:

- Kenneth C. Louden, *Compiler Construction Principles and Practice* ISBN：9780534939724, Publisher: Course Technology, 1997
- Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools, 2ed, Pearson Education, 2006

- Andrew w. Appel, *Modern Compiler Implementation in C*, Cambridge University Press, 1999
- Steven S. Muchnick, *Advanced Compiler Design Implementation*, Academic Press, 1997
- Other suggested books will be introduced at be the beginning of the course.

INTENDED LEARNING OUTCOMES

Upon successful completion of this subject, students will be able to:

1. Understand the basic concepts and principles of compilers.
2. Understand relatively large C program.
3. Implementing different parts of compilers.
4. Learn and apply the knowledge of automata and formal languages that are related to programming language and compilers.
5. Experience working and communication in groups for projects.
6. Understand different aspects of designing a programming language.
7. Apply knowledge and skills of programming language and data structures to construct compilers.
8. Experience cooperation and team works in doing projects.

Weekly Schedule

No.	Topic	Hours	Teaching Method
1	*Introduction*: Paradigms of programming languages, levels of languages, structure of a compiler, brief history of compilers, categories and features of compilers, compiler design issues.	1.5	Lecture and discussion
1	*Language Processors*: Translator & compiler, manipulation of programs, bootstrapping and tombstone diagrams, real and abstract machines.	1.5	Lecture and discussion
2 – 3	*Theory of computation topics*: Natural language & formal language, DFA, NFA, Regular Expressions, regular grammar, formalism of grammar vs. automaton, Chomsky classification, context-free grammars, PDAs, Turing machines.	6	Lecture and discussion
4	*Language Specification*	3	Lecture and

	Backus-Naur Form (BNF), regular expression, Extended BNF, sspects of specification for a programming language, syntax trees.		discussion
5 – 6	*Lexical Analysis* Syntax trees in processing of compiler, from program text to tokens, Disambiguating rules, Finite-state machines and language recognition, Finite state automata, Creating a lexical analyzer, Error handling, Lexical analyzer generator.	6	Lecture and discussion
7 – 8	*Syntactic Analysis* Parsing, Derivation, Construction of parse tree, Handling ambiguity, Top-down parsing, Bottom-up parsing, Error detection and recovery, Recursive-descent parsing, Predictive parsing, LL(*k*) grammar and parsing.	6	Lecture and discussion
9	*Contextual/Semantic Analysis* Identification, Symbol table, Scope, Block structures, Attributes, Standard environment, Type checking, Attribute grammar.	3	Lecture and discussion
10	*Run-time Organization* Data representation, Expression evaluation, Postfix expression, Conversion algorithm, Storage allocation, Routines.	3	Lecture and discussion
11 – 12	*Code Generation* Simple code generation with pure stack machine and pure register machine, Introduction to three-address code. Translation for declaration, assignment, arithmetic expressions, and control flow structures. Syntax-directed translation. Interpretation.	6	Lecture and discussion
13	Advanced topics	3	Lecture and discussion
14 – 15	Review of course content. Discussing the projects of development of a tiny compiler for a specific language.	6	Lecture and discussion on projects

ASSESSMENT APPROACH

Assessment aspect	% weight
Attendance (Class participation)	8%
Assignments and Projects	57%
Final exam	35%
Total	100%

Guideline for Letter Grade:

Marks	Grade
93-100	A+
88-92	A
83-87	A-
78-82	B+
72-77	B
68-71	B-
63-67	C+
58-62	C
53-57	C-
50-52	D
0-49	F

Additional Information

Course-related information and materials, including slides, handouts, notes, and videos, etc., will be distributed online by methods like Moodle and WeChat.