Department of computer science

College of science

University of Cihan / Sulaimaniyah

Subject: Compiler 1 Course Book – Year 3

Lecturer's name: Ardalan Hussein Awlla

Academic Year: 2023/2024

Course Book

1. Course name	Compiler I
2. Lecturer in charge	Mr. Ardalan Hussein Awlla
3. Department/ College	Computer science department /college of science
4. Time (in hours) per week	Theory: 2 hrs. Practical: 2 hrs.
5. Office hours	Monday and Tuesday
6. Course code	CSC3102
7.Teachers academic profile	B.Sc. in Computer Science, University Of Sulaimani, M.Sc. in Computer Science, NUIST. Areas of Specialization: Computer Science , Database management System, Information Security and Programming language
8. Keywords	*Compiler I*

9. Course overview:

The course will enhance students' knowledge on fundamental of Compiler Design will teach students the fundamental concepts and techniques used for building a simple compiler. Focusing on both theory and practice, we will use a sample language to explore the lexical, syntactic and semantic structures of programming languages, and learn to use those structures in implementing a demonstrative compiler. The discussion will also include the examination of intermediate code states, machine code optimization techniques and support. For advanced language features. At the end of the course, students will understand different considerations and phases of compilation, the impact of language attributes upon the compilation process, the effect of hardware feature on the generated code and the practical fundamentals of compiler implementation.

10. Course objective:

To have broad knowledge of principles of basic techniques, theory and tools underline the practice and act of Compiler Construction. This Course introduces the major concept areas of language translation and compiler design.

11. Student's obligation:

Student should be able to contribute significantly to finish his assignments alone and within a group work. Attending lectures will be compulsory to pass this subject.

12. Forms of teaching:

Contact hours: 2 theoretical weekly hours + 2 Lab.

13. Assessment scheme:

Midterm test As (25%)
Midterm Laboratory exam As (15%)
Quizzes and assignments As (10%)
Final Exam theory exam As (35%)
Final laboratory exam As (15 %)

14. **Student learning outcome:**

1. Understand compiler and different phases. Using this translate program from source code to executable code and files. (Knowledge).
2. Able to explain lexical analysis phase and their connection to language definition through regular expressions and grammars. (Comprehensive)
3. Able to explain the syntax analysis phase and differentiate among various parsing techniques and grammar transformation techniques.(Comprehensive)

15. Course Reading List and References:

1. Compilers Principles, Techniques, & Tools, by A.V.Aho, R.Sethi & J.D.Ullman, Pearson Education.
2. Principle of Compiler Design, A.V.Aho and J.D. Ullman, Addition – Wesley.

16. The Topics	**Lecturer's name: Ardalan Hussein Awlla**
Week *1*	**Theory: Introduction to Compilers** 1. What is a compiler 2. Compiler Types? 3. Tasks of Compiler 4. Compiler phases. 5. Why are compilers important?
	Practical: The student will learn the tools that will be used for compiler practical class, and how to compiler and run it.
Week *2*	**Theory: Lexical Analysis** 1. Role of the lexical analyzer. 2. Regular expressions and finite automata. 3. Tokenization and constructing a lexer.
	Practical: Implementing a lexer in C
Week *3*	**Theory: Syntax Analysis** 1. Context-free grammars and BNF notation. 2. Top-down vs. bottom-up parsing. 3. LL(1) and LR(1) parsing
	Practical: We will try to develop programs for recognition (Recursive descent parsing in C)
Week 4	**Theory: Semantic Analysis** 1. Syntax tree and abstract syntax tree (AST). 2. Semantic rules and type checking. 3. Symbol tables and their implementation. 4. Error handling and reporting
	Practical: Error handling and reporting

Week 5	**Theory: Intermediate Representations** 1. Importance of intermediate representations. 2. Three-address code and quadruples. 3. Building an IR for your compiler. 4. Examples of code transformations using IR
	Practical: Examples of code transformations using IR
Week 6	**Theory: Code Generation** 1. Target machine architecture and assembly language. 2. Mapping IR to assembly code. 3. Register allocation and stack management.
	Practical: Generating code in C for a simple language.
Week 7, 8	*Theory:* *Mid-term examination.*
	Practical: *Mid-term examination.*
Week 9	**Theory: Front-End and Back-End** 1. Separation of front-end and back-end in a compiler. 2. Front-end tasks 3. Back-end tasks
	Practical: Modular design in C
Week 10	**Theory: Ambiguity** 1. Parse Tree and Ambiguity. 2. Syntactic Ambiguity. 3. Semantic Ambiguity.
	Practical: implement ambiguity
Week 11	**Theory: Compiler Tools and Libraries** 1. Introduction to tools like Lex and Yacc/Bison.

	2. Using Lex and Yacc/Bison for lexical and syntax analysis. 3. Integration with C code. 4. Advanced tooling for compiler development.
	Practical: Advanced tooling for compiler development
Week 12	**Theory: Associative** 1. Left-Associative (Left-to-Right) 2. Right-Associative (Right-to-Left)
	Practical: We will write programs that simulate left associativity and right associativity.
Week13	**Theory: Case Study - Building a Simple Compiler** 1. Step-by-step implementation of a simple compiler for a custom language. 2. Combining all the concepts learned in previous lectures. 3. Debugging and testing techniques. 4. Performance considerations
	Practical: simple project
Week 14	**Theory : Advanced Topics** 1. Just-in-time (JIT) compilation. 2. Garbage collection. 3. Language-specific challenges (e.g, OOP). 4. Concurrency and parallelism in compilers.
	Practical:
Week15	**Theory: Project Presentation and Discussion** 1. Students present their compiler projects. 2. Discuss challenges faced, design decisions, and future improvements. 3. Q&A and feedback session

17. Peer review

Mr. Ardalan Hussein Awlla