CSC-353: Compiler Construction

General Information

Course Number	CSC-353
Credit Hours	3 (Theory Credit Hour = 3, Lab Credit Hours = 0)
Prerequisite	Theory of Automata (CSC-304)
Course Facilitator	Muhammad Haris

Course Objectives

High-level programming languages like C make programming a breeze, but how do they work? There is a big gap between C and machine instructions for modern computers. This course aims to learn how to translate high-level language programs all the way to low-level language like assembly. This will introduce the principles behind the construction of compilers, including automata, lexical analysis, and syntactic analysis, constructing parser, translation, and code generation. Various features of programming language will be studied through the eye of the compiler designer.

Catalog Description

CSC-353

Assessment Tools & Evaluation Criteria

	Tool Name	Weightage	Criteria
1	Mid Examination	30 Points Each	Correctness of Solutions, Ethics
2	Final Examination	50 Points	Correctness of Solutions, Ethics
3	Assignments / Quizzes /presentations	20 Points	Submission Punctuality

Course Content

Week	Topics	Assignments/ Activity	Suggested Readings
01	• Introduction to the Class & Subject o Why compilers? o A brief history, o Evolution of Programing Languages o The analysis – synthesis model of compilation, o Analysis of the source program, o The phases of a compiler o Types of Compilers o Cousins of Compilers, o Preprocessors, Assemblers o Two-pass Assembly, Loader and link editors		• Book [1], Chapter 1
02	• Compiler Construction Fundamentals o Structure of Compiler o Lexical Analysis o Syntax Analysis o Semantic Analysis o Intermediate Code Generation o Code Optimization o Code Generation o Symbol Table Management o The grouping of phases, o Compiler Construction Tools		• Book [1], Chapter 2

Week 2, 3	• A Simple Syntax-Directed Translator o Introduction, Syntax Definition, o Syntax-Directed Translation, o Parsing, Parse Trees, o Ambiguity o Associativity of Operators, o Precedence of Operators o A Translator for Simple Expressions, o Lexical Analysis, o Symbol Tables o Top-Down Parsing o Predictive Parsing, o When to Use c-Productions, o Designing a Predictive Parser, o Left Recursion, Left Factoring		• Book [1], Chapter 2
Week 04	• Lexical Analysis o The Role of the Lexical Analyzer o Input Buffering o Specification of Tokens o Recognition of Tokens o The Lexical Analyzer Generator Lex o Recognition of Tokens Finite Automata o From Regular Expressions to Automata o Design of a Lexical-Analyzer Generator		• Book [1], Chapter 3
Week 05	• Syntax Analysis o The Role of the Parser o Representative Grammars o Syntax Error Handling o Error-Recovery Strategies o Ambiguity o Context-Free Grammars VS Regular Expressions		Book [1], Chapter 4
Week 07			
Week 08	• Top-Down Parsing o Recursive-Descent Parsing, o FIRST and FOLLOW, o LL(l) Grammars • Bottom up Parsing o Shift-Reduce Parsing o Conflicts During Shift-Reduce Parsing		• Book [1], Chapter 4
Week 09, 10	• LR Parsing o Items and the LR(O) Automaton o The LR-Parsing Algorithm o Constructing SLR-Parsing Tables o Canonical LR(l) Items ▪ Constructing LR(l) Sets of Items ▪ Canonical LR(l) Parsing Tables ▪ Constructing LALR Parsing Tables o Efficient Construction of LALR Parsing Tables		• Book [1], Chapter 4

Week 11	• Syntax Directed Translation ○ Syntax Directed Definition ○ Evaluation Orders of SDDs ○ Applications of Syntax Directed Translation ○ Syntax Directed Translation Schemes ○ Implementing L-Attributed SDDs		• Book [1], Chapter 5
Week 13			
Week 14	• Intermediate Code Generation ○ Variants of Syntax Trees ○ Three-Address Code ○ Types and Declaration ○ Translation of Expressions ○ Type Checking ○ Control Flow ○ Back-patching ○ Switch-Statement		• Book [1], Chapter 6
Week 15, 16	**Project Demonstration and Presentations**		
Week 17	**Final Examination, Practice, Review & Consultation**		

Text Book

1. Aho, A. V., Sethi, R., & Ullman, J. D. (2007). Compilers: principles, techniques, and tools (Vol. 2).
2. Lewis, P. M., Stearns, R. E., & Rosenkrantz, D. J. (1976). Compiler design theory.

Reference Material

1. Cooper, K., & Torczon, L. (2011). Engineering a compiler. Elsevier.
2. Louden, K. C. (1997). Compiler construction. Cengage Learning.
3. Wirth, N., Wirth, N., Wirth, N., Information, S., & Wirth, N. (1996). Compiler construction (Vol. 1). Reading: Addison-Wesley.

CLO – Course Learning Outcomes

1. To understand how compilers translate source code to machine executable.
2. Identify tokens of a typical high-level programming language
3. To comprehend how to perform parsing.
4. To apply parsing algorithms for syntax analysis.
5. To understand how compilers generate machine code.
6. To be familiar with techniques for simple code optimizations.

CLO-GA/PLO Map

CLO ID	GA1	GA2	GA3	GA4	GA5	GA6	GA7	GA8	GA9	GA10	GA11	GA12
CLO 1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLO 2	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLO 3	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLO 4	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLO 5	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
CLO 6	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Approvals

Prepared By	Faryal Shamsi
Approved By	Not Specified
Last Update	15/06/2025