



**KHWAJA FAREED UNIVERSITY OF
ENGINEERING & INFORMATION
TECHNOLOGY (KFUEIT)**

Compiler Construction THEORY CLASS

Course Instructor:

Safdar Hussain

(Lecturer Dept. Computer Science)

safdar2phd@gmail.com

safdar.hussain@kfueit.edu.pk

Audience:

BSCS-7th (A,B & C), MCS-4th (A)

Fall, 2021

Department of Computer Science (KFUEIT)

COURSE DESCRIPTION	
Course Objectives	▪ To teach the students the basic techniques that underlie the practice of Compiler Construction. ▪ Course will introduce the theory and tools to perform syntax-directed translation of a high-level programming language into an executable code. ▪ To train the students for multiple applications including verification and program analysis. ▪ In addition, the course will provide deeper insights into the more advanced semantic aspects of the programming languages.
Learning Outcomes On successful completion of this course, student should be able to	1. Understand the structure of Compilers 2. Understand the basic techniques used in Compiler Construction such as Lexical Analysis, top-down, bottom-up parsing, context-sensitive analysis and intermediate code generation 3. Understand the basic data structures used in compiler construction such as abstract syntax trees, symbol tables, three-address code and stack machines 4. Design and implement a compiler using a software engineering approach 5. Use generators (e.g. Lex and Yacc)
Course Contents	Introduction to Compilers, Lexical Analysis, Syntax Analysis, Parsers Implementation, Semantic Analysis, Intermediate Representation, Code Generation, Code Optimization, Error Detection and Recovery, Error Repair, Compiler Implementation, Compiler Design Options and examples: C, C++, Java and YACC Compilers

METHODS OF TEACHING:

- Lectures
- Presentations
- Assignments
- Group Discussions
- Quizzes
- Case Studies
- Self-Learning

THEORY CLASS SCHEDULE

Lectures Breakdown		
Week#	Lecture#	Description
Week 1 <i>(Introduction)</i>	Lecture 1	Chapter#1 ✓ Introduction to Language Processors (Translator) ✓ Introduction to Compiler, Cousins of Compiler ✓ Compilation, Model of Compilation ✓ Analysis Task for Compilation
	Lecture 2	Chapter#1 ✓ Symbol Table, Error Handler ✓ Synthesis Task for Compilation ✓ Compiler Construction Tools
Week 2 <i>(Introduction)</i>	Lecture 3	Chapter#1 ✓ Challenges to build Compiler ✓ Knowledge required to build Compiler ✓ Compiler Architecture
	Lecture 4	Chapter#1 ✓ Lexical Analysis Notation, Tree Notation ✓ Compiler Phases (Brief look on all 6 phases) ✓ Operations in each phase of Compiler
Week 3 <i>(A Simple Syntax-directed translator)</i>	Lecture 5	✓ Quiz#1 (Tentative) Chapter#2 ✓ Tokens vs Terminals, Tree Terminology, Grammars ✓ Parse Tree, Derivation, Ambiguity, Associativity & Precedence, left-associative & right-associative grammars ✓ Syntax Statements
	Lecture 6	Chapter#2 ✓ Syntax-Directed Translation (SDT), Postfix Notation ✓ Synthesized & Inherited Attributes, Infix to Postfix Translation, Attributed Grammars ✓ Depth-First Traversal, SDD, Translation Schemes
Week 4 <i>(A Simple Syntax-directed translator)</i>	Lecture 7	Chapter#3 ✓ Parsing Types (Top-down & Bottom-Up) ✓ Recursive-descent parsing, Predictive Parsing ✓ Predictive Parser (pseudo code), Tracing input string using predictive parser
	Lecture 8	Chapter#3 ✓ FIRST & FIRST Sets, Left Recursion, Left Factoring ✓ Translator for Simple Expressions, Adding a lexical Analyzer ✓ Token Attributes, Instructions for stack manipulation
Week 5 <i>(Lexical Analysis)</i>	Lecture 9	Chapter#3 ✓ Interaction of Lexical Analyzer & Parser, Lexical Errors ✓ Tokens, lexemes, patterns, attributes of tokens ✓ String & language operations, RE, Regular definitions, Notational shorthands

	Lecture 10	Chapter#3 ✓ Regular Definitions & Grammars, Extra operations of LA ✓ Implementing Transition Diagrams
Week 6 (Lexical Analysis)	Lecture 11	Chap#3 ✓ Lexical Analyzer Generator design, Lex, Flex, YACC ✓ Structure of Lex Programs, Lex predefined variables & functions
	Lecture 12	✓ Quiz#2 (Tentative) Chap#3 ✓ NFA formal definition, RE to NFA (Thompson Algorithm) ✓ Simulating the combined NFA, empty closure & move
Week 7 (Lexical Analysis)	Lecture 13	Chap#3 ✓ Subset Construction Algorithm, Simulating a DFA ✓ NFA to DFA Conversion (Thompson Algorithm)
	Lecture 14	Chap#3 ✓ RE to DFA conversion directly, nullable(), firstpos(), lastpos(), followpos(), DFA State Minimization, Time Space Tradeoffs
Week 8 (Syntax Analysis)	Lecture 15	Chap#4 ✓ Parser, Error Handling, Viable prefix property, ✓ Error Recovery Strategies, Grammars, Derivation ✓ Chomsky Hierarchy, Left Recursion & its Elimination
	Lecture 16	Chap#4 ✓ Left Factoring, Predictive parsing, FIRST & FOLLOW Sets
Mid Term Exams		
Week 9 (Syntax Analysis)	Lecture 17	Chap#4 ✓ LL(1) Grammars, Predictive parsing variants & its program ✓ Error Recovery (Panic Mode, Phrase-level, Adding Error productions)
	Lecture 18	Chap#4 ✓ Bottom-up parsing, LR Methods (Left-to-right, Rightmost derivation), SLR, CLR, LALR, ✓ Shift Reduce parsing, Operator-precedence parsing
Week 10 (Syntax Analysis)	Lecture 19	Chap#4 ✓ Bottom-up parsing, LR Methods (Left-to-right, Rightmost derivation), SLR, CLR, LALR, ✓ Shift Reduce parsing, Operator-precedence parsing
	Lecture 20	✓ Quiz#3 (Tentative) Chap#4 ✓ ANTLR, Yacc & Bison
Week 11 (Syntax Directed Translation)	Lecture 21	Chap#5 ✓ Syntax-Directed Translation, Synthesized vs Inherited attributes
	Lecture 22	Chap#5 ✓ S-Attributed Definitions, Bottom-up Evaluation, Acyclic

		dependency graphs for Parse Tree, L-Attributed definitions
Week 12 (Syntax Directed Translation)	Lecture 23	Chap#5 ✓ Concrete & Abstract Syntax Trees, Generating Abstract Syntax Trees using YACC
	Lecture 24	Chap#5 ✓ Eliminating Left Recursion from a Translation Scheme
Week 13 (Intermediate Code Generation)	Lecture 25	Chap#6 ✓ Intermediate Code Generation, Static checking, ✓ Variants of Syntax Trees, DAG (Directed Acyclic Graph)
	Lecture 26	Quiz#4 (Tentative) Chap#6 ✓ 3-Address Code, Data Structures of 3-Address Code
Week 14 (Intermediate Code Generation)	Lecture 27	Chap#6 ✓ Basic Blocks Construction, Flow Graphs, Optimization of Basic Blocks, Remove Common Sub-expressions
	Lecture 28	Chap#6 ✓ Copy propagation, Dead-Code Elimination, Code Motion, Induction Variables and Reduction in Strength
Week 15 (Code Generation)	Lecture 29	Chap#8 ✓ The Target Machine ✓ Instruction Selection & Register Allocation
	Lecture 30	Chap#8 ✓ Basic blocks and Flow Graphs ✓ A Simple Code Generator
Week 16 (Code Generation)	Lecture 31	Chap#8 ✓ Peephole Optimization ✓ Instruction Selector Generator
	Lecture 32	✓ Graph-coloring Register Allocator
Final Term Exams		

TEXT BOOK (s):

- Alfred V. Aho, Ravi Sethi and Jeffry D. Ulman, “**Compilers, Principles, Techniques and Tools**”, Addison-Wesley, Second Edition, 2006, (ISBN: 0-201-10088-6).

REFERENCE BOOK (s):

- D. Grune, H. E. Bal, C.J. H. Jacobs, K. G. Langendoen, John Wiley, “**Modern Compiler Design**”, Second Edition, Springer, New York, 2012.
- A. W. Appel, M. Ginsburg, “**Modern Compiler Implementation in C**”, Cambridge University Press, 2002.

IMPORTANT:

Office Help Hours	<i>On demand (any time)</i>
Grading	Evaluation by Assignments, Presentations, quizzes.
Problem Session	On demand
Student Evaluation Criteria	(0%) Attendance (4%) Workshops, Assignments, Case Study (4%) Surprise Tests/Sudden Tests, Quizzes (12%) Project Implementation (30%) Mid Term Paper (Closed Book, Closed Notes) (50%) Final Term Paper(Closed Book, Closed Notes)
Total	
Student Responsibilities	▪ Attendance must ≥ 85 ▪ Due respect to Instructor ▪ Due respect to fellows ▪ Due intension to Subject ▪ Cooperation in Class Non-Ethical behavior will lead to 0% award
Course Instructor	Safdar Hussain
Approved By (HoD)	Dr. Hafeez-ur-Rehman
Dean/Chairman/Program Coordinator	Dr. Saleemullah