RKUNIVERSITY

SYLLABUS

Course Title	Compiler Design	
Course Code	CE702	
Course Credit	Theory	: 03
	Practical	: 01
	Tutorial	: 00
	Credits	: 04

Course Learning Outcomes

The objectives of this course are

- **Improve** the theory and practice of compilation, in particular, the lexical analysis, syntax, and semantic analysis, code generation and optimization phases of compilation.
- **Understand**and**Create** lexical rules and grammars for a programming language.
- **Demonstrate**Flex or similar tools to create a lexical analyzer and Yacc/Bison tools to create a parser.
- **Implement** a lexer without using Flex or any other lexer generation tools.
- **Implement**a parser such as a bottom-up SLR parser without using Yacc/Bison or any other compiler-generation tools.
- **Design**semantic rules into a parser that performs attribution while parsing.
- **Design** a compiler for a concise programming language.

Sr. No.	Name of chapter & details	Hours Allotted
	Section – I	
1	**Introduction to Compilers:** Compilers, Analysis of the source program, The phases of a compiler, Cousins of the compiler, The grouping of phases, Compiler Construction tools.	**03**
2	**LEXICAL ANALYSIS:** Introduction, Role of a Lexical Analyzer, Specification and Recognition of Tokens, Regular Expression, Finite Automata, Regular Expression to Finite Automation.	**06**
3	**SYNTAX ANALYSIS:** Role of a Parser, Context Free Grammars, Top-Down Parsing, Bottom-Up Parsing, LR-Parsing.	**10**

4	**Syntax Directed Translation:** Syntax-Directed Definitions, Construction of Syntax Trees, Bottom-Up, Evaluation of S-Attributed Definitions, L-Attributed Definitions, syntax directed, definitions and translation schemes, Type Checking.	**05**
	Section – II	
5	**Run Time Memory Management:** Source Language Issues, Storage Organization, Storage-Allocation Strategies, and Access to Non local Names, Parameter Passing, Symbol Tables, and Language Facilities for Dynamic Storage Allocation, Dynamic Storage Allocation Techniques.	**08**
6	**Intermediate Code Generation:** Different Intermediate Forms, Syntax Directed Translation Mechanisms And Attributed Mechanisms And Attributed Definition.	**06**
7	**Code Generation:** Issues in the Design of a Code Generator, The Target Machine, Run-Time Storage Management, Basic Blocks and Flow Graphs, Next-Use Information, A Simple Code Generator, Register Allocation and Assignment, The DAG Representation of Basic Blocks, Peephole Optimization, Generating Code from DAGs, Dynamic Programming Code-Generation Algorithm, Code-Generator Generators.	**06**
8	**Code Optimization:** Introduction, source of optimizations, optimization of basic blocks, loops, Global Data Flow Analysis, A Few Selected Optimizations like Command Sub Expression Removal, Loop Invariant Code Motion, Strength Reduction.	**04**

Instructional Method and Pedagogy

- Activities to be conducted for the topics like analysis.
- Small group activities to be conducted for case studies
- Feedback by posing a question, quiz, multiple choice questions.
- Group work assigning real world application
- Power point presentations integrated with video lectures.
- Simulators providing a mock scenario

Reference Books
• A.V.Aho, Ravi Sethi, J.D.Ullman, Compiler Tools Techniques, Second Edition, Addison Wesley • Trembley J.P. And Sorenson P.G.,The Theory And Practice Of Compiler Writing, First Edition, Mcgraw-HillRobert • V. Raghavan,Principles of Compiler Design, First Edition, McGrawHill • Dick Grune, Henri E. Bal, Jacob, Langendoen,Modern Compiler Design, Second Edition, WILEY India
Additional Resources
• NPTEL video lectures of Compiler Design course of Computer Science & Engineering by Prof.Y.N.Srikanth, IISc Bangalore • NPTEL slides of Compiler Design course of Computer Science & Engineering by Sanjeev Aggarwal, IIT Kanpur

Tutorial list

Tutorial: - 1

Design Lexical Analyzer and **develop** sample lexical analyzer for C language.

- Lexical analyzer should recognize identifiers
- Lexical analyzer should recognize digits
- Lexical analyzer should recognize operators
- Lexical analyzer should recognize keywords
- Lexical analyzer should recognize special symbols
- Lexical analyzer should recognize white space
- Develop above rules in C

Tutorial: - 2

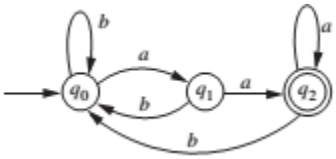
Design regular expression and **implement** sample regular expression for c language.

- Design Regular Expression like (a/b)*ba(a/b)*
- Implement above Regular Expression such that it should accept all the strings belong to that Regular Expression

Tutorial: - 3

Create DFA (Deterministic Finite Automata) for compiler and **demonstrate** that automation in C language.

- Create DFA (Deterministic Finite Automata) like “strings ending with aa”



- Demonstrate above DFA (Deterministic Finite Automata) in C

Tutorial: - 4

Design sample Lexical Analyzer and **implement** the same for C language using LEX tool.

- Lexical analyzer should recognize identifiers
- Lexical analyzer should recognize digits
- Lexical analyzer should recognize operators
- Lexical analyzer should recognize keywords
- Lexical analyzer should recognize special symbols
- Lexical analyzer should recognize white space
- Implement above rules with the help of LEX tool

Tutorial: - 5

Design sample syntax Analyzer and **implement** the same for C language using YACC tool.

- Syntax Analyzer should recognize syntax errors like Missing parenthesis, Missing semicolons etc.
- Implement above rules with the help of YACC tool

Tutorial: - 6

Prepare Research report on compiler Technology.

- **Report** must contain following points
    - Introduction to compiler
    - Architecture of compiler
    - Rules and syntax for compiler
    - Comparison with another compiler