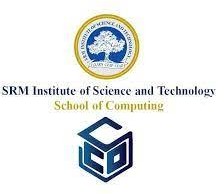
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY FACULTY OF ENGINERING AND TECHNOLOGY SCHOOL OF COMPUTING



COURSE PLAN

21CSC304J COMPILER DESIGN

JANUARY – MAY 2024

## Revision History:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Version | Modification  done | Modified by | Reviewed by | Authorized by |
| 10-01-2024 | 1.0 | Initial Release | K.Shantha Kumari | Dr.S.S. Sirdhar |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Table of Contents

[1.0 General Details 3](#_bookmark0)

[2.0 Reference Books 3](#_bookmark1)

[3.0 Prerequisites 3](#_bookmark2)

[4.0 Instructional Objectives 3](#_bookmark3)

[5.0 Overall Assessment Plan 4](#_bookmark4)

[6.0 Tentative Test Schedule 4](#_bookmark5)

[7.0 Detailed Test Plan 5](#_bookmark6)

[8.0 Compiler Construction challenge / Hackerrank challenges 6](#_bookmark7)

[9.0 QUIZ from GATE Syllabus / Lab Exercises 7](#_bookmark8)

[10.0 Innovative Teaching Pedagogy Planned 7](#_bookmark9)

[11.0 Detailed Session Plan 9](#_bookmark11)

[12.0 Overall Execution Plan](#_bookmark11) 11

# 1.0 General Details

Course Code: 21CSC304T

Course Title: Compiler Design

Semester: VI

Course Time: JANUARY – MAY 2024 Slot: D ; Batch 2

|  |  |  |
| --- | --- | --- |
| Day | Batch 2 | |
| Hour | Timing |
| Day order 1 | - | - |
| Day order 2 | - | - |
| Day order 3 | 9 | 3.15 - 4:05 PM |
| Day order 4 | 1,2 | 8:00 – 9:40 AM |
| Day order 5 | 10 | 4:05 – 4:55 PM |
| P43,44 | 9:45 – 11:30 AM |

Location: UB514 [Theory]; UB815

# 2.0 Reference Books

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques and Tools, Second Edition, Pearson Education, 2011.
2. S. Godfrey Winster, S. Aruna Devi, R.Sujatha,”Compiler Design”, Yesdee Publishing Pvt.Ltd, 2016.
3. K .Muneeswaran,”CompilerDesign”, Oxford Higher Education, Fourth Edition, 2015.
4. David Galles, “Modern Compiler Design”, Pearson Education, Reprit 2012.
5. Raghavan V., “Principles of CompilerDesign”, Tata McGraw Hill Education Pvt. Ltd., 2010.

# 3.0 Prerequisites

Proficiency in a high-level programming language and a solid understanding of formal languages, automata theory, and basic computer science concepts.

# 4.0 Instructional Objectives

1. To articulate the purpose and components of compilers, comprehending each phase, including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization, and code generation.
2. To apply formal language concepts (regular expressions, finite automata, context-free grammars) in designing lexical analyzers and parsers.
3. To implement top-down (recursive descent, LL(1)) and bottom-up (LR(1)) parsing techniques.
4. To perform semantic analysis, including type checking, symbol table management, and scope resolution.
5. To design and generate intermediate code representations using three-address code, quadruples, and triples.
6. To apply optimization techniques (constant folding, dead code elimination, loop optimization) to improve code efficiency.
7. To analyze target machine architecture and implement code generation algorithms, addressing issues like register allocation.

# 5.0 Overall Assessment Plan

|  |  |  |  |
| --- | --- | --- | --- |
| # | Component [CLA 1] | Type | Marks |
| 1 | CLA 1 – T1 | Written Test | 10 |
| Technical Quiz / Review questions | 5 |
| 2 | CLA 1 – T2 | Open Book Test | 10 |
| Compiler Construction challenges 1 | 5 |
| 3 | CLA 1 – T3 | Written Test | 7.5 |
| Compiler Construction challenges 2 | 7.5 |
| Total Marks | | | 45 |

|  |  |  |  |
| --- | --- | --- | --- |
| # | Component [CLA 2]  Lifelong learning | Type | Marks |
| 1 | CLA 2 – T1 | Lab Completion  Lab 1 - Implementation of Lexical Analyzer  Lab 2 conversion from Regular Expression to NFA  Lab 3 Conversion from NFA to DFA  Lab 4 Elimination of Ambiguity, Left Recursion and Left Factoring  Lab 5 -FIRST AND FOLLOW computation | 2.5 |
| Quizzes / Puzzles from GATE Syllabus | 2.5 |
| 2 | CLA 2 – T2 | Lab Completion  Lab 6 Predictive Parsing Table  Lab 7 - Shift Reduce Parsing  Lab 8- Computation of LEADING AND TRAILING  Lab 9 Computation of LR (0) items  Lab 10-Intermediate code generation – Postfix, Prefix | 2.5 |
| Quizzes / Puzzles from GATE Syllabus | 2.5 |
| 3 | CLA 2 – T3 | Lab Completion  Lab 11 Intermediate code generation – Quadruple, Triple, Indirect triple  Lab 12: A simple code Generator  Lab 13 Implementation of DAG  Lab 14: Implementation of Global Data Flow Analysis  Lab 15: Implement any one storage allocation strategies (heap, stack, static) | 2.5 |
| Quizzes / Puzzles from GATE Syllabus | 2.5 |
| Total Marks | | | 15 |

# 6.0 Tentative Test Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Tentative date | Test | Marks | Portion | Duration |
| 1 | 04-03-2024 | CLA 1 – T1 | 15 | Unit 1 and 2 | 100 minutes |
| 05-03-2024 | CLA 2 – T1 | 05 | Lab verification & Quiz | 30 minutes |
| 2 | 04-04-2024 | CLA 1 – T2 | 15 | Unit 3 and 4 | 100 minutes |
| 05-04-2024 | CLA 2 – T2 | 05 | Lab verification & Quiz | 30 minutes |
| 3 | 06-05-2024 | CLA 1 – T3 | 15 | Unit 5 | 50 minutes |
|  | 07-05-2024 | CLA 2 – T3 | 05 | Lab verification & Quiz | 30 minutes |

# 7.0 Detailed Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Tentative Date | Type | Marks | Mode |
|  | 04-03-2024 | Written Test | **Total: 40 Marks**  Exam Pattern:  MCQs – 10 [ 10 Marks]  Concept Understanding Questions – 2  [ 15 Marks]  Scenario based / HOTs Questions – 2 [15 Marks] |  |
| CLA 1 -T1 | Offline |
|  | Technical Quiz / Review Questions | **Total: 10 Marks**  GATE based questions |
| CLA 2 -T1 | 05-03-2024 | Lab verification &Quiz | Total: 5 Marks  3 programs out of 5 + viva | Offline |
|  | 04-04-2024 | Written Test | **Total: 40 Marks**  Exam Pattern:  MCQs – 10 [ 10 Marks]  Concept Understanding Questions – 2  [ 15 Marks]  Scenario based / HOTs Questions – 2 [15 Marks] |  |
| CLA 1 -T2 | Offline |
|  | Compiler Construction challenge 1 | Total: 10 Marks  Review and Explanation |
| CLA 2 -T2 | 05-04-2024 | Lab verification & Quiz | Total: 5 Marks  3 programs out of 5 + viva | Offline |
| CLA 1 -T3 | 06-05-2024 | Written Test | **Total: 20 Marks**  Exam Pattern:  MCQs – 5 [ 5 Marks]  Concept Understanding Questions – 2 [ 8 Marks]  Scenario based / HOTs Questions – 1  [7 Marks] |  |
| Compiler Construction challenge 2 | Total: 10 Marks  Review and Explanation | Offline |
| CLA 2 -T3 | 07-05-2024 | Lab verification & Quiz | Total: 5 Marks  3 programs out of 5 + viva | Offline |

# 8.0 Compiler Construction challenge / Hackerrank challenges

# Compiler for a Toy Language

# Project Description:

# Design and implement a complete compiler for a simple programming language. The language should include features like variables, conditionals, loops, and functions. The compiler should generate intermediate code, perform optimization passes, and produce assembly code for a specific target architecture.

# Deliverables:

# Lexical analyzer, parser, and semantic analyzer

# Intermediate code generator with basic optimizations

# Code generation for a target architecture

# Sample programs in the toy language and their compiled outputs

# Comprehensive documentation detailing the design choices, challenges faced, and lessons learned

# 2. Domain-Specific Language (DSL) Compiler

# Project Description:

# Create a compiler for a domain-specific language tailored for a specific application domain (e.g., scientific computing, game development). The language should have specialized constructs relevant to the chosen domain. Implement lexical analysis, parsing, semantic analysis, optimization, and code generation for a chosen target.

# Deliverables:

# Definition of the DSL and its grammar

# Lexical analyzer, parser, and semantic analyzer for the DSL

# Intermediate code generation and optimization passes

# Code generation for a target architecture

# Case studies demonstrating the DSL's usefulness in the chosen domain

# Documentation detailing the DSL design and compiler implementation

|  |  |  |
| --- | --- | --- |
| Tentative date of  review | Marks | Deliverables |
| 4-4-2024  [CLA1 T2] | Coding – 5  Explanation -5  Total: 10 marks | Lexical analyzer, parser, and semantic analyzer |
| 6-5-2024  [CLA1 T3] | Coding – 5  Explanation -5  Total: 10 marks | Intermediate code representation and optimization passes  Provide a detailed report, along with sample input programs and the corresponding compiled output. |

# 9.0 QUIZ from GATE Syllabus / Lab Exercises

Lexical analysis, parsing, syntax-directed translation. Runtime environments. Intermediate code generation. Local optimization, Data flow analyses: constant propagation, liveness analysis, common sub expression elimination

|  |  |  |
| --- | --- | --- |
| Tentative date of Quiz | Marks | Syllabus |
| 4-3-2024 [CLA1 T1] | 10 Marks | Unit 1 and 2 |
| 5-4-2024 [CLA2 T1] | 2.5 Marks | Lab Exercises 1 to 5 |
| 4-4-2024 [CLA2 T2] | 2.5 Marks | Lab Exercises 6 to 10  Unit 3 and 4 |
| 6-5-2024 [CLA2 T3] | 2.5 Marks | Lab Exercises 11 to 15  Unit 5 |

# 10.0 Innovative Teaching Pedagogy Planned

Five activities will be conducted. One for each unit-wise and score will be calculated for 5 marks.

1. **Role Play**
   1. **Lexer and Parser Interaction: [ Unit 1]**

Scenario: Students can role play as components of the compiler, with some acting as lexers and others as parsers. They can physically interact and simulate the tokenization and parsing process, helping to visualize how these components work together.

* 1. **Compiler Architecture Design: [Unit 1]**

Scenario: Students can engage in a role play where they collectively design the architecture of a compiler. Each student represents a different phase or component of the compiler, and they must collaborate to ensure a seamless integration of these components.

1. **Flipped Classroom**
   1. **Scenario 1: Flipped Classroom for Top-Down Parsing [Unit 2]**

Pre-Class Preparation (Outside of Class):

* + 1. Assigned Readings:

Students are assigned readings, online resources, or video lectures explaining the principles of top-down parsing, including recursive descent parsing and LL(1) parsing.

* + 1. Online Quiz:

Students take an online quiz to assess their understanding of top-down parsing concepts. This can include questions related to First and Follow sets, LL(1) grammars, and parsing table construction.

In-Class Activities:

* + 1. Peer Teaching Sessions:

Students take turns explaining specific concepts related to top-down parsing to their peers. This peer teaching approach reinforces their understanding and encourages active participation.

* + 1. Hands-On Coding:

Students engage in hands-on coding activities where they implement a simple LL(1) parser or analyze and debug existing parser code.

* + 1. Feedback and Reflection:

Towards the end of the class, there is a brief feedback session where students reflect on their learning experience, ask additional questions, and provide input on what topics need further clarification.

* 1. **Scenario 2: Flipped Classroom for Bottom-Up Parsing**

Pre-Class Preparation (Outside of Class):

* + 1. Assigned Videos:

Students watch instructional videos on bottom-up parsing techniques, including LR(0), SLR(1), and LALR(1) parsing. The videos explain shift-reduce and reduce-reduce conflicts, parsing table construction, and conflict resolution strategies.

* + 1. Online Exercises:

Students engage in online exercises, such as constructing LR(0) and SLR(1) parsing tables for given grammars, to reinforce their understanding of bottom-up parsing concepts.

In-Class Activities:

* + 1. Simulation Exercise:

In class, students participate in a simulation exercise where they act as items in a parsing stack. The instructor guides them through the process of a shift-reduce parsing step, providing a physical representation of the parsing process.

* + 1. Hands-On Coding:

Students engage in hands-on coding activities where they implement a simple LR(1) parser or analyze and debug existing parser code. This coding session allows them to apply theoretical knowledge to practical scenarios.

* + 1. Interactive Discussions:

Class time is devoted to interactive discussions on challenging aspects of bottom-up parsing, addressing questions raised by students during the pre-class preparation.

1. **Interactive Simulations:** Use online tools or simulations to visually represent abstract concepts such as lexical analysis, parsing, and code generation. [ Unit 2,3 and 4]
2. **Project-Based Learning (PBL)** : Compiler Construction tools [ Unit 1 to 5]
3. **Gamification Elements** : Introduce gamification elements, such as coding challenges, quizzes, or leaderboards, to make learning more engaging and competitive. [ Unit 1 to 5]

# 11.0 Detailed Session Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Topics to be covered | Hours | Ref | Teaching method | Testing method |
| Unit 1 – Introduction | | | | | |
| 1 | Compilers-Phases of Compiler-Cousins of the Compiler | 1 |  | PPT | Illustration using example |
| 2 | Grouping of Phases-Compiler construction tools | 1 |  | PPT | Illustration using example |
| 3 | Lexical Analysis-Role of Lexical Analyzer | 1 |  | PPT | Illustration using example |
| 4 | Input Buffering -Specification of Tokens | 1 |  | PPT | Illustration using example |
| 5 | LEX -Finite Automata | 1 |  | PPT | Illustration using example |
| 6 | Regular Expressions to Automata | 1 |  | PPT | Illustration using example |
| 7 | Minimizing DFA. | 1 |  | PPT | Illustration using example |
| 8 | Summary with examples /Role play | 1 |  | BB | Group discussion |
| 9 | Quiz/Puzzles/Review Questions | 1 |  | - | Gamification elements |
| Unit 2 - Top-Down Parsing | | | | | |
| 10 | Role of Parser-Grammars-Error Handling | 1 |  | PPT | Illustration using example |
| 11 | Context-Free Grammars-Writing a grammar | 1 |  | BB | Illustration using example |
| 12 | Elimination of Ambiguity-Left Recursion- Left Factoring | 1 |  | BB | Illustration using example |
| 13 | Top-Down Parsing — Recursive Descent Parser | 1 |  | PPT | Flipped Classroom  Illustration using example |
| 14 | Predictive Parser-LL (1) Parser- Computation of FIRST Computation of FOLLOW | 1 |  | BB | Flipped Classroom, Illustration using example |
| 15 | Construction of a predictive parsing table-Predictive Parsers LL(1) Grammars | 1 |  | PPT | Interactive Examples |
| 16 | Predictive Parsing Algorithm- Problems related to Predictive Parser | 1 |  | PPT | Illustration using example |
| 17 | Error Recovery in Predictive Parsing | 1 |  | PPT | Illustration using example |
| 18 | Summary and Review | 1 |  | - | Gamification elements |
| Unit 3 - Bottom-Up Parsing | | | | | |
| 19 | Bottom Up Parsing-Reductions-Handle Pruning | 1 |  | PPT | Flipped Classroom Illustration using example |
| 20 | Shift Reduce Parser-Problems related to Shift Reduce Parsing | 1 |  | BB | Flipped Classroom Illustration using example |
| 21 | Operator Precedence Parser, LEADING, TRAILING | 1 |  | BB | Illustration using example |
| 22 | LR Parser- LR Parsers- Need of LR Parsers | 1 |  | PPT | Interactive Examples |
| 23 | LR (0)Item-Closure of Item Sets | 1 |  | PPT | Illustration using example |
| 24 | Construction of SLR Parsing Table -Problems related to SLR | 1 |  | PPT | Interactive Examples |
| 25 | Construction of Canonical LR(1)- Problems related to CLR | 1 |  | PPT | Interactive Examples |
| 26 | LALR Parser — Problems related to LALR | 1 |  | BB | Interactive Examples |
| 27 | YACC. | 1 |  | BB | Gamification elements |
| Unit 4-Code Generation | | | | | |
| 28 | Intermediate Code Generation- prefix – postfix notation- Quadruple - triple - indirect triples Representation- | 1 |  | BB | Interactive Examples |
| 29 | Syntax tree- Evaluation of expression - Three-address code | 1 |  | BB | Illustration using example |
| 30 | Synthesized attributes – Inherited attributes | 1 |  | PPT | Illustration using example |
| 31 | Intermediate languages–Declarations- Assignment Statements | 1 |  | PPT | Illustration using example |
| 32 | Boolean Expressions- Case Statements- Back patching- Procedure calls | 1 |  | BB | Interactive Examples |
| 33 | Code Generation- Issues in the design of code generator- The target machine | 1 |  | PPT | Illustration using example |
| 34 | Runtime Storage management- A simple Code generator | 1 |  | PPT | Illustration using example |
| 35 | Code Generation Algorithm | 1 |  | PPT | Illustration using example |
| 36 | Register and Address Descriptors | 1 |  | PPT | Illustration using example |
| Unit 5 - Code Optimization | | | | | |
| 37 | Code optimization -Principal Sources of Optimization | 1 |  | PPT | Illustration using example |
| 38 | Function Preserving Transformation | 1 |  | PPT | Illustration using example |
| 39 | DAG- Basic Blocks- Flow Graphs | 1 |  | BB | Interactive Examples |
| 40 | Global Data Flow Analysis — Efficient Data Flow Algorithm | 1 |  | PPT | Illustration using example |
| 41 | Runtime Environments- Source Language issues | 1 |  | PPT | Illustration using example |
| 42 | Storage Organization- Activation Records | 1 |  | PPT | Illustration using example |
| 43 | Storage Allocation strategies. | 1 |  | PPT | Illustration using example |
| 44 | Quiz and review questions | 1 |  | PPT | Gamification elements |
| 45 | Summary and review | 1 |  | - | Gamification elements |

**Lab Exercises**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Topics to be covered | Hours | Ref | Teaching method |
| 1 | Lab 1 - Implementation of Lexical Analyzer | 1 |  | Turbo C / C++ programming environment  Or  Code blocks |
| 2 | Lab 2 conversion from Regular Expression to NFA | 1 |  |
| 3 | Lab 3 Conversion from NFA to DFA | 1 |  |
| 4 | Lab 4 Elimination of Ambiguity, Left Recursion and Left Factoring | 1 |  |
| 5 | Lab 5 -FIRST AND FOLLOW computation | 1 |  |
| 6 | Lab 6 Predictive Parsing Table | 1 |  |
| 7 | Lab 7 - Shift Reduce Parsing | 1 |  |
| 8 | Lab 8- Computation of LEADING AND TRAILING | 1 |  |
| 9 | Lab 9 Computation of LR (0) items | 1 |  |
| 10 | Lab 10-Intermediate code generation – Postfix, Prefix | 1 |  |
| 11 | Lab 11 Intermediate code generation – Quadruple, Triple, Indirect triple | 1 |  |
| 12 | Lab 12: A simple code Generator | 1 |  |
| 13 | Lab 13 Implementation of DAG | 1 |  |
| 14 | Lab 14: Implementation of Global Data Flow Analysis | 1 |  |
| 15 | Lab 15: Implement any one storage allocation strategies (heap, stack, static) | 1 |  |

**12. Overall Execution Plan**:

|  |  |  |
| --- | --- | --- |
| **#** | **Activity** | **Target Dates** |
| 1 | Power Point Preparation | 11-01-2024 |
| 2 | Lab Program Exercises Questions Preparation | 11-01-2024 |
| 3 | Question Bank Preparation | 13-02-2024 |
| 4 | Question Bank Scrutiny | 20-02-2024 15-03-2024 21-04-2024 |
| 5 | Cycle Test | 04-03-2024 05-04-2024 06-05-2024 |
| 6 | Course File Preparation | 13-03-2024 25-04-2024 15-05-2024 26-05-2024 |
| 7 | Review questions | 14-02-2024 07-03-2024 21-03-2024 07-04-2024 01-05-2024 |
| 8 | Feedback Collection and Minutes of Meeting | 17-03-2024 14-04-2024 18-05-2024  30/05/2024 |