RV UNIVERSITY Go, change the world an initiative of RV EDUCATIONAL INSTITUTIONS	**School of Computer Science & Engineering** **B.Tech(H) Program** **Academic Year: Jan-May 2025** **Semester 6 Version 3.0**

COURSE DESIGN, DELIVERY AND ASSESSMENT

Course Code: CS3704	**Course Name: Compiler Design**	
Semester: 6	**Area:** MAJOR COMPUTER SCIENCE CORE	**SEE Type:** Theory (30 Marks)
Level:	**Credits: 3(**2: 1: 0)	**Contact Hours:** 45 Hours
Prerequisite (Course/Skill/Knowledge): • Knowledge of Regular expression, finite automata and formal languages		

Course Faculty:

Sl#	Section	Course Faculty Name	Contact: Email/Contact Number	Sign with Date
1	A	Uma Shankari B	umashankarib@rvu.edu.in/ 7025236832	
2	C	Uma Shankari B	umashankarib@rvu.edu.in/ 7025236832	
3	C	Uma Shankari B	umashankarib@rvu.edu.in/ 7025236832	

Course Lead (Name, Sign, Date): **Uma Shankari B**

1. Course Context & Overview

This course delves into the theory and practical aspects of designing and implementing compilers. These essential tools translate source code written in a high-level programming language into a form executable by a computer. This will provide deeper insights into the more advanced semantics aspects of programming languages, code generation, machine-independent optimizations, dynamic memory allocation, types and their inferences, object orientation.

2. Course Contents

Unit 1: Introduction to Compiler **9 Hours**

Definition of compiler, interpreter and its differences, language processing system, the phases of a compiler, role of lexical analyzer, regular expressions, finite automata, finite automata to regular expressions, bootstrapping, Lexical Analyzer Generator.

Unit 2: Parsing Techniques **9 Hours**

Context free grammar, derivations, parse trees, ambiguity, eliminating ambiguity, elimination of left recursion, left factoring, top-down parsing: backtracking, recursive descent parsing, predictive parsers, LL(1) grammars, bottom up parsing: LR parsers, Simple LR, Canonical LR(CLR) and Look Ahead LR (LALR) parsers, Operator precedence parsing.

Unit 3: Syntax-directed Translation **9 Hours**

Syntax directed definition, construction of syntax trees, S-attributed and L-attributed definitions, Syntax-directed Translation schemes and applications, Intermediate code, postfix notation, Parse trees & syntax trees, three address code, quadruple & triples, Translation of simple statements and control flow statements.

Unit 4: Symbol Tables & Run-Time Environments **9 Hours**

Symbol Tables: Data structure for symbols tables, representing scope information. Run-Time Environments: Implementation of simple stack allocation scheme. Error Detection & Recovery: Lexical Phase errors, syntactic phase errors, and semantic errors.

Unit 5: Code Optimization & Generation **9 Hours**

Design Issues, the Target Language. Addresses in the Target Code, Basic Blocks and Flow Graphs, Optimization of Basic Blocks, Code Generator. Code optimization: Machine Independent Optimizations, Loop optimization, DAG representation of basic blocks, value numbers and algebraic laws, Data-Flow analysis: constant propagation, liveness analysis.

Course Outcomes: After completing the course, the students will be able to:	
CO1	Develop skills to devise, select, and apply appropriate tools and techniques for effective compiler design.
CO2	Apply context-free grammars (CFG) to develop language specifications.
CO3	Analyze syntax-directed translation schemes for various programming constructs and generate intermediate code.
CO4	Develop knowledge about run-time data structures like symbol table organization and different techniques.
CO5	Apply advanced knowledge of compiler optimization and code generation to practical scenarios.

(Tick✓ the Relevant Methodologies)

Assignments✓	Closed book tests✓	Open book tests
Case study	Student Presentation✓	Mini projects / Model Building✓
MOOC	Quiz✓	Peer Review

Textbooks (With ISBN No)

1. Aho, Sethi & Ullman, "Compilers: Principles, Techniques and Tools”, Pearson Education,ISBN- 978-9332518667

2. V Raghvan, "Principles of Compiler Design", TMH, ISBN: 978-0070144712 Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, First Edition, June 2009. ISBN-13: 978-0596516499.

Reference Material (With ISBN No)

1. 1. Kenneth Louden, "Compiler Construction: Principles and Practice", Course Technology Inc. ISBN: 978-0534939724
2. 2. Charles Fischer and Ricard LeBlanc," Crafting a Compiler with C", Pearson Education ISBN: 978-8131708132
3. Compiler design - Course - Swayam – NPTEL. https://archive.nptel.ac.in/courses/106/105/106105190/#

3. CO Mapping: Cognitive Levels, Knowledge Category, Contact & Activity Hours

Sl.No	Course Outcomes	Cognitive Level	Knowledge Category	PO (optional)	Class Conceptual hours	Class Activity Hours	Weightage of CO%
1	Develop and utilize skills to devise, select, and apply appropriate tools and techniques for effective compiler design.	Apply	Procedural		7	2	20
2	Apply context-free grammars (CFG) to develop language specifications.	Apply	Procedural		7	2	20
3	Analyze syntax-directed translation schemes for various	Analyze	Procedural		7	2	20

	programming constructs and generate intermediate code.						
4	Develop knowledge about run-time data structure like symbol table organization and different techniques used in that.	Apply	Procedural		7	2	20
5	Apply advanced knowledge of compiler optimization and code generation to practical scenarios	Apply	Procedural		7	2	20
	TOTAL: 45 Contact Hours (15 Weeks)				35	10	

4. Course Plan

Week 1	Outcome	Topics	Activity Based Teaching Learning Process
Week 1	CO1	Definition of compiler, interpreter and its differences, language processing system the phases of a compiler	Practical demo and group activity

Week 2	CO1	role of lexical analyzer, regular expressions,finite automata, finite automata to regular expressions	Lexical Analyzer Simulation,Simulation of NFA and DFA
Week 3	CO1	Lexical Analyzer Generator, bootstrapping	Hands-On Practice with Lex/Flex
Week 4	CO2	Context free grammar, derivations, parse trees, ambiguity, eliminating ambiguity, elimination of left recursion, left factoring	Group Exercise: Refactor a Complex Grammar
Week 5	CO2	top-down parsing: backtracking, recursive descent parsing, predictive parsers, LL(1) grammars	Case Study: LL(1) Parsing in Real Compilers
Week 6	CO2	bottom up parsing: LR parsers, Simple LR, Canonical LR(CLR) and Look Ahead LR (LALR) parsers, Operator precedence parsing.	Group activity and case study
Week 7	CO3	Syntax directed definition, construction of syntax trees, S-attributed and L-attributed definitions	Hands on Implementing SDDs for Expressions
Week 8	CO3	Syntax-directed Translation schemes and applications, Intermediate code, postfix notation	Group activity and case study
Week 9	CO3	Parse trees & syntax trees, three address code, quadruple & triples, Translation of simple statements and control flow statements	Group Exercise: Translation of Complex Statements

Week 10	CO4	Symbol Tables: Data structure for symbols tables, representing scope information	Practical demo
Week 11	CO4	Run-Time Environments: Implementation of simple stack allocation scheme. Error Detection & Recovery	Group activity
Week 12	CO4	Lexical Phase errors, syntactic phase errors, and semantic errors.	Practical demo
Week 13	CO5	Design Issues, the Target Language. Addresses in the Target Code, Basic Blocks and Flow Graphs	Practical demo
Week 14	CO5	Optimization of Basic Blocks, Code Generator. Code optimization: Machine Independent Optimizations, Loop optimization	Hands on activity on code optimization
Week 15	CO5	DAG representation of basic blocks, value numbers and algebraic laws, Data-Flow analysis: constant propagation, liveness analysis	Hands on activity on DAG

5. Partial Delivery

Sl.No:	Description	Topics Beyond Syllabus/ Industry Visit/ Guest Lectures/ Technical Talks/ Workshops/ NPTEL etc.
1.	Compiler design - Course - Swayam – NPTEL. https://archive.nptel.ac.in/courses/106/105/106105190/#	NPTEL

6. Instructional Methodologies (Tick the relevant)

Blackboard & chalk ✓	PowerPoint presentations ✓	Student seminars
Mini-Projects ✓	Industry Guest Lectures	Flipped Classroom ✓
Web resources ✓/certification	MOOC	Any other (Specify) ANTLR,LEX,YACC,LLVM

7. Assessment Methodologies - Indirect (Tick the relevant)

Student Feedback on Course (Exit Survey) ✓	Feedback from Industry Expert
Feedback from Alumni	If any other (Please Specify)

8. Course Outcomes to Program Outcome Mapping

Program Outcome / Course Outcome		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	Develop and utilize skills to devise,	3	3	3	-	1	-	-	-	2	-	-	2

	select, and apply appropriate tools and techniques for effective compiler design.												
CO2	Apply context-free grammars (CFG) to develop language specifications.	3	3	3		1	-	-	-	2	-	-	2
CO3	Analyze syntax-directed translation schemes for various programming constructs and generate intermediate code.	3	3	3	2	1	-	-	-	2	-	-	2

CO4	Develop knowledge about run-time data structure like symbol table organization and different techniques used in that.	3	3	3	2	1	-	-	-	2	-	-	2
CO5	Apply advanced knowledge of compiler optimization and code generation to practical scenarios	3	3	3	2	1	-	-	-	2	-	-	2

9. Justification of CO-PO Mapping [For all the CO's]

CO1: Develop and utilize skills to devise, select, and apply appropriate tools and techniques for effective compiler design.

PO	Level of Correlation	Justification
PO1	High	Use mathematical concepts such as automata theory and discrete mathematics to design lexical analyzers and parsers. Apply knowledge of computer architecture to generate optimized machine code. Leverage engineering principles to analyze and handle complex language constructs.
PO2	High	Identify and analyze ambiguities or inefficiencies in grammar and language specifications.Use research literature and formal methods to derive solutions for challenging problems like context-sensitive parsing or runtime optimization.

PO3	High	Develop compilers that meet specific needs, such as targeting embedded systems, high-performance computing, or language translation.Ensure safety and correctness by incorporating error detection and recovery mechanisms.
PO5	Low	Compiler design involves using modern engineering tools to solve complex problems effectively,Tools like Flex, ANTLR
PO9	Medium	Designing compilers that adhere to industry standards and legal requirements.Protecting intellectual property by respecting the open-source licenses of tools and libraries.
PO12	Medium	Ensures that learners can adapt to new technologies and trends, maintaining their relevance and expertise.

CO2: Apply context-free grammars (CFG) to develop language specifications

PO	Level of Correlation	Justification
PO1	High	Applying CFGs requires a strong understanding of mathematical and engineering principles.
PO2	High	Emphasizes identifying and solving language specification challenges using analytical approaches.
PO3	High	CFGs are used to design robust language specifications.
PO5	Low	CFGs are closely tied to modern tools used in language specification and parser development
PO9	Medium	Ensuring grammars are not biased or restrictive for certain users or regions
PO12	Medium	New programming paradigms (e.g., quantum programming, domain-specific languages) require updated knowledge of language specifications.

CO3: Analyze syntax-directed translation schemes for various programming constructs and generate intermediate code.

PO	Level of Correlation	Justification
PO1	High	Analyzing and designing syntax-directed translation schemes require foundational knowledge in mathematics ,science and engineering fundamentals

PO2	High	Decomposing complex language constructs and formulating efficient SDT schemes enhances problem-solving.
PO3	High	Designing SDT schemes and generating intermediate code ensures accurate translation of high-level constructs.
PO4	Medium	Investigating optimization techniques for SDT and intermediate code ensures validity and efficiency.
PO5	Low	Tools like LLVM and intermediate representation frameworks assist in the implementation of SDT schemes
PO9	Medium	Ensures generated intermediate code is secure, fair, and adheres to ethical norms in software development.
PO12	Medium	Continuous learning allows students to adapt SDT techniques to emerging paradigms and tools

CO4: Develop knowledge about run-time data structure like symbol table organization and different techniques used in that.

PO	Level of Correlation	Justification
PO1	High	Apply engineering knowledge, particularly in data structures and compiler theory, to solve complex problems in symbol table design.
PO2	High	Analyze problems involving symbol tables, choosing the best data structure based on the problem's needs This includes understanding the trade-offs involved in solving real-world problems in software design and compiler construction.
PO3	High	Design efficient symbol table organizations for compilers, considering aspects such as scope management, memory allocation, and performance. T
PO4	Medium	Conduct experiments and analyze different techniques for symbol table optimization .
PO5	Low	Exposes students to modern tools such as ANTLR, LLVM, and other compiler design tools that are used for symbol table creation and language processing.
PO9	Medium	correct design and implementation of symbol tables in compilers, ensuring they handle identifiers responsibly, avoid security risks (e.g., variable shadowing), and maintain privacy and transparency in software systems.
PO12	Medium	Engage in continuous learning to stay updated on emerging techniques in runtime data structures.

CO5: Apply advanced knowledge of compiler optimization and code generation to practical scenarios

PO	Level of Correlation	Justification
PO1	High	Leverage engineering knowledge to solve practical problems in optimization and code generation.
PO2	High	Analyze performance bottlenecks, identify optimization opportunities, and select appropriate techniques based on the given problem
PO3	High	Designing solutions that are both technically sound and ethically responsible.
PO4	Medium	Engage in experiments to explore the performance impacts of different optimization techniques
PO5	Low	Encourages the use of modern tools to enhance the optimization process
PO9	Medium	Ethical responsibilities in ensuring safe and secure code generation.
PO12	Medium	New languages, techniques, and architectures emerge regularly, requiring practitioners to stay updated on the latest research and developments.

Assessment Plan

Internal Assessment Plan: 70 Marks				
Sl#	**Component**	**Marks**	**Type of Assessment**	**Timeline**
Continuous Internal Evaluation-1 (20 Marks)				
1	CO1	5	Graded Component 1 (Problem statement for Mini project)	Week 3
2	CO2	15	Graded Component 1 (Quiz)	Week 5
Continuous Internal Evaluation-2 (30 Marks)				
3	CO1 - C03	25	Mid Sem Examination (Theory)	Week 9
Continuous Internal Evaluation-3 (20 Marks)				

4	CO4 - CO5	15	QUIZ + Written test	Week 13
	CO1-CO5	10	Mini Project	Week 14, 15

Mid Sem Assessment Pattern: 25 Marks		
Sl#	**Content**	
Part A- 5 Marks (2 questions of 2.5 Marks each)		
1	5 Questions of 2 mark each	10
Part B- 20 Marks (Compulsory 4 questions of 5 marks each, there can be max 2 sub divisions)		
2	4 Questions of 5 marks each with 1 or 2 sub divisions possibly	20

SEE Assessment Pattern: 30 Marks		
Sl#	**Content**	
Part A - 10 Marks (2 Questions, 5 Marks Each, max 3 Subdivisions)		
1	2 Questions of 5 marks each	10
Part B - 20 Marks (2 Questions, 10 Marks Each, Max Four Subdivisions per Question)		
2	2 Questions of 10 marks each	20

10. Rubrics for Assessment Components

Assessment Component	Type of Component	Rubrics for Assessment
1.	CP1	Quiz will be conducted for 15 marks. Each right answer carries one mark each. No negative marking for wrong answers.
		Problem statement of Mini project evaluated for 5 marks.

		Students work individually on their assigned tasks.
2.	Mid Sem	Test will be conducted for 25 marks and will be evaluated according to Scheme of Evaluation prepared by team of course faculty.
3.	CP3	Quiz + Written test = 15 marks Mini Project = 10 marks
4.	SEE	Test will be conducted out of 30 marks and will be evaluated according to Scheme of Evaluation prepared by team of course faculty.

11. Detailed Rubrics of Assessment Components

Mini Project Assessment

Parameters	Excellent (5)	Good (4)	Satisfactory (2-3)	Not Satisfactory (1)
Problem-Solving (5 Marks)	Uses advanced problem-solving skills to create innovative and effective solutions.	Uses good problem-solving skills to create effective solutions.	Uses basic problem-solving skills with partially effective solutions.	Uses poor problem-solving skills with ineffective or incorrect solutions.
Project demonstrations and viva voce (5 Marks)	Present the whole project in organized manner and competently address the and handle the questions from audiences. Show the demonstration without struggling.	Present the whole project in organized manner and competently address the and handle the questions from audiences. Show the demonstration with struggling.	Present the whole project in un organized manner and competently address the and handle the questions from audiences.	Presentation is not clear and lacked in answering the questions

Report (5 marks)	All the required information is included, report is organized according to the templates and submitted with the dead line.	Most of the required information is included, report is organized according to the templates and submitted with the dead line.	Necessary information is included with some irrelevant information. Report partially follows the given template and submission exceeded the given dead line.	The report is not organized and important information is missing. submission exceeded the given dead line.

12. Course Policy

- All the students
  - should bring their personal computers (fully charged) to the classroom.
  - should have an RVU mail ID.
  - should be able to connect to both RVCE and RVU WiFi.
- Use of mobile phones is not allowed during class hours. Also, they should not be connected to WiFi.
- In general, late submission by one day, without prior permission, will result in a penalty of 25%. Late submission beyond two days will not be accepted.
- Being late by more than 5 minutes in the first session and by more than 2 minutes in the remaining sessions will not be given attendance. Also, latecomers are required to not disturb others in the classroom.

Reviewed By (Name, Affiliation & Date):

Program Head (Name, Sign, Date): **SoCSE Dean (Sign & Date)**

Version History	Prepared by	Status

1:0 /	Dr. Mydhili K Nair	Valid till odd sem 2023
2:0/	Dr. Merin Thomas	Valid till odd sem 2024
3:0/	Dr. K Sailaja Kumar	Action
4.0		