INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad - 500 043

COMPUTER SCIENCE AND ENGINEERING

COURSE DESCRIPTION FORM

Course Title	**COMPILER DESIGN**			
Course Code	**A50514**			
Regulation	**R15**			
Course Structure	Lectures	Tutorials	Practicals	Credits
	4	-	-	4
Course Coordinator	Ms. B Ramyasree, Assistant Professor.			
Team of Instructors	Ms. N Mamatha Assistant Professor, Mr. N Poorna Chandra Rao, Assistant Professor.			

I. COURSE OVERVIEW:

This course is intended to make the students learn the basic techniques of compiler construction and tools that can used to perform syntax-directed translation of a high-level programming language into an executable code. It also discuss various aspects of the run-time environment into which the high-level code is translated. This will provide deeper insights into the more advanced semantics aspects of programming languages, code generation, machine independent optimizations, dynamic memory allocation, and object orientation.

II. PREREQUISITE(S):

Level	Credits	Periods/ Week	Prerequisites
UG	4	4	Computer Programming, Formal Languages Automata Theory

III. MARKS DISTRIBUTION:

Sessional Marks	University End Exam marks	Total marks
Midterm Test There shall be two midterm examinations. Each midterm examination consists of essay paper, objective paper and assignment. The essay paper is for 10 marks of 60 minutes duration and shall contain 4 questions. The student has to answer 2 questions, each carrying 5 marks. The objective paper is for 10 marks of 20 minutes duration. It consists of 10 multiple choice and 10 fill-in-the blank questions, the student has to answer all the questions and each carries half mark. First midterm examination shall be conducted for the first two and half units of syllabus and second midterm examination shall be conducted for the remaining portion. Five marks are earmarked for assignments. There shall be two assignments in every theory course. Assignments are usually issued at the time of commencement of the semester. These are of problem solving in nature with critical thinking. Marks shall be awarded considering the average of two midterm tests in each course.	75	100

IV. EVALUATION SCHEME:

S. No	Component	Duration	Marks
1.	I Mid Examination	80 minutes	20
2.	I Assignment	-	5
3.	II Mid Examination	80 minutes	20
4.	II Assignment	-	5
5.	External Examination	3 hours	75

V. COURSE OBJECTIVES:

I. Be familiar with major concepts of language translation and compiler design.

II. Learn the various parsing techniques and different levels of translation

III. Extend the knowledge of parser by parsing LL parser and LR parser.

IV. Enrich the knowledge in various phases of compiler ant its use, code optimization techniques, machine code generation, and use of symbol table.

V. Be familiar with compiler architecture and with register allocation.

VI. Learn how to optimize and effectively generate machine codes

VII. Provide practical programming skills necessary for constructing a compiler.

VI. COURSE OUTCOMES:

At the end of the course the students are able to:

1. **Understand** the design of a compiler and the phases of program translation from source code to executable code and the files produced by these phases.
2. **Relate** finite state automata, push-down automata and their connection to language definition through regular expressions and grammars.
3. **Use** the powerful compiler generation tools such as Lex and YACC for generating the parser.
4. **Identify** the analysis phase, similarities and differences among various parsing techniques and grammar transformation techniques.
5. **Implement** major parsing techniques ranging from the recursive decent methods to the computationally more intensive LR techniques that have been used in parser generator.
6. **Apply** the formal attributed grammars for specifying the syntax and semantics of programming languages.
7. **Implement** the static semantic checking and type checking using syntax directed definition (SDD) and syntax directed translation (SDT).
8. **Translate** common programming language constructs into intermediate code.
9. **Understand** the storage organization used to support the run-time environment of a program.
10. **Identify** the effectiveness of optimization and effectively generate machine codes.
11. **Implement** the global optimization using data flow analysis such as basic blocks and DAG.
12. **Apply** the several algorithms for collecting and optimizing the information using data flow analysis.
13. **Understand** the code generation techniques to generate target code.
14. **Differentiate** between machine-dependent and machine-independent optimizations.

VII. HOW PROGRAM OUTCOMES ARE ASSESSED:

Program Outcomes		Level	Proficiency assessed by
PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	H	Exercises
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.	H	Exercises
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.	H	Assignments
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.	S	Projects
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.	H	Mini Projects
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.	N	--
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.	N	--
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	N	--
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	H	Projects
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.	N	--
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.	S	Projects
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.	N	--

N - None S - Supportive H - Highly Related

VIII. HOW PROGRAM SPECIFIC OUTCOMES ARE ASSESSED:

Program Specific Outcomes		Level	Proficiency assessed by
PSO1	**Professional Skills:** The ability to research, understand and implement computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient analysis and design of computer-based systems of varying complexity.	S	Lectures, Assignments
PSO2	**Problem-solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.	H	Projects
PSO3	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths, to be an entrepreneur, and a zest for higher studies.	S	Guest Lectures

N - None S - Supportive H - Highly Related

IX. SYLLABUS:

UNIT – I

Overview of Compilation: Phases of Compilation – Lexical Analysis, Regular Grammar and regular expression for common programming language features, pass and Phases of translation, interpretation, bootstrapping, data structures in compilation – LEX lexical analyzer generator

Top down Parsing: Context free grammars, Top down parsing – Backtracking, LL (1), recursive descent parsing, Predictive parsing, Preprocessing steps required for predictive parsing.

UNIT – II

Bottom up parsing: Shift Reduce parsing, LR and LALR parsing, Error recovery in parsing , handling ambiguous grammar, YACC – automatic parser generator.

UNIT – III

Semantic analysis: Intermediate forms of source Programs – abstract syntax tree, polish notation and three address codes. Attributed grammars, Syntax directed translation, Conversion of popular Programming languages language Constructs into Intermediate code forms, Type checker.

Symbol Tables: Symbol table format, organization for block structures languages, hashing, tree structures representation of scope information. Block structures and non block structure storage allocation: static, Runtime stack and heap storage allocation, storage allocation for arrays, strings and records.

UNIT – IV

Code optimization: Consideration for Optimization, Scope of Optimization, local optimization, loop optimization, frequency reduction, folding, DAG representation.

Data flow analysis: Flow graph, data flow equation, global optimization, redundant sub expression elimination, Induction variable elements, Live variable analysis, Copy propagation.

UNIT – V

Object code generation: Object code forms, machine dependent code optimization, register allocation and assignment generic code generation algorithms, DAG for register allocation.

Text books:

1. A.V. Aho . J.D.Ullman, “Principles of compiler design”, Pearson Education.
2. Andrew N. Apple, “Modern Compiler Implementation in C”, Cambridge University Press.

References:

1. John R. Levine, Tony Mason, Doug Brown, “Lex & yacc”, O’reilly.
2. Dick Grune, Henry E. Bal, Cariel T. H. Jacobs, “Modern Compiler Design” Wiley dreamtech.
3. Cooper & Linda, “Engineering a Compiler”, Elsevier.
4. Louden, “Compiler Construction”, Thomson.

X. COURSE PLAN:

At the end of the course, the students are able to achieve the following course learning outcomes.

Lecture No.	Topics to be covered	Course Learning Outcomes	Reference
1-4	Introduction, structure, Phases of Compilation.	**Understand** the basic compilers and compilation process	T1:1.1-1.8
5-8	Construction of regular grammar from regular expression NFA,DFA, common programming features	**Relate** regular grammar to programming feature	T1:35-3.7
9	Concept of pass and difference between pass and phase.	**Differentiate** Pass and Phases of translation	T1:3.3,T2:2.3
10	Bootstrapping and types of compiler.	**Design** of compiler for a language	T1:4.1
11-13	Lex-Lexical analyzer generator Derivations and parse tree regular expressions v/s context free grammar.	**Identify** Data structure in compilation Using lexical analyzer	T1:4.1
14-16	Backtracking, LL(1),Recursive decent parsing Finding FIRST and FOLLOW.	**Understand** Top down parsing techniques	T1:3.8,T2:2.5 T1:4.1, T2:3.1
17-20	Construction of parse tables, Predictive parsing.	**Construct** the parsing table for given inputs	T1:4.3
21-22	Shift reduce parsing, operator precedence parsing	**Understand** bottom up parsing techniques	T1:4.4
23-25	LR-SLR,LR(0)	**Differentiate** types of LR(0) parsers	T1:4.5
26-28	LALR,CLR.	**Differentiate** types of LR(1) parsers	T1: 4.6
29	Description of error recovery	**Construct** a parse tree for ambiguous grammar	T1: 4.7
30	Yacc parser generator	**Implement** parser generator	T1:4.8
31-32	Abstract syntax tree, three address code	**Implement** the construction of syntax trees	T1:4.9
33-35	Introduction to attributes grammars Syntax directed definitions, applications of SDD, implementing L-attributed SDD's	**Recognize** the semantics of grammar	T1:5.2
36-37	Control flow, back patching, switch statements	**Describe** the forms of intermediate code generation phases	T1:5.1.5.3,5.4
38-40	Rules, type conversions, Overloading, type inference and polymorphic functions.	**List** different types of language constructs	T1:8.1-8.6
41-43	Symbol table format, ordered and unordered symbol tables. Organization for block structures languages	**Summarize** the symbol table	T1:6.1-6.6
44-46	Static, runtime stack and heap storage allocations	**List** different types of storage allocation	T1:7.6
47-48	Storage allocation for arrays, strings and	**Understand** storage allocations	T1: 7.7

	records	for data structures	
48-50	Introduction for optimization. Local, global and scope optimization	**Understand** Various optimization techniques	T1:7.8-7.9
51-53	Basic blocks, flow graphs, loops, code motion, induction variables, reduction in strength	I**mplementation** of basic block optimization techniques	T1:10.1-10.2
54-55	DAG construction, applications	**Construction** of DAG	T1:10.3-10.4
55-57	Data flow analysis of flow graphs. Flow graph, loops in flow graphs Representing data flow information, data flow equations for programming constructs	**Understand** the Data flow analysis	T1:10.5
58-60	Examples for sub expression elimination, Live variable analysis copy propagation and examples	**Implement** optimization on data flow graphs	T1:10.6-10.8 T2:9.1
61-62	Introduction, issues in code generation, object code forms Need of machine dependent code optimization, peephole optimization	**Understand** various code generation techniques	T1:10.9-10.13
63-65	Global register allocation, register assignment for outer loops Rearranging the order, heuristic ordering for DAGs, optimal ordering and labeling algorithm	**Implement** machine dependent optimizations	T1:10.12

XI. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course Objectives	Program Outcomes												Program Specific Outcomes		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
I	S	S	H								S		H	S	
II		S	H	S	S								S	H	S
III	H			S	H						S		S	S	S
IV	S			S	H						S		S	S	
V		S		S	H								H	H	S
VI		H	S						S					S	S
VII		S			S						H		H	S	

S - Supportive **H - Highly Related**

XII. MAPPING COURSE OUTCOMES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Course Outcomes	Program Outcomes												Program Specific Outcomes		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
1	H			S							S		H	S	
2	S	S	H	S	S						S		S	H	
3			H		H						S			S	H
4	S	S		S					S		H		H	S	
5		S	S	H	H				S		S		S	H	S
6	S				H									S	H
7				H							S			H	S
8	S			S	H						S		S	S	
9		S		S	H								H	H	S
10		H	S						S					S	S
11					S				H		S		S	H	S
12		H	S						S					S	S
13		H	S						S					S	S
14		H	S						S					S	S

S - Supportive **H - Highly Related**

Prepared by: Ms. B Ramyasree, Assistant Professor,
Ms.N Mamatha, Assistant Professor,
Mr.N Poornachandra Rao, Assistant Professor.

HOD, COMPUTER SCIENCE AND ENGINEERING