INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
Dundigal, Hyderabad -500 043

INFORMATION TECHNOLOGY

COURSE DESCRIPTOR

Course Title	**COMPILER DESIGN**				
Course Code	AIT004				
Programme	B.Tech				
Semester	V	CSE | IT			
Course Type	Core				
Regulation	IARE - R16				
Course Structure	**Theory**			**Practical**	
	Lectures	**Tutorials**	**Credits**	**Laboratory**	**Credits**
	3	1	4	-	-
Chief Coordinator	Dr. K Srinivasa Reddy, Professor				
Course Faculty	Ch. Suresh Kumar Raju, Assistant Professor				

I. COURSE OVERVIEW:

This course deals with the basic techniques of compiler construction and tools that can be used to perform syntax-directed translation of a high-level programming language into an executable code. 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 and object orientation.

II. COURSE PRE-REQUISITES:

Level	Course Code	Semester	Prerequisites	credits
UG	ACS001	I	Computer Programming	3
UG	ACS002	II	Data Structures	4
UG	AHS013	III	Discrete Mathematical Structures	4
UG	AIT002	IV	Theory of Computation	3

III. MARKS DISTRIBUTION:

Subject	SEE Examination	CIA Examination	Total Marks
Compiler Design	70 Marks	30 Marks	100

IV. DELIVERY / INSTRUCTIONAL METHODOLOGIES:

✔	Chalk & Talk	✔	Quiz	✔	Assignments	✘	MOOCs
✔	LCD / PPT	✔	Seminars	✘	Mini Project	✔	Videos
✘	Open Ended Experiments						

V. EVALUATION METHODOLOGY:

The course will be evaluated for a total of 100 marks, with 30 marks for Continuous Internal Assessment (CIA) and 70 marks for Semester End Examination (SEE). Out of 30 marks allotted for CIA during the semester, marks are awarded by taking average of two CIA examinations or the marks scored in the make-up examination.

Semester End Examination (SEE): The SEE is conducted for 70 marks of 3 hours duration. The syllabus for the theory courses is divided into FIVE modules and each module carries equal weightage in terms of marks distribution. The question paper pattern is as follows. Two full questions with "either" or "choice" will be drawn from each module. Each question carries 14 marks. There could be a maximum of two sub divisions in a question.

The emphasis on the questions is broadly based on the following criteria:

50 %	To test the objectiveness of the concept.
50 %	To test the analytical skill of the concept OR to test the application skill of the concept.

Continuous Internal Assessment (CIA):

CIA is conducted for a total of 30 marks (Table 1), with 25 marks for Continuous Internal Examination (CIE), 05 marks for Quiz/Alternative Assessment Tool (AAT).

Table 1: Assessment pattern for CIA

Component	Theory		Total Marks
Type of Assessment	CIE Exam	Quiz / AAT	
CIA Marks	25	05	30

Continuous Internal Examination (CIE):

Two CIE exams shall be conducted at the end of the 8th and 16th week of the semester respectively. The CIE exam is conducted for 25 marks of 2 hours duration consisting of two parts. Part–A shall have five compulsory questions of one mark each. In part–B, four out of five

questions have to be answered where, each question carries 5 marks. Marks are awarded by taking average of marks scored in two CIE exams.

Quiz / Alternative Assessment Tool (AAT):

Two Quiz exams shall be online examination consisting of 25 multiple choice questions and are be answered by choosing the correct answer from a given set of choices (commonly four). Marks shall be awarded considering the average of two quizzes for every course. The AAT may include seminars, assignments, term paper, open ended experiments, five minutes video and MOOCs.

VI. HOW PROGRAM OUTCOMES ARE ASSESSED:

Program Outcomes (POs)		Strength	Proficiency assessed by
PO 1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.	3	Assignments
PO 2	**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	3	Seminars
PO 3	**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	3	Assignments
PO 4	**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.	2	Seminars

3 = High; 2 = Medium; 1 = Low

VII. HOW PROGRAM SPECIFIC OUTCOMES ARE ASSESSED:

Program Specific Outcomes (PSOs)		Strength	Proficiency assessed by
PSO 1	**Professional Skills:** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design,	2	Assignments

Program Specific Outcomes (PSOs)		Strength	Proficiency assessed by
	big data analytics, and networking for efficient design of computer-based systems of varying complexity.		
PSO 2	**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.	2	Seminars
PSO 3	**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.	-	-

3 = High; 2 = Medium; 1 = Low

VIII. COURSE OBJECTIVES :

The course should enable the students to:	
I	Apply the principles in the theory of computation to the various stages in the design of compilers.
II	Demonstrate the phases of the compilation process and able to describe the purpose and operation of each phase.
III	Analyze problems related to the stages in the translation process.
IV	Exercise and reinforce prior programming knowledge with a non-trivial programming project to construct a compiler.

IX. COURSE OUTCOMES (COs):

COs	Course Outcome	CLOs	Course Learning Outcome
CO 1	Understand the various phases of compiler and design the lexical analyzer.	CLO 1	Define the phases of a typical compiler, including the front and backend.
		CLO 2	Recognize the underlying formal models such as finite state automata, push-down automata and their connection to language definition through regular expressions and grammars.
		CLO 3	Identify tokens of a typical high-level programming language; define regular expressions for tokens and design and implement a lexical analyzer using a typical scanner generator.
CO 2	Explore the similarities and differences among various parsing techniques and grammar transformation techniques	CLO 4	Explain the role of a parser in a compiler and relate the yield of a parse tree to a grammar derivation
		CLO 5	Apply an algorithm for a top-down or a bottom-up parser construction; construct a parser for a given context-free grammar.
		CLO 6	Demonstrate Lex tool to create a lexical analyzer and Yacc tool to create a parser.
CO 3	Analyze and implement syntax directed translations schemes and intermediate code generation.	CLO 7	Understand syntax directed translation schemes for a given context free grammar.
		CLO 8	Implement the static semantic checking and type checking using syntax directed definition (SDD) and syntax directed translation (SDT).

COs	Course Outcome	CLOs	Course Learning Outcome
		CLO 9	Understand the need of intermediate code generation phase in compilers.
		CLO 10	Write intermediate code for statements like assignment, conditional, loops and functions in high level language.
		CLO 11	Explain the role of a semantic analyzer and type checking; create a syntax-directed definition and an annotated parse tree; describe the purpose of a syntax tree.
		CLO 12	Design syntax directed translation schemes for a given context free grammar.
CO 4	Describe the concepts of type checking and analyze runtime allocation strategies.	CLO 13	Explain the role of different types of runtime environments and memory organization for implementation of programming languages.
		CLO 14	Differentiate static vs. dynamic storage allocation and the usage of activation records to manage program modules and their data.
		CLO 15	Understand the role of symbol table data structure in the construction of compiler.
CO 5	Demonstrate the algorithms to perform code optimization and code generation.	CLO 16	Learn the code optimization techniques to improve the performance of a program in terms of speed & space.
		CLO 17	Implement the global optimization using data flow analysis such as basic blocks and DAG.
		CLO 18	Understand the code generation techniques to generate target code.
		CLO 19	Design and implement a small compiler using a software engineering approach.
		CLO 20	Apply the optimization techniques to intermediate code and generate machine code

X. COURSE LEARNING OUTCOMES (CLOs):

CLO Code	CLO's	At the end of the course, the student will have the ability to:	PO's Mapped	Strength of Mapping
AIT004.01	CLO 1	Define the phases of a typical compiler, including the front and backend.	PO 1, PO 2	3
AIT004.02	CLO 2	Recognize the underlying formal models such as finite state automata, push-down automata and their connection to language definition through regular expressions and grammars.	PO 1, PO 4	3
AIT004.03	CLO 3	Identify tokens of a typical high-level programming language; define regular expressions for tokens and design and implement a lexical analyzer using a typical scanner generator.	PO 3	3
AIT004.04	CLO 4	Explain the role of a parser in a compiler and relate the yield of a parse tree to a grammar derivation.	PO 1, PO 2	3
AIT004.05	CLO 5	Apply an algorithm for a top-down or a bottom-up parser construction; construct a parser for a given context-free grammar.	PO 2	2
AIT004.06	CLO 6	Demonstrate Lex tool to create a lexical analyzer and Yacc tool to create a parser.	PO 1, PO 4	3
AIT004.07	CLO 7	Understand syntax directed translation schemes for a given context free grammar.	PO 1, PO 4	3
AIT004.08	CLO 8	Implement the static semantic checking and type checking using syntax directed definition (SDD) and syntax directed translation (SDT).	PO 1, PO 2	3

CLO Code	CLO's	At the end of the course, the student will have the ability to:	PO's Mapped	Strength of Mapping
AIT004.09	CLO 9	Understand the need of intermediate code generation phase in compilers.	PO 3, PO 4	3
AIT004.10	CLO 10	Write intermediate code for statements like assignment, conditional, loops and functions in high level language.	PO 1, PO 4	3
AIT004.11	CLO 11	Explain the role of a semantic analyzer and type checking; create a syntax-directed definition and an annotated parse tree; describe the purpose of a syntax tree.	PO 4	2
AIT004.12	CLO 12	Design syntax directed translation schemes for a given context free grammar.	PO 1, PO 3	3
AIT004.13	CLO 13	Explain the role of different types of runtime environments and memory organization for implementation of programming languages.	PO 1	2
AIT004.14	CLO 14	Differentiate static vs. dynamic storage allocation and the usage of activation records to manage program modules and their data.	PO 2	3
AIT004.15	CLO 15	Understand the role of symbol table data structure in the construction of compiler.	PO 1	2
AIT004.16	CLO 16	Learn the code optimization techniques to improve the performance of a program in terms of speed & space.	PO 2	3
AIT004.17	CLO 17	Implement the global optimization using data flow analysis such as basic blocks and DAG.	PO 1	2
AIT004.18	CLO 18	Understand the code generation techniques to generate target code.	PO 3	3
AIT004.19	CLO 19	Design and implement a small compiler using a software engineering approach.	PO 1, PO 3	3
AIT004.20	CLO 20	Apply the optimization techniques to intermediate code and generate machine code	PO 1, PO 4	3

3= High; 2 = Medium; 1 = Low

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

Course Outcomes (COs)	Program Outcomes (POs)				Program Specific Outcomes (PSOs)	
	PO1	PO2	PO3	PO4	PSO1	PSO2
CO 1	3	2	3	2	3	3
CO 2	3	3		3	3	3
CO 3	3	3	3	3	2	3
CO 4	2	3			2	2
CO 5	3	3	3	2	2	3

3 = High; 2 = Medium; 1 = Low

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

Course Learning Outcomes (CLOs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CLO 1	3	2											3	2	

Course Learning Outcomes (CLOs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CLO 2	3			2									2		
CLO 3			3												
CLO 4	2	3											2		
CLO 5		2												3	
CLO 6	3			2									3		
CLO 7	2			3									2		
CLO 8	2	3													
CLO 9			3	3									2	3	
CLO 10	3			2											
CLO 11				2									3		
CLO 12	3		3										2		
CLO 13	2													2	
CLO 14		3											2		
CLO 15	2													2	
CLO 16		3											2		
CLO 17	2													2	
CLO 18			3												
CLO 19	3		3										2		
CLO 20	3			2									2	3	

3 = High; 2 = Medium; 1 = Low

XIII. ASSESSMENT METHODOLOGIES – DIRECT

CIE Exams	PO1, PO2, PO3,PO4, PSO1,PSO2	SEE Exams	PO1, PO2, PO3,PO4, PSO1,PSO2	Assignments	PO1,PO3, PSO1	Seminars	PO2,PO4, PSO2
Laboratory Practices	-	Student Viva	-	Mini Project	-	Certification	-
Term Paper	PO1, PO2, PO3,PO4, PSO1,PSO2						

XIV. ASSESSMENT METHODOLOGIES - INDIRECT

✔	Early Semester Feedback	✔	End Semester OBE Feedback
✘	Assessment of Mini Projects by Experts		

XV. SYLLABUS

UNIT-I	INTRODUCTION TO COMPILERS AND PARSING
Introduction to compilers: Definition of compiler, interpreter and its differences, the phases of a compiler, role of lexical analyzer, regular expressions, finite automata, from regular expressions to finite automata, pass and phases of translation, bootstrapping, LEX-lexical analyzer generator; Parsing: Parsing, role of parser, context free grammar, derivations, parse trees, ambiguity, elimination of left recursion, left factoring, eliminating ambiguity from dangling-else grammar, classes of parsing, top-down parsing: backtracking, recursive-descent parsing, predictive parsers, LL(1) grammars.	
UNIT-II	**BOTTOM-UP PARSING**
Bottom-up parsing: Definition of bottom-up parsing, handles, handle pruning, stack implementation of shift- reduce parsing, conflicts during shift-reduce parsing, LR grammars, LR parsers-simple LR, canonical LR and Look Ahead LR parsers, error recovery in parsing, parsing ambiguous grammars, YACC-automatic parser generator.	
UNIT-III	**SYNTAX-DIRECTED TRANSLATION AND INTERMEDIATE CODE GENERATION**
Syntax-directed translation: Syntax directed definition, construction of syntax trees, S-attributed and Lattributed definitions, translation schemes, emitting a translation. Intermediate code generation: Intermediate forms of source programs– abstract syntax tree, polish notation and three address code, types of three address statements and its implementation, syntax directed translation into three-address code, translation of simple statements, Boolean expressions and flow-of control statements	
UNIT-IV	**TYPE CHECKING AND RUN TIME ENVIRONMENT**
Type checking: Definition of type checking, type expressions, type systems, static and dynamic checking of types, specification of a simple type checker, equivalence of type expressions, type conversions, overloading of functions and operators; Run time environments: Source language issues, Storage organization, storage- allocation strategies, access to nonlocal names, parameter passing, symbol tables, and language facilities for dynamic storage allocation.	
UNIT-V	**CODE OPTIMIZATION AND CODE GENERATOR**
Code optimization: The principle sources of optimization, optimization of basic blocks, loops in flow graphs, peephole optimization; Code generator: Issues in the design of a code generator, the target machine, runtime storage management, basic blocks and flow graphs, a simple code generator, register allocation and assignment, DAG representation of basic blocks.	
Text Books:	
1. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, ―Compilers–Principles, Techniques and Tools‖, Pearson Education, Low Price Edition, 2004	
Reference Books:	
1. Kenneth C. Louden, Thomson, ―Compiler Construction– Principles and Practice‖, PWS Publishing 1st Edition ,1997 2. Andrew W. Appel, ―Modern Compiler Implementation C‖, Cambridge University Press, Revised Edition, 2004. 3. Richard Arnold Johnson, Irwin Miller and John E. Freund, "Probability and Statistics for Engineers", Prentice Hall, 8th Edition, 2013.	

XVI. COURSE PLAN:

The course plan is meant as a guideline. Probably there may be changes.

Lecture No	Topics to be covered	Course Learning Outcomes (CLOs)	Reference
1-4	Introduction, Analysis of the source program, Difference of compiler and interpreter, Phases of compilation, Grouping of phases, role of lexical analyzer.	CLO 1	T1:1.1-1.5 R1:1.1
5-6	Construction of regular grammar from regular expression, NFA,DFA.	CLO 2	T1: 3.6-3.7 R1:2.2-2.4
7	Concept of pass and difference between pass and phase.	CLO 1	T1: 1.5
8	Bootstrapping and types of compiler.	CLO 3	T1: 1.1 R1:1.6
9-11	Lex-Lexical analyzer generator, Derivations and parse tree, regular expressions v/s context free grammar.	CLO 6	T1: 3.8-4.3 R1:3.1-3.3
12-15	Backtracking, eliminating ambiguity from dangling-else grammar, Elimination of left recursion and left factoring, Recursive decent parsing, Finding FIRST and FOLLOW.	CLO 4	T1: 4.3-4.4 R1:4.1
16-18	Construction of parse tables, Predictive parsing, LL(1) grammar.	CLO 4	T1: 4.5-4.7 R1:4.3-4.5
19-21	Handles, handle pruning, Shift reduce parsing, Conflicts during shift-reduce parsing, LR parsers- Goto and closure functions.	CLO 5	T1: 4.5-4.7 R1:5.1-5.2
22-24	LR(0) and SLR and construction of parser table for SLR.	CLO 5	T1: 4.7 R1:5.3
25-27	CLR operations and construction of parser table for LALR., LALR operations and construction of parser table for LALR.	CLO 5	T1: 4.7 R1:5.4-5.5
28	Description of error recovery.	CLO 11	T1: 4.7 R1:5.6
29	Yacc parser generator.	CLO 6	T1: 4.9 R1:5.5
30	Abstract syntax tree, three address code.	CLO 9	T1: 4.9
31-32	Introduction to attributes grammars, Syntax directed definitions, applications of SDD, Implementing L-attributed SDD's.	CLO 8	T1: 5.1-5.4 R1:6.1
33	Control flow, back patching, translation of simple statements, Boolean expressions.	CLO 10	T1:8.4-8.6
34-35	Type checking, type expressions, type systems, Type conversions, Overloading.	CLO 11	T1: 6.1 R1:6.4-6.5
36-37	Source language issues, Storage organization, storage-allocation strategies. Access to nonlocal names, parameter passing.	CLO 14	T1: 7.1-7.5 R1:7.1
38-39	Symbol tables, and language facilities for dynamic storage allocation.	CLO 15	T1: 7.6-7.7
40	Principle sources of optimization.	CLO 16	T1: 10.2
41-47	Optimization of basic blocks - Local, global and scope optimization, Loops in flow graphs, peephole optimization.	CLO 17	T1:10.1-10.2 T1: 10.4,9.9
48-49	Introduction, issues in code generation, , the target machine.	CLO 18	T1: 9.1-9.2
50	Runtime storage management.	CLO 13	T1: 9.3 R1:7.6
51-52	Basic blocks and flow graphs.	CLO 17	T1: 9.4
53-54	A simple code generator, register allocation and assignment.	CLO 20	T1: 9.6-9.7 R1:8.1-8.8
55	DAG construction, applications.	CLO17	T1: 9.8

XVII. GAPS IN THE SYLLABUS-TO MEET INDUSTRY / PROFESSION REQUIREMENTS:

S NO	Description	Proposed Actions	Relevance With POS	Relevance With PSOS
1	ANother Tool for Language Recognition (ANTLR)	Seminars / Guest Lectures	PO1, PO3, PO4	PSO 1
2	Java Compiler Compiler(JAVACC)	Seminars / Guest Lectures	PO 1, PO 3,PO4	PSO 3
3	Familiarization Lexer and Parser Tools	Seminars	PO 1	PSO 1
4	Awareness on Computer Architecture for fine tuning Target Codes	Seminars / NPTEL Video Lectures/ Moocs	PO 3	PSO 2

Prepared by:
Ms. E Uma Shankari, Assistant Professor

HOD, IT