JNTUA COLLEGE OF ENGINEERING (AUTONOMOUS): PULIVENDULA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
B.Tech
SUBJECT: Compiler Design (CD)

LESSON PLAN

Course Code	15ACS24			
Course Title	Compiler Design			
Course Structure	Lectures	Tutorials	Practical's	Credits
	3	1	0	3
Course Coordinator	Sri M. Chenna Keshava			
Team of Instructors	Sri G. Murali			

I. Course Overview:

The course is intended to teach the students the basic techniques that underlie the practice of Compiler Construction. The course will introduce the theory and tools that can be tenderly employed in order to perform syntax-directed translation of a high-level programming language into an executable code.

These techniques can also be employed in wider areas of application, whenever we need a syntax-directed analysis of symbolic expressions and languages and their translation into a lower-level description. They have multiple applications for man-machine interaction, including verification and program analysis.

In addition to the exposition of techniques for compilation, the course will 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, such as recursion, dynamic memory allocation, types and their inferences, object orientation, concurrency and multi-threading.

II. Prerequisite(s):

Level	Credits	Periods / Week	Prerequisites
UG	3	3	Mathematical background, Logical Thinking, Automata Theory

III. Assessment:

FORMATIVE ASSESMENT	
Mid Semester Test I for 20 Marks in first 2 units is conducted at the end of 9th week. Mid Semester Test II for 20 Marks in last three units is conducted at the end of the course work. 80% Marks taken from the test which secured highest marks	20 Marks

and 20% marks from the other test is taken as final	
Multiple Choice Mid Semester Test I for 10 Marks form first 2 units is conducted at the end of 9th week. Multiple Choice Mid Semester Test II for 10 Marks from last 3 units is conducted at the end of the course work. 80% Marks taken from the test which secured highest marks and 20% marks from the other test is taken as final	10 Marks
Total (Formative)	30 Marks
SUMMATIVE ASSESMENT	
End Semester Examination in all units is conducted for 70 Marks	70 marks
Grand Total	100 Marks

IV. Course Objectives:

1. Realize that computing science theory can be used as the basis for real applications introduce the major concept areas of language translation and compiler design. Learn how a compiler works.
2. Know about the powerful compiler generation tools and techniques, which are useful to the other non-compiler applications.
3. Know the importance of optimization and learn how to write programs that execute faster.

V. Course Outcomes:

1. Student can able to design a compiler for a simple programming language.
2. Student can able to use the tools related to compiler design effectively and efficiently can write an optimized code.

VI. Program outcomes:

a An ability to apply knowledge of computing, mathematical foundations, algorithmic principles, and computer science and engineering theory in the modeling and design of computer-based systems to real-world problems (fundamental engineering analysis skills)
b An ability to design and conduct experiments, as well as to analyze and interpret data (information retrieval skills)
c An ability to design , implement, and evaluate a computer-based system, process, component, or program to meet desired needs, within realistic constraints such as economic, health and safety, manufacturability, and sustainability (Creative Skills)
d An ability to function effectively on multi-disciplinary teams (team work)
e An ability to analyze a problem, identify, formulate and use the appropriate computing and engineering skills for obtaining its solution (engineering problem solving skills)
f Obtaining the knowledge of algorithmic skills regarding data structures. (program oriented skills)
g An ability to communicate effectively both in writing and orally (speaking / writing skills)
h The broad education necessary to analyze the local and global impact of computing and engineering solutions on individuals, organizations, and society (engineering impact assessment skills)

i Recognition of the need for, and an ability to engage in continuing professional development and life-long learning (continuing education awareness)
j A Knowledge of structural skills which are related to theoretical skills for programming (detailed subject oriented skills).
k An ability to use current techniques, skills, and tools necessary for computing and engineering practice (practical engineering analysis skills)
l An ability to apply design and development principles in the construction of software and hardware systems of varying complexity (software hardware interface)
m An ability to recognize the importance of professional development by pursuing postgraduate studies or face competitive examinations that offer challenging and rewarding careers in computing (successful career and immediate employment).

VII. Syllabus:

COMPILER DESIGN

L	T	P	C
3	1	0	3

UNIT – I

Overview of Compilation and Language processing:Preprocessor-Compiler-assembler-interpreters-pre-processors-linkers and loaders-structure of a compiler- 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.

UNIT – II

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

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.

UNIT – IV

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.

Intermediate code Generation: Intermediate languages, Declarations, Assignment statements, Boolean expressions, back patching.

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

UNIT – V

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

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. Principles of compiler design -A.V. Aho .J.D.Ullman; Pearson Education. (Text book edition)
2. Modern Compiler Implementation in C- Andrew N. Appel, Cambridge University Press.
3. Compilers Principles, Techniques and Tools-Alfred V.Aho, Ravi Sethi, JD Ullman, Pearson Education, 2007.

REFERENCES:

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

VIII. Course Plan:

The course plan is meant as a guideline. There may probably be changes.

Lecture No.	No. of periods required	Course Learning Outcomes	Topics to be covered	Reference
1	2	Phases of compilation	Introduction to compiler design	T1:1,T2:1.6,R1:1.4
2	2	Lexical analysis	Concepts of compiler design	T1:1.1,R2:1.5
3	2	Regular grammar	Explanation about regular grammar expressions	T1:1.2,R2:1.7,R3:1.5
4	2	Pass and phases of translation.	Phases of compilation	T1:1.3,T2:1.4
5	2	Lex lexical analyzer	Knowing the concept of Lexical analyzer	T1:1.4,T2:2.2,T3:2.3
6	2	List out the building blocks of software quality.	Architectural conception in Absence of Experience	T1:1.5,R1:2.4
7	2	Context free grammars	Writing Context free grammars solutions	T2:2.2,T3:2.5,R3:2.6
8	2	Top down parsing	Important characteristics of Top down parsing	T1:2.3,T2:2.7
9	2	Different types of Parsers	Backtracking and LL(1) parsers explanations	T1:2.4,T2:2.8
10	2	Recursive descent parsing	Knowing the solutions for problems regarding RDP	T1: 2.5, T2:2.6,T3:2.9
11	2	Predictive parsing	Evaluating expressions of predictive parser	T1:2.7,R1:2.6
12	2	Preprocessing steps for predictive parser.	Rules of predictive parser	T1.3.1,T3:4.7,R2:3.2
13	2	Shift reducing	Rules for shift reduce	T1:3.2,R1:3.3

		parsing.	parser	
14	2	Definition of LR and LALR parsing	Rules for LR and LALR parsers	T1:3.3,T2:3.4
15	2	Error recovery in parsing	Rules for error recovery	T1:3.4,R3:3.6
16	2	Definition of YACC	Knowing the automatic parser generator	T1:3.5-3.6-3.7,R2:4.1
17	2	Illustrate the principles which lead to abstract syntax tree	Process of abstract syntax tree	T1:3.8-3.9,R1:3.9
18	2	Three address code	Techniques for three address code	T1:3.10,R1:3.12
19	2	Describing attributed grammars.	Explanation of attributed grammars	T1.3.11-3.12,R3:4.7
20	2	Illustrate the Syntax directed translation	Basic definitions of syntax directed translation	T1:4.1-4.2,R2:5.1
21	2	Describing the type checker	Basic definitions of the type checker	T1:4.3,T3:4.6,R3:5.1
22	2	Describe the Intermediate code forms	Rules of intermediate code forms	T1:4.4-4.5,R1:5.4
23	2	Illustrate the Symbol table format	Explanation about symbol table	T1:4.6,T3:3.2,R1:3.8
24	2	Organization for block structures languages.	Definition of block structures language	T2:6.2,R3:4.8
25	2	Explanation of building the software engineering teams.	Building software engineering teams, project scheduling and tracking	T1:4.7-4.8,R2:6.2
26	2	Describe the Hashing	Hashing techniques	T2:2.2,R1:3.6,R2:6.7
27	2	Code optimization	Consideration of optimization techniques	T1:4.9,T2:5.1
28	2	Scope of optimization	Contents of optimization.	T1:5.4,T2:4.2,R1:4.4
29	2	Local and loop optimizations.	Explanation of types of optimizations.	T1:5.6,R2:6.3,T3:6.8
30	2	Describe the Flow graph	Explanation about flow graphs.	T1:5.7,R2:6.4,T3:6.7
31	2	Data flow equation	Rules regarding data flow equation	T1:5.8,T3:4.2,R1:5.1
32	2	Global optimization redundant sub expressions	Explanation of global optimization techniques and sub expressions	T1:6.1,T3:5.2,R1:5.9
33	2	Introduction to variable elements and live variable	Explanation about live variable analysis	T1:6.2,R1:5.2,R2:6.6

		analysis		
34	2	Introduction to Copy propagation	Explanation of copy propagation	T1:6.4,T3:5.8,R1:6.2
35	2	DAG for register allocation	Explanation of DAG	T1:6.5,T3:6.2,R2:6.7

IX. Mapping course outcomes leading to the achievement of the program outcomes:

Course Outcomes	Program Outcomes												
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
1	H		S		H						H	S	
2	H	S	S		S						H		

S = Supportive **H = Highly Related**

Justification of Course syllabus covering Course Outcomes:

By covering the syllabus a student can understand the major concepts are as of language translation and compiler design. Student is able to develop the real time projects by using powerful compiler generation tools and techniques.

Justification of CO's –PO's Mapping Table:

1. By mapping CO-1 to the PO's A, C, E, K, and L which are related to the course CO1: Student can able to design a compiler for a simple programming language.

2. By mapping CO-2 to the PO's A, B, C, E, and K, which are related to the course CO2: Student can able to use the tools related to compiler design effectively and efficiently has been write an optimized code.