GPCET
Pioneering Innovative Education G.PULLAIH COLLEGE OF ENGINEERING & TECHNOLOGY

Nandikotkur Road, Kurnool– 518002

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Semester – VI (2017-2018)

COURSE DESCRIPTION

Course Code	:	**15A05601**
Course Title	:	**COMPILER DESIGN**
Course Structure	:	**Lectures 3 · Tutorials 1 · Practical - · Credits 3**
Course Coordinator	:	**R.Sandeep Kumar ,Asst Professor**
Team of Instructors	:	**N.Poorna Chandra Rao**

I. Course Overview:

The main objective of this course is to introduce the major concept areas of language translation and compiler design and to develop an awareness of the function and complexity of modern compilers. This course is a study of the theory and practice required for the design and implementation of interpreters and compilers for programming languages.

II .Prerequisite(s)

Level	Credits	Periods/ Week	Prerequisites
UG	4	5	Formal languages and automata, comparative languages(Experience with programming in imperative languages such as C/C++

III. Marks Distribution:

Sessional Marks	University End Exam Marks	Total Marks
For theory subjects, during the semester, there shall be two midterm examinations and for first year there shall be three midterm examinations. Each midterm examination consists of objective paper for 10 marks and subjective paper for 20 marks with duration of 1hour 50 minutes (20 minutes for objective and 90 minutes for subjective paper). Objective paper shall be for 10 marks. Subjective paper shall contain 5 questions of which student has to answer 3 questions evaluated* for 20 marks.	70	100

IV. Evaluation Scheme:

S.No	Component	Duration (hours)	Marks
I	I MID EXAMINATION	1hr 50 min	30
2	II MID EXAMINATION	1hr 50 min	30
3	EXTERNAL EXAMINATION	3hrs	70

V. Course Educational Objectives:

1. To learn the process of translating a modern high-level language to executable code.
2. To provide a student with an understanding of the fundamental principles in compiler design and to provide the skills needed for building compilers for various situations that one may encounter in a career in Computer Science.
3. To develop an awareness of the function and complexity of modern compilers.
4. To apply the code generation algorithms to get the machine code for the optimized code.
5. To represent the target code in any one of the code formats
6. To understand the machine dependent code
7. To draw the flow graph for the intermediate codes.
8. To apply the optimization techniques to have a better code for code generation

VI. Course Outcomes:

By the end of the course, the successful student will be able to do:

- To realize basics of compiler design and apply for real time applications.
- To introduce different translation languages
- To understand the importance of code optimization
- To know about compiler generation tools and techniques
- To learn working of compiler and non compiler applications
- Design a compiler for a simple programming language

VII. How Course Outcomes are assessed:

By the end of this course the student should be capable of

- Able to convert any instruction of a program to convert from source language to target language and should be recognize what happens at each and every phase of a compiler.
- Student should be in a position to understand the different types of parsing techniques and should be in a position to solve the problem.
- Student should analyze the program and minimize the code by using optimizing techniques which helps in reducing the no. of instructions in a program and also utilization of registers in an effective way.
- Able to write the code by using YACC and lex.

VII. **How Course Outcomes are assessed:**

Outcome		Level	Proficiency assessed by
A	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering Fundamentals and an engineering specialization to the solution of complex engineering problems.	H	--
B	**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	--
C	**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	Tests and assignments
D	**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	N	--

	conclusions.		
E	**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	Tests and assignments
F	**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	--
G	**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		
H	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.	N	--
I	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.	H	--
J	**Communications:** 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 receive clear instructions.	N	--
K	**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.	H	Tests and assignments
L	**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. Syllabus:

UNIT I

Introduction: Language processors, Phases of a compiler, Pass and phase, Bootstrapping, Compiler construction tools, Applications of compiler technology, Programming language basics
Lexical Analysis: Role and Responsibility, Input buffering, Specification of tokens, Recognition of tokens, LEX tool, Design of a Lexical Analyzer generator

UNIT II

Syntax Analysis: Role of the parser, Context Free Grammars - Definition, Derivations, Parse trees, Ambiguity, Eliminating ambiguity, Left recursion, Left faltering.

TOP Down Parsing: Recursive descent parsing, Non-recursive predictive parsing, LL(1) grammars, Error recovery in predictive parsing.

Bottom Up Parsing: Handle pruning, Shift-Reduce parsing, Conflicts during shifts- reduce parsing, SLR Parsing, Canonical LR(1) parsers, LALR parsers, Using ambiguous grammars, YACC tool.

UNIT III

Syntax Directed Translation: Syntax Directed Definitions, Evaluation orders for SDD‘s, Application of SDT, SDT schemes, Implementing L-attribute SDD‘s.

Intermediated Code Generation: Need for intermediate code, Types of intermediate code, Three address code, Quadruples, Triples, Type expressions, Type equivalence, Type checking, Translation of expressions, control flow statements, switch statement, procedures, back patching.

UNIT IV

Run Time Storage Organization: Scope and Life time of variable, Information associated with symbols in symbol table, Data Structures for symbol Table, Static vs dynamic storage allocation, Stack allocation of space, Access to non-local data on stack, Heap management, Introduction to garbage collection

Optimization: Need and objective of optimization, Places of optimization, Optimization at user level, Construction of Basic blocks and Processing, Data Flow analysis using flow graph, Data flow equations for blocks with back ward flow control, Principles source of optimization and transformations, Alias, Loops in flow graphs, Procedural optimization, Loop optimization

UNIT V

Code Generation: Issues in code Generation, Target machine architecture, Subsequent Use Information, Simple code generator, Register allocation, DAG representation of basic blocks, Code Generation from intermediate code, Peephole optimization, Code scheduling

Text Books :

1. Compilers Principles, Techniques and Tools, Second Edition, Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman., Pearson.

2. Compiler Design, K. Muneeswaran., Oxford University Press, 2012

Reference Books :

1. Compiler Construction, K.V.N Sunitha, Pearson, 2013

2. Engineering a Compiler, Second Edition, Keith D. Cooper & Linda Torczon., Morgan Kaufmann, Elsevier.

3. Compilers Principles and Practice, Parag H. Dave, Himanshu B. Dave., Pearson

4. Compiler Design, Sandeep Saxena, Rajkumar Singh Rathore., S.Chand publications
5. Compiler Design, Santanu Chattopadhyay
6. Principals of Compiler Design, Nadhni Prasad, Elsevier.

VIII. Course Plan:

Lect. No	Learning Objective	Topics to be covered	Reference
1-2	To convert an instruction to machine code	Phases of a compilation	T1:1.3
3-4	Identifying the tokens in the first phase	Lexical Analysis	T1:2.6
5-6	Identifying the grammar is in CFG or not and Conversions from NFA to DFA	Regular Grammar,Regualr Expressions	T2
7	Knowing the difference between and pass and a phase	Pass and phases of a translation	T1:1.5
8	Knowing the types of a compiler	Bootstapping	T1:11.2
9	How the LEX identifies the tokens in a program	LEX analyzer generator	T1:3.8
10	Each and every grammar should be in an unique format by Removing useless productions, useless Symbols, removing epsilon productions	Context free grammars & Top-down Parsing LMD,RMD	
11-12	A program written for any given grammar	Recursive descent parsing	T1:4.4
13-16	Finding FIRST AND FOLLOW functions	Predictive parsing	T1:4.4
17-18	To generate a parsing table and check whether the given string is accepted or not	Preprocessing steps required for predictive parsing	T1:4.4
19	To determine whether the string should be push or pop from the stack	Shift Reduce parsing	T1:4.5
20-23	To solve how the parsing is done in bottom up parsers	SLR, LALR and CALR parsing	T1
24	Removing the conflicts occurred in LR parsers	Handling ambiguous grammar	T1:4.8
25-26	To write parsing program using YACC(it is a parser generator)	YACC automatic generation parsing algorithm	T1:4.9
27	To Know the intermediate forms that has generated from parsers	Intermediate forms of source program	T1:8.1

28	**Generating the tree by using intermediate forms**	**Abstract syntax tree**	**T1:5.2**
29-30	**To draw the tree diagram for a given expression in specific format**	**Polish notation and three address codes**	**T1:8.1**
31-32	**To know about synthesized and Inherited attribute**	**Attributed grammars**	**T1:5.1**
33-34	**An augmented CFG can be generated**	**Syntax directed translation**	**T1:5.5**
35-36	**Conversions from programming language into intermediate code forms**	**Conversions from programming language into intermediate code forms**	**T1:5.2**
37-38	**To check the types of a variables**	**Type checker**	**T1:6.2**
46	**To know how the data is stored in symbol table**	**Symbol table format**	**T1**
47	**To know how the data table entries are formed in the different types of block formats**	**Organization for block structures language**	**T1**
48		**Hashing**	**T1**
49	**A tree diagram for a variables located either for global or local**	**Tree structure representation of scope information**	**T1**
50	**To determine how the data is stored in block or non block structure storage allocation**	**Block structure and non block structure storage allocation**	**T1**
51	**To know how the data is allocated dynamically**	**Static ,Run time storage heap allocation memory**	**T1**
52-53	**To know the various types of data will be allocated in symbol table**	**Storage allocation for arrays and strings and records**	**T1**
54-55	**To minimize the code based on whether the scope lies in static or dynamic**	**Scope of optimization**	**T1**
56-57	**To know the various techniques or steps in reducing the code**	**Local Optimization**	**T1**
58	**Different techniques of reduction methods**	**Frequency reduction**	**T1**
59-60	**To minimize the code if any code is repeated in an expression**	**Folding ,DAG representation**	**T1**
61-62	**To know about how the control flow of information is added to blocks**	**Flow graph**	**T1**
63	**The way of representing the expressions in flow graph**	**Data Flow Equation**	**T1**
64	**Optimization that can be applied to procedures or functions which is not possible in local**	**Global Optimization**	**T1**

	optimization		
65	Reducing the code by using some common techniques	Redundant sub expression evaluation	T1
66	Understanding how to compute the data flow equations	Live variable analysis	T1
67	A flow graph which contains ud chains in blocks	Copy propagation	T1
68	To know the different forms of object code whether it is in absolute or relocatable machine code or assembler code	Object code forms	T1
69-70	To know the different techniques how register is allocated for the expression	DAG for register allocation	T1

VIII. Mapping of course objectives to the achievements of course outcomes:

Course Objectives	Program Outcomes												
	a	b	c	D	e	f	g	h	i	j	k	L	m
I	S				H		S				S		
II		H							H				
III										H			
IV	H	S									H		
V					H			H					
VI	S						H						

IX. Mapping of 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								
2	S				H								
3	S				H						S		
4					H		S						
5					S						H		
6	S												H
7		S			H								

Prepared By : R.Sandeep Kumar

Date : 07-01-2014