PRINCIPLES OF COMPILER DESIGN

Semester	Subject Code	Category	Lecture Hrs		Theory Hrs		Practical		Credits
			Per week	Per Sem	Per week	Per Sem	Per week	Per Sem	
II	21CPCS2C	CORE - VI	5	75	5	75	0	0	4

COURSE OBJECTIVE

- This paper helps the students to know about how to compile and parsing the software programs.

COURSE OUTCOME

successful completion of the course, students will be able to

CO Number	CO Statement	Knowledge Level (K1-K4)
CO1	Understand the phases and tools available in compiler	K2
CO2	Design and implement a lexical analyzer	K3
CO3	Compare and analyze different types of compilers	K4
CO4	Specify appropriate translations to generate Intermediate code for the given Programming language Construct	K3
CO5	Identify sources for Code Optimization	K4

Knowledge Level – K1-Remember, K2- Understand, K3-Apply, K4-Analyze

MAPPING WITH PROGRAMME OUTCOME

COS	PO1	PO2	PO3	PO4	PO5	PO6
CO1	S	M	M	S	S	M
CO2	S	S	S	M	M	S
CO3	S	S	S	S	S	M
CO4	S	S	S	S	M	M
CO5	S	S	S	M	S	S

S-Strong, M-Medium and L-Low

UNIT I – INTRODUCTION TO COMPILERS **14 Hours**

Introduction to compiling - Compilers - Analysis of the source program - Phases of a compiler. Grouping of Phases - Compiler construction tools.

UNIT II – LEXICAL ANALYSIS **15 Hours**

Lexical Analysis - Role of the lexical Analyzer - specification and Recognition of tokens - Language for specifying lexical Analyzer - Finite Automata -regular expression to NFA - Design of lexical Analyzer generator - Optimization of DFA - based pattern matchers.

UNIT III – SYNTAX ANALYSIS **15 Hours**

Syntax Analysis - Role of parser - context free grammars - Top down parsing - Bottom up parsing - Operator precedence parsing - LR Parsers.

UNIT IV – SYNTAX DIRECTED TRANSLATION **15 Hours**

Syntax Directed Translation: Syntax directed Definitions - construction of syntax trees - Bottom up evaluation of attributed definition - Bottom up evaluation of inherited attributes - Recursive evaluators.

UNIT V -INTERMEDIATE CODE GENERATION AND OPTIMIZATION 16 Hours

Intermediate Code generation: Intermediate languages - Declaration - Assignment statements. Procedure calls - Runtime storage management. Code generation and optimization: Basic blocks and Flow graphs - DAG representation.

Distribution of Marks: Theory 75% and Problem 25%

TEXTBOOKS

S. NO	AUTHORS	TITLE	PUBLISHERS	YEAR OF PUBLICATION
1	Alfred Aho Ravi Sethi JeffyD.Ullman	Compilers-Principles,Techniques and Tools	Pearson	1986
2	Dick Grune, Bal, Langendoen,Jacobs	Modern Compiler Design	Wiley	2012
3	K.Muneeswaran	Compiler Design	Oxford University Press	2013

REFERENCEBOOKS

S. NO	AUTHORS	TITLE	PUBLISHERS	YEAR OF PUBLICATION
1	David Galles	Modern Compiler design	Pearson education Asia	2001
2	Steven S. Muchnick	Advanced Compiler Design and implementation	Morgan Kaufmann Publishers	2000
3	C.N.FisherR.J.Le Blanc	Crafting a compiler with c	Pearson Education	2000

WEB RESOURCES

1.https://www.geeksforgeeks.org/compiler-lexical-analysis

2.https://ieeexplore.ieee.org/document/7779385/

3.https://www.tutorialspoint.com/compiler_design/compiler_design_tutorial.pdf **TEACHING METHODOLOGY**

- Class room teaching & Group discussions
- Seminars & Smart Class room
- Chart/Assignment & Simulation Model

SYLLABUS DESIGNER

- Mrs.G.SANGEETHA LAKSHMI, Assistant professor & HOD, Dept of Computer Science & Applications
- Dr. R HAMSAVENI, Assistant professor, Dept of Computer Science & Applications