COURSE CODE	COURSE TITLE	L	T	P	C
UCS1602	COMPILER DESIGN	3	0	2	4

OBJECTIVES

- To learn various phases of a compiler
- To learn various parsing techniques
- To understand intermediate code generation
- To learn to implement code generator and optimize the code.

UNIT I INTRODUCTION TO COMPILERS 9

Language processors -- Phases of compiler -- Role of lexical analyzer -- Input buffering -- Specification of tokens -- Recognition of tokens; Lexical analyzer generator: Structure of lex program – Look ahead operator and conflict resolution.

UNIT II SYNTAX ANALYSIS 9

Role of Parser -- Writing grammars for language constructs -- Types of grammars: Ambiguity -- Deterministic and recursive; Top down parsers: Recursive descent parser -- Predictive parser; Bottom up parsers: SLR Parser -- CLR Parser -- LALR Parser; Error handling and recovery in syntax analyzer; Syntax analyzer generator: Structure of yacc program -- Creating =yacc= lexical analyzers with =lex=.

UNIT III INTERMEDIATE CODE GENERATION 9

Syntax directed definitions: Synthesized attribute -- Inherited attribute -- Dependency graph -- Evaluation order of syntax directed definitions; Intermediate languages: Syntax tree -- Three address code; SDD for type checking -- Declarations -- Evaluation of expressions and flow of control statements -- Bottom-up evaluation of S-attribute definitions.

UNIT IV RUNTIME ENVIRONMENTS AND CODE GENERATION 9

Source language issues -- Storage organization -- Storage allocation strategies: Static, Stack and Heap -- Implementation of symbol table -- Issues in code generation -- Design of a simple code generator.

UNIT V CODE OPTIMIZATION 9

Principal sources of optimization -- DAG -- Optimization of basic blocks -- Global data flow analysis -- Introduction to Low Level Virtual Machine (LLVM) -- Design of LLVM -- Core libraries -- Developing plugin in LLVM.

TOTAL PERIODS(THEORY): 45

SUGGESTIVE EXPERIMENTS

1. Implementation of Lexical Analyzer using Lex Tool
2. Implementation of Arithmetic Calculator using LEX and YACC
3. Generation of TAC for a simple program using LEX and YACC
4. Consider a simple program as an input and process this code to print the intermediate code after every phase. It is necessary to print the output of lexical, syntax, semantic, intermediate code generation, code optimization and code generation phases
5. Study of LLVM framework.

TOTAL PERIODS(PRACTICAL): 30
TOTAL PERIODS: 75

OUTCOMES

On successful completion of this course, the student will be able to:

- Design a lexical analyzer for a sample language (K3)
- Apply different parsing algorithms to develop the parsers for the given grammar (K3)
- Write syntax directed translation for programming language constructs (K3)
- Understand and implement a simple code generator (K3)
- Understand and implement code optimization techniques (K3).

TEXTBOOKS

1. Alfred V Aho, Monica S Lam, Ravi Sethi, Jeffrey D. Ullman,"Compilers: Principles, Techniques and Tools", 2nd Edition,Pearson Education, 2009.

REFERENCE BOOKS

1. Randy Allen, Ken Kennedy, "Optimizing Compilers for Modern Architectures:A Dependence BasedApproach", Morgan Kaufmann Publishers,2002.
2. Steven S. Muchnick, "Advanced Compiler Design and Implementation", Morgan Kaufmann Publishers – Elsevier Science, India, Indian Reprint, 2003.
3. Keith D Cooper, Linda Torczon, "Engineering a Compiler", Morgan Kaufmann Publishers Elsevier Science, 2004.
4. Andrew W. Appel,"Modern Compiler Implementation in C", CambridgeUniversity Press, 1st edition, 2004.
5. Watson,Des, "A PracticalApproach to Compiler Construction", 1st edition, Springer-Verlog, 2017.