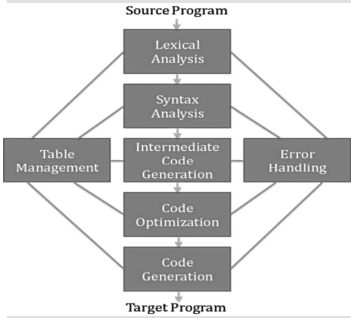
19CS303 COMPILER DESIGN



source :
https://www.tutorialspoint.com/compiler_design

Hours Per Week :

L	T	P	C
3	-	-	3

Total Hours :

L	T	P	CS	WA/RA	SSH	SA	S	BS
45	-	-	5	5	30	20	5	5

PREREQUISITE COURSES: Programming for Problem Solving- I & II and Formal Languages and Automata Theory.

COURSE DESCRIPTION AND OBJECTIVES:

This course introduced the concepts of lexical analyser, parser, code generation and code optimization techniques. The objective of this course is to enable the student to acquire the knowledge of various phases of compiler such as lexical analyser, parser and code optimization.

The student should be able to Gain the foundation knowledge for understanding the theory and practice of compilers and compiler design concepts; language recognition, symbol table management, compiler parsing techniques, semantic analysis and code generation.

COURSE OUTCOMES:

Upon completion of the course, the student will be able to achieve the following outcomes:

COs	Course Outcomes	POs
1	Understand the different phases of compiler with various examples.	1
2	Apply different Parsing and optimization techniques in the design of compiler.	1
3	Analyze the code optimaization techniques..	2
4	Design and implement an algorithm for compiler segments and evaluate the algorithm for optimized code generation.	3

SKILLS:

- ✓ *Design parsers using top-down and bottom-up approaches.*
- ✓ *Usage of tools like LEX and YACC.*
- ✓ *Design a simple compiler.*

UNIT – I **L- 9**

INTRODUCTION: The evolution of programming languages and basic language processing system; The structure of a compiler; Bootstrapping; Lexical analysis-the role of lexical analyzer; Input buffering; Specifications and recognition of tokens; Lexical analyzer generator- LEX tool.

UNIT – II **L- 9**

SYNTAX ANALYSIS : The role of the parser ; Context-free grammars; Top down parsing-Recursive-descent parsing, predictive parsing; Bottom-up parsing-handle pruning, shift reduce parsing and operator precedence parsing, LR Parsers - SLR, CLR and LALR; Parser generators - Yacc Tool.

UNIT – III **L- 9**

SEMANTIC ANALYSIS: Type checking; Syntax directed definition(SDD) and translation schemes(TS); Application of SDD and TS; Translation of expressions and control flow statements.

INTERMEDIATE REPRESENTATIONS: Three address code; Syntax tree; DAG.

UNIT – IV **L- 9**

RUN-TIME ENVIRONMENT: Storage organization; Stack allocation of space - activation trees, activation record and control stack; Access to non local data on the stack.

CODE IMPROVEMENT: Basic blocks and flow graphs; The principal sources of optimization - local optimization; Global optimization and loop optimization.

UNIT – V **L- 9**

CODE GENERATION: Issues in the design of code generator; Code-generation algorithm - register allocation and assignment and peephole optimization.

TEXT BOOKS:

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi and Jeffrey D. Ulman, “Compilers: Principles, Techniques and Tools”, 2nd edition, Pearson Education, 2012
2. Thomson, “Introduction to Theory of Computation”, 2nd edition, Sipser, 2006.

REFERENCE BOOKS:

1. V. Raghavan, “Principles of Compiler Design”, 2nd edition, McGrawHill, 2010.
2. John R.Levin, Tony Mason and Doug Brown, “Lex & YAAC”, 2nd edition, O Reilly, 2012.
3. Kenneth C Louden, “Compiler Construction”, 1st edition,Course Technology Inc,1997.
4. Henry E Bal and Carriel T H Jacobs, "Modern Compiler Design", 2nd edition, Springer, 2012.