© STD - 0431 - 2670142
FAX - 0431 - 2670143

DHANALAKSHMI SRINIVASAN
INSTITUTE OF TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to Anna University)
NH - 45, Trichy - Chennai Trunk Road,
SAMAYAPURAM,TRICHY - 621 112.
E.mail:dsit2011@gmail.com Website:www.dsit.ac.in

COURSE PLAN

Subject code: CS8602	**Branch/Year/Sem/Section: B.E CSE/III/VI**
Subject Name: Compiler Design	**Batch:2017-2021**
Staff Name:S.SRILEKAA	**Academic year:2019-2020**

COURSE OBJECTIVE

- To learn the various phases of compiler.
- To learn the various parsing techniques.
- To understand intermediate code generation and run-time environment.
- To learn to implement front-end of the compiler.
- To learn to implement code generator.

TEXT BOOK:

T1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques and Tools‖, Second Edition, Pearson Education, 2009.

REFERENCES:

R1.Randy Allen, Ken Kennedy, Optimizing Compilers for Modern Architectures: A Dependence based Approach, Morgan Kaufmann Publishers, 2002.
R2. Steven S. Muchnick, Advanced Compiler Design and Implementation‖, Morgan Kaufmann Publishers - Elsevier Science, India, Indian Reprint 2003.
R3. Keith D Cooper and Linda Torczon, Engineering a Compiler‖, Morgan Kaufmann Publishers Elsevier Science, 2004.
R4. V. Raghavan, Principles of Compiler Design‖, Tata McGraw Hill Education Publishers, 2010.
R5. Allen I. Holub, Compiler Design in C‖, Prentice-Hall Software Series, 1993.

WEB RESOURCES

W1: http://nptel.ac.in/syllabus/syllabus_pdf/106108052.pdf
W2:www.wikipedia.org
W3: http://studentsfocus.com/

TEACHING METHODOLOGIES:

➢ BB	- BLACK BOARD
➢ VIDEO	- VIDEO TUTORIAL
➢ PPT	- POWER POINT PRESENTATION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS8602 COMPILER DESIGN L T P C

3 0 2 4

1. **UNIT I INTRODUCTION TO COMPILERS 9**

   Structure of a compiler – Lexical Analysis – Role of Lexical Analyzer – Input Buffering –Specification of Tokens – Recognition of Tokens – Lex – Finite Automata – Regular Expressions to Automata – Minimizing DFA.

2. **UNIT II SYNTAX ANALYSIS 12**

   Role of Parser – Grammars – Error Handling – Context-free grammars – Writing a grammar –Top Down Parsing - General Strategies Recursive Descent Parser Predictive Parser-LL(1)Parser-Shift Reduce Parser-LR Parser-LR (0)Item Construction of SLR Parsing Table -Introduction to LALR Parser - Error Handling and Recovery in Syntax Analyzer-YACC.

3. **UNIT III INTERMEDIATE CODE GENERATION 8**

   Syntax Directed Definitions, Evaluation Orders for Syntax Directed Definitions, Intermediate Languages: Syntax Tree, Three Address Code, Types and Declarations, Translation of Expressions, Type Checking.

4. **UNIT IV RUN-TIME ENVIRONMENT AND CODE GENERATION 8**

   Storage Organization, Stack Allocation Space, Access to Non-local Data on the Stack, Heap Management - Issues in Code Generation - Design of a simple Code Generator.

5. **UNIT V CODE OPTIMIZATION 8**

   Principal Sources of Optimization – Peep-hole optimization - DAG- Optimization of Basic BlocksGlobal Data Flow Analysis - Efficient Data Flow Algorithm.

TOTAL: 45 PERIODS

Topic No	Topic Name	Books For reference	Page No	Teaching Methodology	No of periods required	Cumulative periods
UNIT I	**INTRODUCTION TO COMPILERS**					**(9)**
1.	Structure of a compiler	T1	4-12	BB	1	1
2.	Lexical Analysis, Role of Lexical Analyzer	T1	109-114	BB	1	2
3.	Input Buffering	T1	115-116	BB	1	3
4.	Specification of Tokens	T1	116-125	BB	1	4
5.	Recognition of Tokens	T1	130-136	BB	1	5
6.	Lex	T1	140-144	BB	1	6
7.	Finite Automata	T1	147-149	BB	1	7
8.	Regular Expressions to Automata	T1	152-166	BB	1	8
9	Minimizing DFA	T1	180-186	BB	1	9
LEARNING OUTCOME: **At the end of unit , the students will be able to** • Know the fundamentals of Compiler Design. • Understand the structure of a compiler. • Gain the knowledge about Lexical Analyser.						
UNIT II	**SYNTAX ANALYSIS**					**(12)**
1	Role of Parser – Grammars	**T1**	191-193	BB	1	10
2	Error Handling – Context-free grammars	T1	194-206	BB	1	11
3	Writing a grammar	T1	209-215	BB	1	12
4	Top Down Parsing	T1	217-219	BB	1	13
5	General Strategies Recursive Descent Parser Predictive Parser	T1	219-220	BB	1	14
6	LL(1)Parser	T1	222-228	BB	1	15
7	Shift Reduce Parser	T1	236-238	BB	1	16
8	LR Parser	T1	241-242	BB	1	17
9	LR (0)Item Construction of SLR Parsing Table	T1	242-252	BB	1	18
10	Introduction to LALR Parser Analyzer	T1	266-270	BB	1	19
11	Error Handling and Recovery in Syntax	T1	281-285	BB	1	20
12	YACC	T1	287-295	BB	1	21

LEARNING OUTCOME:

At the end of unit , the students will be able to

- Define the Role of Parser.
- Understand the design principles of syntax analyzer.
- Gain the knowledge about types of the parser.

UNIT – III INTERMEDIATE CODE GENERATION (8)

1	Syntax Directed Definitions	T1	303-306	BB	1	22
2	Evaluation Orders for Syntax Directed Definitions	T1	309-314	BB	1	23
3	Intermediate Languages	T1	357-358	BB	1	24
4	Syntax Tree	T1	358-360	BB	1	25
5	Three Address Code,	T1	363-369	BB	1	26
6	Types and Declarations	T1	370-376	BB	1	27
7	Translation of Expressions	T1	378-383	BB	1	28
8	Type Checking	T1	386-395	BB	1	29

LEARNING OUTCOME:

At the end of unit , the students will be able to

- Understand the concept of SDD.
- Gain knowledge about Code generation.
- Define the Code optimization.

UNIT IV RUN-TIME ENVIRONMENT AND CODE GENERATION (8)

1	Storage Organization	T1	427	BB	1	30
2	Stack Allocation Space	T1	430-438	BB	1	31
3	Stack Allocation Space	T1	--	BB	1	32
4	Access to Non-local Data on the Stack	T1	441-449	BB	1	33
5	Heap Management	T1	452-460	BB	1	34
6	Issues in Code Generation	T1	505-511	BB	1	35
7	Design of a simple code generator	T1	542-547	BB	1	36
8	Design of a simple code generator	T1	--	BB	1	37

LEARNING OUTCOME:

At the end of unit , the students will be able to

- Understand the concept of Storage Organization.
- Known about the code generator.

UNIT V	CODE OPTIMIZATION					(8)
1	Principal Sources of Optimization	T1	512-516	BB	1	38
2	Principal Sources of Optimization	T1	--	BB	1	39
3	Peep-hole optimization	T1	549-582	BB	1	40
4	DAG	T1	533-535	BB	1	41
5	Optimization of Basic Blocks	W3	533-541	BB	1	42
6	Basic Blocks Examples	W3	525-531	BB	1	43
7	Global Data Flow Analysis	W3	--	PPT	1	44
8	Efficient Data Flow Algorithm.	W3	--	PPT	1	45

LEARNING OUTCOME:
At the end of unit , the students will be able to

- Understand the concept of Optimization.
- Gain knowledge about Risk management

COURSE OUTCOME

At the end of the course, the student should be able to:

- Understand the different phases of compiler.
- Design a lexical analyzer for a sample language.
- Apply different parsing algorithms to develop the parsers for a given grammar.
- Understand syntax-directed translation and run-time environment.
- Learn to implement code optimization techniques and a simple code generator.
- Design and implement a scanner and a parser using LEX and YACC tools.

CONTENT BEYOND THE SYLLABUS

Various code optimization technique and its complexity

CONTINUES INTERNAL ASSESSMENT DETAILS

ASSESMENT NUMBER	I	II	MODEL
(UNIT)	(1st & 2nd units)	(3rd & 4th units)	(units 1-5)

ASSIGNMENT DETAILS

ASSIGNMENT NUMBER	I	II	III
TOPIC NUMBER FOR REFERENCE	1-18 (1st & 2nd units)	19-36 (3rd & 4th units)	1-45 (units 1-5)
DEAD LINE			

ASSIGNMENT NUMBER	BATCH	DESCRIPTIVE QUESTIONS/TOPIC (Minimum of 8 Pages)
I	B1 (R.Nos 1-18)	1. Analysis-Synthesis model of Compilation 2. Various Phases of a Compiler 3. Tool based approach to Compiler Construction
	B2 (R.Nos 19-36)	1. Lexical Analysis 2. Parser and Symbol Table,Token 3. Lexeme and Patterns
	B3 (R.Nos 37-302)	1. Error Reporting and Implementation 2. Regular definition 3. Transition diagrams
II	B1 (R.Nos 1-18)	1. LEX 2. Syntax analysis 3. Context free Grammers
	B2 (R.Nos 19-36)	1. Top Down Parsing 2. Recursive Descent Parsing 3. Bottom Up Parsing
	B3 (R.Nos 37-302)	1. LR Parsers (SLR, LALR, LR) 2. YACC 3. L- and S-Attributed Definitions
III	B1 (R.Nos 1-18)	1. DAG Representation of Programs 2. Code Generation from Dags 3. Peep Hole Optimization
	B2 (R.Nos 19-36)	1. Type Checking ,Run Time System 2. Intermediate Code Generation 3. Code Generation and Instruction Selection
	B3 (R.Nos 37-302)	1. Global Dataflow Analysis 2. Code Improving Transformations 3. Data Flow Analysis of Structured Flow Graphs

LIST OF EXPERIMENTS:

1. Develop a lexical analyzer to recognize a few patterns in C. (Ex. identifiers, constants, comments, operators etc.). Create a symbol table, while recognizing identifiers.
2. Implement a Lexical Analyzer using Lex Tool
3. Implement an Arithmetic Calculator using LEX and YACC
4. Generate three address code for a simple program using LEX and YACC.
5. Implement simple code optimization techniques (Constant folding, Strength reduction and Algebraic transformation)
6. Implement back-end of the compiler for which the three address code is given as input and the 8086 assembly language code is produced as output.

Session No	Experimental concepts to be covered	Teaching Aid	No. of Hours	Cumulative No. of Hours
1	Development of a lexical analyzer to recognize a few patterns in C.	PPT	2	5
2	Implementation of Lexical Analyzer using Lex Tool.		2	10
3	Implementation of Calculator using LEX and YACC.		2	15
4	Implementation of three address code using LEX and YACC		2	20
5	Implementation of Simple Code Optimization Techniques.		2	25
6	Implementation of back end of the compiler.		2	30

OUTCOMES:

Upon the completion of Compiler Design practical course, the student will be able to:

1. Understand the working of lex and yacc compiler for debugging of programs.

2. Understand and define the role of lexical analyzer, use of regular expression and transition diagrams.

3. Understand and use Context free grammar, and parse tree construction.

4. Learn & use the new tools and technologies used for designing a compiler.

5. Develop program for solving parser problems.

6. Learn how to write programs that execute faster.

PREPARED BY **VERIFIED BY**

S.SRILEKAA, AP/CSE **HOD/CSE**

APPROVED BY

PRINCIPAL