DESIGN OF LANGUAGE PROCESSORS

Semester II (Computer Engineering)

SUB CODE: MECE201

Teaching Scheme (Credits and Hours):

Teaching scheme				**Total**	**Evaluation Scheme**					
L	**T**	**P**	**Total**	**Credit**	**Theory**		**Mid Sem Exam**	**CIA**	**Pract.**	**Total**
Hrs	**Hrs**	**Hrs**	**Hrs**		**Hrs**	**Marks**	**Marks**	**Marks**	**Marks**	**Marks**
04	**00**	**02**	**06**	**05**	**3**	**70**	**30**	**20**	**30**	**150**

LEARNING OBJECTIVES:

The objective of this course is to introduce students to the following concepts underlying the design and implementation of language processors.

- Describe the steps and algorithms used by language translators.
- Recognize the underlying formal models such as finite state automata, push-down automata and their connection to language definition through regular expressions and grammars.
- Discuss the effectiveness of optimization.
- Explain the impact of a separate compilation facility and the existence of program libraries on the compilation process.
- To study different language processors and their contribution in language processing system.

OUTLINE OF THE COURSE:

Unit No	**Topics**
1	Introduction to Language Processing
2	Language Processors
3	Lexical Analyzer
4	Syntax Analyzer
5	Semantic Analyzer
6	Run Time System
7	Intermediate Code Generation
8	Code Optimization
9	Code Generation
10	Introduction to Natural Language Processing

Total hours (Theory): 60

Total hours (Practical): 30

Total hours: 90

DETAILED SYLLABUS:

Sr. No	Topic	Lecture Hours	Weight age (%)
Module I : Introduction			
1	**Introduction to Language Processing** • Language Processing Activities • Fundamentals of Language Processing • Fundamentals of Language Specifications	05	10
2	**Language Processors** • Macro Processors - Macro Definition and Call, Macro Expansion, Nested Macro Calls, Advanced Macro Facilities in 'C' • Assemblers - Elements of Assembly Language Programming, Assembly Scheme, single pass Assembler, Detailed Design of two pass assembler • Loader and Linkers - Relocation of Linking Concept, Design of Linker, Linker for MS DOS, Linking for overlays, Design of absolute loaders, Design of direct linking loaders	08	15
Module II : Language Analyzer			
3	**Lexical Analyzer** • Lexical Analysis • Specification of tokens • Recognition of tokens • Regular Expression • Finite automata • NFA • Lex Implementation	08	15
4	**Syntax Analyzer** • Syntax analysis • Types of Grammar • CFG, CFL, PDA & Turing Machine • Top down parsing • Bottom up parsing • YACC	12	15

5	**Semantic Analyzer** • Syntax directed Translation • L- attributed and S-attributed definitions with their implementation • Type checking	06	10
6	**Run Time System:** • storage organization • activation tree • activation record • parameter passing	02	05
Module III : Code Generation & Optimization			
7	**Intermediate Code Generation** • Run-Time Environment: issues and design • Intermediate Languages • Implementation of Three Address Code	05	10
8	**Code Optimization** • Optimization of basic blocks • Loops in flow graphs • Global data flow analysis • Code generation	05	10
9	**Code Generation** • Issues in the Design of a Code Generator • The Target Machine • Run-Time Storage Management • Basic Blocks and Flow Graphs • Next-Use Information • A Simple Code Generator • Register Allocation and Assignment • The DAG Representation of Basic Blocks • Peephole Optimization • Generating Code from DAGs • Dynamic Programming Code-Generation Algorithm	04	05
Module IV : Natural Language Processing			
10	**Introduction to Natural Language Processing** • NLP tasks in syntax, semantics, and pragmatics • Applications such as information extraction, question	05	05

	answering, and machine translation • The problem of ambiguity • Linguistics Essentials • Language Models		

INSTRUCTIONAL METHOD AND PEDAGOGY (Continuous Internal Assessment (CIA) Scheme)

- At the start of course, the course delivery pattern, prerequisite of the subject will be discussed.
- Lectures will be conducted with the aid of multi-media projector, black board, OHP etc.
- Attendance is compulsory in lecture and laboratory which carries 10 marks in overall evaluation.
- One internal exam will be conducted as a part of internal theory evaluation.
- Assignments based on the course content will be given to the students for each unit and will be evaluated at regular interval evaluation.
- Surprise tests/Quizzes/Seminar/tutorial will be conducted having a share of five marks in the overall internal evaluation.
- The course includes a laboratory, where students have an opportunity to build an appreciation for the concepts being taught in lectures.
- Experiments shall be performed in the laboratory related to course contents.

STUDENTS LEARNING OUTCOMES:

On successful completion of the course, the student will:

1. Understand how the design of a compiler requires most of the knowledge acquired during their study.
2. Develop a firm and enlightened grasp of concepts learned earlier in their study like higher level programming, assemblers, automata theory, and formal languages, languages, languages specifications, data structure and algorithms, operating systems.
3. Apply the ideas, the techniques, and the knowledge acquired for the purpose of other language processor design.
4. Working skills in theory and application of finite state machines, recursive descent, production rules, parsing, and language semantics.
5. Know about the powerful compiler generation tools, which are useful to the other non-compiler applications

REFERENCE MATERIAL:

Books:

1. Compilers, Principles, Techniques and Tools by A.V. Aho, R. Sethi and J.D.Ullman, Pearson
2. Advanced compiler Design Implementation by Steven S. Muchnick
3. The Compiler Design handbook: Optimization and Machine Code Generation by Y. N. Shrikant and Priti Shankar, Second Edition
4. System Programming and Operating Systems by D M Dhamdhere, TMH
5. Systems Programming by John J. Donovan

6. Charles N. Fischer, Richard J. leBlanc, Jr.- Crafting a Compiler with C, Pearson Education, 2008.

Articles:

1. Joshi A K, “Natural Languagae Processing“, Journal of Science, 253(5025):1242-1249, 1991
2. Gobinda G. Chowdhury, “Natural Languagae Processing“, Annual review of Computer Scienece and Technology, 37(1): 51-89, 2003

3. http://nptel.iitm.ac.in/courses.php?disciplineId=106

   http://symbolaris.com/course/Compilers/waitegoos.pdf

LIST OF PRACTICALS:

Sr. No	Name of Experiment
1	Implement a C program to identify keywords and identifiers using finite automata.
2	Implement a C program to find FIRST and FOLLOW set of given grammar.
3	Implement a C program to eliminate Left Recursion from given grammar.
4	Implement a C program to perform Left factoring in given grammar.
5	Implement any one shift reduce parser.
	Lex Programs
1	Write a lex program to identify numbers, words and other characters and generate tokens for each.
2	Write a lex program to count the number of characters, words and lines in the given input.
3	Write a lex program to remove empty lines.
4	Write a lex program to display the comments from given input file.
5	Write a lex program to identify all the lexemes from input file that follow the given RE. Provide the RE as command line argument.
6	Generate a lexer for C program.
	Yacc programs
1	Write a Yacc program for desktop calculator with ambiguous grammar.
2	Write a Yacc program for calculator with unambiguous grammar.
3	Write a Yacc program to convert infix into postfix expression.