Department of Computer Science

COURSES OFFERED BY DEPARTMENT OF COMPUTER SCIENCE

(Provide the details of the Discipline Specific Courses offered by your department for the UG Programme with your discipline as the Single Core Discipline)
[UG Programme for Bachelor in Computer Science (Honours) degree]

DISCIPLINE SPECIFIC CORE COURSE -19 (DSC-19) : Compiler Design

CREDIT DISTRIBUTION, ELIGIBILITY AND PRE-REQUISITES OF THE COURSE

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSC19 Compiler Design	**4**	**3**	**0**	**1**	Pass in Class XII	One course in any Programming Language

Learning Objectives

The basic objective of the compiler design course is to understand the basic principles of compiler design, its various constituent parts, algorithms, and data structures required to be used in the compiler. It also aims to understand the use of basic compiler-building tools.

Learning Outcomes

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

1. Describe the concepts and different phases of compilation.
2. Represent language tokens using regular expressions and context free grammars.
3. Describe the working of lexical analyzers.
4. Understand the working of different types of parsers and parse a particular string.
5. Describe intermediate code representations using syntax trees and DAG's as well as use this knowledge to generate intermediate code in the form of three address code representations.

6. Apply optimization techniques to intermediate code and generate machine code for high level language program.
7. Use Lex and Yacc automated compiler generation tools.

Syllabus

Unit 1 Introduction: Overview of compilation, Phases of a compiler.

Unit 2 Lexical Analysis: Role of a Lexical analyzer, Specification and recognition of tokens, Symbol table, Error reporting, Regular expressions and definitions , Lexical Analyzer Generator-Lex.

Unit 3 Syntax Analysis: CFGs, left recursion, left factoring, Top-down parsing- LL parser, Bottom-up parsing- LR parser, Parser Generator-yacc.

Unit 4 Intermediate representations: Syntax Directed Definitions, Evaluation Orders for Syntax Directed Definitions, Intermediate Languages: Syntax Tree, Three Address Code, Types and Declarations, Translation of Expressions, loops and conditional statements, Type Checking.

Unit 5 Storage organization & Code generation: Activation records, stack allocation, Issues in Code Generation – Design of a simple Code Generator.

Unit 6 Code optimization : Principal sources of optimization, Peephole optimization.

References

1. Aho, A., Lam, M., Sethi, R., & Ullman, J. D. *Compilers: Principles, Techniques, and Tools*, 2nd edition, Addison Wesley, 2006.

Additional References

(i) V Raghvan, *Principles of Compiler Design*, TMH, 2010.

(ii) Santanu Chattopadhayay, *Compiler Design*, PHI, 2005.

Suggested Practical List

1. Write a Lex program to count the number of lines and characters in the input file.
2. Write a Lex program to count the number of vowels and consonants in a given string
3. Write a Lex program that implements the Caesar cipher: it replaces every letter with the one three letters after in alphabetical order, wrapping around at Z. e.g. a is replaced by d, b by e, and so on z by c.
4. Write a Lex program that finds the longest word (defined as a contiguous string of upper and lower case letters) in the input.
5. Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.
6. Write a Lex program to count the number of words, characters, blank spaces and lines in a C file.
7. Write a Lex specification program that generates a C program which takes a string "abcd" and prints the following output

   abcd
   abc
   a
8. Write a Lex program to recognize a valid arithmetic expression.

9. Write a YACC program to find the validity of a given expression (for operators + - * and /)A program in YACC which recognizes a valid variable which starts with a letter followed by a digit. The letter should be in lowercase only.
10. Write a program in YACC to evaluate an expression (simple calculator program for addition and subtraction, multiplication, division).
11. Write a program in YACC to recognize the string „abbb", „ab" „a" of the language (an b n , n>=1).
12. Write a program in YACC to recognize the language (an b , n>=10). (output to say input is valid or not)

Additional Suggestive list of Practicals (Can be implemented in C++/Python)

1. Write a program to implement DFAs that recognize identifiers, constants, and operators of the mini language.
2. Write a program Design a Lexical analyzer for the above language. The lexical analyzer should ignore redundant spaces, tabs and newlines. It should also ignore comments. Identifiers may be of restricted length.
3. Write a program to check the types of expressions in a language.
4. Write a translator to translate a 3-address code into assembly code.

COMMON POOL OF DISCIPLINE ELECTIVE COURSES (DSE) COURSES

Computer Science Courses for all Undergraduate Programmes of study with Computer Science as Discipline Elective

DISCIPLINE SPECIFIC ELECTIVE COURSE: Data Analysis and Visualization

Credit distribution, Eligibility and Pre-requisites of the Course

Course title & Code	Credits	Credit distribution of the course			Eligibility criteria	Pre-requisite of the course (if any)
		Lecture	Tutorial	Practical/ Practice		
DSE7a: Digital Image Processing	**4**	**3**	**0**	**1**	Pass in Class XII	One course in any Programming Language

Course Objective

This course introduces students to the fundamentals of digital image processing, It introduces image processing in the Spatial anrequency domains including techniques for various image