SATYUG DARSHAN INSTITUTE OF ENGINEERING & TECHNOLOGY

Department of Computer Science & Engineering

Subject: Compiler Design

Subject Outline

Semester: 6th B. Tech 3rd Year

Session: 2020-2021

SATYUG DARSHAN INSTITUTE OF ENGINEERING & TECHNOLOGY

COURSE OUTLINE B.TECH-CSE 3rd YEAR 6th SEMESTER (2021)

Compiler Design (PEC-CS-605)

Introduction

The course is intended to teach the students the basic techniques that underlie the practice of Compiler Construction. The course will introduce the theory and tools that can be standard employed in order to perform syntax-directed translation of a high-level programming language into an executable code.

These techniques can also be employed in wider areas of application, whenever we need a syntax-directed analysis of symbolic expressions and languages and their translation into a lower-level description. They have multiple applications for man-machine interaction, including
verification and program analysis.

In addition to the exposition of techniques for compilation, the course will also discuss various aspects of the run-time environment into which the high-level code is translated. This will provide deeper insights into the more advanced semantics aspects of programming languages, such as recursion, dynamic memory allocation, types and their inferences, object orientation, concurrency and multi-threading.

Syllabus

MODULE 1:

Introduction: Phases of compilation and overview.

Lexical Analysis (scanner): Regular languages, finite automata, regular expressions, from regular expressions to finite automata, scanner generator (lex, flex).

MODULE 2:

Syntax Analysis (Parser): Context-free languages and grammars, push-down automata, LL(1) grammars and top-down parsing, operator grammars, LR(O), SLR(1), LR(1), LALR(1) grammars and bottom up parsing, ambiguity and LR parsing, LALR(1) parser generator (yacc, bison)

MODULE 3:

Semantic Analysis: Attribute grammars, syntax directed definition, evaluation and flow of attribute in a syntax tree.

Symbol Table: Symbol table and its structure, symbol attributes and management.

MODULE 4:

Run-time environment: Procedure activation, parameter passing, value return, memory allocation, and scope.

Intermediate Code Generation: Translation of different language features, different types of intermediate forms.

MODULE 5:

Code Improvement (optimization): Analysis: control-flow, data-flow dependence etc.; Code improvement local optimization, global optimization, loop optimization, peep-hole optimization etc. **Architecture dependent code improvement:** instruction scheduling (for pipeline), loop optimization (for cache memory) etc. Register allocation and target code generation.

MODULE 6:

Advanced topics:

Type systems, data abstraction, compilation of Object Oriented features and non-imperative programming languages.

Faculty: Mr. Ashok Madaan

Course Objective

Objective: Students successfully completing this course will be able to:

1. To learn the process of translating a modern high-level language to executable code.
2. To provide a student with an understanding of the fundamental principles in compiler design and to provide the skills needed for building compilers for various situations.
3. To develop an awareness of the function and complexity of modern compilers.
4. To apply the code generation algorithms to get the machine code for the optimized code.
5. To represent the target code in any one of the code formats
6. To understand the machine dependent code
7. To draw the flow graph for the intermediate codes.
8. To apply the optimization techniques to have a better code for code generation

Books /Study material

"Compilers Principles, Techniques and Tools, Second Edition, By Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman., Pearson.

Further readings:

All the additional readings are available under their respective sections.

1. **Principles of Compiler Design, By Narosa Publication**
2. **"Compiler Design, K. Muneeswaran., Oxford University Press, 2012.**
3. **"Compiler Construction, K.V.N Sunitha, Pearson, 2013**
4. **"Compiler Design, Sandeep Saxena, Rajkumar Singh Rathore., S.Chand publications.**
5. **"Engineering a compiler", K.D. Cooper and L. Torczon, Elsevier,2004.**
6. **"Elements compiler Design", Dr. M. Joseph, University Science Press.**

.

List of Programs

Certain set of experiments based on case studies, as per university syllabus shall be conducted.

Objective:

1. Write a program to design a DFA in which all the strings contains '01'
2. Write a program to design a DFA in which Strings end with 001.
3. Write a program to check whether entered string is identifier or constant.
4. Write a program to Implement Top down Parsing.
5. Write a program to Implement Left Recursion.
6. Write a program to Implement Operator grammar.
7. Write a program to Implement Left Factoring
8. Write a program to calculate leading of input string.
9. Write a program to calculate Trailing of input string.
10. Write a program to calculate First of Input Production.