Assignment No : B6

Roll no : 4351

1 Title

Abstract Syntax Tree

2 Problem Statement

Write a LEX and YACC program to generate abstract syntax tree for the given expression

3 Learning Objectives

1. To learn the concepts of Compiler Design
2. To study various code generation techniques
3. To understand the code generation phase of the compiler

4 Learning Outcome

1. Compiler code generation techniques were successfully learnt.
2. Implementation of abstract syntax tree in Lex and Yacc
3. Various Code generation techniques were learnt

5 Theory

Abstract Syntax Tree

Abstract syntax trees are data structures widely used in compilers, due to their property of representing the structure of program code. An AST is usually the result of the syntax analysis phase of a compiler. It often serves as an intermediate representation of the program through several stages that the compiler requires, and has a strong impact on the final output of the compiler.
The AST is used intensively during semantic analysis, where the compiler checks for correct usage of the elements of the program and the language. The compiler also generates symbol tables based on the AST during semantic analysis. A complete traversal of the tree allows verification of the correctness of the program.
After verifying correctness, the AST serves as the base for code generation. The AST is often used to generate the 'intermediate representation' '(IR)', sometimes called an intermediate language, for the code generation.

Core elements of AST Design

- Variable types must be preserved, as well as the location of each declaration in source code.
- The order of executable statements must be explicitly represented and well defined.
- Left and right components of binary operations must be stored and correctly identified.
- Identifiers and their assigned values must be stored for assignment statements.

Code Generation

Code generation is the process by which a compiler's code generator converts some intermediate representation of source code into a form (e.g., machine code) that can be readily executed by a machine.
The input to the code generator typically consists of a parse

tree or an abstract syntax tree. The tree is converted into a linear sequence of instructions, usually in an intermediate language such as three-address code.

6 Mathematical Model

Let S be the solution perspective of the system such that
S={ s, e , X, Y, Fme , DD, NDD ,Ffriend | ϕs }
where
DD = Deterministic Data
NDD = Non - Deterministic Data = User Input

ϕs = Constraints on System S

s is start state

e is end state=Syntax Tree is generated

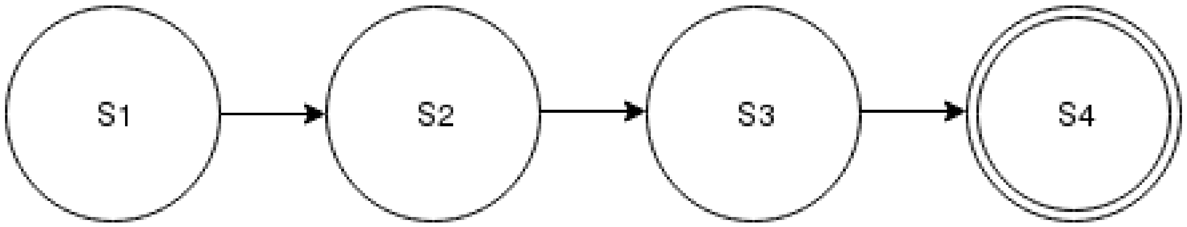
X=set of input parameters
X = X1
where, X1=Expression whose syntax tree needs to be generated

Y=set of output parameters
Y = Y1
where Y1 = Abstract Syntax Tree
Fme is the set of main functions
Fme = f1, f2
f1 = Parse the input
f2 = Generate Syntax Tree

Success Case = Right Solution

Failure Case = Wrong solution

7 State Diagram



s1 is the start state
s2 is the input state
s3 is the syntax tree generating state
s4 is the final state

8 Conclusion

We successfully implemented Abstract Syntax Tree