Name : Prajakta Keer
Roll No : 33231

ASSIGNMENT 6

Aim : Intermediate code generation using LEX and YACC for control flow and switch case statements

Objectives : • To understand LEX and YACC language
• To understand intermediate code generation process
• To apply LEX and YACC for intermediate code generation

Theory: **LEX** – Several tools have been built for constructing lexical analyzers from special purpose notations based on regular expression • Several algorithms exist for compiling regular expressions into pattern matching algorithms • Lex is tool which uses such algorithm.

Lex Specification – Lex program contains 3 parts

declarations
%% %%
transition rules
%% %%
Auxiliary procedures

- **Declarations** are surrounded by special bracket %{ and %} • Anything appearing between these brackets is copied directly into lex.yy.c and isn't treated as a part of regular definitions or translation rules.

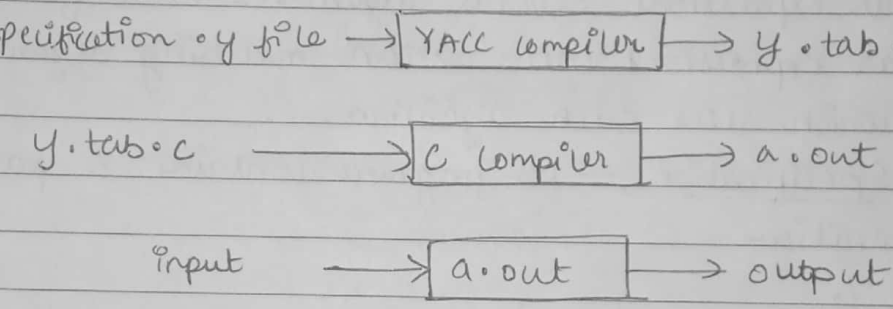
- **Regular Definitions** – Each such definition consists of a name and a regular expression denoted by that name.

- **Transition Rules** – Structure of LA is such that it keeps trying to recognize tokens until action associated with one found causes a return.

yylval is a variable whose definition appears in Lex

output lex.yy.c and which is also available to parser. Purpose of yylval is to hold the lexical value returned.

* Parser Generators : YACC – It is yet another compiler. It can be used to facilitate construction of frontend of compiler. It makes use of shift reduce parser. Ambiguity in grammar can be resolved by specifying the operators and their associativity either left or right.



* Communication between LEX and YACC – yylex() produces pairs containing token and its associated value. If token number is return token NUMBER must be declared in first section of YACC. The attribute value associated with a token is communicated to the parser through YACC defined variable yylval.

* Intermediate code Generation – It uses the structure produced by syntax analyzer to create a stream of simple instructions. many styles of intermediate code are possible

* Intermediate Representations – • span the gap between source and target programs.
  - High level representations – closer to source language
  - Low level representations – closer to target machine

Conclusion : I learnt about generation of Intermediate code using LEX and YACC for control statement.