D.Y. Patil College of Engineering,

Akurdi, Pune – 44Department of Information Technology

NAME: Aditya Somani

PRN: 71901204L ROLL: T1851061

SL-5(Group'A')

Assignment No.7

Aim- Write a program for Intermediate code generation using LEX &YACC for Control Flow statement.

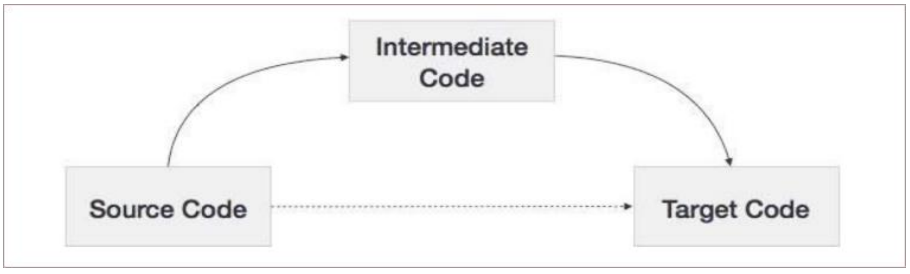
Objective:

1. To study the concept of LEX and YACC.

2. To implement Intermediate code generation using LEX and YACC.

Theory-

• If a compiler translates the source language to its target machine language without having the option for generating intermediate code, then for each new machine, a full native compiler is required.



• Intermediate code eliminates the need of a new full compiler for every unique machine by keeping the analysis portion same for all the compilers.

• The second part of compiler, synthesis, is changed according to the target machine.

• It becomes easier to apply the source code modifications to improve code performance by applying code optimization techniques on the intermediate code.

Intermediate Representation

Intermediate codes can be represented in a variety of ways and they have their own benefits.

• High Level IR - High-level intermediate code representation is very close to the source language itself. They can be easily generated from the source code and we can easily apply code modifications to enhance performance. But for target machine optimization, it is less preferred.

• Low Level IR - This one is close to the target machine, which makes it suitable for register

and memory allocation, instruction set selection, etc. It is good for machine- dependent optimizations.

Intermediate code can be either language specific (e.g., Byte Code for Java) or language

independent (three-address code).

Three-Address Code

Intermediate code generator receives input from its predecessor phase, semantic analyzer, in the form of an annotated syntax tree. That syntax tree then can be converted into a linear representation, e.g., postfix notation. Intermediate code tends to be machine independent code. Therefore, code generator assumes to have unlimited number of memory storage (register) to generate code.

For example:

The intermediate code generator will try to divide this expression into sub-expressions and then generate the corresponding code.

r being used as registers in the target program.

A three-address code has at most three address locations to calculate the expression.

Conclusion-

Thus we have implemented Intermediate code generation using LEX and YACC.

a = b + c * d;

r1 = c * d;

r2 = b + r1;

a = r2

OUTPUT:-