# ASSIGNMENT - 12

TITLE: LEX AND YACC TO GENERATE INTERMEDIATE CODE.

PROBLEM STATEMENT: WAP for intermediate code generation using YACC & LEX for control-flow statement (while or switch case)

- THEORY:
- INTERMEDIATE LANGUAGES:

3 ways of intermediate representation:
- syntax tree
- postfix notation
- 3 address code.

Steps to execute program:
1. $ lex. filename.l
2. $ yacc_d filename.y
3. $ cc lex.yy.c y.tab.c -ll -ly -lm
4. $ ./a.out
5. (eg: comp.l)
6. (eg: comp.y)

- ALGORITHM:

Write a LEX & YACC program to generate IL for arithmetic expression LEX program.

i. Declaration of header files, specially y.tab.h which contains declaration for letter, digit, expr
ii. End declaration with '%.%'
iii. Match regular expression
iv. If match found, then convert it into char store it into char & ~~.~~ yylval.p where p is

Scanned with CamScanner

the pointer declared in YACC-

v. Return token

vi. If input contains new line character ("\n") then return 0

vii. If input contains "." then return yytext[0]

viii. End rule section with "%.%"

ix. Declare main function

x. Open file given at command line

xi. If any error occurs, then print error & exit

xii. Assign file pointer fp to yyin

xiii. call function yylex until file ends

xiv. End

YACC program

i. Declaration of header files

ii. Declare structure for 3 address code representation having fields of argument 1, argument 2, operator, result

iii. Declare pointer of char type in union

iv. Declare token expr. of type pointer p

v. Give precedence to *, /

vi. Give precedence to +, -

vii. End declaration section with %%

viii. If final expr. evaluates, then add it to the table of 3 address code

ix. If input type is of the form

x. exp "+" exp, then add to the table argument 1, argument 2, operator

xi exp "-" exp, then add to table arg 1, arg 2, operator

xii exp "*" exp, then add to table arg1, arg 2, operator

xiii exp "/" exp, then add to table arg1, arg2, operator

Scanned with CamScanner

xiv. "(" exp ")" then assign $2 to $$
xv. digit or letter then assign $1 to $$
xvi. End section with %%
xvii. Declare file *yyin externally
xviii. Declare main function & call yyparse function until yyin ends.
xix. Declare yyerror if any error occurs
xx. Declare char pointer to print error
xxi. Print error message
xxii. End of program

- CONCLUSION :
  Thus, we have successfully implemented program for IC generation using LEX & YACC for control flow.

Scanned with CamScanner