**LPCC ASSIGNMENT-5**

****

**SUBMITTED TO:**

**PROF. PALLAVI REGE**

**VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

**COMPUTER ENGINEERING DEPARTMENT**

**BY:**

**NAME: ABHISHEK MORE**

**G.R No.: 21810033**

**ROLL NO.: 323036**

**CLASS: T.Y COMP**

**BATCH: COMP C2**

**ASSIGNMENT-5**

**AIM:**

Write a program to generate three address code for the simple expression.

**THEORY:**

Three address code is a type of intermediate code which is easy to generate and can be easily converted to machine code.It makes use of at most three addresses and one operator to represent an expression and the value computed at each instruction is stored in temporary variable generated by compiler. The compiler decides the order of operation given by three address code.

General representation : a = b op c

Where a, b or c represents operands like names, constants or compiler generated temporaries and op represents the operator

e.g. e= g + h

1. T1 = g + h

2. e = T1

**SOURCE CODE:**

package lpcc.assignment5;

import java.util.\*;

public class Assgn5 {

static Scanner sc = new Scanner(System.in);

public static void main(String[] args) {

Map<String,Integer> operator = new HashMap<>();

Stack<String> operatorStack = new Stack<>();

Stack<String> varStack = new Stack<>();

Stack<String> exStack = new Stack<>();

operator.put("=",0);

operator.put("+",1);

operator.put("-",1);

operator.put("\*",2);

operator.put("/",2);

operator.put("%",2);

operator.put(")",3);

operator.put("(",3);

String[] t\_list = {"T1","T2","T3","T4","T5","T6"};

List<String> tList = Arrays.asList(t\_list);

String exp = sc.nextLine();

String ele,ch;

String[] expArr = exp.split("[ ]\*");

for(int i=0;i<expArr.length;i++){

ch = expArr[i];

if(!ch.equals(" ")){

if(operator.containsKey(ch)){

if(ch.equals("(")){

operatorStack.push(ch);

}

else if(ch.equals(")")){

ele = operatorStack.pop();

while(!ele.equals("(")){

varStack.push(ele);

ele = operatorStack.pop();

}

}

else if(operatorStack.size()>0 && !operatorStack.elementAt(operatorStack.size() - 1).equals("(")){

ele = operatorStack.pop();

if(operator.get(ch)<=operator.get(ele)){

operatorStack.push(ch);

varStack.push(ele);

}

else{

operatorStack.push(ele);

operatorStack.push(ch);

}

}

else{

operatorStack.push(ch);

}

}

else{

if(ch.equals("=")){

}

varStack.push(ch);

}

}

}

for(int i=0;i<operatorStack.size();i++){

ele = operatorStack.pop();

varStack.push(ele);

}

int t =0;

String op1,op2;

for(int i=0;i<varStack.size();i++){

ch = varStack.elementAt(i);

if (operator.containsKey(ch)) {

op1 = exStack.pop();

op2 = exStack.pop();

if(ch.equals("=")){

System.out.println(op2 +" "+ch+" "+op1);

continue;

}

exStack.push(tList.get(t));

System.out.println(tList.get(t)+"="+" "+op2+" "+ch+" "+op1);

t+=1;

}

else

exStack.push(ch+"");

}

System.out.println(exStack.elementAt(0)+" = "+exStack.elementAt(3) );

}

}

**OUTPUT:**

Enter String : B = A-D

T1= A - D

B = T1