18CSC304J Compiler Design Lab

Exercise 11:

Intermediate code generation – Quadruple, Triple, Indirect triple

Submitted To:-

Dr.M.Kanchana

Submitted By:-

Name:- Puneet Sharma

Reg.No. :- RA1911003010331

CODE:-

```
OPERATORS = set(['+', '-', '*', '/', '(', ')'])
PRI = {'+':1, '-':1, '*':2, '/':2}

### INFIX ===> POSTFIX ###
def infix_to_postfix(formula):
    stack = [] # only pop when the coming op has priority
    output = ''
    for ch in formula:
        if ch not in OPERATORS:
            output += ch
        elif ch == '(':
            stack.append('(')
        elif ch == ')':
            while stack and stack[-1] != '(':
                output += stack.pop()
            stack.pop() # pop '('
        else:
            while stack and stack[-1] != '(' and PRI[ch]
<= PRI[stack[-1]]:
                output += stack.pop()
            stack.append(ch)
    # leftover
    while stack:
     output += stack.pop()
    print(f'POSTFIX: {output}')
```

```
    return output


### INFIX ===> PREFIX ###
def infix_to_prefix(formula):
    op_stack = []
    exp_stack = []
    for ch in formula:
        if not ch in OPERATORS:
            exp_stack.append(ch)
        elif ch == '(':
            op_stack.append(ch)
        elif ch == ')':
            while op_stack[-1] != '(':
                op = op_stack.pop()
                a = exp_stack.pop()
                b = exp_stack.pop()
                exp_stack.append( op+b+a )
            op_stack.pop() # pop '('
        else:
            while op_stack and op_stack[-1] != '(' and
PRI[ch] <= PRI[op_stack[-1]]:
                op = op_stack.pop()
                a = exp_stack.pop()
                b = exp_stack.pop()
                exp_stack.append( op+b+a )
            op_stack.append(ch)
```

```
    # leftover
    while op_stack:
        op = op_stack.pop()
        a = exp_stack.pop()
        b = exp_stack.pop()
        exp_stack.append( op+b+a )
    print(f'PREFIX: {exp_stack[-1]}')
    return exp_stack[-1]

### THREE ADDRESS CODE GENERATION ###
def generate3AC(pos):
     print("### THREE ADDRESS CODE GENERATION ###")
     exp_stack = []
     t = 1

     for i in pos:
          if i not in OPERATORS:
               exp_stack.append(i)
          else:
               print(f't{t} := {exp_stack[-2]} {i}
{exp_stack[-1]}')
               exp_stack=exp_stack[:-2]
               exp_stack.append(f't{t}')
               t+=1

expres = input("INPUT THE EXPRESSION: ")
pre = infix_to_prefix(expres)
```

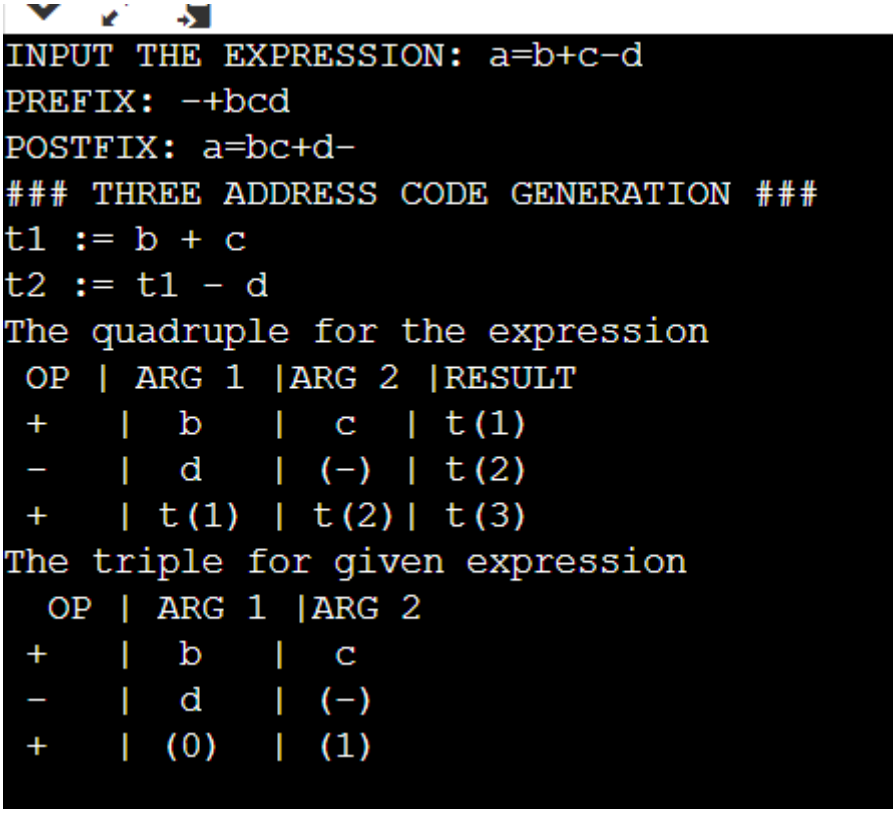
```
pos = infix_to_postfix(expres)
generate3AC(pos)
def Quadruple(pos):
  stack = []
  op = []
  x = 1
  for i in pos:
    if i not in OPERATORS:
       stack.append(i)
    elif i == '-':
        op1 = stack.pop()
        stack.append("t(%s)" %x)
        print("{0:^4s} | {1:^4s} |
{2:^4s}|{3:4s}".format(i,op1,"(-)"," t(%s)" %x))
        x = x+1
        if stack != []:
          op2 = stack.pop()
          op1 = stack.pop()
          print("{0:^4s} | {1:^4s} |
{2:^4s}|{3:4s}".format("+",op1,op2," t(%s)" %x))
          stack.append("t(%s)" %x)
          x = x+1
    elif i == '=':
      op2 = stack.pop()
      op1 = stack.pop()
      print("{0:^4s} | {1:^4s} |
{2:^4s}|{3:4s}".format(i,op2,"(-)",op1))
    else:
```

```
      op1 = stack.pop()
      op2 = stack.pop()
      print("{0:^4s} | {1:^4s} |
{2:^4s}|{3:4s}".format(i,op2,op1," t(%s)" %x))
      stack.append("t(%s)" %x)
      x = x+1
print("The quadruple for the expression ")
print(" OP | ARG 1 |ARG 2 |RESULT  ")
Quadruple(pos)

def Triple(pos):
        stack = []
        op = []
        x = 0
        for i in pos:
          if i not in OPERATORS:
            stack.append(i)
          elif i == '-':
            op1 = stack.pop()
            stack.append("(%s)" %x)
            print("{0:^4s} | {1:^4s} |
{2:^4s}".format(i,op1,"(-)"))
            x = x+1
            if stack != []:
              op2 = stack.pop()
              op1 = stack.pop()
              print("{0:^4s} | {1:^4s} |
{2:^4s}".format("+",op1,op2))
```

```
                stack.append("(%s)" %x)
                x = x+1
            elif i == '=':
              op2 = stack.pop()
              op1 = stack.pop()
              print("{0:^4s} | {1:^4s} |
{2:^4s}".format(i,op1,op2))
            else:
              op1 = stack.pop()
              if stack != []:
                op2 = stack.pop()
                print("{0:^4s} | {1:^4s} |
{2:^4s}".format(i,op2,op1))
                stack.append("(%s)" %x)
                x = x+1
print("The triple for given expression")
print("  OP | ARG 1 |ARG 2  ")
Triple(pos)
```

OUTPUT:-


INPUT THE EXPRESSION: a=b+c-d
PREFIX: -+bcd
POSTFIX: a=bc+d-
THREE ADDRESS CODE GENERATION
t1 := b + c
t2 := t1 - d
The quadruple for the expression
OP | ARG 1 |ARG 2 |RESULT
+ | b | c | t(1)
- | d | (-) | t(2)
+ | t(1) | t(2)| t(3)
The triple for given expression
OP | ARG 1 |ARG 2
+ | b | c
- | d | (-)
+ | (0) | (1)