



University of Asia Pacific

Department of Computer Science & Engineering

Course Title: Compiler Design Lab

Course Code: CSE 430

Lab Exercise-05 Solution

Submitted by:-

Abid Al Mohtasin

20101089

B1

Description of the exercise:

Write a program to generate three address code from a given expression. Three-address code (often abbreviated to TAC or 3AC) is an intermediate code used by optimizing compilers to aid in the implementation of code-improving transformations. Three address code 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 a temporary variable generated by the compiler. The compiler decides the order of operation given by three address code.

Sample Inputs:

$$-(a * b) + (c + d) - (a + b + c + d)$$

Sample Outputs:

$$T1 = a \times b$$

$$T2 = \text{uminus } T1$$

$$T3 = c + d$$

$$T4 = T2 + T3$$

$$T5 = a + b$$

$$T6 = T3 + T5$$

$$T7 = T4$$

Code:

```
Assignment5_TAC.py X input.txt
Assignment5_TAC.py > ...
1 import sys
2 from collections import defaultdict, deque
3
4 f = open("input.txt", "r")
5 exp = f.readline().rstrip().split(" ")
6
7 #operator
8
9 op = {'^':0, '*': 1, '/': 1, '%': 1, '+': 2, '-': 2}
10 stack = []
11 tac = deque() |
12
13 counter = 1
14
15 #bracket
16 for c in exp:
17     if c=="(":
18         temp = []
19         while stack and stack[-1] != "(":
20             temp.append(stack.pop())
21         if stack[-1]=='(':
22             stack.pop()
23         tac.append((counter, temp[::-1]))
24         stack.append("(" + str(tac[counter-1][0]))
25         counter+=1
26     else:
27         stack.append(c)
28 tac.append((counter, stack))
29
30 result = deque()
31
32 counter = 1
33 t = defaultdict(int)
34 while tac:
35     current = tac.popleft()
36     index = current[0]
37     current = current[1]
38
39     #update t
40     for i in range(len(current)):
```

Ln 11, Col 15

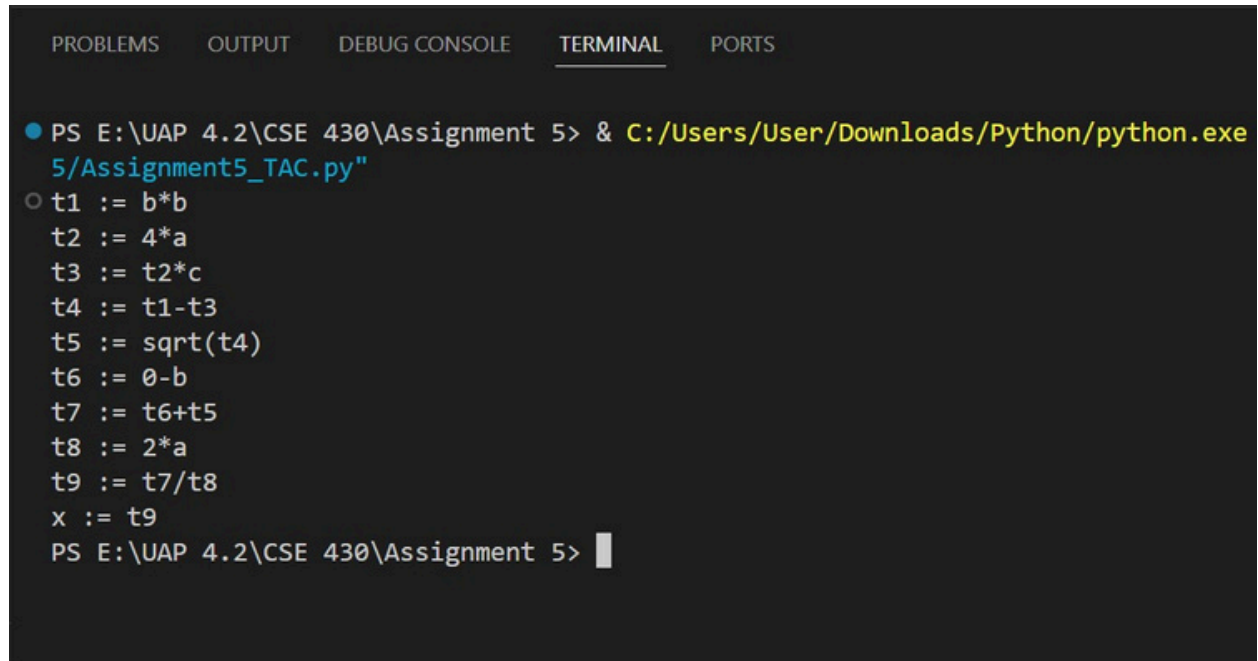
```
Assignment5_TAC.py X input.txt
Assignment5_TAC.py > ...
41     if current[i][0]=='t':
42         current[i] = 't'+str(t[int(current[i][1])])
43
44
45     while len(current)>2:
46         #sqrt
47         for i in range(len(current)):
48             if current[i]=='sqrt':
49                 result.append(("t"+str(counter),"sqrt("+current[i+1]+")"))
50                 current = current[:i]+["t"+str(counter)]+current[i+2:]
51                 counter+=1
52                 break
53
54         # ^
55         for i in range(len(current)):
56             if current[i]=='^':
57                 temp = "".join([current[i-1] for j in range(int(current[i+1]))])
58                 temp = deque(temp)
59                 while len(temp)>2:
60                     for j in range(len(temp)):
61                         if temp[j]=='*':
62                             result.append(("t"+str(counter), temp[j-1]+"*"+temp[j+1]))
63                             temp.popleft()
64                             temp.popleft()
65                             temp.popleft()
66                             temp.appendleft("t"+str(counter))
67                             counter+=1
68                             break
69                 current = [result[-1][0]]+current[i+2:]
70                 break
71
72         # *
73         flag = False
74         for i in range(len(current)):
75             if current[i]=='*' or current[i]=='/' or current[i]=='%':
76                 result.append(("t"+str(counter),current[i-1]+current[i]+current[i+1]))
77                 current = current[:i-1]+["t"+str(counter)]+current[i+2:]
78                 counter+=1
79                 flag = True
80                 break
```

```
Assignment5_TAC.py X input.txt
Assignment5_TAC.py > ...
81     if flag:
82         continue
83
84     # + -
85     for i in range(len(current)):
86         if current[i]=='+' or current[i]=='-':
87             if current[i]=='-' and i==0:
88                 result.append(("t"+str(counter)+'0'+current[i]+current[i+1]))
89                 current = ["t"+str(counter)]+current[i+2:]
90             else:
91                 result.append(("t"+str(counter),current[i-1]+current[i]+current[i+1]))
92                 current = current[:i-1]+["t"+str(counter)]+current[i+2:]
93             counter+=1
94             break
95
96     # =
97     for i in range(len(current)):
98         if current[i]=='=':
99             result.append((current[i-1],current[i+1]))
100             current = current[:i-1]+["t"+str(counter)]+current[i+2:]
101             counter+=1
102             break
103
104     t[index] = len(result)
105
106 for item in result:
107     print(f"{item[0]} := {item[1]}")
```

Sample Input:

```
Assignment5_TAC.py input.txt
input.txt
1  x = ( - b + sqrt ( b ^ 2 - 4 * a * c ) ) / ( 2 * a )
```

Observed Output:



The image shows a screenshot of a terminal window with a dark background. At the top, there are five tabs: "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL" (which is selected and underlined), and "PORTS". The terminal content shows a command prompt "PS E:\UAP 4.2\CSE 430\Assignment 5>" followed by a command to run a Python script: "& C:/Users/User/Downloads/Python/python.exe 5/Assignment5_TAC.py". Below this, a list of Python code statements is displayed, each preceded by a small circle icon. The statements are: "t1 := b*b", "t2 := 4*a", "t3 := t2*c", "t4 := t1-t3", "t5 := sqrt(t4)", "t6 := 0-b", "t7 := t6+t5", "t8 := 2*a", "t9 := t7/t8", and "x := t9". The terminal ends with the prompt "PS E:\UAP 4.2\CSE 430\Assignment 5>" and a cursor.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS E:\UAP 4.2\CSE 430\Assignment 5> & C:/Users/User/Downloads/Python/python.exe
5/Assignment5_TAC.py"
○ t1 := b*b
  t2 := 4*a
  t3 := t2*c
  t4 := t1-t3
  t5 := sqrt(t4)
  t6 := 0-b
  t7 := t6+t5
  t8 := 2*a
  t9 := t7/t8
  x := t9
PS E:\UAP 4.2\CSE 430\Assignment 5> █
```