Name: Saurabh Khandagale
Roll No: 46

Practical 5

Theory:

Aim: Write a program to generate three address code for while loop and Boolean expressions.

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.

Implementation of Three Address Code –
There are 3 representations of three address code namely
1. Quadruple
2. Triples
3. Indirect Triples

1. Quadruple –
It is structure with consist of 4 fields namely op, arg1, arg2 and result. op denotes the operator and arg1 and arg2 denotes the two operands and result is used to store the result of the expression.
Advantage –
- Easy to rearrange code for global optimization.
- One can quickly access value of temporary variables using symbol table.

Disadvantage –
- Contain lot of temporaries.
- Temporary variable creation increases time and space complexity.

2. Triples –
This representation doesn't make use of extra temporary variable to represent a single operation instead when a reference to another triple's value is needed, a pointer to that triple is used. So, it consist of only three fields namely op, arg1 and arg2.
Disadvantage –
- Temporaries are implicit and difficult to rearrange code.
- It is difficult to optimize because optimization involves moving intermediate code. When a triple is moved, any other triple referring to it must be updated also. With help of pointer one can directly access symbol table entry.

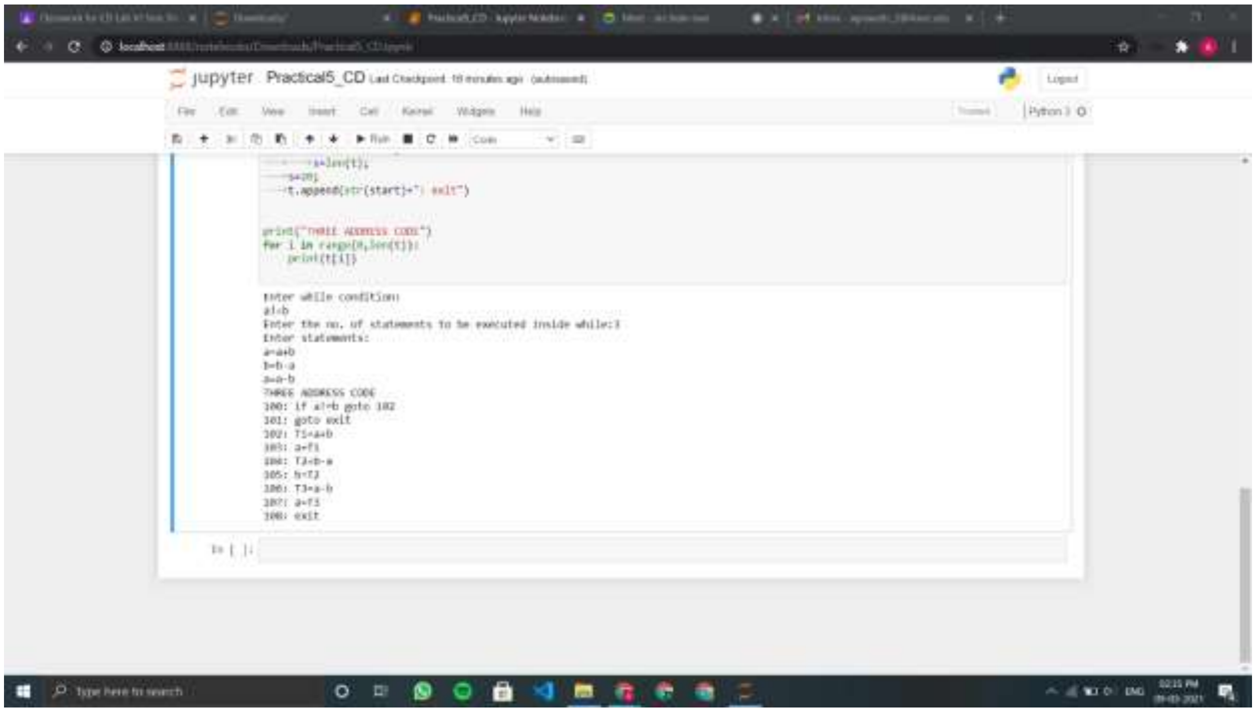
Code:

```
print("Enter while condition:")
wc=[]
wc.insert(0,input())
t=[]
n=int(input("Enter the no. of statements to be executed inside while:"))
s=0;
start=100;
addr=100;
arr=[]
m1=start;
t2=[];

if wc[0].find("and")!=-1:
   e=wc[0].split("and");
   exp=str(start)+": if "+str(e[0])+" goto "+str(start+2)
   t.append(exp)
   exp=str(start+1)+": goto exit"
   t.append(exp)
   start=start+2

   exp=str(start)+": if "+str(e[1])+" goto "+str(start+2)
   t.append(exp)
   exp=str(start+1)+": goto exit"
   t.append(exp)
   start=start+2

elif wc[0].find("or")!=-1:
   e=wc[0].split("or");
   exp=str(start)+": if "+str(e[0])+" goto "+str(start+4)
   t.append(exp)
   exp=str(start+1)+": goto "+str(start+2)
   t.append(exp)
   start=start+2

   exp=str(start)+": if "+str(e[1])+" goto "+str(start+2)
   t.append(exp)
   exp=str(start+1)+": goto "+" exit"
   t.append(exp)
   start=start+2

elif wc[0].find("not")!=-1:
   e=wc[0].split("not");
   t.append(str(start)+": if "+str(e[1])+" goto exit")
```

```
    t.append(str(start+1)+" goto "+str(start+2))
    start=start+2

else:
    exp=str(start)+": if "+wc[0]+" goto "+str(start+2)
    t.append(exp)
    exp=str(start+1)+": goto exit"
    t.append(exp)
    start=start+2

m2=start;
s=len(t)
while s<20:
    print("Enter statements: ");
    for i in range(1,n+1):
        wc.append(input())
        arr=wc[i].split("=")
        exp=str(start)+": T"+str(i)+"="+arr[1]
        t.append(exp);
        exp=str(start+1)+": "+str(arr[0])+"="+"T"+str(i)
        t.append(exp);
        start=start+2;
        s=len(t);
    s=20;
    t.append(str(start)+": exit")

print("THREE ADDRESS CODE")
for i in range(0,len(t)):
    print(t[i])
```

Output:

Case 1: No Boolean expression

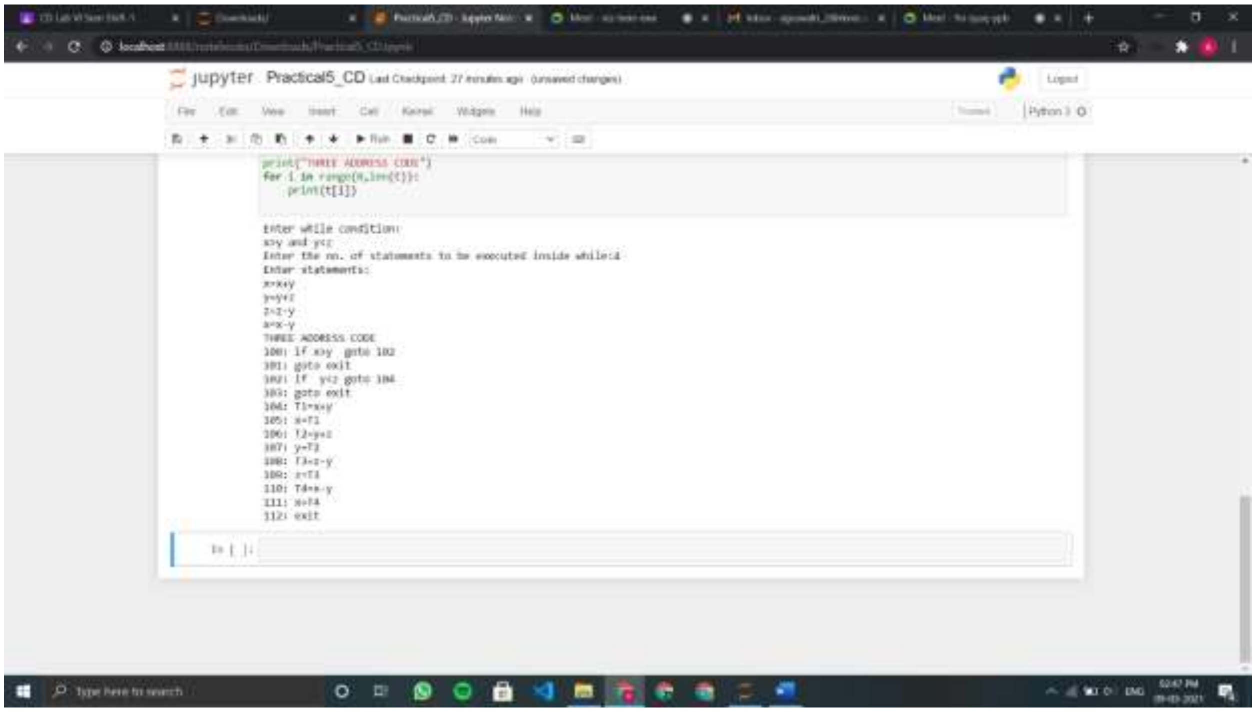


jupyter Practical5_CD Last Checkpoint: 18 minutes ago (autosaved)

```
            s=len(t);
        s=20;
    t.append(str(start)+": exit")


print("THREE ADDRESS CODE")
for i in range(0,len(t)):
    print(t[i])
```

```
Enter while condition:
a!=b
Enter the no. of statements to be executed inside while:3
Enter statements:
a=a+b
b=b-a
a=a-b
THREE ADDRESS CODE
100: if a!=b goto 102
101: goto exit
102: T1=a+b
103: a=T1
104: T2=b-a
105: b=T2
106: T3=a-b
107: a=T3
108: exit
```

Case 2: AND

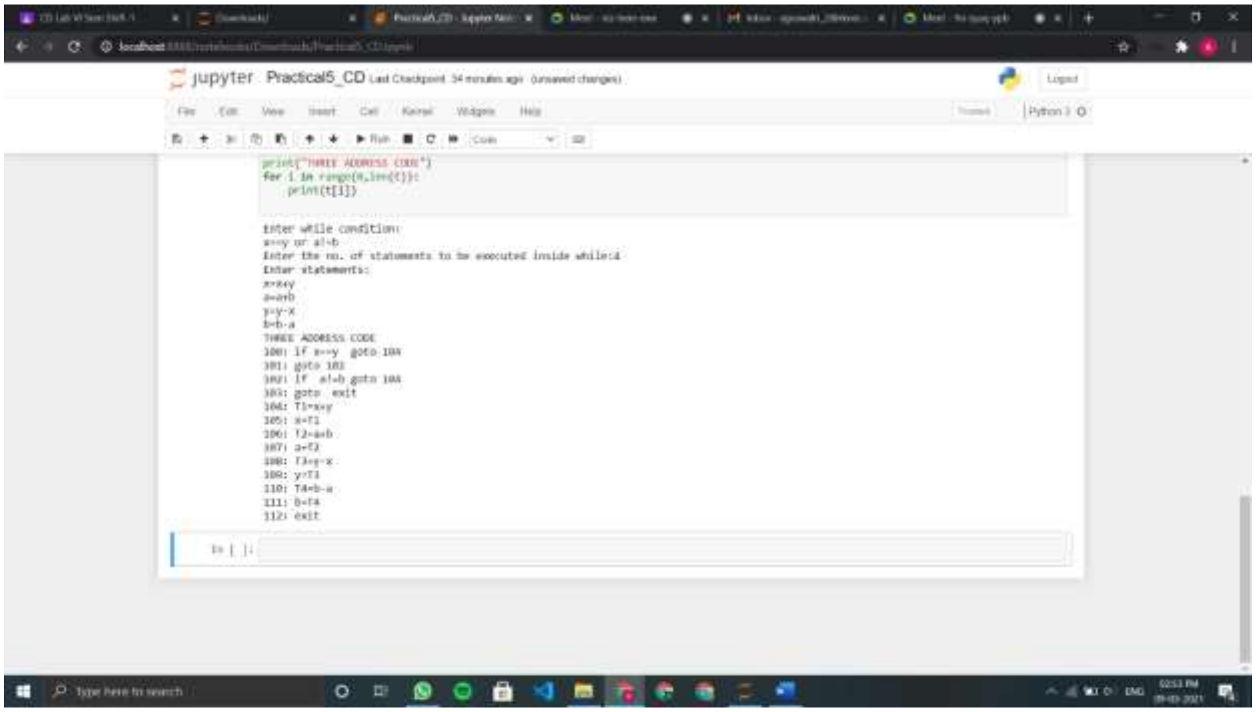


jupyter Practical5_CD Last Checkpoint: 27 minutes ago (unsaved changes)

```
print("THREE ADDRESS CODE")
for i in range(0,len(t)):
    print(t[i])
```

```
Enter while condition:
x>y and y<z
Enter the no. of statements to be executed inside while:4
Enter statements:
x=x+y
y=y+z
z=z-y
x=x-y
THREE ADDRESS CODE
100: if x>y  goto 102
101: goto exit
102: if  y<z goto 104
103: goto exit
104: T1=x+y
105: x=T1
106: T2=y+z
107: y=T2
108: T3=z-y
109: z=T3
110: T4=x-y
111: x=T4
112: exit
```

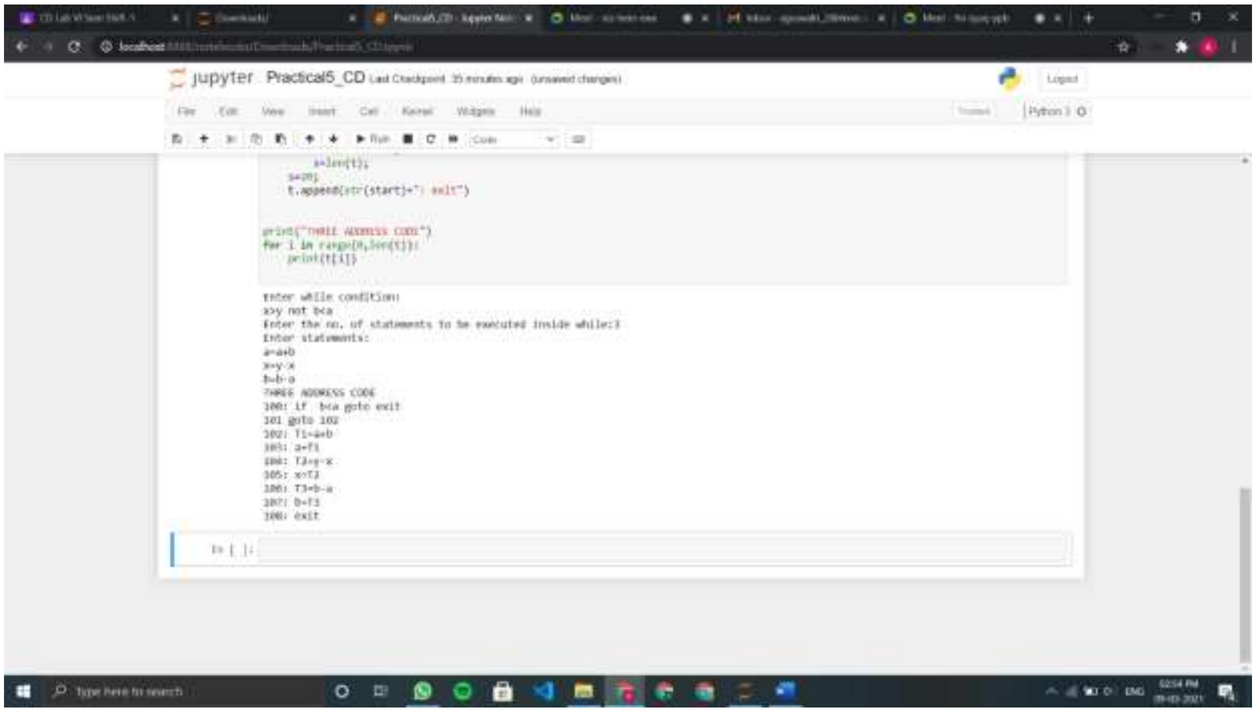
Case 3: OR


jupyter Practical5_CD Last Checkpoint 34 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Python 3
Type here to search

```
print("THREE ADDRESS CODE")
for i in range(0,len(t)):
    print(t[i])
```

```
Enter while condition:
x==y or a!=b
Enter the no. of statements to be executed inside while:4
Enter statements:
x=x+y
a=a+b
y=y-x
b=b-a
THREE ADDRESS CODE
100: if x==y  goto 104
101: goto 102
102: if  a!=b goto 104
103: goto  exit
104: T1=x+y
105: x=T1
106: T2=a+b
107: a=T2
108: T3=y-x
109: y=T3
110: T4=b-a
111: b=T4
112: exit
```

Case 4: NOT


jupyter Practical5_CD Last Checkpoint 35 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Python 3
Type here to search

```
        s=len(t);
    s=20;
    t.append(str(start)+": exit")


print("THREE ADDRESS CODE")
for i in range(0,len(t)):
    print(t[i])
```

```
Enter while condition:
x>y not b<a
Enter the no. of statements to be executed inside while:3
Enter statements:
a=a+b
x=y-x
b=b-a
THREE ADDRESS CODE
100: if  b<a goto exit
101 goto 102
102: T1=a+b
103: a=T1
104: T2=y-x
105: x=T2
106: T3=b-a
107: b=T3
108: exit
```