**Name: Atharva Paliwal**

**Roll No: 40**

**Practical No. 6**

**Topic**: Code Optimisation

**Platform**: Windows or Linux

**Language to be used**: Python or Java (based on the companies targeted for placement)

**CO Mapped**: CO4- Learn three address code generation and implement code

Optimization techniques for improving the performance of a program

segment.

**Aim**: Write a program to perform loop detection by finding leader, basic blocks and program flow graph &amp; natural loop.

=============================================================================

**Theory**

=============================================================================

Flow graph is a directed graph. It contains the flow of control information for the set of basic block.

A control flow graph is used to depict that how the program control is being parsed among the blocks. It is useful in the loop optimization.

In optimization, high-level general programming constructs are replaced by very efficient low-level programming codes. A code optimizing process must follow the three rules given below:

* The output code must not, in any way, change the meaning of the program.
* Optimization should increase the speed of the program and if possible, the program should demand less number of resources.
* Optimization should itself be fast and should not delay the overall compiling process.

Efforts for an optimized code can be made at various levels of compiling the process.

* At the beginning, users can change/rearrange the code or use better algorithms to write the code.
* After generating intermediate code, the compiler can modify the intermediate code by address calculations and improving loops.
* While producing the target machine code, the compiler can make use of memory hierarchy and CPU registers.

Optimization can be categorized broadly into two types : machine independent and machine dependent.

=============================================================================

**Program Code**

=============================================================================

tac = ['']

with open("C:\\Users\\ACER\\Desktop\\code.txt") as f:

    for line in f:

        tac.append(line)

leader = [1]

block = {}

for i in range(2,len(tac)):

    if 'goto' in tac[i]:

        j = int(tac[i].split('goto')[1].strip())

        if j not in leader:

            leader.append(j)

        if i+1 not in leader:

            leader.append(i+1)

leader.sort()

for i in range(len(leader)):

    try:

        block[i+1] = (leader[i],leader[i+1]-1)

    except IndexError:

        block[i+1] = (leader[i],leader[i])

#flow diagram

flow = []

for k in block:

    if 'if' in tac[block[k][0]]:

        i = int(tac[block[k][0]].split('goto')[1].strip())

        for B in block:

            if block[B][0]==i:

                b = B

        flow.append('B'+str(k)+' -> B' + str(b))

        flow.append('B'+str(k)+' -> B' + str(k+1))

    elif 'goto' in tac[block[k][0]]:

        i = int(tac[block[k][0]].split('goto')[1].strip())

        for B in block:

            if block[B][0]==i:

                b = B

        flow.append('B'+str(k)+' -> B' + str(b))

    elif k != len(block):

        flow.append('B'+str(k)+' -> B' + str(k+1))

print('Leaders:')

for l in leader:

    print(l)

print('\nBlocks:')

for b in block:

    print('B'+str(b)+' ->',block[b])

print('\nProgram flow graph:')

for f in flow:

    print(f)

=============================================================================

**Program Output**

=============================================================================

Leaders:

1

3

4

12

Blocks:

B1 -> (1, 2)

B2 -> (3, 3)

B3 -> (4, 11)

B4 -> (12, 12)

Program flow graph:

B1 -> B2

B2 -> B4

B2 -> B3

B3 -> B4

**\*\*\*END\*\*\***