Compiler Design

Static Single Assignment (SSA) :-

1) It is a 3-Address code

2) SSA form is used to represent an intermediate code.

* 3) A variable can not be used more than once in the LHS

* 4) Every variable has a Single Assignment (Definition)

Ex:

$a = b + c$

$d = a - b$

$a = b + d$

$d = a - c$

SSA

SSA

$a_1 = b + c$

$d_1 = a_1 - b$

$a_2 = b + d_1$

$d_2 = a_2 - c$

1, 2

Compiler Design

Question:- Find minimum No. of variables to represent 3-Address code & SSA

$X = a * b + a * b$

3 Address code

$a = a * b$

$a = a + a$

Optimized Best 3-Address code.

2 variables required in 3 Address (a, b)

SSA:

$a_1 = a * b$

$a_2 = a_1 + a_1$

minimum variable in SSA = 4 variable

(a_1, a_2, a_3, a_4) =

Compiler Design

Question:- Consider the following code segment

```
x = u - t
y = x * v
x = y + w
y = t - z
y = x * y
```

5 + 5 -> 10

The mini. No. of total variables required to convert the above code segment to Static single Assignment form is ?

Compiler Design

Question: 3 Address code:

$t_1 = a * a$

$t_2 = a * b$

$t_3 = t_1 * t_2$

$t_4 = t_3 + t_2$

$t_4 = t_3 + t_2 = t_1 * t_2 + a * b$

$= (a*a) * (a*b) + a * b$

$b = a * b$

$a = a * a$

$a = a * b$

$a = a + b$

no of variables $= 2 (a, \& b)$

i) How many minimum of variables in equivalent 3 Address code:

$\Rightarrow$ 2 Ans

ii) Mini. No of variable in equivalent SSA ?

$b_1 = a * b$

$a_1 = a * a$

$a_2 = a_1 * b_1$

$a_3 = a_2 + b_1$

$\Rightarrow$ 6 variables

Compiler Design

Question:-

$t_1 = b * c$

$t_2 = a + t_1$

$t_3 = b * c$

$t_4 = d * t_3$

$t_5 = t_2 + t_4$

Stmt → 1

$t_5 = t_2 + t_4$

$= (a + t_1) + (d * t_3)$

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

$b = b * c$

i) Mini variables in 3 Address Code ? → 4

ii) Mini variable in SSA Code ? SSA → 8

i) $b_1 = b * c$

2) $a_1 = a + b_1$

3) $b_2 = d * b_1$

4) $a_2 = a_1 + b_2$

8

Any

Compiler Design

Basic Block: Basic block is a set of statements that always executes in a sequence one after the other.

The characteristics of basic blocks are-

-They do not contain any kind of jump statements in them.
-There is no possibility of branching or getting halt in the middle.
-All the statements execute in the same order they appear.
-They do not lose the flow control of the program.

Ex: $a = b + c * d$

Basic Block

$t_1 = c * d$
$t_2 = b + t_1$
$a = t_2$

if A < B then 1 else 0

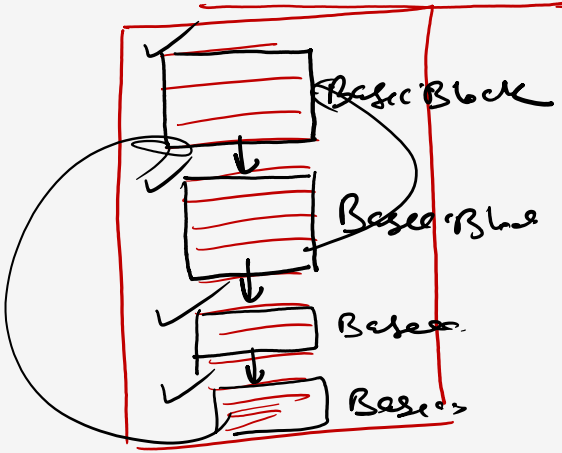
1) if A<B goto 4
2) $t_1 = 0$
3) goto 5
4) $t_1 = 1$
5) ——

Basic Block

Compiler Design

How to find the Basic Blocks

1. First, find the set of leaders from intermediate code, the first statements of basic blocks. The following are the steps for finding leaders:
    1. The first instruction of the three-address code is a leader.
    2. Instructions that are the target of conditional/unconditional goto are leaders.
    3. Instructions that immediately follow any conditional/unconditional goto/jump statements are leaders.
2. For each leader found, its basic block contains itself and all instructions up to the next leader.



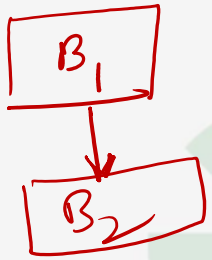
Compiler Design

Control Flow Graph(CFG): used in Code optimization & Code generation

→ A CFG gives a flow information b/w the Basic Blocks

→ Each Basic Block is a node in CFG

→ A CFG is a directed graph b/w the Basic Blocks.
There will be a directed edge from Basic Block B_1 to the Basic Block B_2 if B_2 comes immediately after B_1

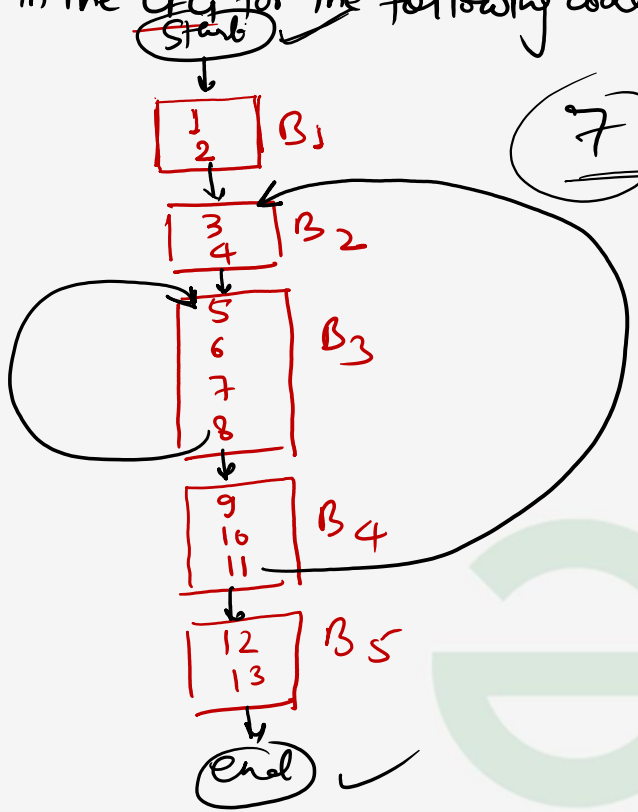


Compiler Design

Question: Find the No. of nodes & edges in the CFG for the following code segment

```
1 - a = 1
2 - i = 0
3 - t1 = a+5
4 - t2 = x*a
5 - t3 = 4*b
6 - a = a+t3
7 - i = i+1
8 - if i<t3 goto5
9 - J = J+1
10 - K = K+1
11 - if J<K goto3
12 - i = J+K
13 - K = 0
```

CFG:



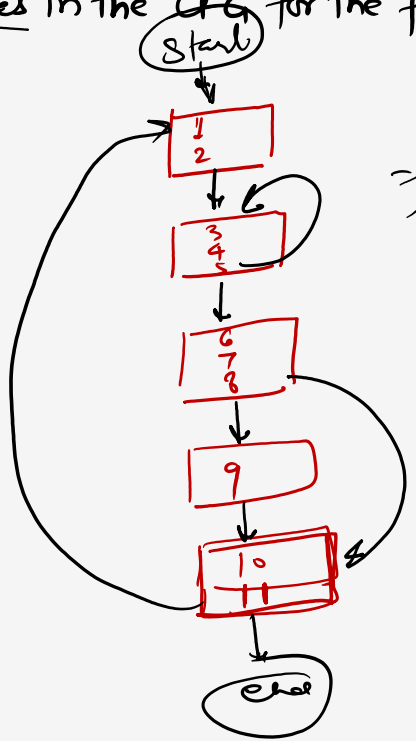
7 nodes & 8 edges

7

Compiler Design

Question:- Find the No. of nodes & edges in the CFG for the following code segment

1- a = b
2- b = a+5
3- c = a-b
4- a = a+5
5- if a>d goto 3
6- b = c*a
7- d = b+c
8- if b<a goto 10
9- c = a-b
10- a = a*10
11- if a<100 goto (1)



⇒ 7 nodes & 9 edges

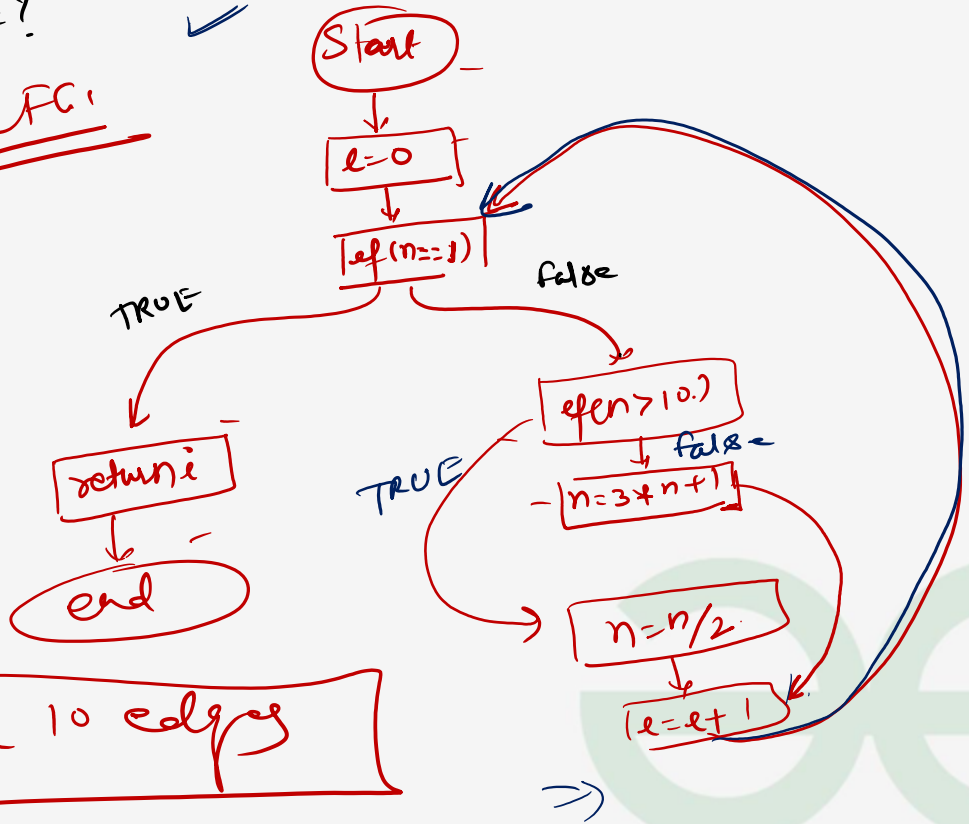
Compiler Design

CFG:

Question:- no of nodes & edges = ?

```
i = 0
L1: if(n==1) goto L4
    if(n>10) goto L2
    n = 3*n+1
    Goto L3
L2: n = n/2
L3: i = i+1
    goto L1
L4: return i
```

CFG:



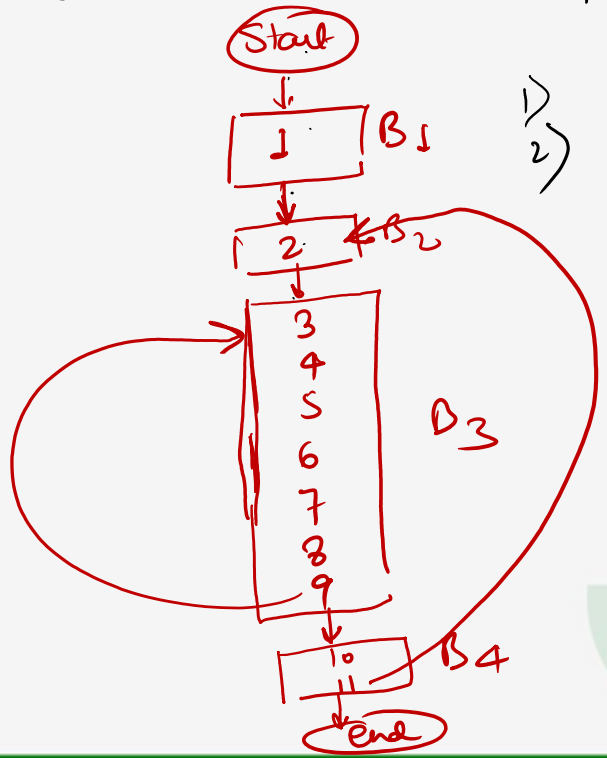
9 nodes & 10 edges

Compiler Design

Question:- Consider the following intermediate code given below. The no. of nodes & edges in the Control flow graph constructed for the code respectively are ?

a) 5 and 7
b) 6 and 7
c) 5 and 5
d) 7 and 8

6 & 7



1) $i = 1$
2) $J = 1$
3) $t_1 = 5 * i$
4) $t_2 = t_1 + J$
5) $t_3 = 4 * t_2$
6) $t_4 = t_3$
7) $a[t_4] = -1$
8) $J = J + 1$
9) if ($J <= 5$) goto 3
10) $i = i + 1$
11) if ($i < 5$) goto 2

Compiler Design

Thank You !