Homework #5 Solution

1.Convert the following program (Fig. 9.10) to SSA form:

Show your work after each stage:

a. Add a start node containing initializations of all variables.

Add Block B0: {a= , b= , c= , d= , e= }

Add Edges: ENTRY -> B0, B0->B1

b. Draw the dominator tree.

```
ENTRY -> B0 -> B1 -> B2 -> B3 -> B4
                           |
                           ---> B5 -> B6 -> EXIT
```

c. Calculate dominance frontiers for each node.

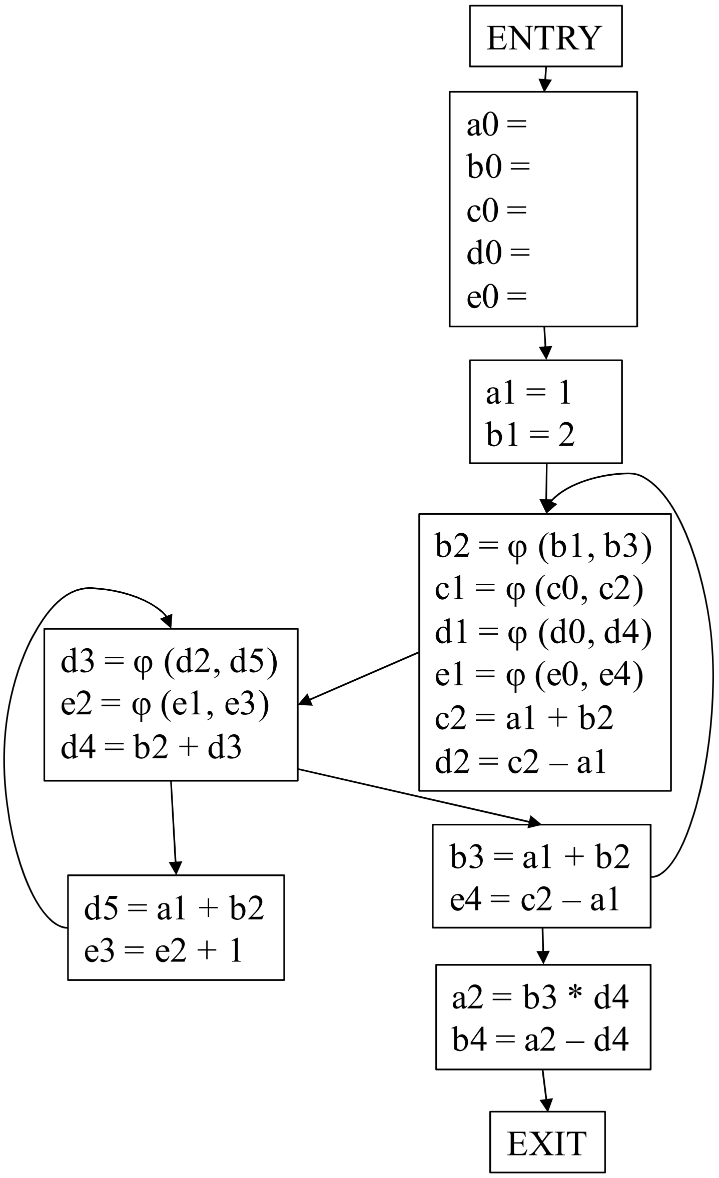
Node	DF
Entry	{ }
B0	{ }
B1	{ }
B2	{ B2}
B3	{ B2, B3 }
B4	{ B3 }
B5	{ B2 }
B6	{ }
EXIT	{ }

d. Insert φ-functions.

Add φ-function for {b, c, d, e} in B2

Add φ-function for {d, e} in B3

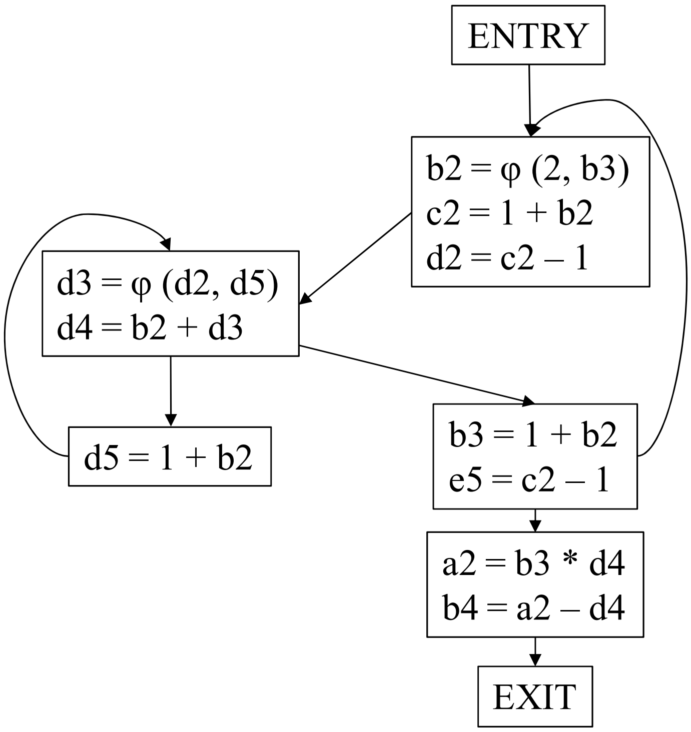
e. (Renaming) Add subscripts to variables.



f. Perform constant propagation and dead code elimination optimizations on the SSA form.

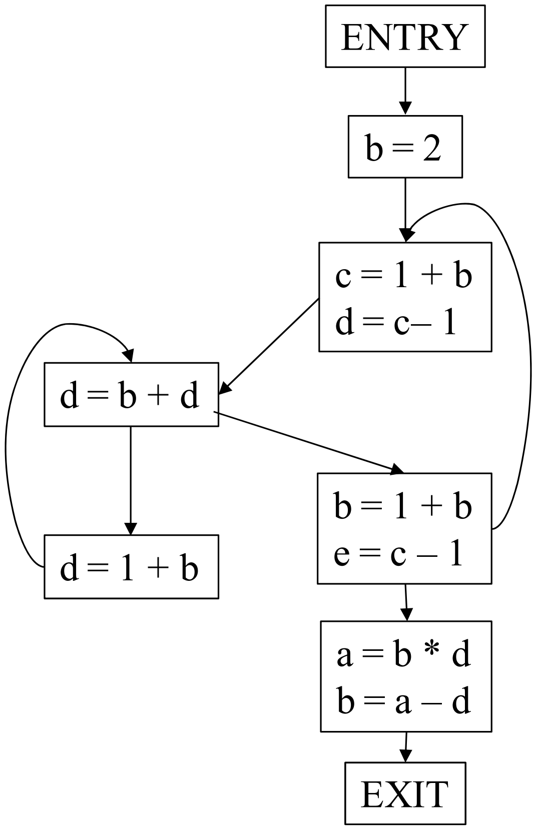
Assume the last instances of each variable (a2, b4, c1, d3, e4) are live after EXIT.

First, perform optimization based on SSA:



```
ENTRY

b2 = φ (2, b3)
c2 = 1 + b2
d2 = c2 – 1

d3 = φ (d2, d5)
d4 = b2 + d3

d5 = 1 + b2

b3 = 1 + b2
e5 = c2 – 1

a2 = b3 * d4
b4 = a2 – d4

EXIT
```

Next, transform the code out of SSA after optimization (optional):



```
ENTRY

b = 2

c = 1 + b
d = c– 1

d = b + d

d = 1 + b

b = 1 + b
e = c – 1

a = b * d
b = a – d

EXIT
```