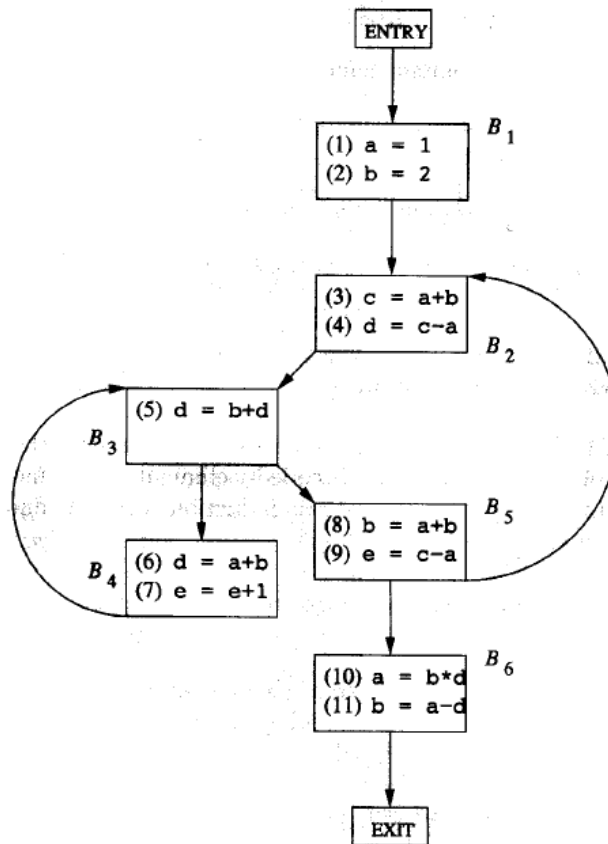


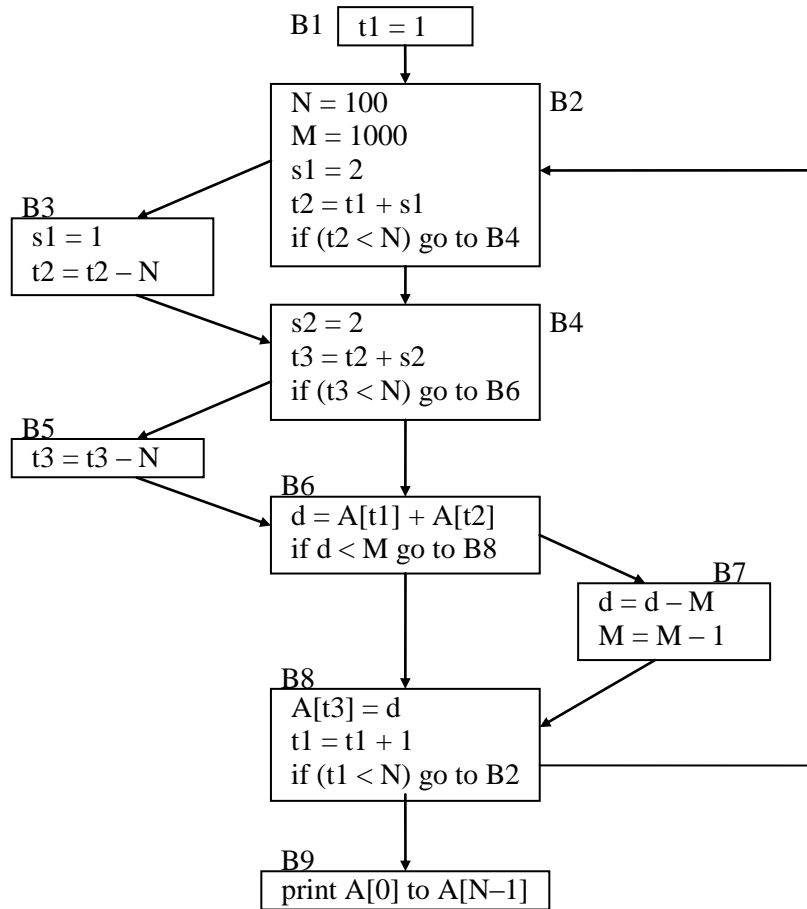
CS 6353 Compiler Construction, Homework #4

1. Consider the flow graph in Dragon book, Figure 9.10 (also given as follows).



- Compute Gen and Kill sets for each block in the flow graph.
- Compute the In and Out sets for each block in the flow graph.
- Perform constant propagation and constant folding.
- Perform common subexpression elimination.

2. Consider the following flow graph.



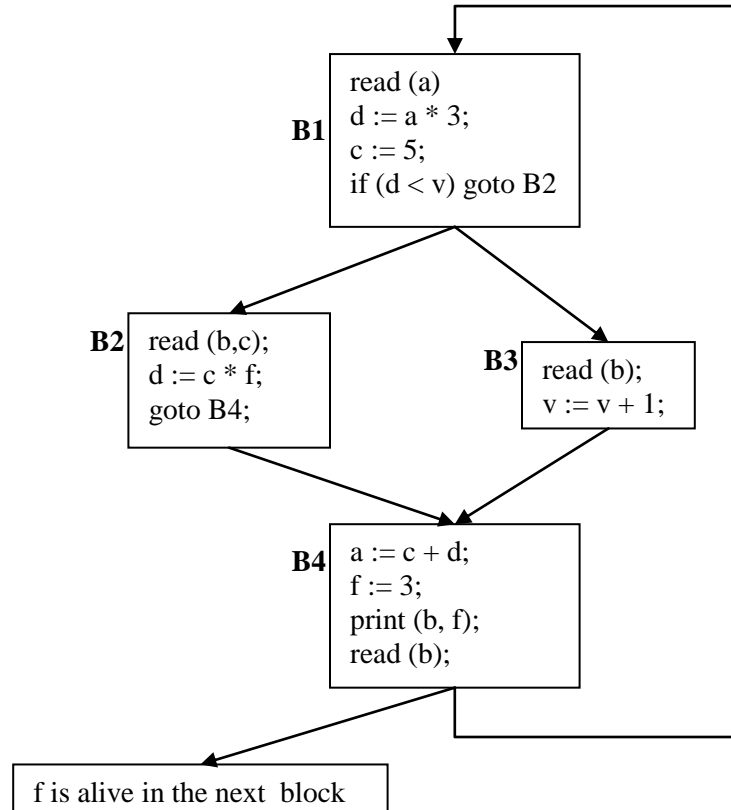
- Identify and mark define-use links within the loop based on data flow analysis results (based on the In and Out sets). You need to consider the block level define-use relations as well as define-use relations within the block.
- Identify all the loop invariants based on the define-use links computed in (b). Assume that all the loop invariants can be moved out of the loop. So you need to find loop invariants repetitively till a fixed point is reached. In each round, you need to pretend to move out those loop invariants you found in the previous rounds.
- For each loop invariant, determine whether it can actually be moved out of the loop. If so, move it out of the loop. If not, state the reason why it cannot be moved out. Generate the new code after code motion.

3. Consider the following three address code in a basic block.

- (1) $t1 = j - 1$
- (2) $t2 = 4 * t1$
- (3) $temp = A[t2]$
- (4) $t3 = j$
- (5) $t4 = j + 1$
- (6) $t5 = 4 * t3$
- (7) $t6 = A[t5]$
- (8) $t7 = j - 1$
- (9) $t8 = 4 * t7$
- (10) $A[t8] = t6$
- (11) $t9 = j$
- (12) $t10 = j + 1$
- (13) $t11 = 4 * t9$
- (14) $A[t11] = temp$

- (b) Perform copy propagation. You need to do necessary data flow analysis to make sure that the propagation can be done correctly.
- (c) Perform dead code elimination. You need to do necessary analysis to make sure that the elimination can be done correctly. Also, assume that after the basic block, $A[i]$, for all i , are alive and no other variables are alive.

4. Consider the following CFG.



- Perform liveness analysis. Show the Live sets at each point of the CFG. Construct the interference graph.
- Assume that the system has five registers. Perform color assignment. Determine the register for each variable.
- Assume that the system has only three registers. Perform color assignment and determine the register for each variable. Note that you need to spill and rewrite the code and redo the register assignment.

5. Consider the following three address code.

```
t1 := j
t1 := t1 * dim2
t1 := t1 + k
t1 := t1 * w
t2 := j * k
t3 := j
t3 := t3 * dim2
t3 := t3 + k
t3 := t3 * w
t4 := B [ t3 ]
t5 := t2 + t4
A [ t1 ] := t5
```

- (a) Construct a dag from the three address code such that subexpressions and redundant variables are eliminated.
- (b) Schedule the dag based on the numbering scheme. Run a global register allocation algorithm. Generate code based on the register allocation and instruction schedule you derived. Your code should make sure that the content of arrays A and B are permanently modified.

6. incomplete