CS243 Final Examination

Winter 2001-2002

You have 3 hours to work on this exam. The examination has 180 points. Please budget your time accordingly.

Write your answers in the space provided on the exam. If you use additional scratch paper, please turn that in as well.

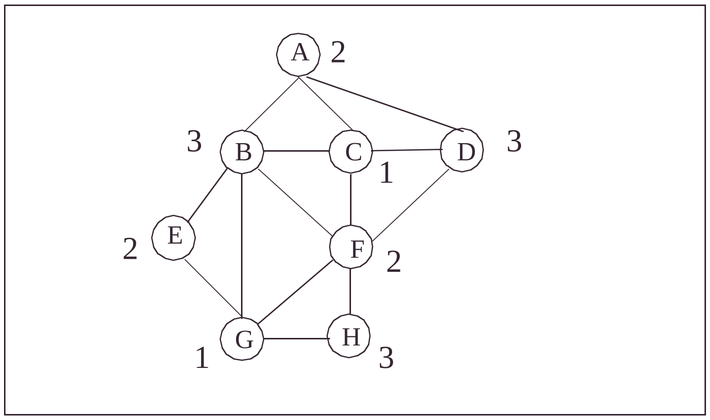
Your Name: ___________________________

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

Signature: ___________________________

Problem	Points	Score
1	20	__________________
2	20	__________________
3	30	__________________
4	20	__________________
5	30	__________________
6	20	__________________
7	40	__________________
Total	180	__________________

1. (20 points) Consider the following interference graph:



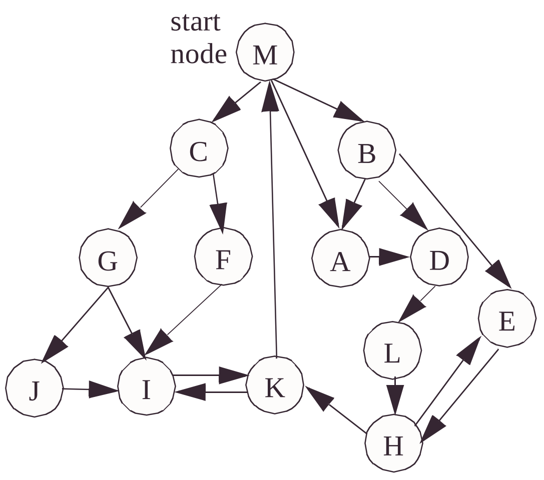
a. Give a machine with 3 registers, is it possible to find an allocation for the graph above without spilling? You may answer one of yes, no, “I don’t know”. If yes, give a coloring.

Yes. See diagram for an example.

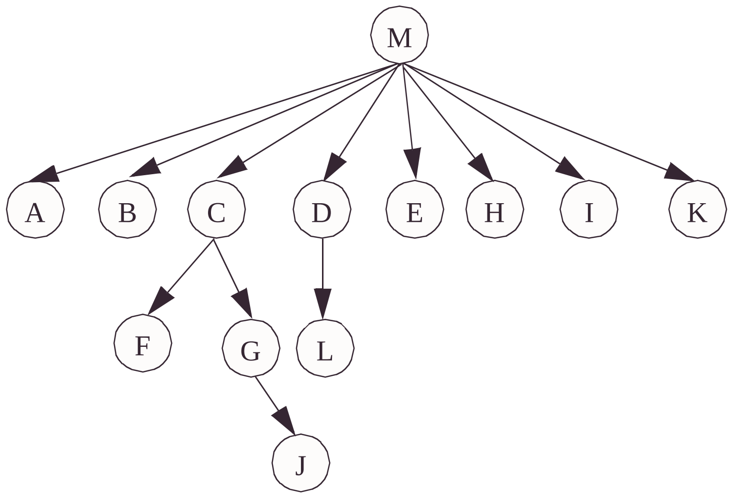
b. How does the “improved Chaitin’s algorithm” behave when given this input? Recall that the improved Chaitin’s algorithm keeps variables that may need to be spilled on the “stack”, and tries to assign a register to them in the assignment phase.

It may or may not find a coloring -- it depends on the node it picks to “spill”, which should be guided by the frequency of execution as well. After eliminating E, H, G, it is left with A, B, C, D, F. If it picks C to eliminate next, then it is possible for it to assign values 1,2,3 to A, D, F and fail to assign C. If it picks any of the other nodes, then A and F (or B and D) will be given the same color, and it can allocate all variables without spilling.

2. (20 points) Consider the following control flow graph:



And here is its dominator tree.



What are the natural loops in this flow graph? (Show intermediate steps for partial credit.)

There is only 1 natural loop from K to M; all nodes are included in the loop.

3. (30 points) Suppose you have carefully defined a forward data flow algorithm for a framework that is distributive and has only finite descending chains.

   a. A colleague of yours accidentally deletes the code that initializes the entry node. Without telling you, he initializes the entry node to $\perp$ (bottom). Does that change the result of the analysis? Why?

      It depends on whether the boundary should be set to bottom or not. If yes, then the result does not change, otherwise, it will most likely unless the value is not really used in the data flow solution. (Note that the last clause is not necessary to get full credit).

   b. Suppose instead your colleague initializes all the *internal* nodes to $\perp$ (bottom).

      i. Will this algorithm give a *safe* answer for *all* flow graphs?

         Yes. (An explanation was not asked for here...)
         Because of monotonicity, having injected a lower value into the flow graph equations will keep the answer safe.

      ii. Will this new algorithm give a *safe* answer for *som*e flow graphs? If so, which ones?

         Naturally, since the above is yes.

      iii. Will this new algorithm give the *meet-over-paths* answer for *all* flow graphs?

         No. It is not guaranteed that we find the max fixpoint.

      iv. Will this new algorithm give the *meet-over-paths* answer for *some* flow graphs? If so, which ones?

         Yes, any graphs that do not have cycles.

4. (20 points) Consider the following program fragment:

```
int i, t, a[10000], b[10000]
for (i = 0; i<n; i++) {
    t = b[i];
    a[i] = t + t;
}
/* t is not used after this point */
```

It is obvious that we can rewrite the code as:

```
for (i = 0; i<n; i++) {
    int t;
    t = b[i];
    a[i] = t + t;
}
```

This transformation of shrinking the scope of a scalar variable to that of a loop body is known as *scalar privatization*. Giving each iteration a private version of the variable enables the iterations in this loop to be parallelized.

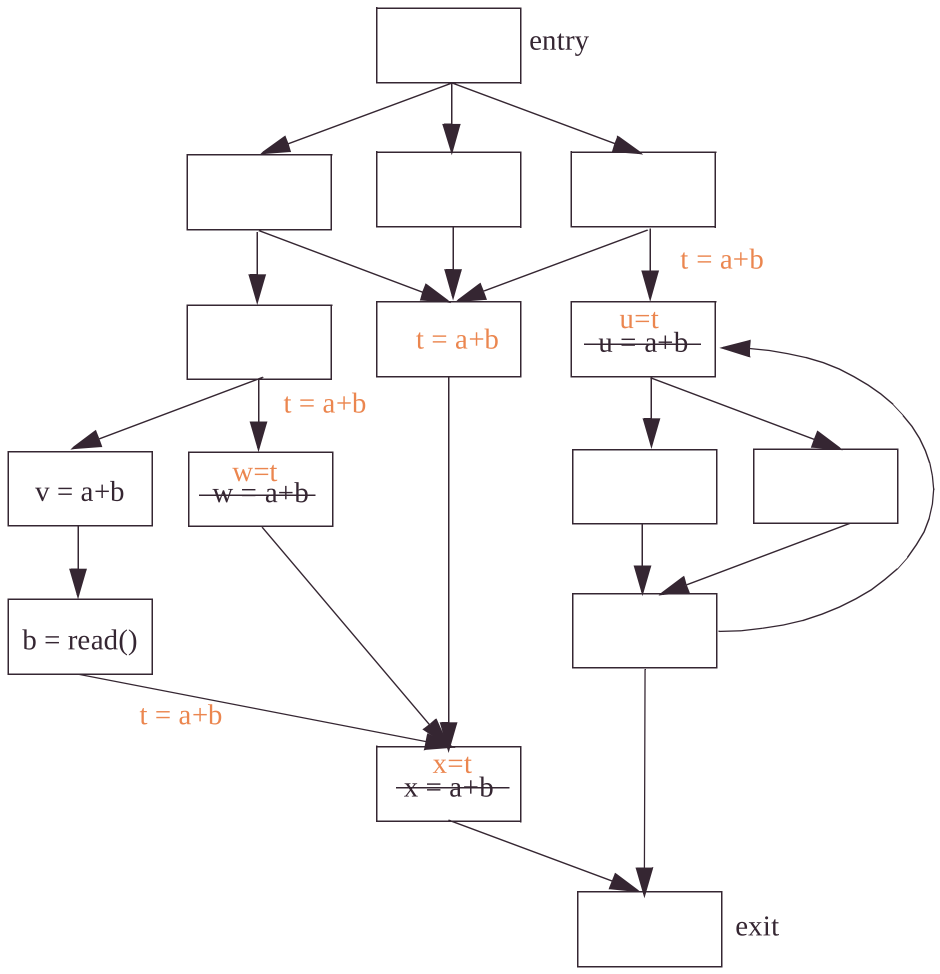
a. When is it legal to privatize a variable?

When the value is not live at the beginning of each iteration.

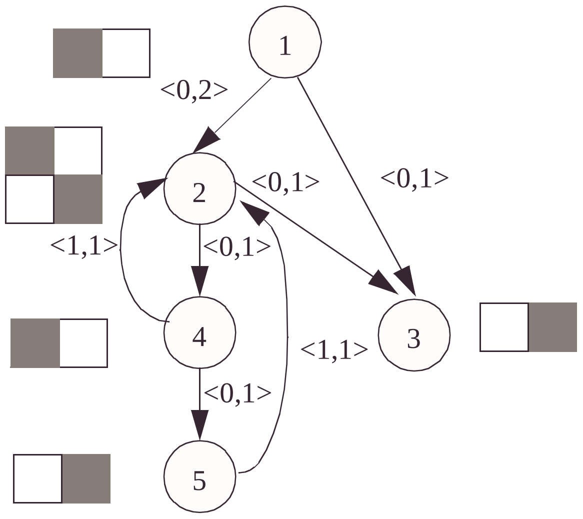
b. Describe an algorithm to detect when a scalabe variable in a loop is privatizable.

```
Run liveness analysis, and check if the variable is live at
the beginning of the iteration.
```

5. (30 points) Apply lazy code motion to the following program. Show the optimized code in the figure. You may add basic blocks to the flow graph, but only show those that are not empty in your solution.



6. (20 points) What is the best software pipelined schedule that you can create for the following precedence graph. This machine has two resources, R0, R1. The edges are labeled by the <iteration difference, latency>; each node is represented by its respective resource reservation table. If an instruction uses resource Rj in cycle i after it has been issued, (i = 0, 1), it is indicated by a black square in row i and column j in the resource reservation table. For example, node 4 uses R0 in cycle 0, and R1 in cycle 1 after the node is issued.



a. What is the bound of the initiation interval?
3.
Resource constraints: R0: 3 units, R1, 3 units
Precedence constraints: 2 cycles: 2/1, 3/1
Therefore max of all constraints are 3.

b. Find the best software pipelined schedule. What is the initiation interval of your schedule? Show the schedule of one iteration.

The best initiation interval is 3:
The schedule is

cycle	
0	1
1	
2	
3	
4	2
5	4
6	5
7	3

7. (40 points) Consider the following program:

```
m = 100;
k = 0;
FOR i = 2 TO (m-1) {
    p = k;
    r = 20;
    FOR j = 2 TO (m-1) {
        if (B[i,j] > C[i,j]) {
            A[p] = A[r + 40*j];
        }
        E[i,j] = D[i,q];
        q = q+2;
        D[i,q-1] = E[i,j];
        p = p+1;
        r = r+1;
    }
}
```

You may use any of the techniques discussed in this course to answer the question, but you must specify the analyses you use to compute the answer.

a. Is the outermost loop parallelizable? Why? Explain how you get your answer; in particular, specify the data dependence tests performed.

No.
Constant propagation will set all the loop bounds to 2:99, and p = 0
Induction analysis of p:
at the beginning of the inner loop:
p = 0 + x1
r = 20 + x1
q = q0 + 2*x1

Dependence tests formulated for the innermost loop for A:
0 <= x0 <= 97
0 <= x1 <= 97
2 <= j <= 99
A is read and written conditionally, we must assume the worst case and say that you may be reading and writing A in each iteration.
A[x1] = A[20*x1 + 40*j]
Dependence test:
does there exist x0r, x1r, x0w, x1w such that
0 <= x0r, x0w <= 97
0 <= x1r, x1w <= 97
2 <= j <= 99
x1r = 20*x1w + 40*j
x0r ≠ x1r

There are many solutions. Just set x1r = x1w and give x0r, x1r different values.
The loop is not parallelizable.

b. Is the innermost loop parallelizable? Why? Explain how you get your answer; in particular, specify the data dependence tests performed.

Yes.

Dependence tests formulated for the innermost loop for A:
0 <= x0 <= 97
0 <= x1 <= 97
2 <= j <= 99

A[x1] = A[20*x1 + 40*j]
Dependence test:
does there exist x0r, x1r, x0w, x1w such that
0 <= x0r, x0w <= 97 (similar to loop test replacement)
0 <= x1r, x1w <= 97
2 <= j <= 99
x1r = 20*x1w + 40*j
x1r ≠ x1w
x0r = x1r

Since the range of x1r <= 97, and 20*x1w + 40*j >= 100, there is no dependence

For D[i,q] and D[i,q-1], with induction variable analysis we get
D[i,q0+2*xi] and D[i, q0+2*xi+1]
Dependence test:
does there exist, x0r, x1r, x0w, x1w such that
0 <= x1r, x1w <= 97
2 <= j <= 99
q0+2*x1r = q0+2*x1w +1
there is no solution because either you are reading even elements and writing odd elements, or vice versa.
We need to apply more tests on E, and output dependence tests on A and D, and there will be no dependence.
Therefore the inner loop is parallelizable.