CS745
Optimizing Compilers
Spring 2001

Goldstein & Mowry
Date: 3/21/01

CS745 Exam

March 21, 2001

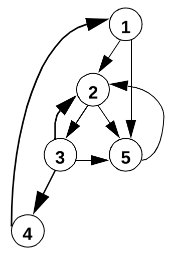
Name ____

This exam is *closed book/notes*. There are XXX points in total, and XXX minutes for the exam (roughly one minute per point). Write your answers in the spaces provided, or on the backs of the pages if you need more room.

Problem #	Total Points	Score
1	10	
2	10	
3	10	
4	25	
5	25	
Total	XXX	

1. Natural Loops (10 points)

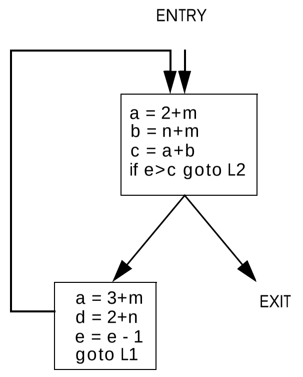
(i) What are the natural loop(s) in the following code? (Show intermediate steps for partial credit.)


$$\begin{tikzcd}
 & & 1 \arrow[dl] \arrow[ddr] & \\
 & 2 \arrow[dl] \arrow[drr] & & \\
3 \arrow[ur, bend left] \arrow[rrr] \arrow[d] & & & 5 \arrow[ull, bend right] \\
4 \arrow[uuurr, bend left] & & &
\end{tikzcd}$$

(ii) Insert preheaders for whatever natural loops exist above.

2. Loop-Invariant Code Motion (10 points)

Apply loop-invariant code motion to the following program:



```
a = 2+m
b = n+m
c = a+b
if e>c goto L2
```

```
a = 3+m
d = 2+n
e = e - 1
goto L1
```

3. Properties of Transfer Functions (10 points)

Explain the difference between *monotone* transfer functions and *distributive* transfer functions in data flow analysis. What is the significance of the difference?

3. Register Allocation (20 points)

Assume you are doing register allocation for a machine that has 4 registers. R0 is used to pass the first argument to a call and to return the result from a caller to the callee, R1 & R2 are callee save registers, and R3 is a caller save register.

(a) Draw the interference graph for the following code fragment. When drawing the graph use solid edges to represent interference. Use dashed edges to represent registers which are candidates for coalescing. Before drawing the interference graph construct webs and rename temporaries as appropriate. Use a separate table to indicate the mapping that results from web creation.

```
    B <- r0
    if B < 5 goto L1
    D <- B + 1
    E <- D
    goto L2

L1: i <- 0

L3: D <- B + i
    if i < 5 goto L2
    st   D
    E <- D + i
    i <- i + 1
    goto L3

L2: r0 <- E
    call foo
    f <- r0
    i <- 0

L4: if i < B goto L5
    F <- F + i
    i <- i + 1
    goto L4
```

```
L5: r0 <- F
    ret
```

(b) Show the interference graph and the stack of simplified nodes after the first round of coalescing has been completed.

3. Properties of Transfer Functions (10 points)

Explain the difference between *monotone* transfer functions and *distributive* transfer functions in data flow analysis. What is the significance of the difference?

4. Data Flow Analysis (25 points)

Consider a language that allows only boolean variables (a boolean variable can hold either the value TRUE or FALSE), and allows only the following types of assignment statements:

```
a = TRUE
a = FALSE
a = b
a = NOT b
```

Devise a data flow algorithm that determines, for all points in the program, whether each variable can *possibly* hold a TRUE value.

(a) What is the direction of your data flow analysis?

(b) Draw a diagram of the lattice for one variable, identifying the top and bottom elements clearly.

(c) How do you initialize the information at the entry/exit nodes?

(d) How do you initialize the iterative algorithm?

(e) Define the transfer function of a basic block.

(f) Is the data flow framework monotone? Why?

(g) Is the data flow framework distributive? Why?

(h) Will your algorithm necessarily converge? If so, why?

5. Short Answer (25 points)

(a) (5 points)
What is the difference between partial redundancy elimination and global common subexpression elimination?

(b) (5 points)
How does partial redundancy elimination relate to loop invariant code motion?

(c) (5 points)
What advantage does interval analysis offer over iterative data flow techniques? When would you choose to use interval analysis rather than iterative data flow?

(d) (5 points)
Briefly explain the difference between how basic and non-basic induction variables are optimized under strength reduction.

(e) (5 points)
Regarding locality analysis (which attempts to predict when cache misses occur), why is it that data *reuse* does not necessarily imply data *locality*?

(f) (5 points)
The dominance frontier is used for performing aggressive dead-code elimination. What is it used for?

(g) (5 points)
In what way does SSA-form make the aggressive dead-code elimination algorithm easier to implement?

6. SSA (15 points)

(a) For the following code fragment indicate the dominance frontiers for each basic block.

```
L0:i <- 0
j <- 0
if j < 10 goto L5

L1:if i > j goto L2
j <- j + 1

L2:if i < 5 goto L3
```

```
k <- 0
goto L4

L3:k <- 1

L4:i <- call foo(k)
j <- j + 1
goto L1

L5:return i
```

(b) Using your answer to part a, insert the minimal number of PHI functions into the code fragment above. Draw the resulting control flow graph. You do not have to renumber the variables.