CMSC430 Spring 2011 Midterm 3 Solutions

1. (12 pts) Compiler optimizations
   a. (4 pts) Give an example of how compiler optimizations improve programmer productivity by supporting high-level language abstractions. Name a specific programming language construct.

      Examples: 1) Compilers can efficiently implement multidimensional arrays by optimizing array address calculations. 2) Compilers can efficiently implement inheritance in object oriented languages by using prefixing to reduce the cost of field and method accesses.

   b. (4 pts) Explain why compiler optimizations are typically organized as passes over the program.

      Examples: 1) So the same optimization may be applied to the code multiple times. 2) So optimizations may be easily added / removed / reordered.

   c. (4 pts) Describe partial redundancy elimination and how it improves performance.

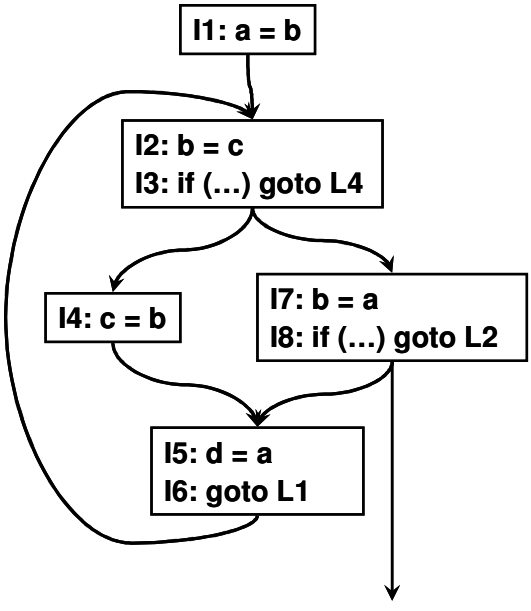
      Compute an expression on path(s) where it is not available to create more available expressions and enable common subexpression elimination.

2. (10 pts) Control flow analysis

   Consider the following code:

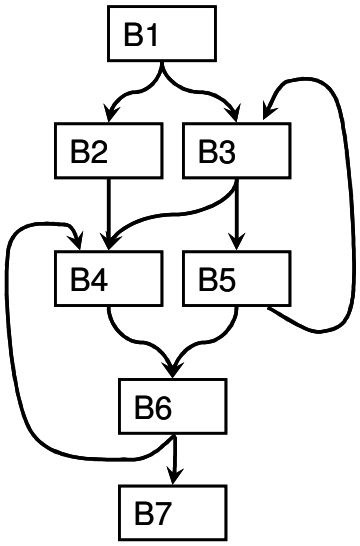
```
<I1>          a := b
<I2>    L1:   b := c
<I3>          if (...) goto L4
<I4>          c := b
<I5>    L2:   d := a
<I6>    L3:   goto L1
<I7>    L4:   b := a
<I8>    L5:   if (...) goto L2
```

   Find basic blocks and draw the control flow graph (CFG).



3. (12 pts) CFG ordering, dominators, loops

Consider the following control flow graph, where B1 is the start of the program.



a. (6 pts) Give 2 different reverse Postorder numberings of the control flow graph.

Possible rPostorder traversals:

1 2 3 5 6 4 7	**1 2 3 5 6 7 4**
1 2 3 5 4 6 7	**1 3 5 2 4 6 7**

b. (2 pts) List all basic blocks dominated by B3.

B3, B5.

c. (2 pts) List all basic blocks dominated by B4.

B4.

d. (2 pts) List all the loops in the control flow graph. For each loop list all basic blocks in the loop.

The only loop is B3, B5. The cycle at B4 is not a loop since B4 does not dominate B6 (i.e., can enter cycle without going through B4).

4. (8 pts) Data-flow equations

Consider the control flow graph (CFG) in Problem 3. Assume you are given IN/OUT for all the basic blocks *except* for B4, and GEN/KILL for B4. Show the data-flow equations for IN/OUT for B4 (e.g., IN(B4) = OUT(B1)).

a. (4 pts) For forward data-flow problems

IN(B4) = OUT(B2) ∧ OUT(B3) ∧ OUT(B6)
OUT(B4) = GEN(B4) ∪ (IN(B4) – KILL(B4))

b. (4 pts) For backwards data-flow problems

OUT(B4) = IN(B6)
IN(B4) = GEN(B4) ∪ (OUT(B4) – KILL(B4))

5. (10 pts) Local information
   Consider the following basic block for available expressions:

```
I1:  b = a+c
I2:  d = b*d
I3:  a = b+c
I4:  e = b*a
I5:  a = a+b
```

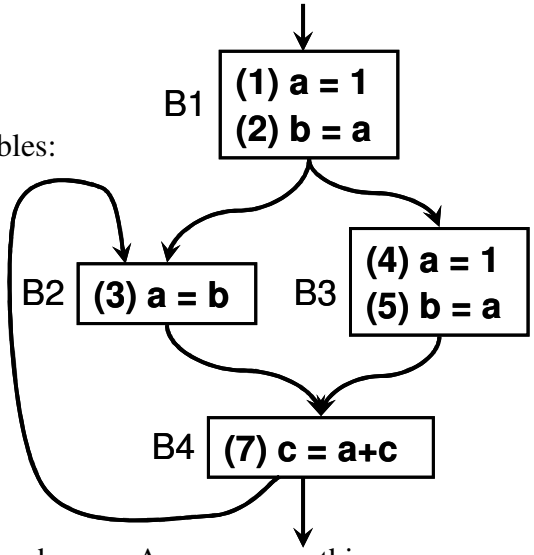
   a. (5 pts) What is GEN for the basic block?

      GEN = { b+c }

   b. (5 pts) What is KILL for the basic block?

      OUT = { a+c, b*d, b+c, b*a, a+b }

6. (32 pts) Data flow analysis:
   Consider the following control flow graph for live variables:



   a. (8 pts) Calculate GEN/KILL for each basic block

	GEN	KILL
B1	∅	**a, b**
B2	**b**	**a**
B3	∅	**a, b**
B4	**a, c**	**c**

   b. (24 pts) Solve live variables, showing IN/OUT for each pass. Assume everything is initialized to Ø.

		Init	Pass1	Pass2	Pass3
B4	OUT	∅	∅	**b, c**	**b, c**
	IN	∅	**a, c**	**a, b, c**	**a, b, c**
B3	OUT	∅	**a, c**	**a, b, c**	**a, b, c**
	IN	∅	**c**	**c**	**c**
B2	OUT	∅	**a, c**	**a, b, c**	**a, b, c**
	IN	∅	**b, c**	**b, c**	**b, c**
B1	OUT	∅	**b, c**	**b, c**	**b, c**
	IN	∅	**c**	**c**	**c**

7. (16 pts) Data-flow analysis frameworks

   Recall that $\wedge$ is used in data-flow iterative analysis to combine information where paths merge.

   a. (4 pts) How is the $\wedge$ operator used to define the $\leq$ and $<$ operators?

      $\leq$ is defined using $\wedge$ as $x \leq y$ if $x \wedge y = x$

      $<$ is defined using $\wedge$ as $x < y$ if $x \wedge y = x$ AND $x \neq y$

   b. (4 pts) For very busy expressions, pick values for a, b, c for which a < b and b < c.

      Example: a = { x+y }, b = { x+y, m+n }, c = {x+y, m+n, x*m }

   c. (4 pts) For reaching definitions, pick values for a, b for which neither $a \leq b$ or $b \leq a$ are true.

      Example: a = { x = … } and b = { y = …}

   d. (4 pts) How can iterative dataflow analysis problems be formulated so that they are guaranteed to eventually stabilize and terminate?

      By ensuring the dataflow framework is monotonic, and lattices have finite height.