CSE P 501 18sp Homework 4

Due: Monday, May 21 by 11 pm. *No late submissions accepted this time* so we can distribute sample solutions the next day before class. As with previous assignments, please use Gradescope (linked from the CSE P 501 web page) to submit your homework online.

- Unreadable solutions cannot be graded--no blurry photos, poor contrast, or illegible handwriting, please.
- Type-written solutions are encouraged but not required.
- If possible, don't split the solution to a problem across a page break.

We suggest you show your work to help us award partial credit if appropriate, and for TA sanity. You should do this assignment individually.

1. (value numbering, based on Cooper/Torczon ex 1, p. 471) For the following program, (i) apply local value numbering to the statements in the block, and (ii) rewrite the code to eliminate redundant expressions using the value numbering information.

$$
\begin{aligned}
t_1 &= a + b \\
t_2 &= t_1 + c \\
t_3 &= t_2 + d \\
t_4 &= b + a \\
t_5 &= t_3 + e \\
t_6 &= t_4 + f \\
t_7 &= a + b
\end{aligned}
$$

2. (dataflow; a similar problem appears as #8 on the CSE P 501 08wi exam) Consider the following small program that we used as a dataflow example for live variable analysis. This time all of the statements are labeled and we want to compute reaching definitions.

```
L0:  a = 0
L1:  b = a + 1
L2:  c = c + b
L3:  a = b * 2
L4:  if a < N goto L1
L5:  return c
```

The *reaching definitions* dataflow problem is to determine for each variable definition which other blocks in the control flow graph could potentially see the value of the variable that was assigned in that definition. To simplify things we will treat each individual statement above as a separate block, and use the statement labels as the names of both the blocks and the definitions in them. So, for example, reaching definition analysis would allow us to determine that definition L0, which assigns to a, can reach block L1.

A definition *d* in block *p* *reaches* block *q* if there is *at least* one path from *p* to *q* along which definition *d* is not killed.

This can be set up as a dataflow problem as follows: For each block *b* in the control flow graph, define GEN(*b*) to be the set of definitions generated in that block and not subsequently killed in the block, and KILL(*b*) to be the definitions killed by that block. These sets can be computed once, statically, for each block. If block *b* contains *d*: x = a op b, then GEN(*b*) contains *d*, provided that *d* is not killed later in block *b*. KILL(*b*) contains all other definitions *d'* elsewhere in the program that define the same variable x.

CSE P 501 18sp Homework 4

Given the GEN and KILL sets for the blocks, we can compute the IN and OUT sets of definitions that reach each block as follows:

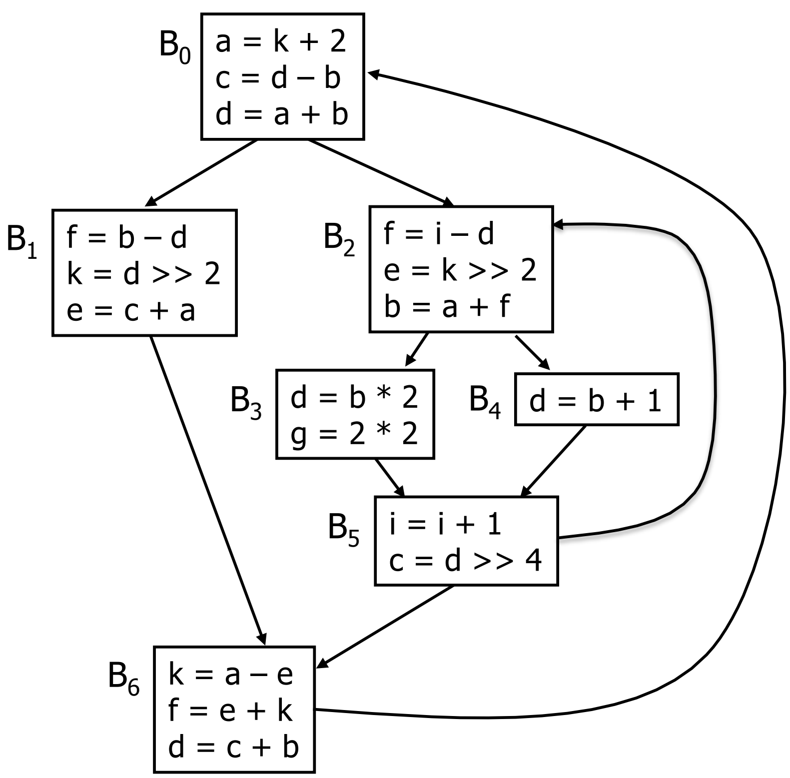
$$IN(b) = \bigcup_{p \in pred(b)} OUT(p)$$

$$OUT(b) = GEN(b) \cup (IN(b) - KILL(b))$$

For this problem, compute the reaching definitions for the blocks in the given program (treating each statement as a separate block). You should give a table with a row for each block showing the GEN and KILL sets for that block, and then compute IN and OUT sets using successive iterations until you there are no further changes to any IN or OUT set.

Note that this is a forward dataflow problem so the answer will converge faster if you start computing with L0, L1, …

3. (loops) Compute the dominator tree for the following CFG, then compute the dominance frontiers for nodes B2, B5, and B6.



4. (ssa, Cooper/Torczon ex. 6, p. 536-537). Translate the code (CFG) from the previous problem into SSA form. You only need to show the final code after both Φ-insertions and renaming. (If you want to edit your answer on a computer, the assignment page contains a link to the original ppt slide with the above image. However, don't feel obligated to do this – it might turn into quite a time sink.)