CSE 401 - Section 9 – Data Flow & SSA - Solutions

1. **Reaching Definitions**

   Consider the following small program that we used as a dataflow example for live variable analysis in lecture. This time all the statements are labeled individually, 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 redefined.

   Dataflow sets for this task:

   GEN(*b*): the definitions assigned and not killed in block b
   KILL(*b*): the definitions of variables overwritten in block b
   IN(*b*): the definitions that are reaching upon entering block b
   OUT(*b*): the definitions that are reaching upon exiting block b

   Equations for IN(*b*) and OUT(*b*) in terms of other sets and other basic blocks:

   $IN(b) = \bigcup_{p \in pred(b)} OUT(p)$
   $OUT(b) = GEN(b) \cup (IN(b) - KILL(b))$

a) Compute the reaching definitions for the blocks in the given program, treating each statement as a separate block. In the following table, compute the GEN and KILL sets for each block, and then use those answers to compute successive iterations of the IN and OUT sets until there are no more changes to be made.

Note that this is a forward dataflow analysis problem, so the answer will converge faster if you compute from beginning to end (i.e. starting with L0).

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0	L0	L3		L0		L0
L1	L1		L0	L0, L1	L0, L1, L2, L3	L0, L1, L2, L3
L2	L2		L0, L1	L0, L1, L2	L0, L1, L2, L3	L0, L1, L2, L3
L3	L3	L0	L0, L1, L2	L1, L2, L3	L0, L1, L2, L3	L1, L2, L3
L4			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3
L5			L1, L2, L3	L1, L2, L3	L1, L2, L3	L1, L2, L3

b) Now that we have completed our dataflow analysis, we want to apply optimizations to the code. After noticing that the definition L0 of the variable *a* is a constant value, we wonder if it is possible to use constant propagation to replace uses of the variable *a* with the constant 0.

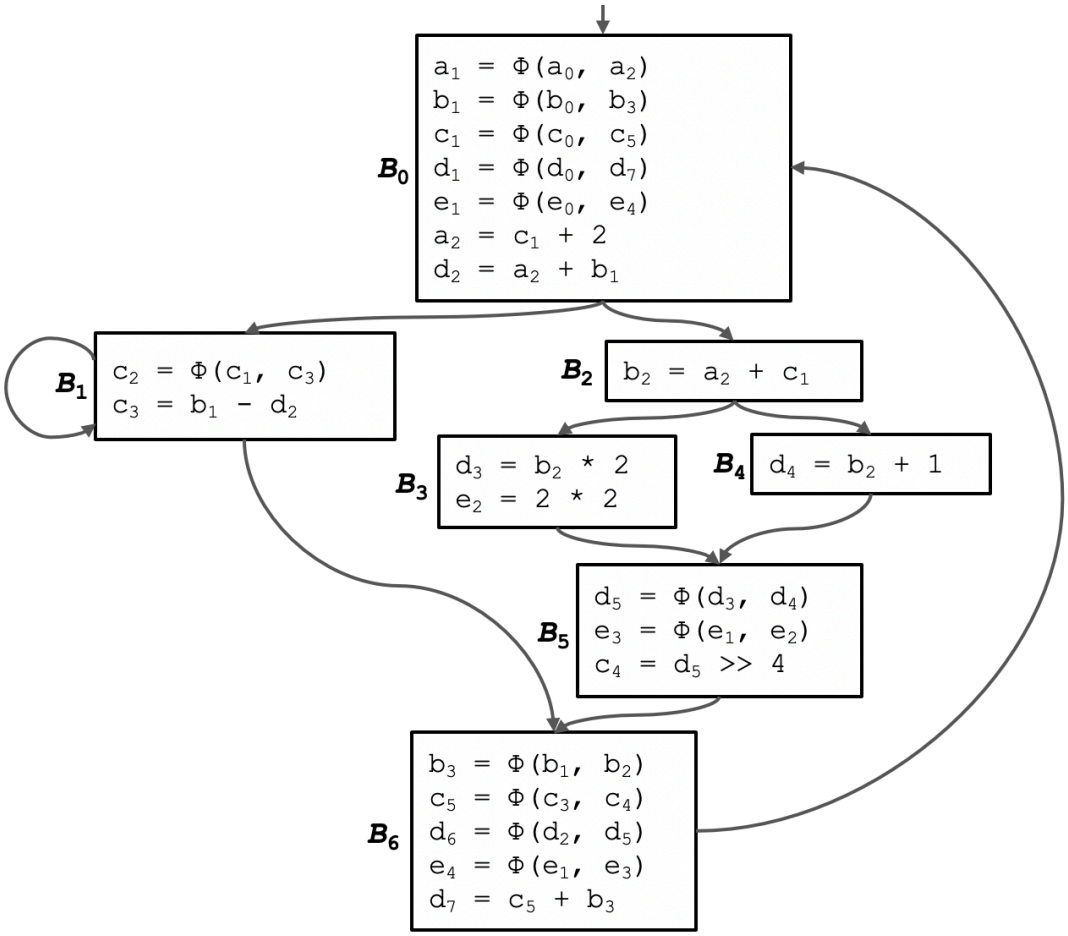
Is it possible to replace the use of *a* in block L1 with the constant 0? Justify your answer using evidence from the sets that you computed during dataflow analysis.

No, it is not possible. To determine this, we would look at the IN set for block L1 -- the fact that the IN set contains two definitions of ‘a’ (L0 and L3) means we cannot perform this constant propagation. In other words, more than one definition of ‘a’ is a reaching definition to block L1, and therefore performing constant propagation would only preserve one possible value of ‘a’ and the generated code would not be equivalent.

2. **Convert to SSA Form**

a) Convert this code into Single Static Assignment form, i.e., add phi functions at merge points and number the variables.

b) The final SSA control flow graph becomes:



The steps taken to compute this final control flow graph will be elaborated upon in upcoming lectures and the next section.