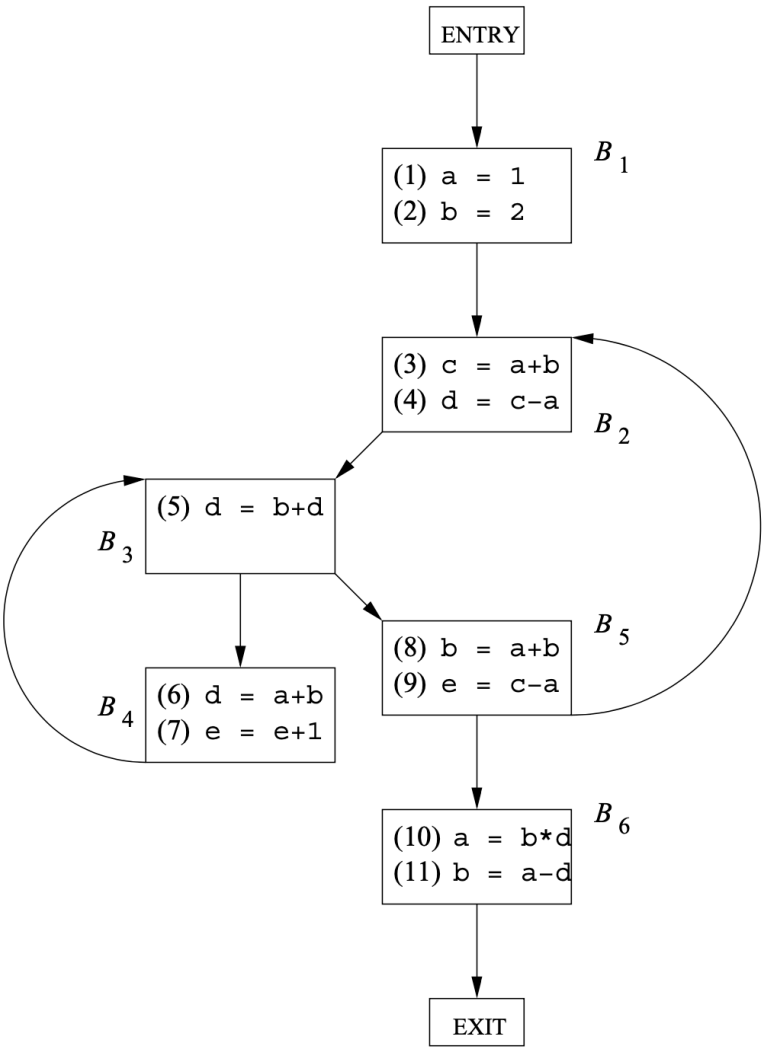
6.1100 Spring 2024 Miniquiz #5

Please submit your answers on Gradescope by April 4th, 2024, 11:59pm.

Name: 6.110 Staff

Email: 6.110-staff@mit.edu

Consider the following control flow graph:



Perform Reaching Definitions analysis, Available Expressions analysis, and Live Variables analysis. Fill your final answers in the table on the following pages.

1. Reaching Definitions

Identify variable definitions by the statement numbers. The first row was filled in for you.

Block	Gen	Kill	In	Out
B_1	{1, 2}	{8, 10, 11}	{}	{1, 2}
B_2	{3, 4}	{5, 6}	{1, 2, 3, 5, 8, 9}	{1, 2, 3, 4, 8, 9}
B_3	{5}	{4, 6}	{1, 2, 3, 4, 6, 7, 8, 9}	{1, 2, 3, 5, 7, 8, 9}
B_4	{6, 7}	{4, 5, 9}	{1, 2, 3, 5, 7, 8, 9}	{1, 2, 3, 6, 7, 8}
B_5	{8, 9}	{2, 7}	{1, 2, 3, 5, 7, 8, 9}	{1, 3, 5, 8, 9}
B_6	{10, 11}	{1, 2, 8}	{1, 3, 5, 8, 9}	{3, 5, 9, 10, 11}

Note: we accidentally published the wrong kill set originally. The correction is marked above. Sorry for any confusion that may have caused while completing the quiz.

Explanation: **Gen** is produced by including all the definitions inside the block. **Kill** is produced by finding all the other definitions of the variables (in other blocks) that are defined in a given block. Then the update rule Out = U[IN] - Kill + Gen is applied. Note that it has to be continually applied until you reach a fixed point.

2. Available Expressions

Consider only the following expressions, numbered in the order we provided:

1) a+b
2) c-a
3) b+d

Block	Gen	Kill	In	Out
B_1	{}	{1, 2, 3}	{}	{}
B_2	{1, 2}	{3}	{}	{1, 2}
B_3	{3}	{3}	{1, 2}	{1, 2}
B_4	{1}	{3}	{1, 2}	{1, 2}
B_5	{1, 2}	{1, 3}	{1, 2}	{2}
B_6	{}	{1, 2, 3}	{2}	{}

Explanation: **Gen** is produced by including all the expressions computed inside the block. **Kill** is produced by finding all expressions that involve variables that are *assigned values* inside the block. Note that the expression c-a inside B_2 is a tricky case of this rule. The expression computed inside the block is computed after the assignment, so the expression is still available. If the assignment came afterwards, it would not have been available. Then the update rule Out = U[IN] + Gen - Kill is applied (order of operations matters here, subtracting the Kill set must come last). Note that it has to be continually applied until you reach a fixed point.

3. Live Variables

Assume all variables are local, i.e. no variables are live upon exiting B_6.

Block	Gen (Use)	Kill (Def)	In	Out
B_1	{}	{a, b}	{e}	{a, b, e}
B_2	{a, b}	{c, d}	{a, b, e}	{a, b, c, d, e}
B_3	{b, d}	{d}	{a, b, c, d, e}	{a, b, c, d, e}
B_4	{a, b, e}	{d, e}	{a, b, c, e}	{a, b, c, d, e}
B_5	{a, b, c}	{b, e}	{a, b, c, d}	{b, d}
B_6	{b, d}	{a, b}	{b, d}	{}

Explanation: **Gen (Use)** is every variable that is used in an expression in the basic block. **Kill (Def)** is every variable that is assigned to in the basic block. A tricky case comes in B_6 with variable a. While a is used to compute an expression in this block, it uses the definition created earlier in the block, so it is not included in the **Gen** set. The update rule is In = U[Out] - Kill + Gen is applied (Note that now the order of operations is reversed, and + Gen must come after - Kill). Note that it has to be continually applied until you reach a fixed point.

4. Optimization

Optimize the given control flow graph as best you can, using copy propagation, common subexpression elimination, and dead code elimination. You may perform the optimizations by hand and introduce new temporaries as needed.

There are many potential solutions depending on the optimizations you use. An example is shown below: