

# Solutions to Written Assignment 8

1. This problem asked you to apply a sequence of optimizations to a basic block.

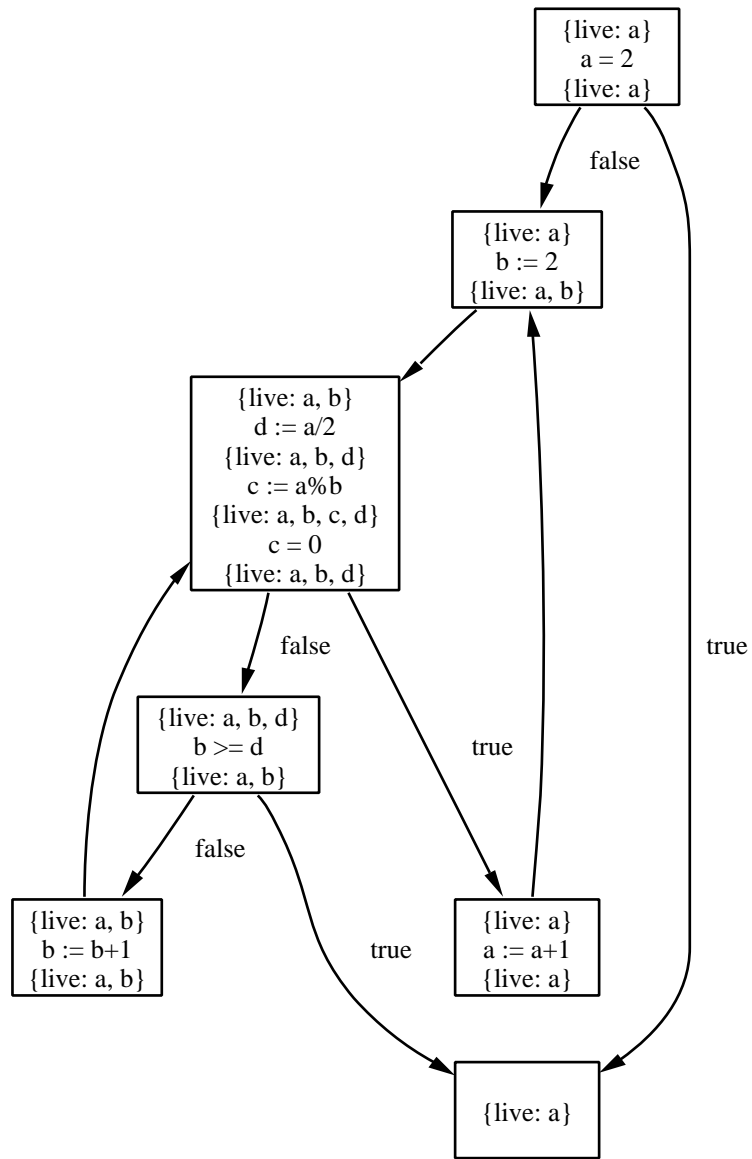
<p>Original code:</p> <pre>a := b + c z := a ** 2 x := 0 * b y := b + c w := y * y u := x + 3 v := u + w</pre>	<p>(c) Result of copy propagation</p> <pre>a := b + c z := a * a x := 0 y := a w := a * a u := x + 3 v := u + w</pre>
<p>(a) Result of algebraic simplification</p> <pre>a := b + c z := a * a x := 0 y := b + c w := y * y u := x + 3 v := u + w</pre>	<p>(d) Result of constant folding/propagation</p> <pre>a := b + c z := a * a x := 0 y := a w := a * a u := 3 v := 3 + w</pre>
<p>(b) Result of common sub-expression elimination</p> <pre>a := b + c z := a * a x := 0 y := a w := y * y u := x + 3 v := u + w</pre>	<p>(e) Result of dead code elimination</p> <pre>a := b + c z := a * a w := a * a v := 3 + w</pre>

Notice that when we're done with (e), the code still will calculate  $a*a$  twice. We can apply another round of common sub-expression elimination, copy propagation, and dead code elimination to remove  $w$  completely from the basic block. The result is

```
a := b + c
z := a * a
v := 3 + z
```

This is a general feature of this style of optimization: In general these optimizations have to be repeated multiple times for optimal effect.

2. (a) Here is the control-flow graph annotated with live variable sets.



(b) See (a)

(c) Assuming  $a$  is a natural number greater than 1, this program computes the smallest prime that is greater than or equal to  $a$ .