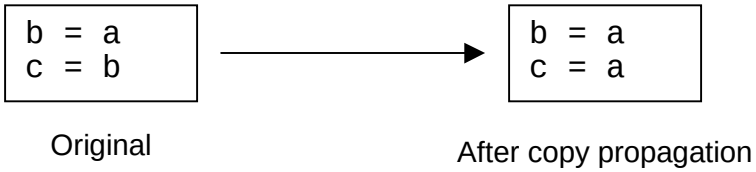
N. Optimization: Copy Propagation

Copy propagation is a transformation that replaces a use of a copy of a variable with a use of the variable itself. For example:



(a) By itself, copy propagation does not decrease the size of the code or the number of instructions to be executed. However, copy propagation may enable other optimizations which do.

Name another optimization which may be enabled by copy propagation.

(b) Show the result of applying this optimization to the code above "After copy propagation".

(c) Copy propagation may be performed as a local optimization. Apply copy propagation to the basic block below.

```
t0 = arg0 * 4
t1 = t0 + BASE
x = load(t1)
t2 = x – 1
y = x
z = t2 * y
z = z * y
x = x + 1
z = z * 2
a = y
```

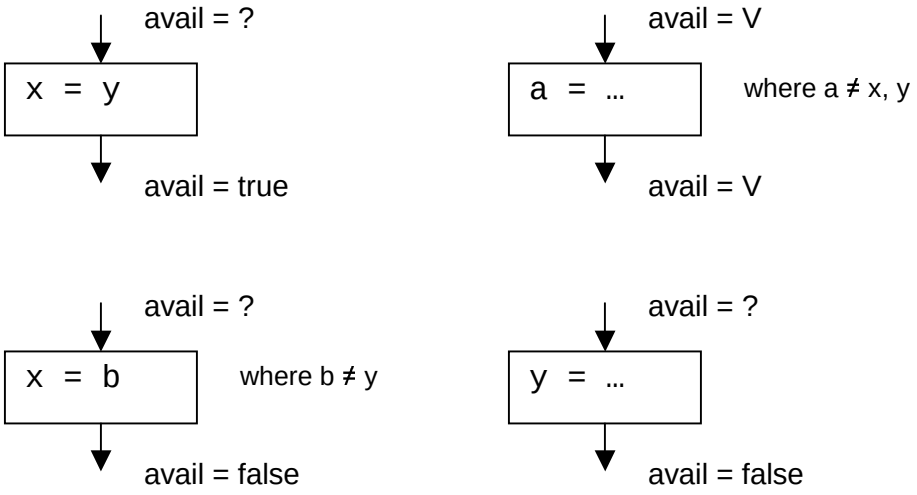
(d) Copy propagation may also be applied as a global optimization using a data-flow analysis called *available copies*. We say that a copy statement x = y is *available* at a program point P if all paths from entry to P have these properties:

(i) x = y is on the path
(ii) x is not modified between x = y and P
(iii) y is not modified between x = y and P

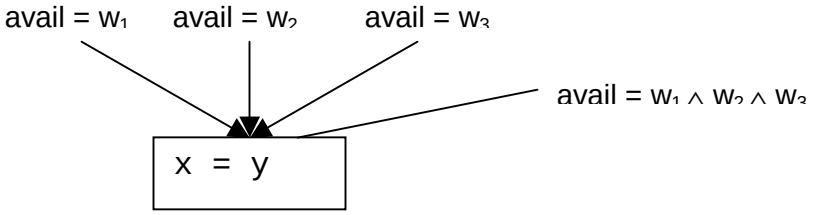
Our analysis for available copies of x = y will yield either **true** (indicating that x = y is available) or **false** at every program point.

Keeping in mind that we need to do a conservative analysis, what does **false** mean? (One sentence at most.)

(e) Copy propagation is a *forward* problem. The **transfer function** for available copies of x = y is defined by these rules:



The **merge function** is **and**, as shown in this diagram:



Using these rules, perform the available copies analysis for b = a on the following CFG. Show the data flow facts before and after **every** statement.