```
define i32 @example(i32 %n) {
entry:
  %y = alloca i32 ① allocates stack space, %y points to this storage
  @dbg.declare(i32* %y, "y" l3) ② source var y is stored at %y
  store i32 0, i32* %y, l3
     ③ stores constant (0) for source var y

for.body:
  %3 = load i32, i32* %x, l5
  %add = add i32 %3, 4, l5
  %4 = load i32, i32* %n.addr, l5
  %add1 = add i32 %add, %4, l5
  %5 = load i32, i32* %y, l5
  %add2 = add i32 %5, %add1, l5
  store i32 %add2, i32* %y, l5
     ④ stores %add2 for source var y
```

At source line 3:
`y = 0`

At source line 5:
```
y = (Add 4 (Add
  (Mul 2 n) n))
```

Unoptimised LLVM IR (O0)

```
define i32 @example(i32 %n) {
entry:
  @dbg.value(i32 0, "y" l3)
     ① source var y = constant (0)
for.cond.cleanup.loopexit:
  %0 = add i32 %n, -1, l4
  %add = add i32 %n, 4
  %mul = shl i32 %n, 1, l2
  %add1 = add i32 %add, %mul
  %1 = mul i32 %0, %add1, l4
  %2 = mul i32 %n, 3, l4
  %3 = add i32 %1, %2, l4
  %4 = add i32 %3, 4, l4
     ② should be mapped to y, but debug mapping lost!
  @dbg.value(i32 undef, "y" l3)
     ③ dead debug mapping without an input value
```

At source line 3:
`y = 0`

Value mapping lost, should be:
```
y = %4 = (Add 4
 (Add
  (Mul (Add -1 n)
   (Add 4
    (Add n (Shl n 1))))
  (Mul 3 n)))
```

Optimised LLVM IR (O1)