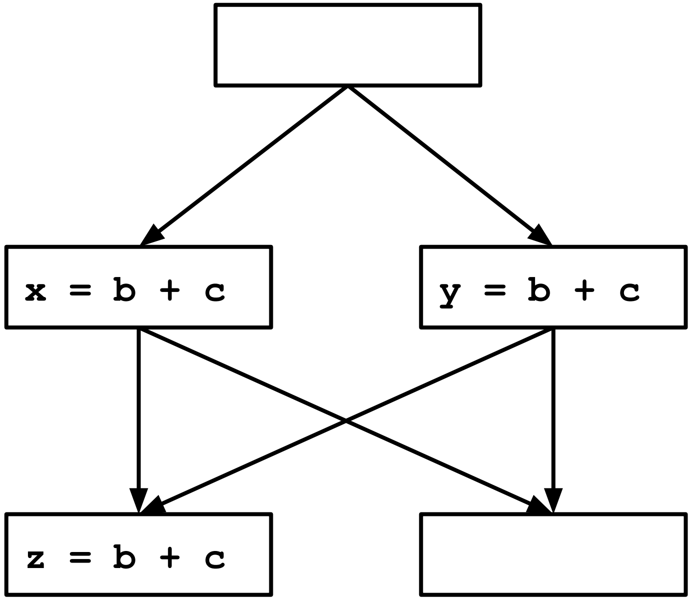
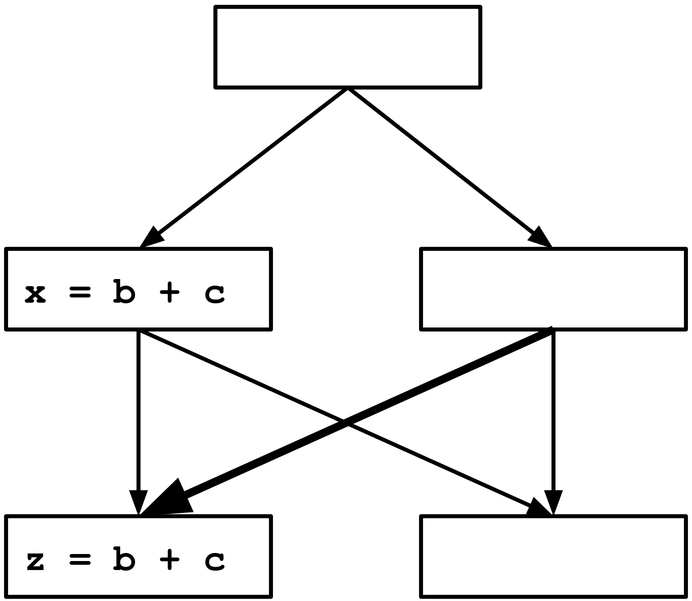
Partial Redundancy Elimination

CS243 Review Session

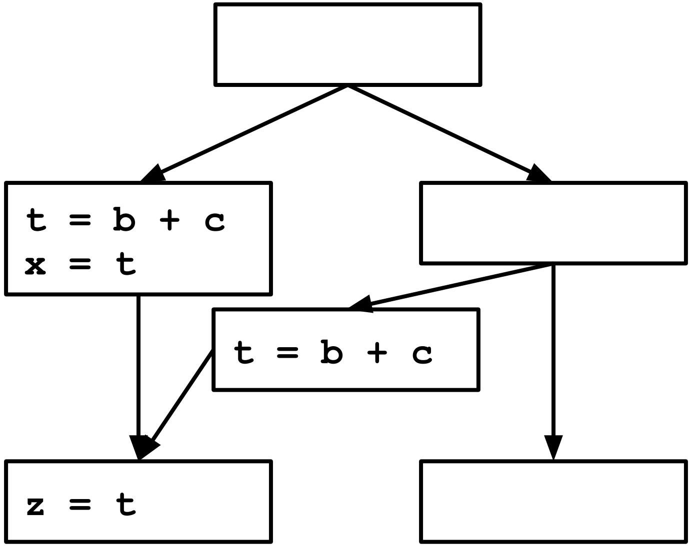
Full Redundancy

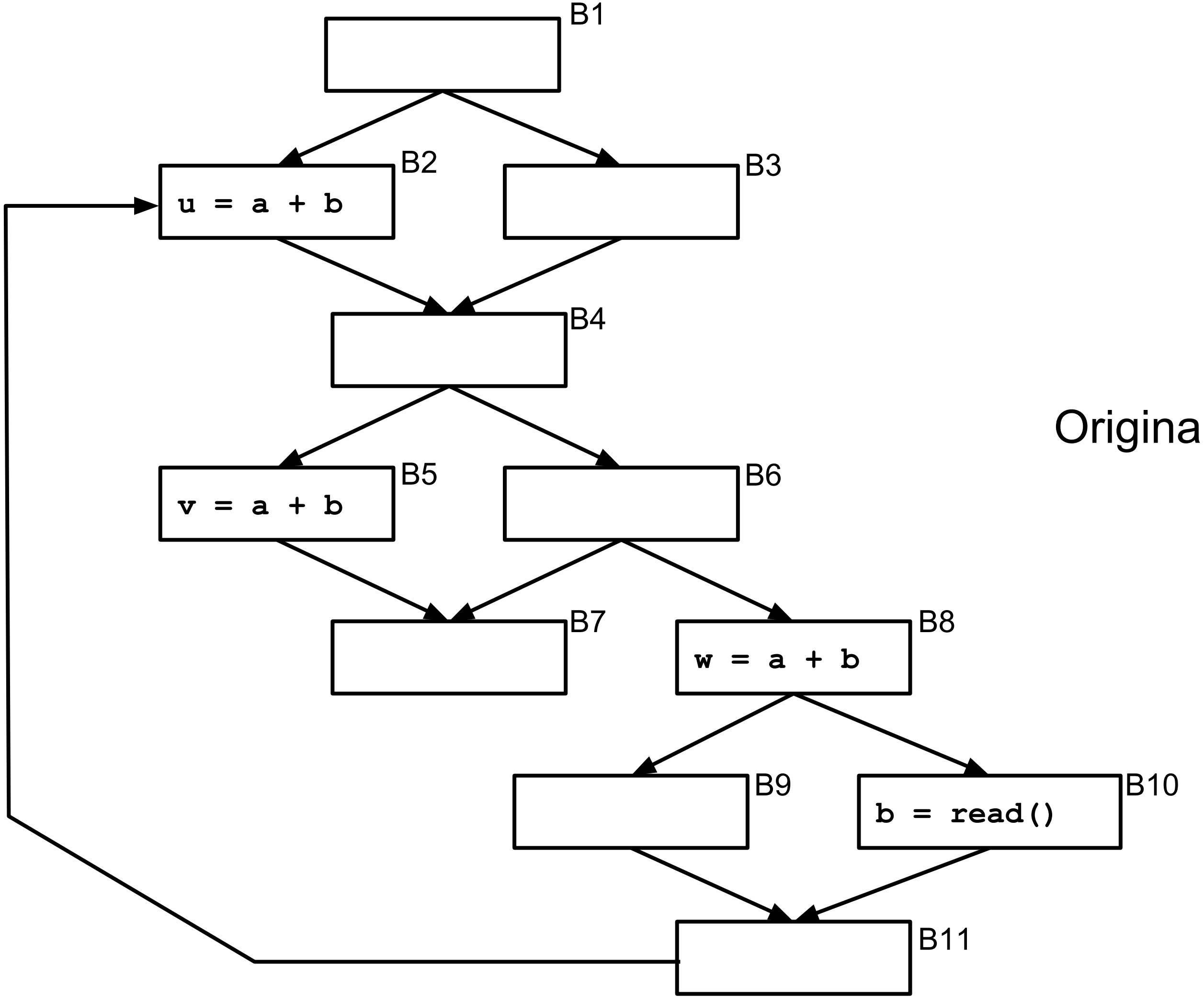


Partial Redundancy

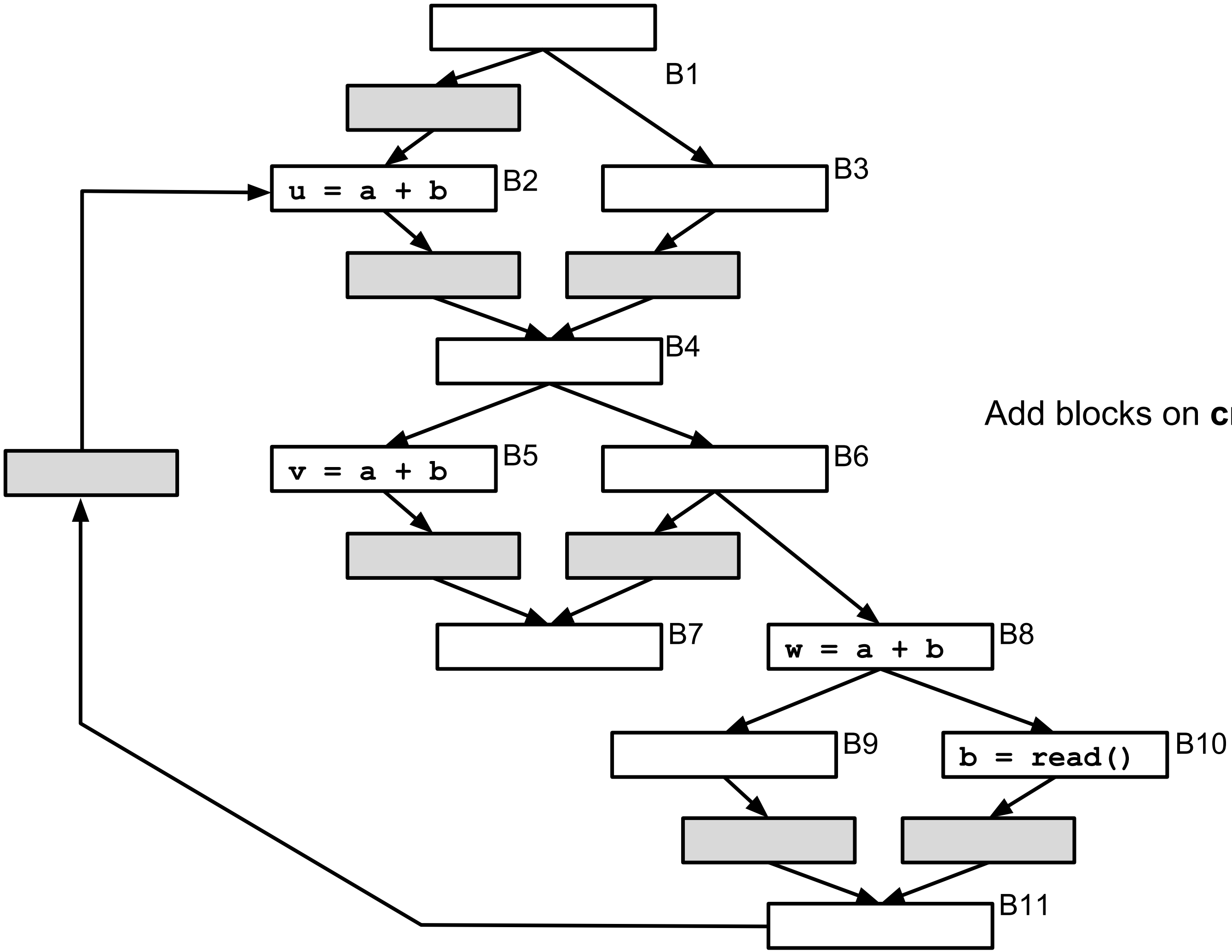


Partial Redundancy

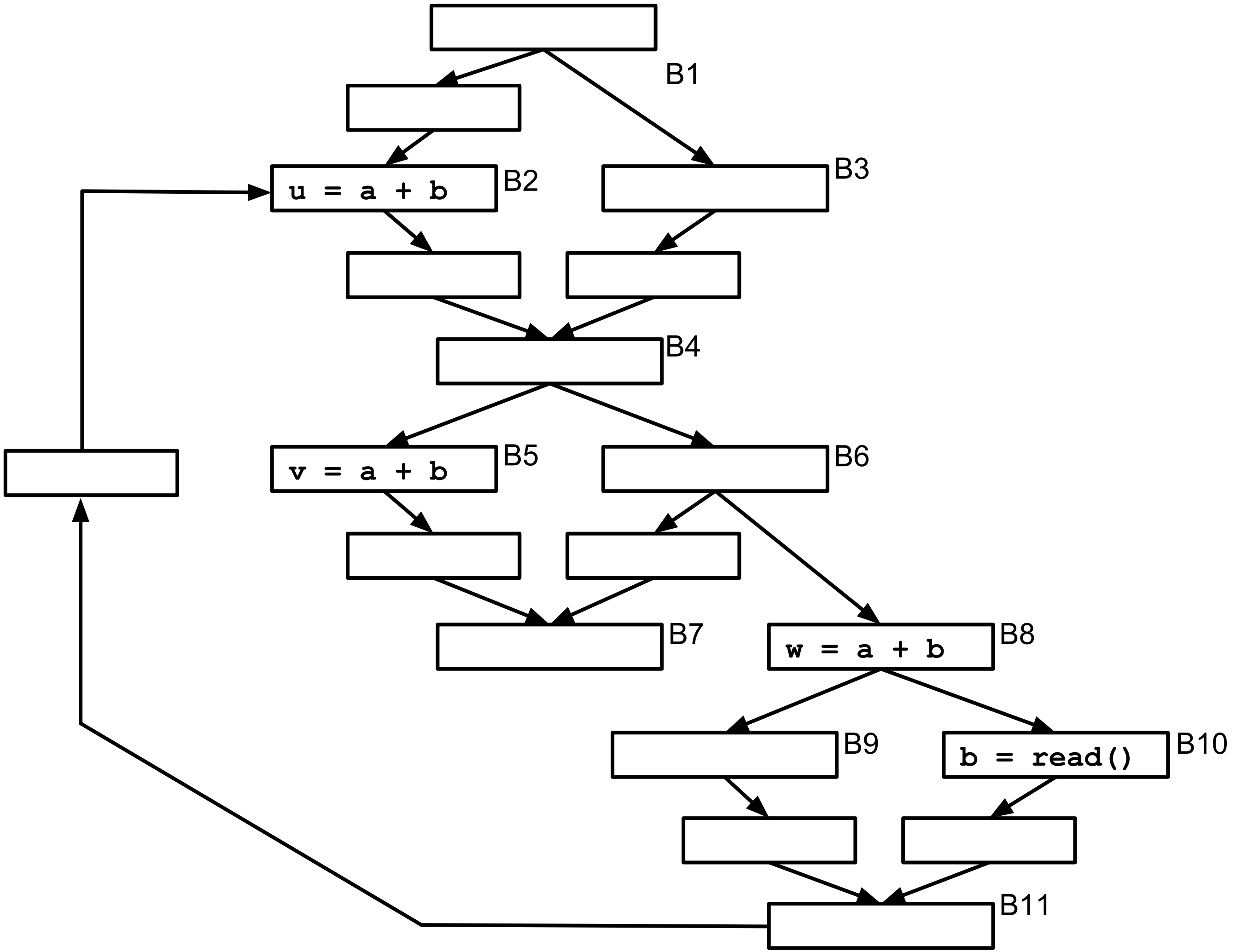




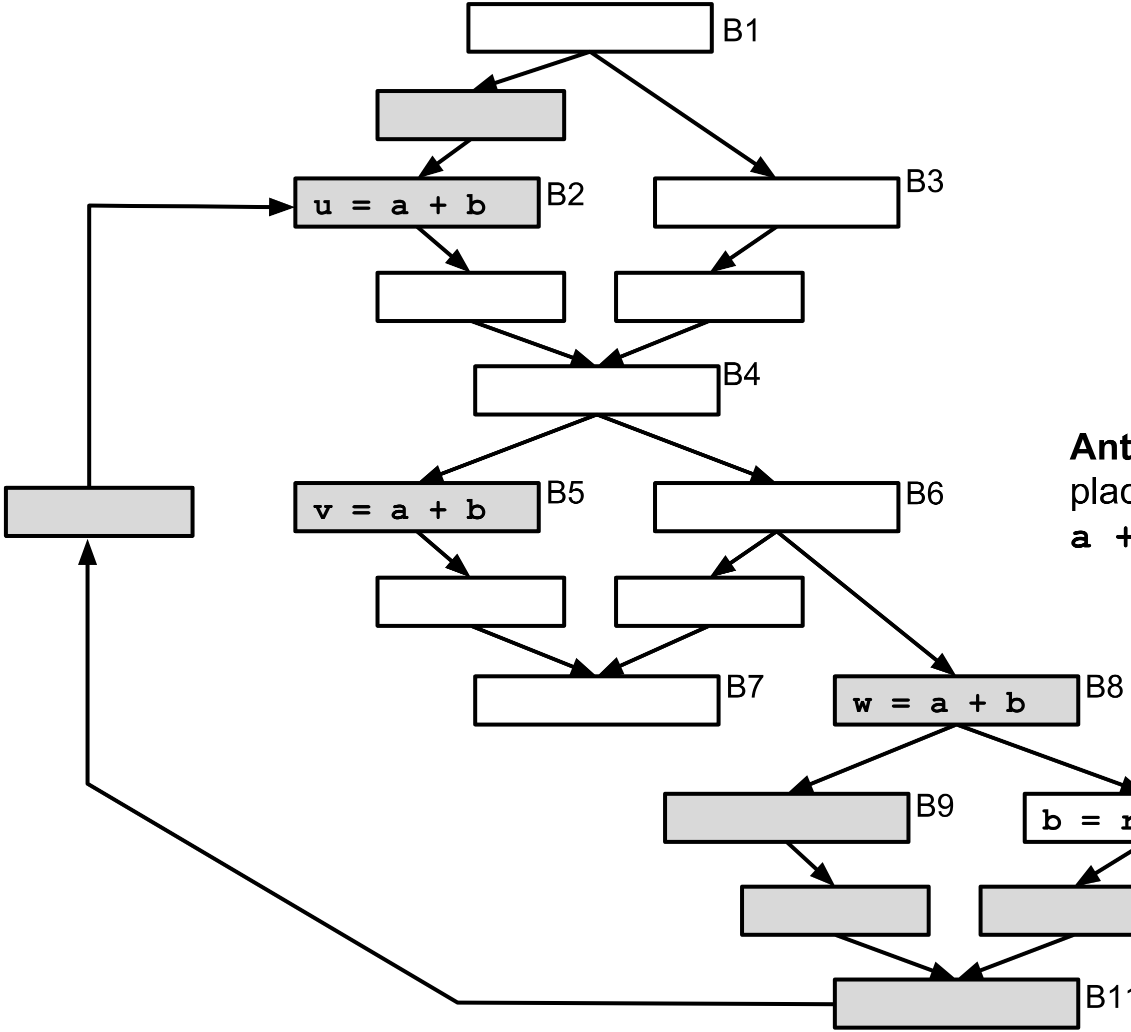
Original graph



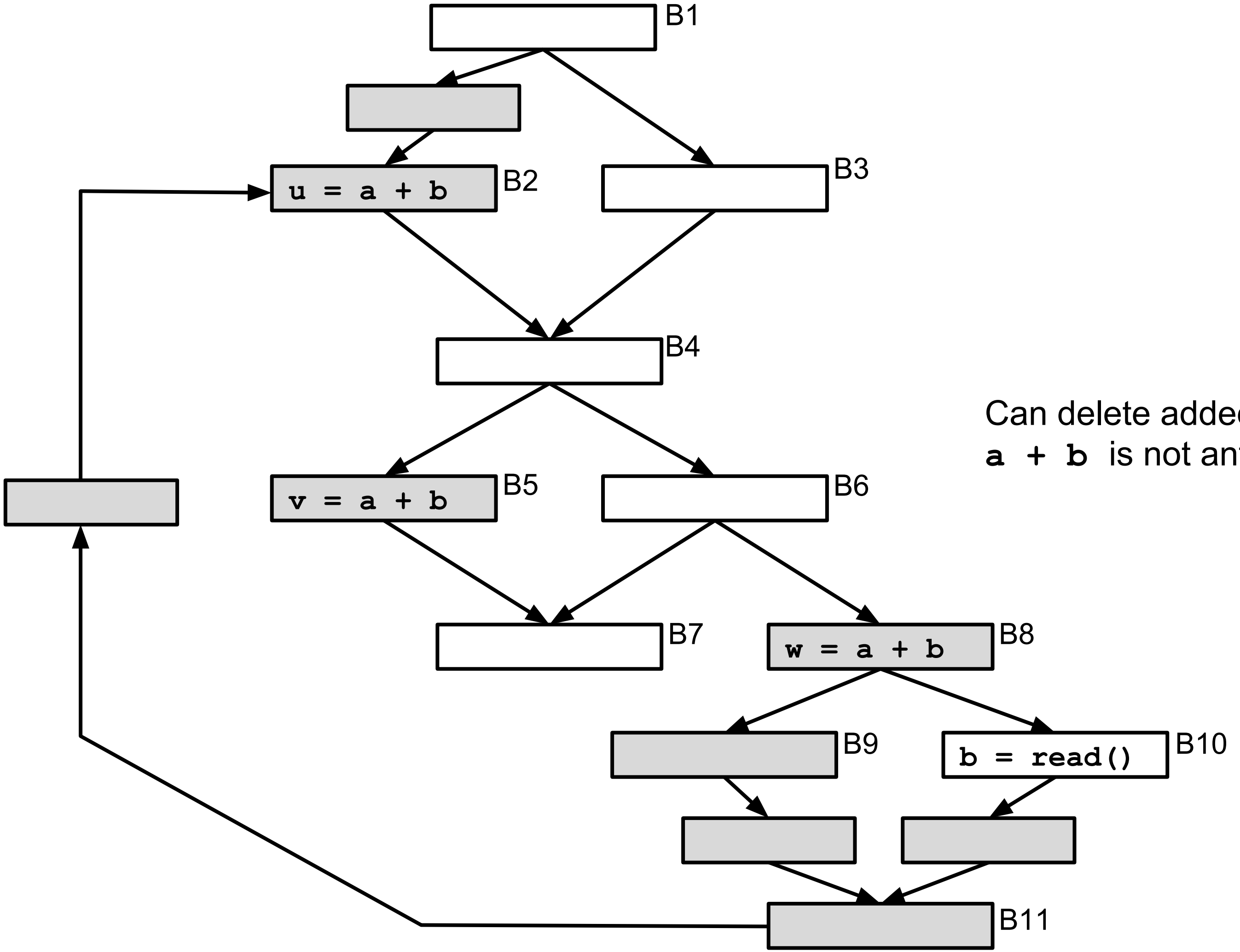
Add blocks on **critical** edges



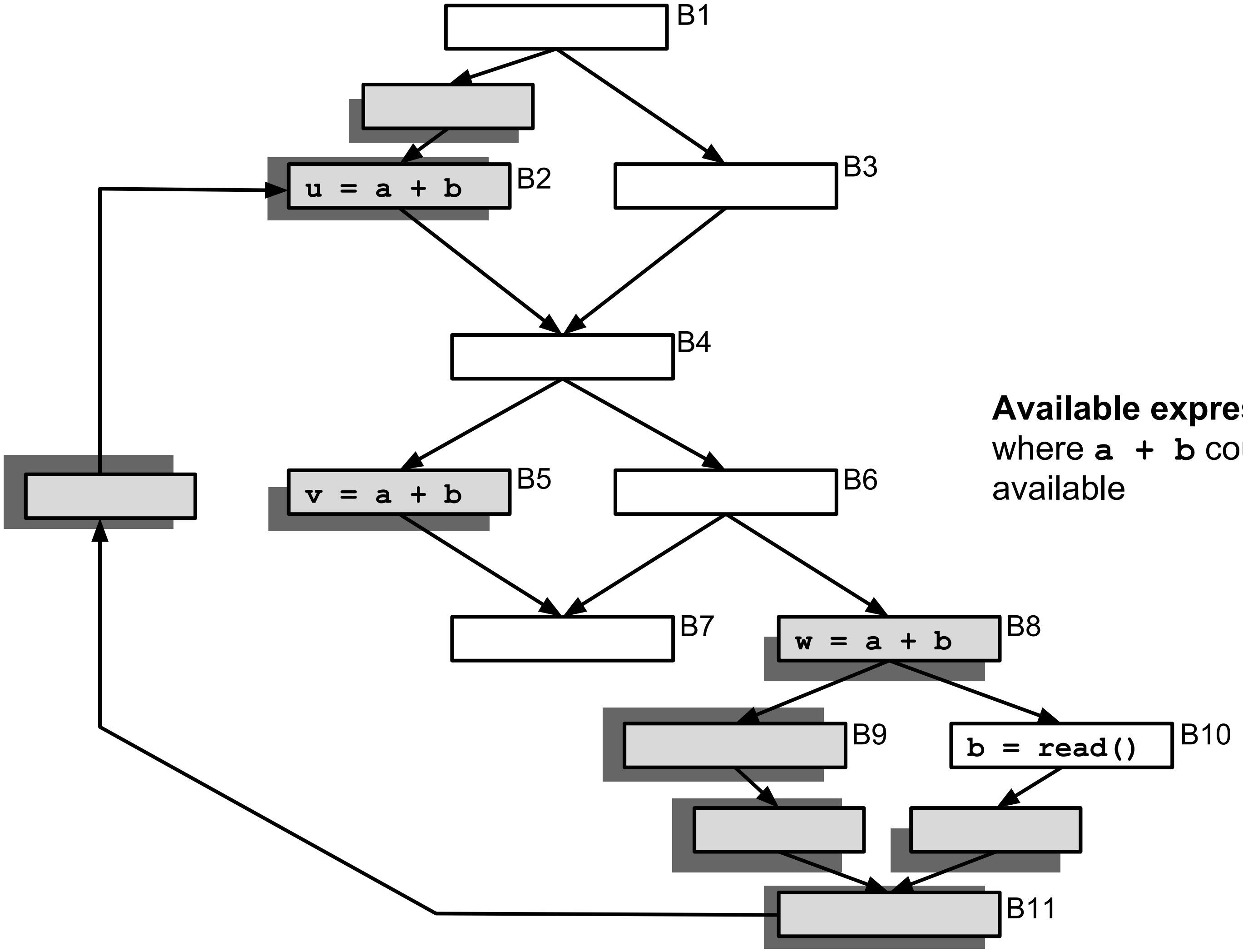
B1
u = a + b
B2
B3
B4
v = a + b
B5
B6
B7
w = a + b
B8
B9
b = read()
B10
B11



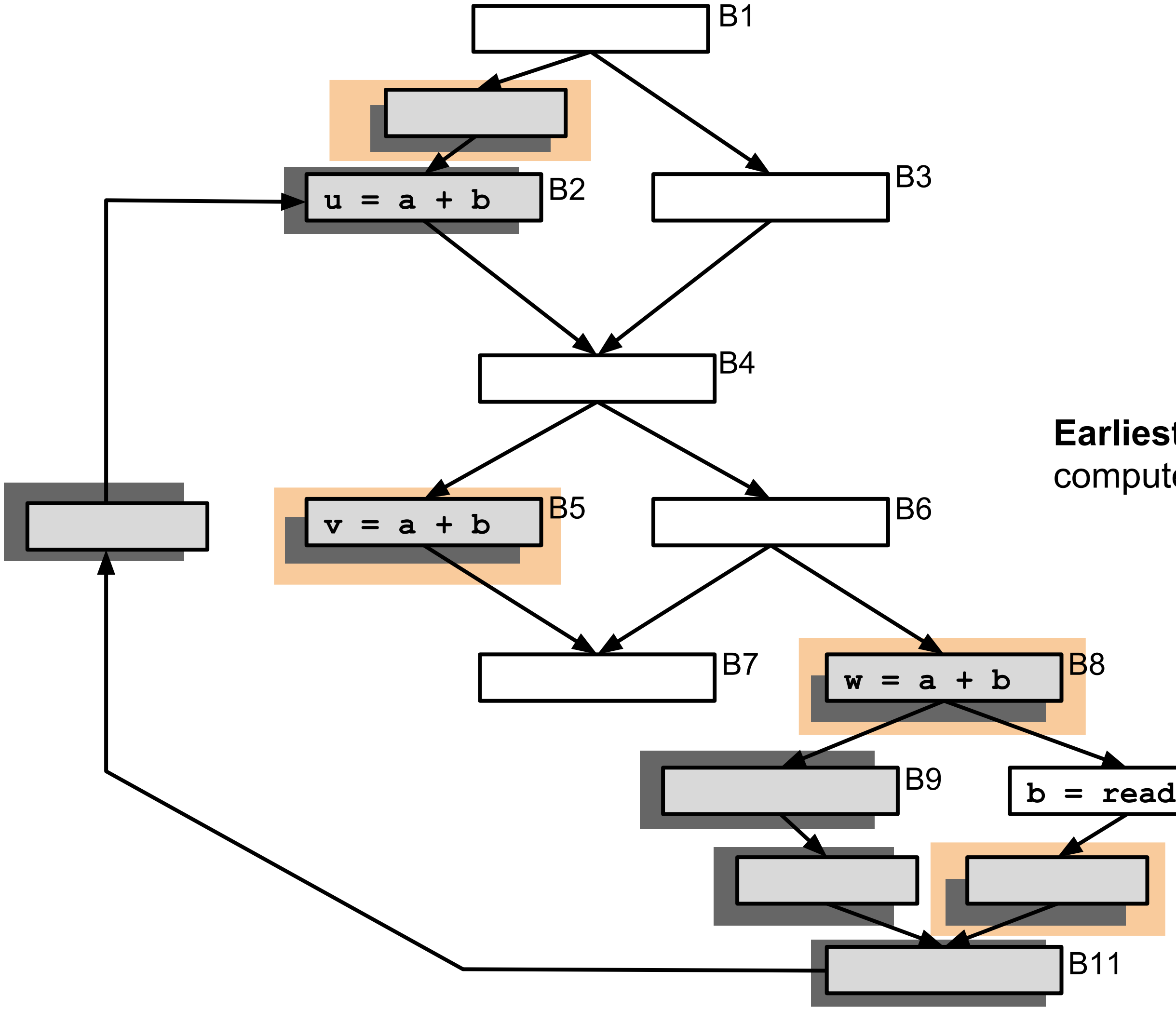
Anticipated expressions:
places where it is **safe** to place
`a + b`



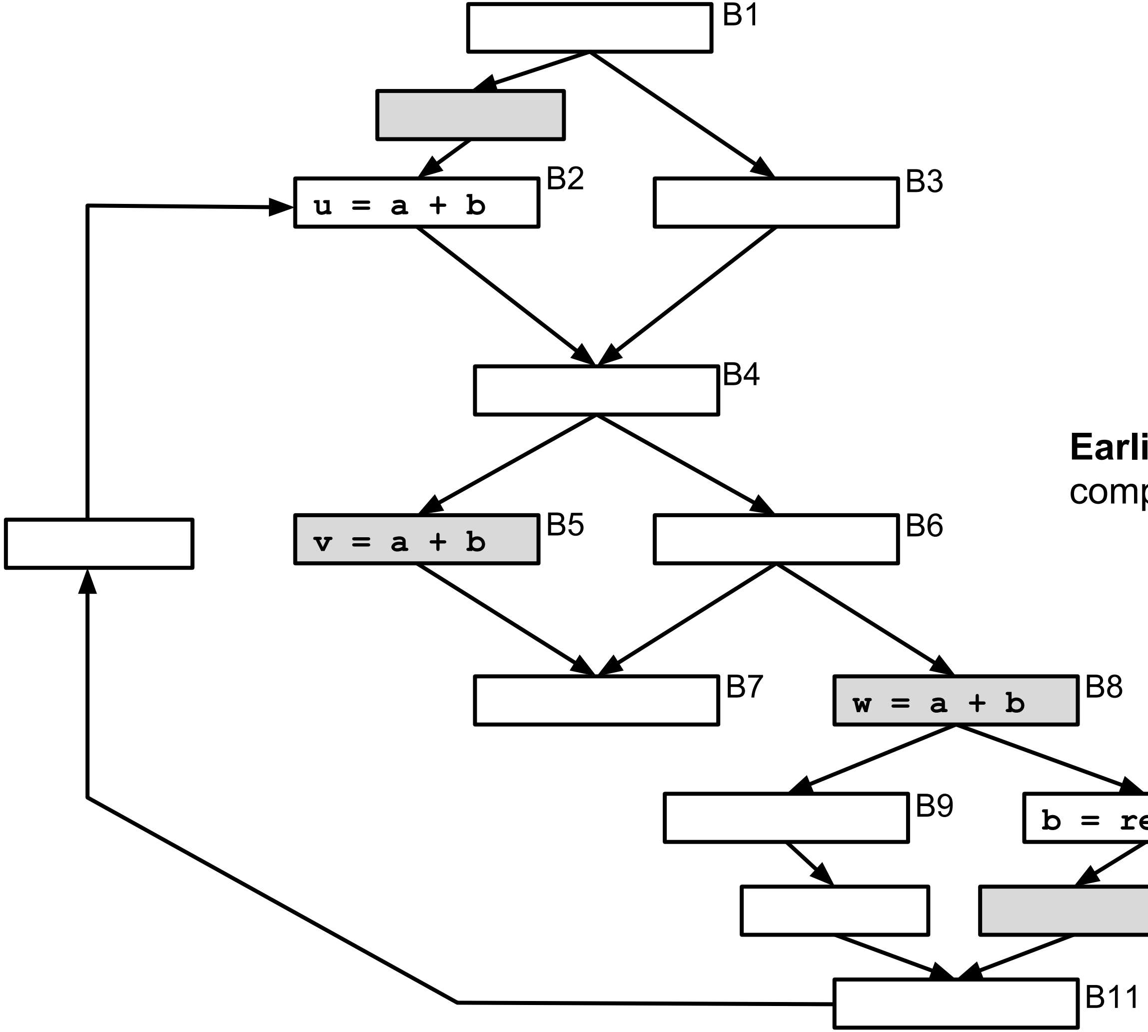
Can delete added blocks where **a + b** is not anticipated



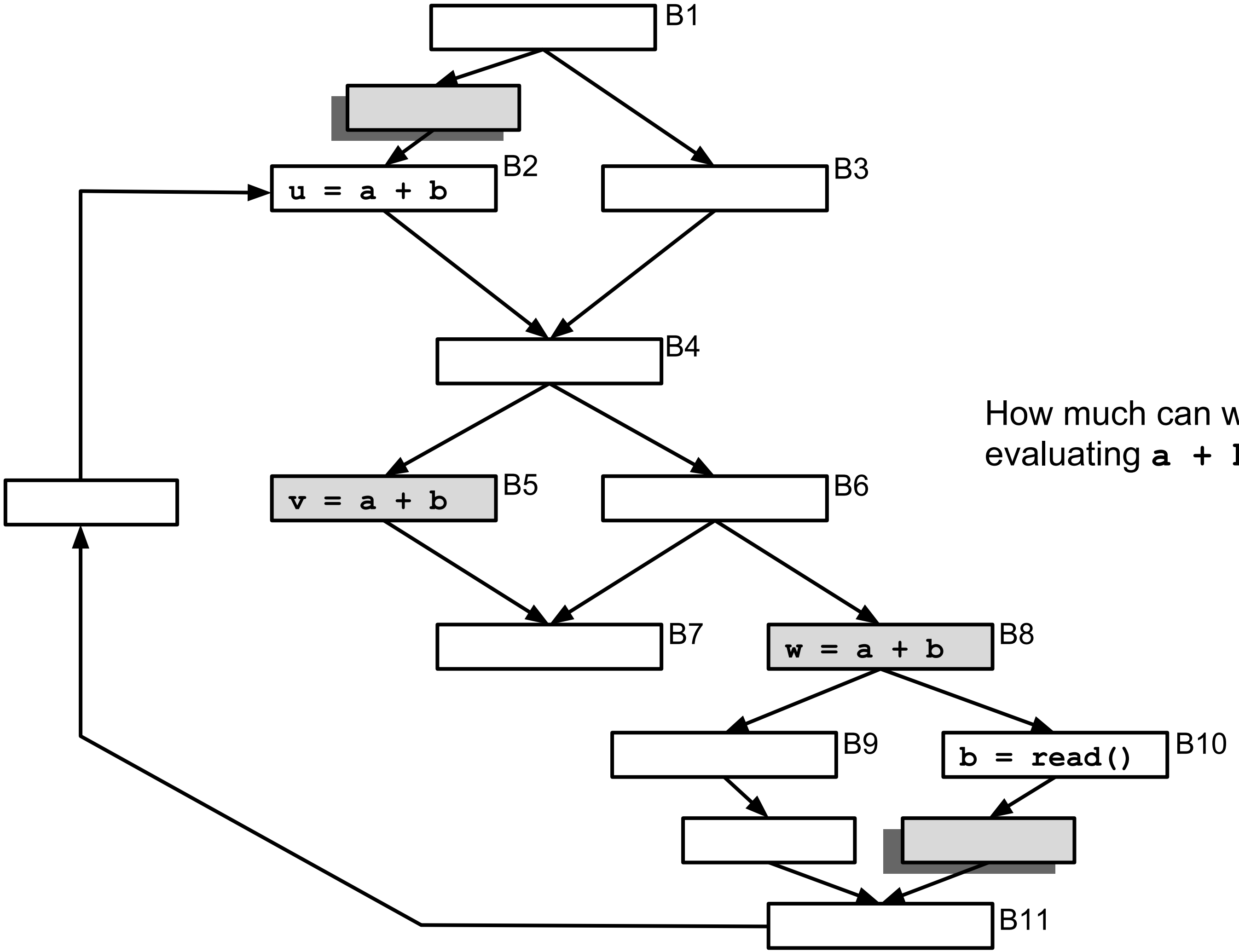
Available expressions: points where `a + b` could be made available



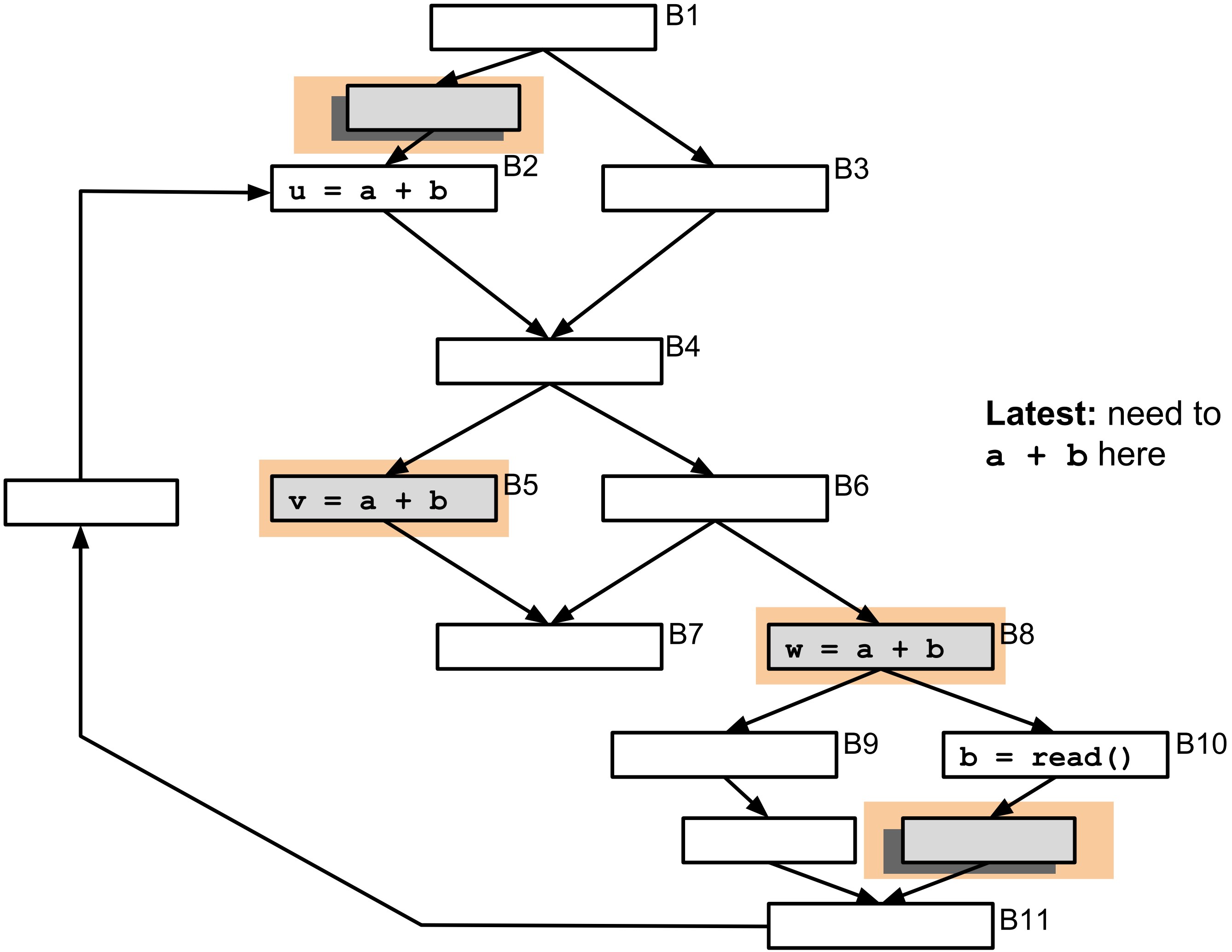
Earliest: when can we earliest compute **a + b**



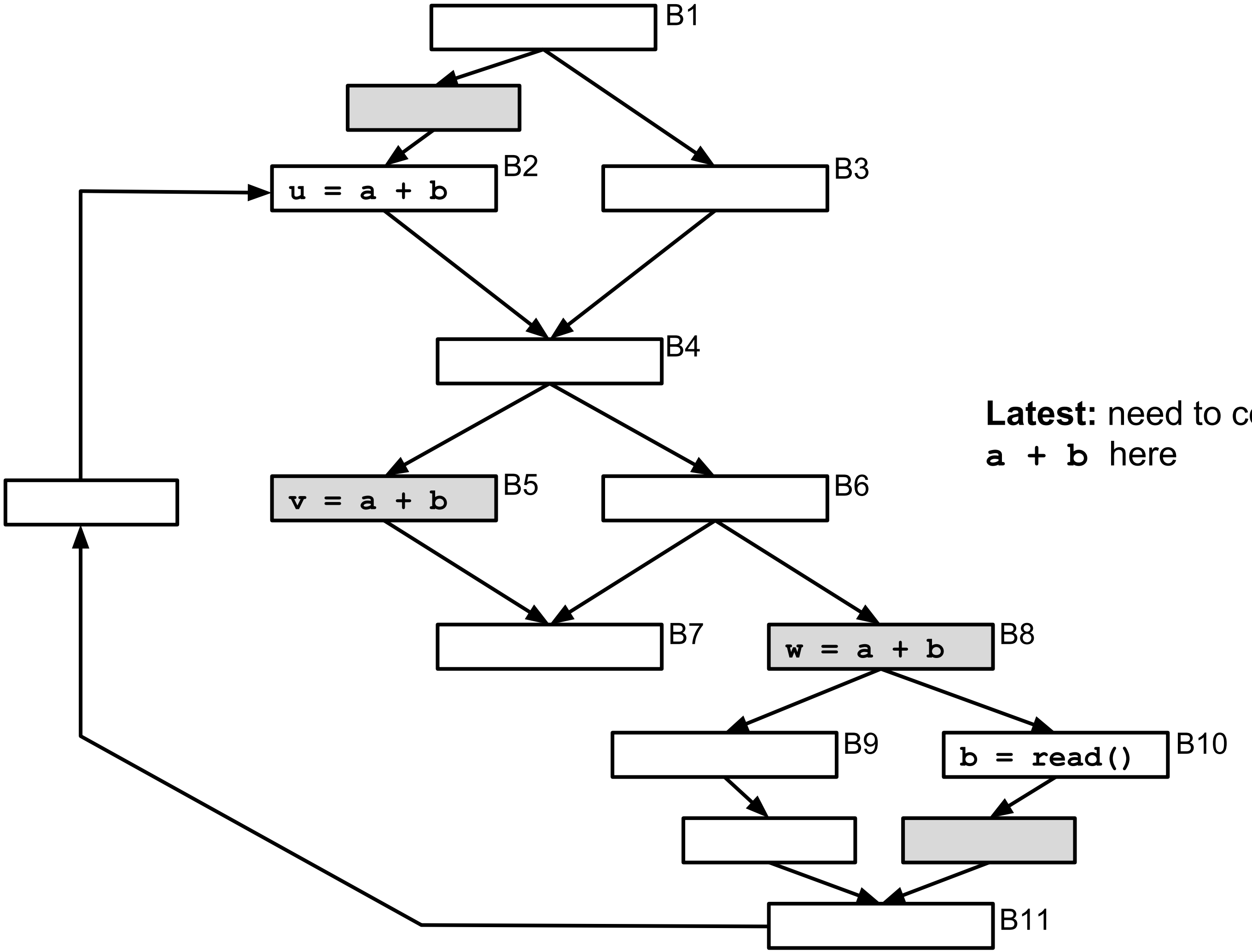
Earliest: when can we earliest compute `a + b`



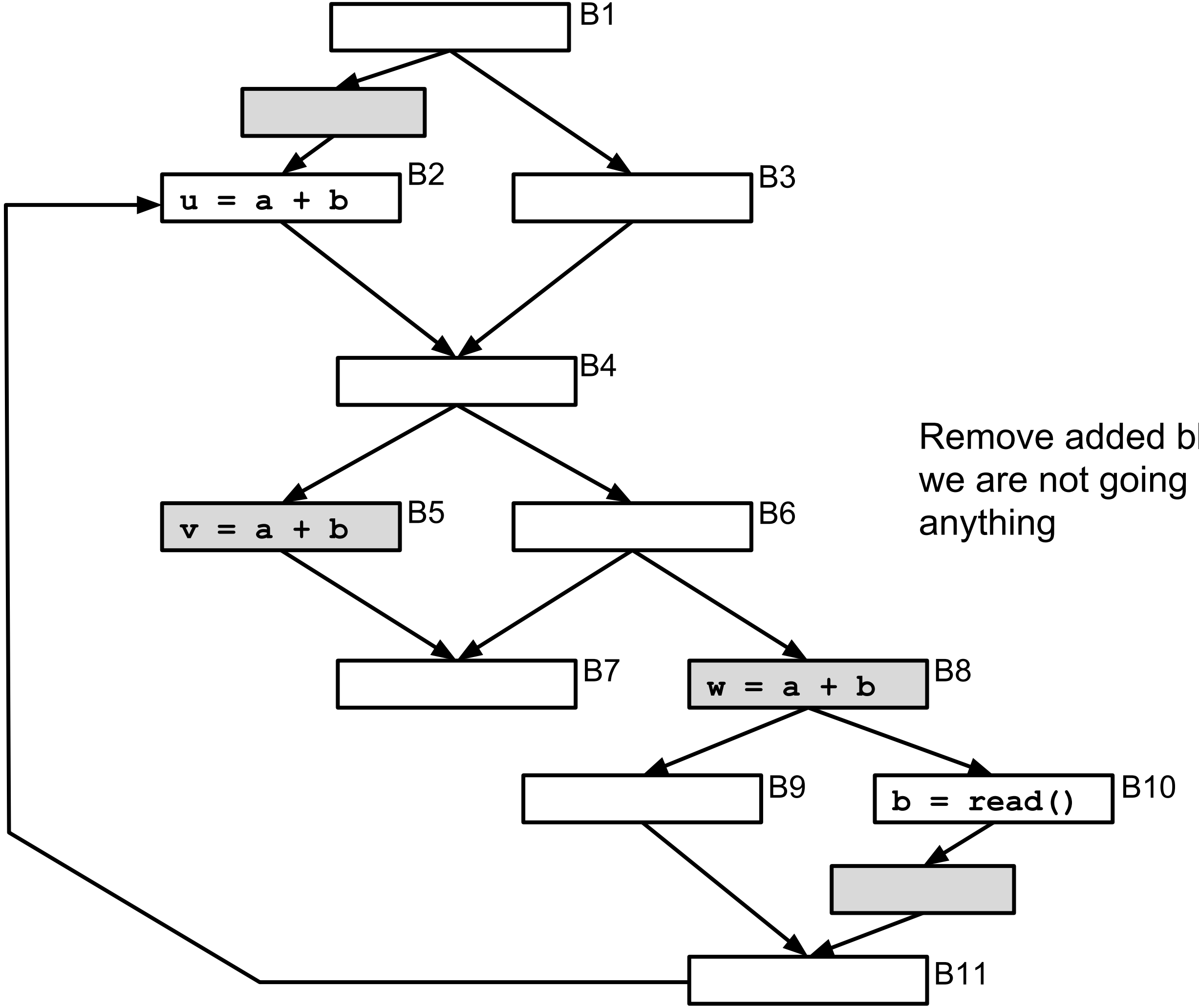
How much can we postpone evaluating **`a + b`**?



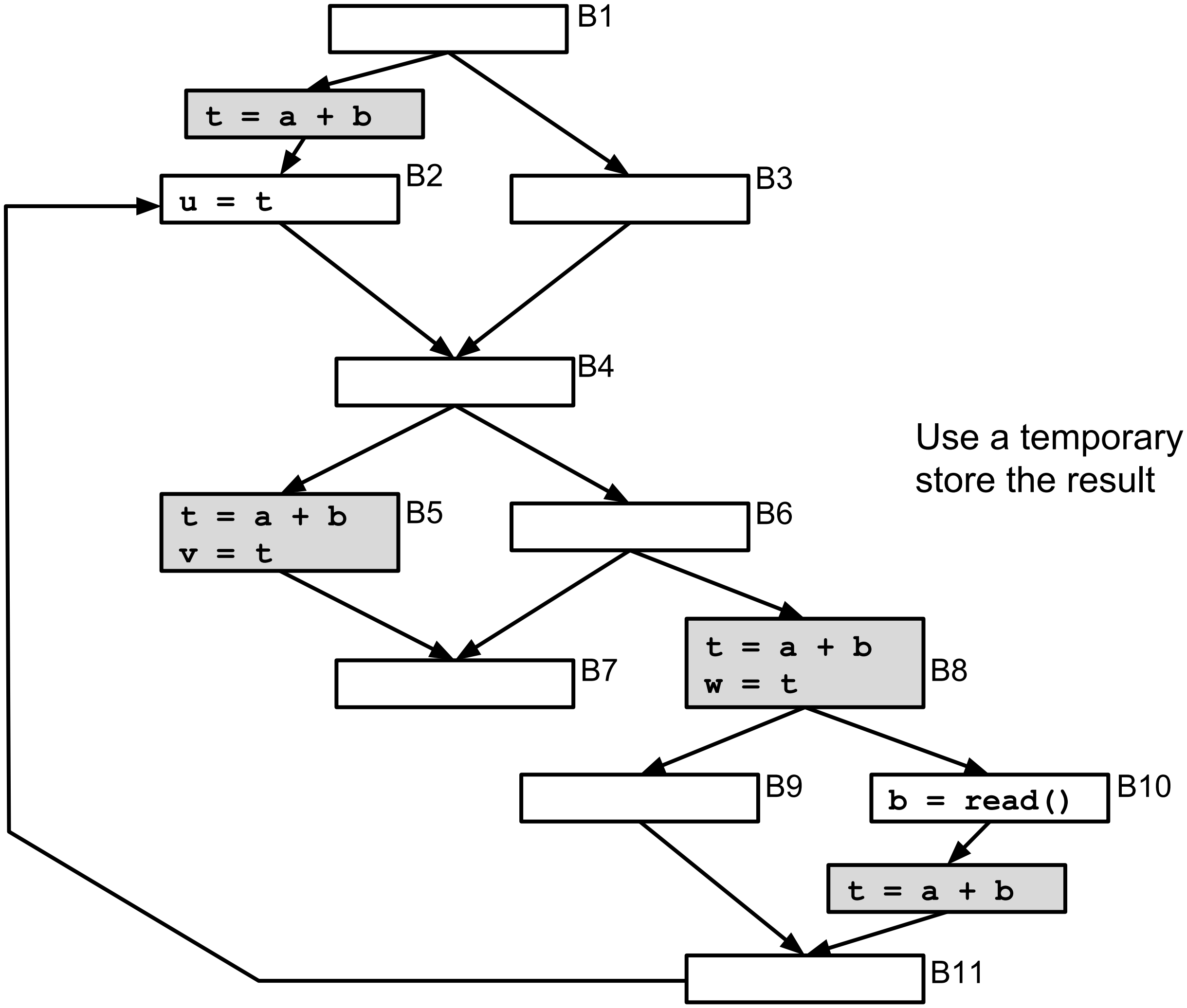
Latest: need to compute `a + b` here



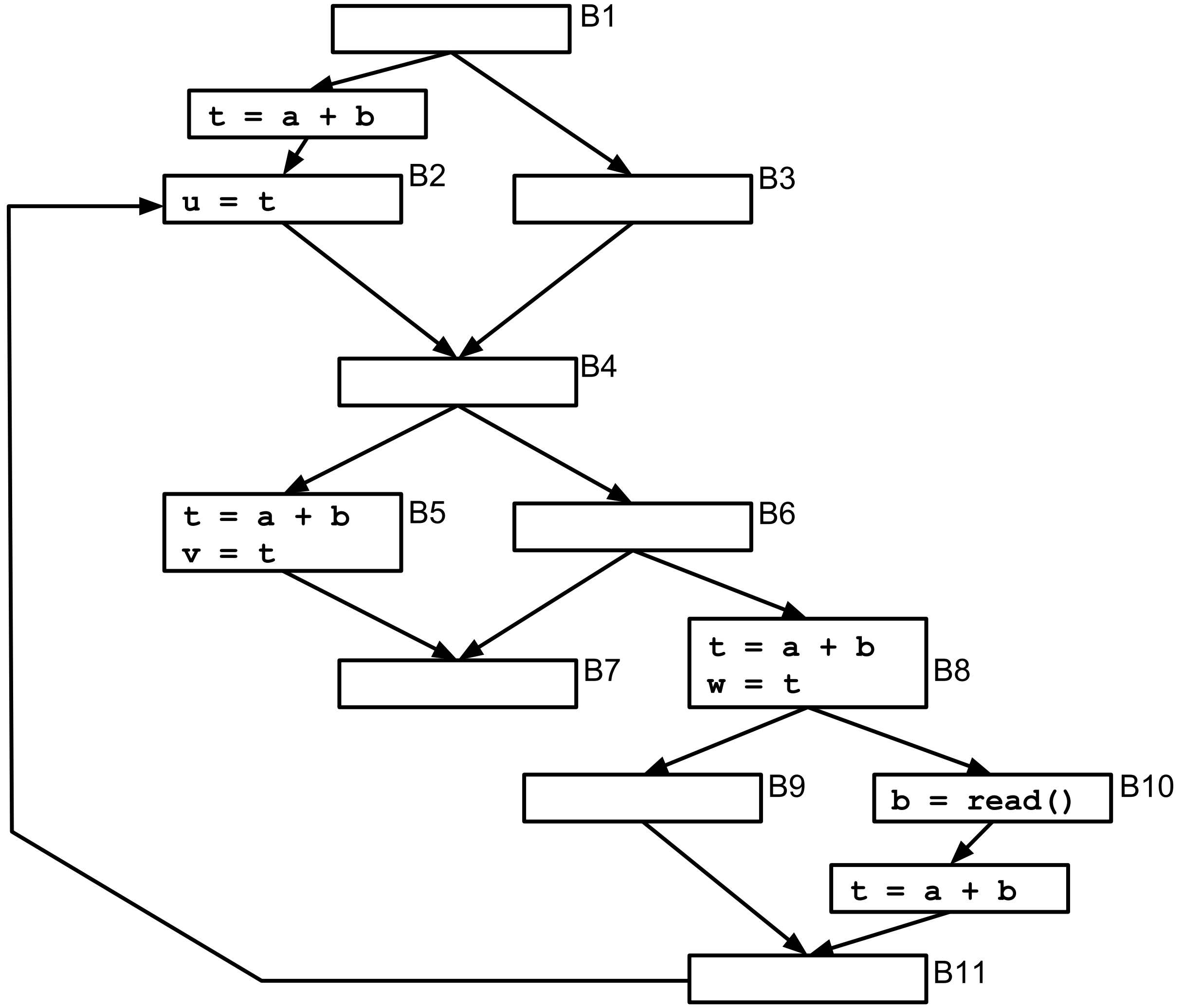
Latest: need to compute
`a + b` here

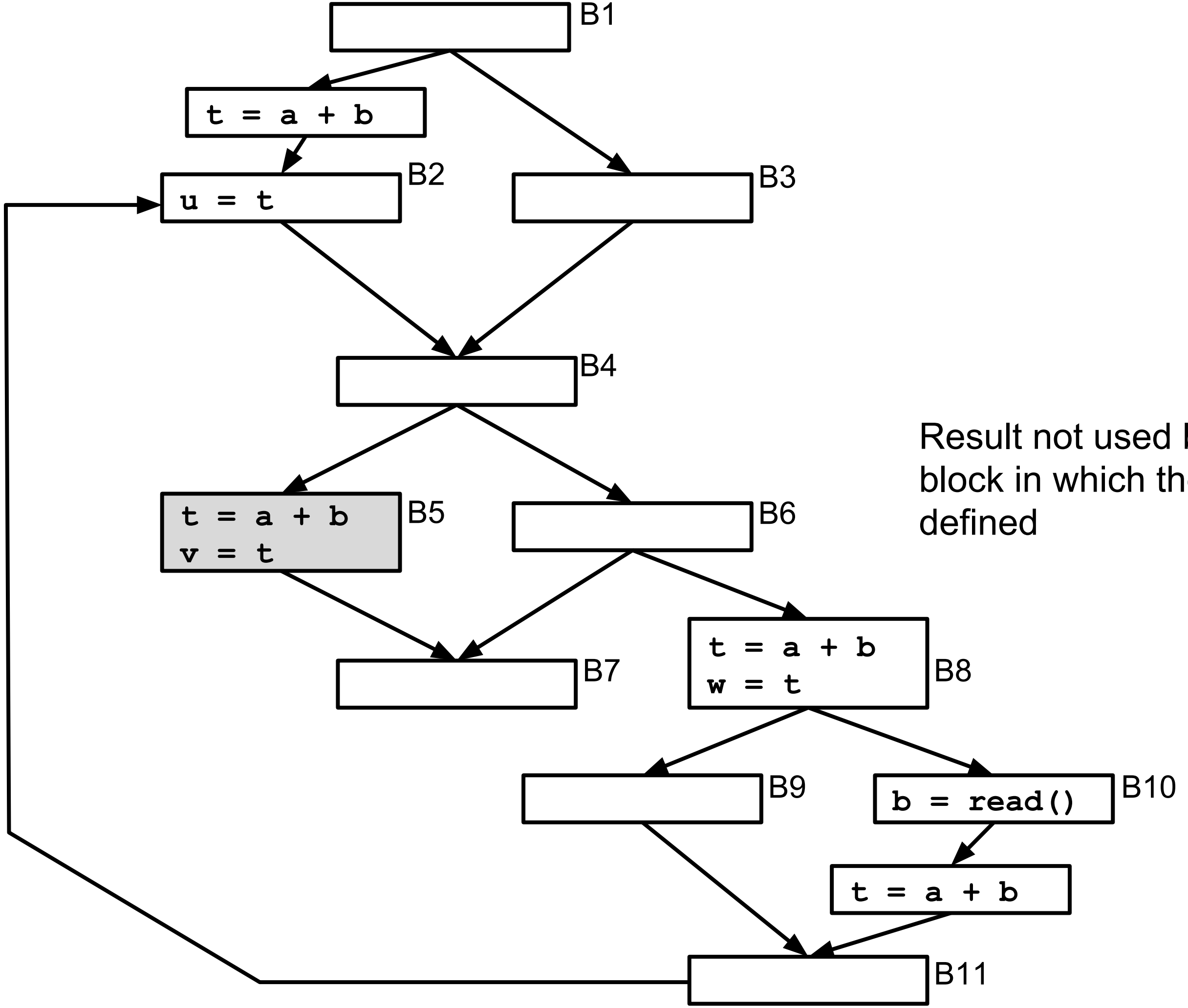


Remove added blocks where we are not going to compute anything

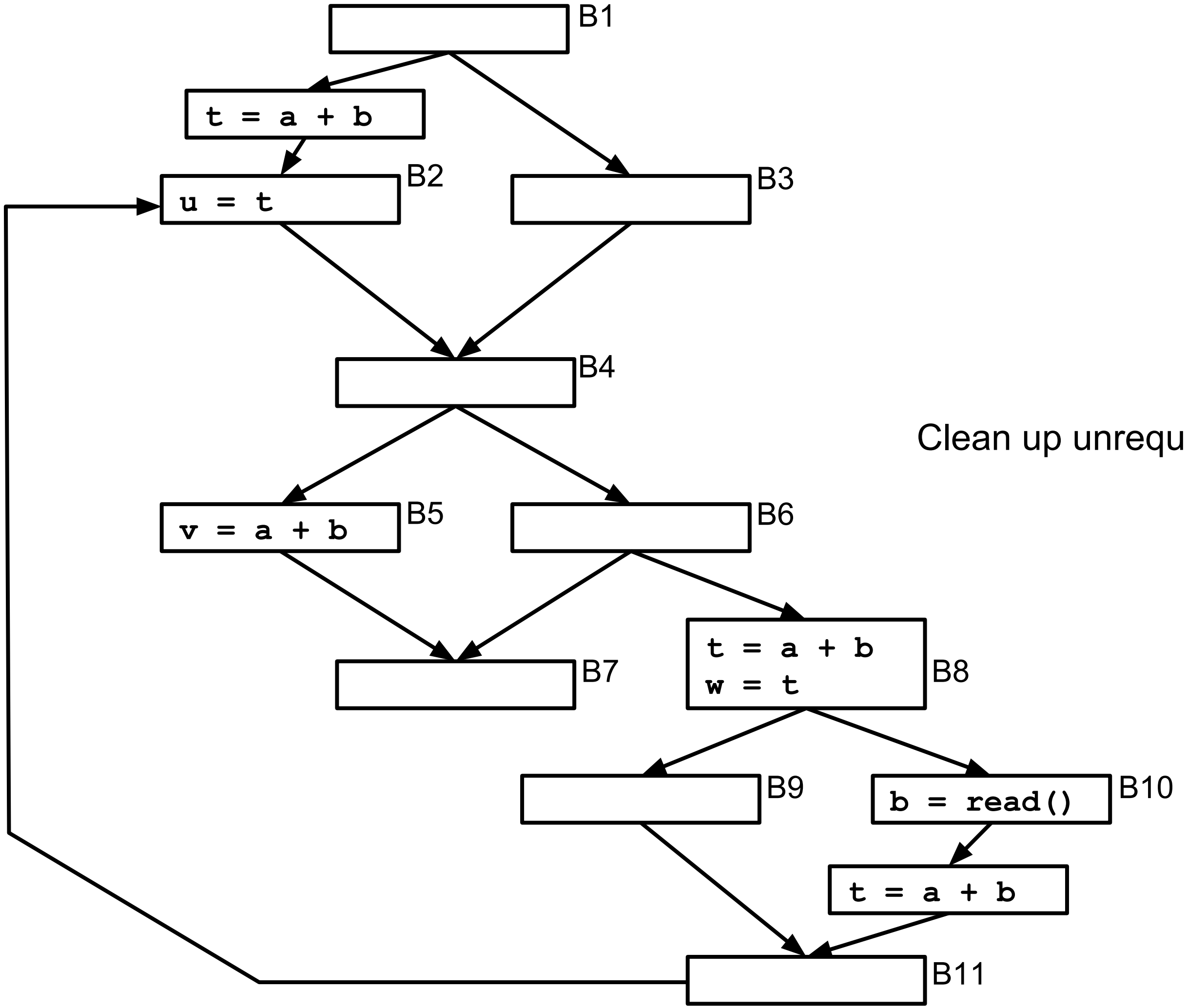


Use a temporary variable to store the result

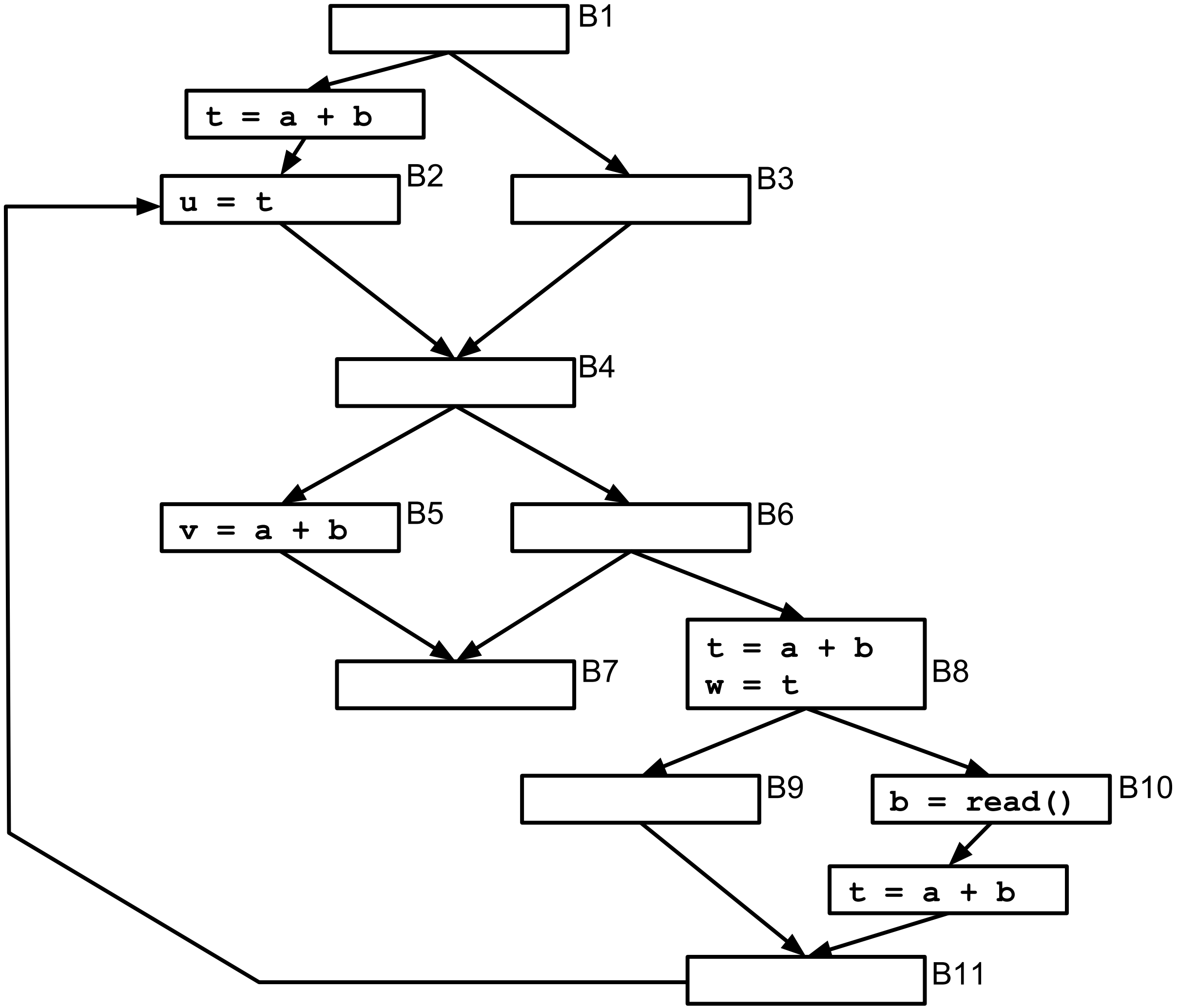

B1
t = a + b
u = t
B2
B3
B4
t = a + b
v = t
B5
B6
B7
t = a + b
w = t
B8
B9
b = read()
B10
t = a + b
B11



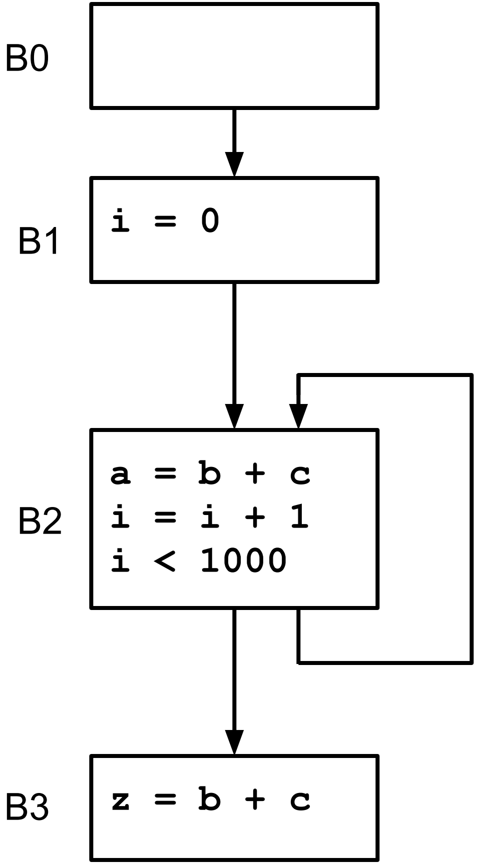
Result not used beyond the block in which the variable is defined

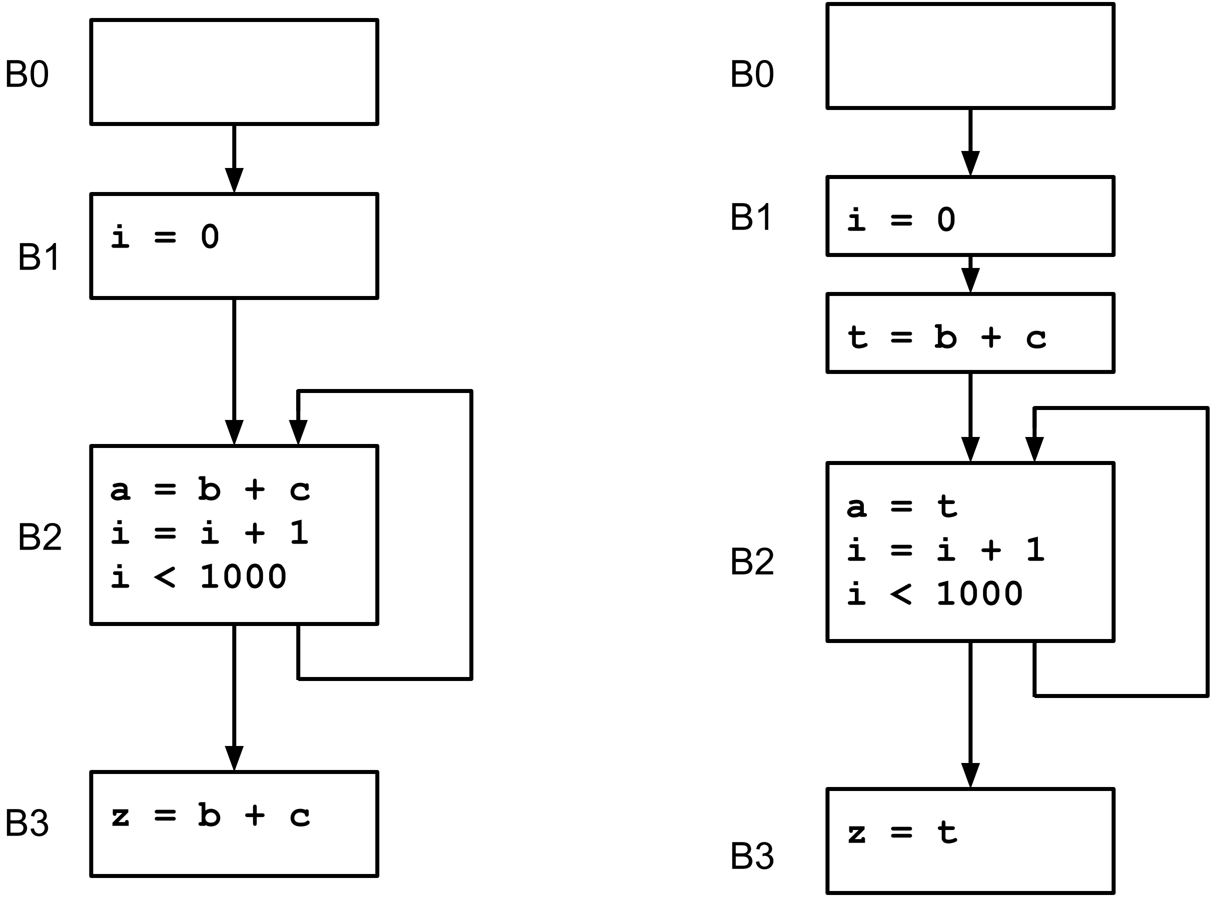


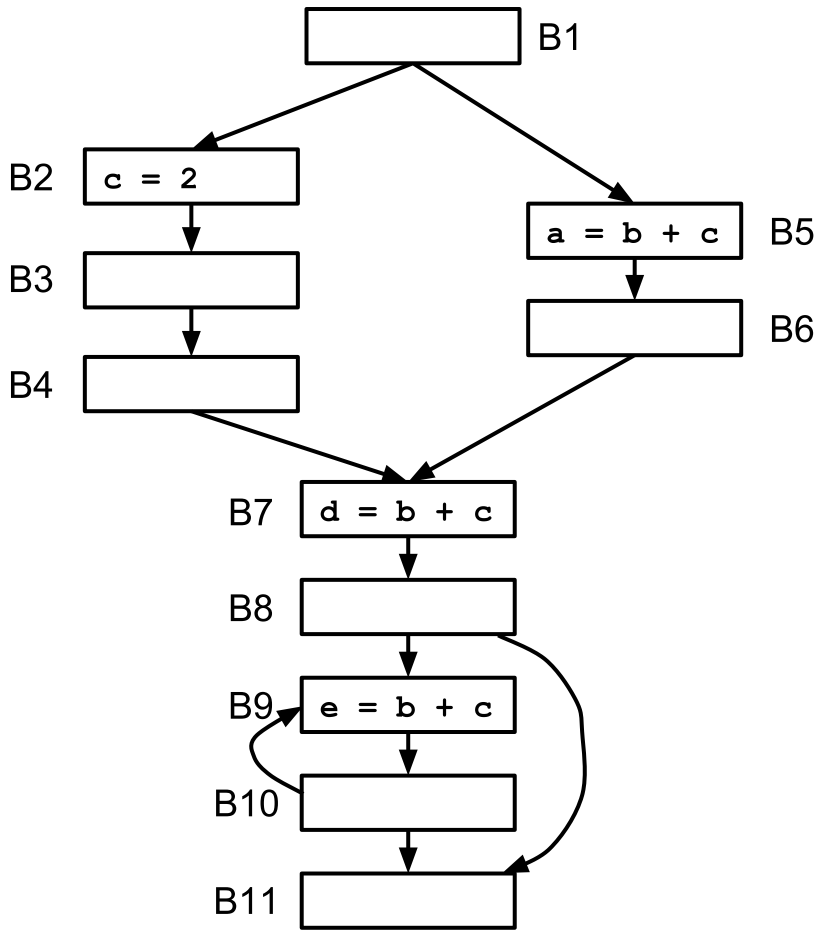
Clean up unrequired temporaries

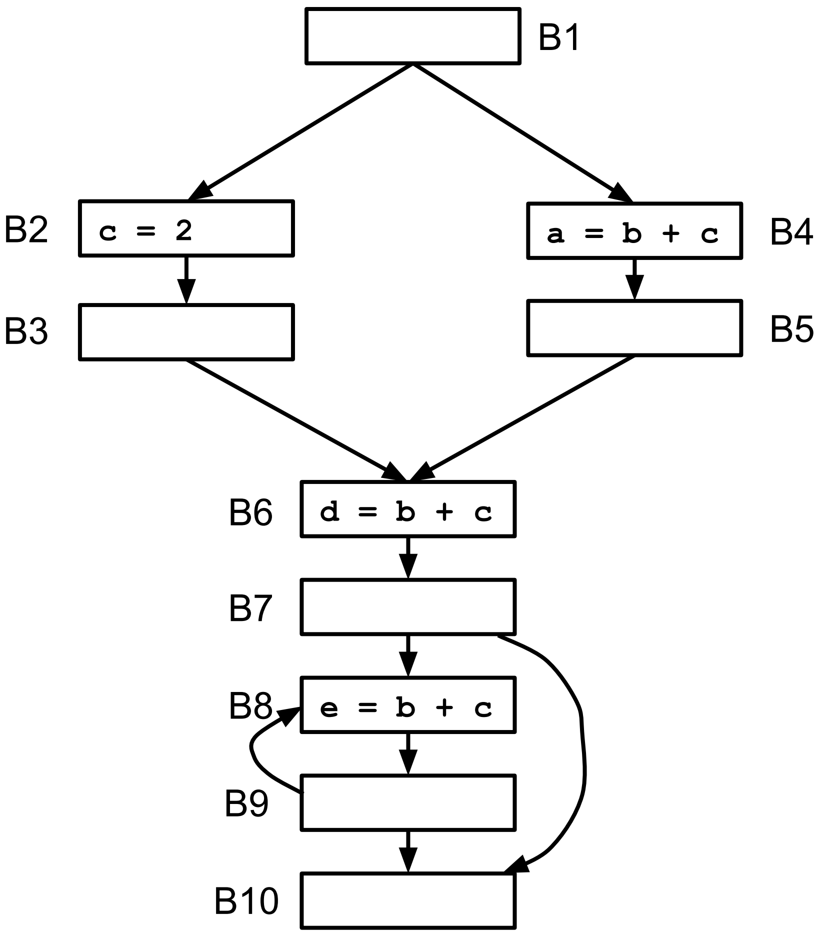

B1
t = a + b
B2
u = t
B3
B4
B5
v = a + b
B6
B7
t = a + b
w = t
B8
B9
B10
b = read()
t = a + b
B11

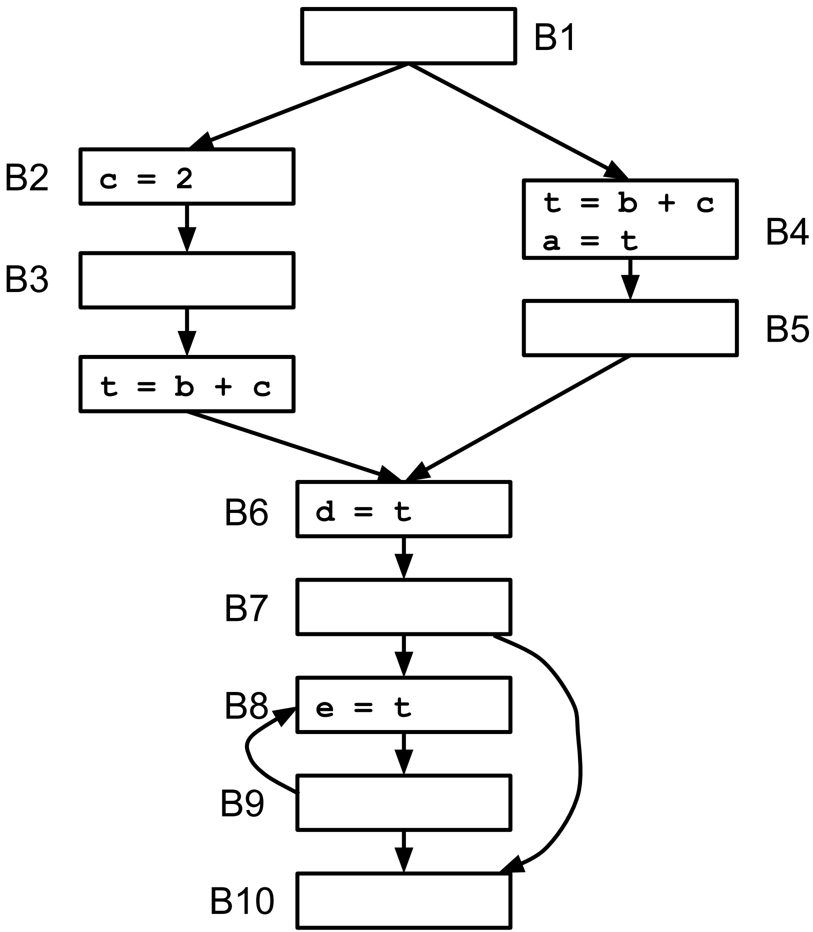
More Examples


B0
B1
i = 0
B2
a = b + c
i = i + 1
i < 1000
B3
z = b + c


B0
B1
i = 0
B2
a = b + c
i = i + 1
i < 1000
B3
z = b + c
B0
B1
i = 0
t = b + c
B2
a = t
i = i + 1
i < 1000
B3
z = t


B1
B2
c = 2
B3
B4
a = b + c
B5
B6
B7
d = b + c
B8
B9
e = b + c
B10
B11

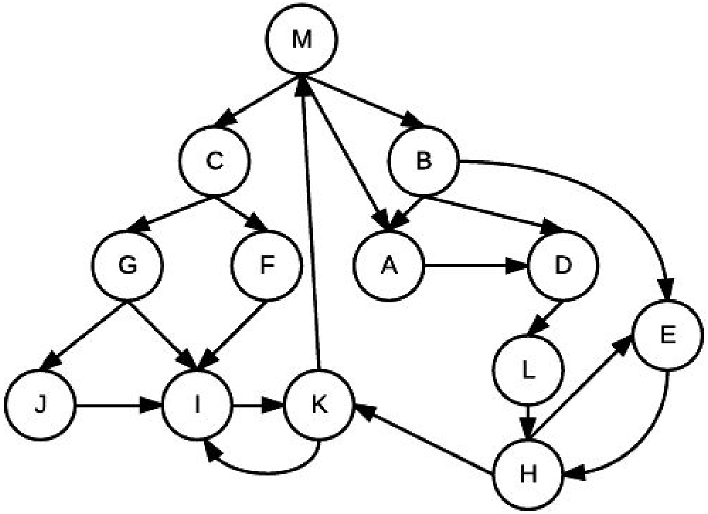

B1
B2
c = 2
B3
a = b + c
B4
B5
B6
d = b + c
B7
B8
e = b + c
B9
B10


B1
B2
c = 2
B3
t = b + c
t = b + c
a = t
B4
B5
B6
d = t
B7
B8
e = t
B9
B10

Dominators

CS243 Review Session

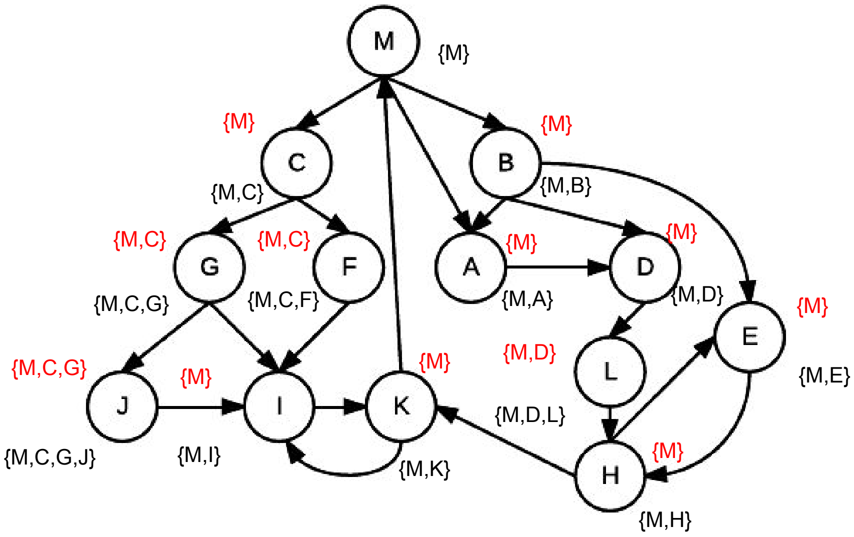
Example



Draw the dominator tree for this control flow graph.

Example

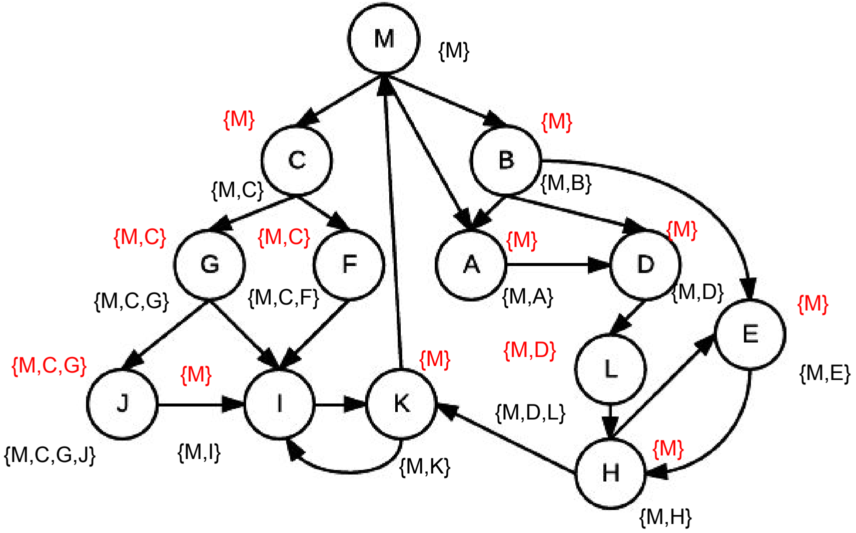
IN = RED
OUT = BLACK



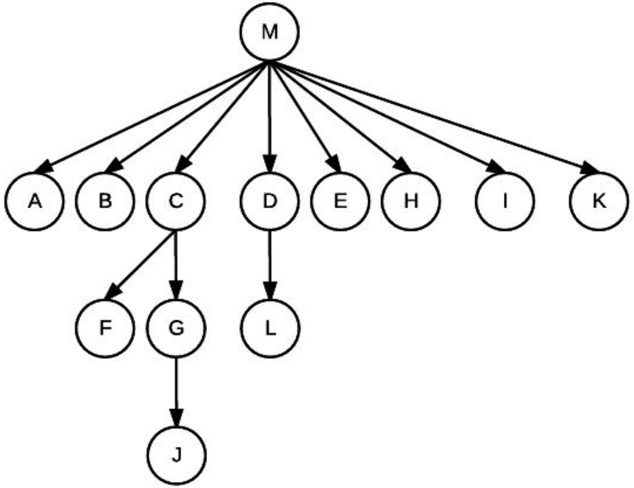
Draw the dominator tree for this control flow graph.

Example

IN = RED
OUT = BLACK



Dominator Tree



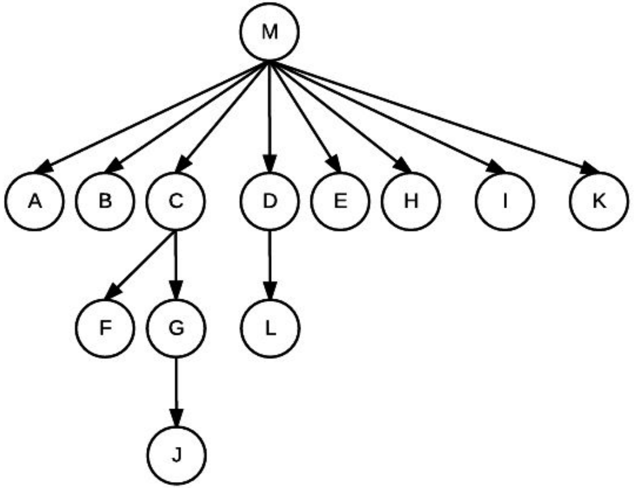
Draw the dominator tree for this control flow graph.

Example

Aside: there are algorithms for constructing the dominator tree directly

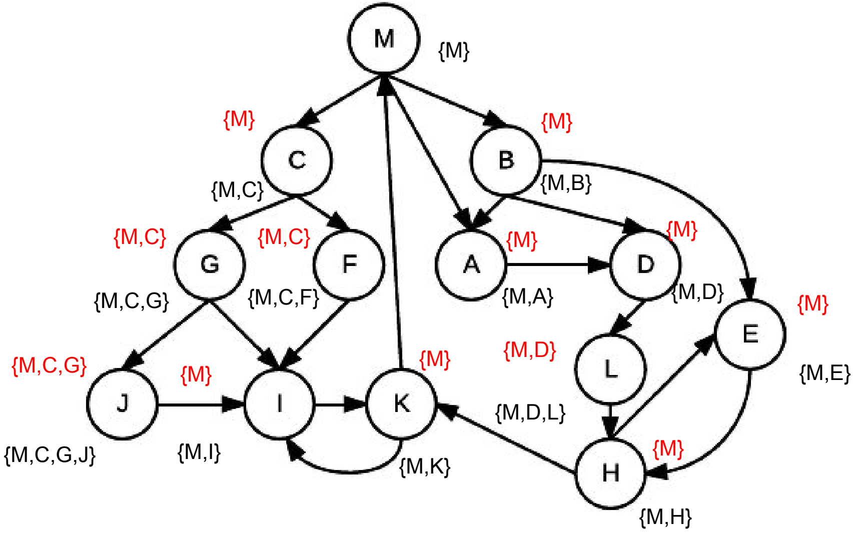
- Tarjan's algorithm (based on DFS)
- Buchsbaum's algorithm

Dominator Tree

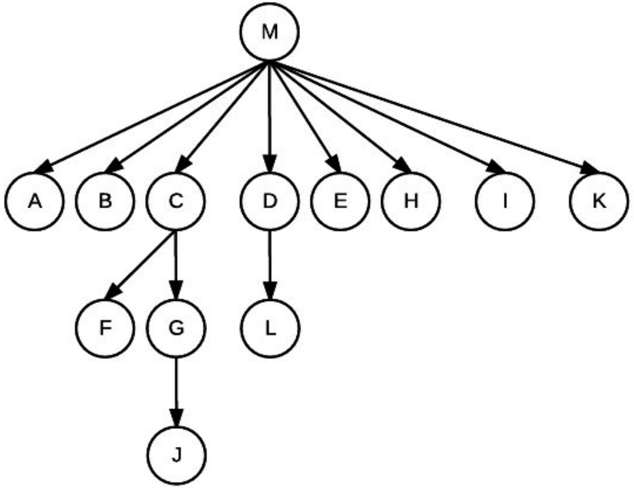


Example

IN = RED
OUT = BLACK



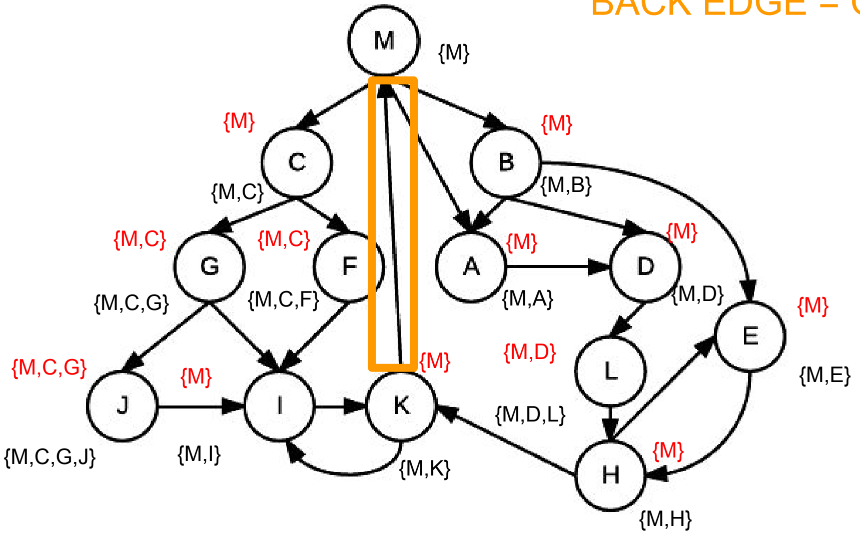
Dominator Tree



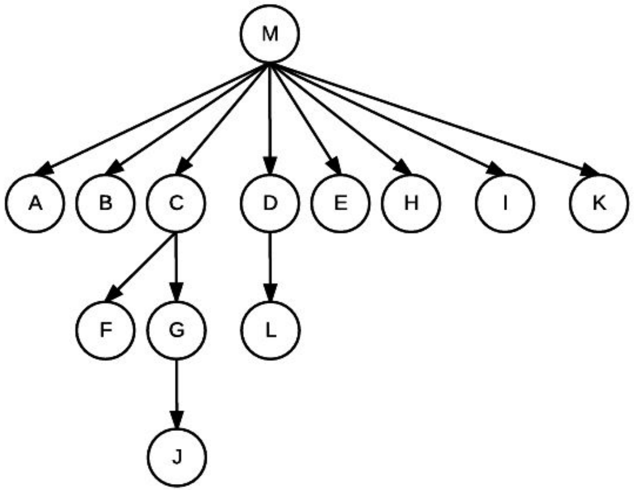
Find the back edges and natural loops in this graph.

Example

IN = RED
OUT = BLACK
BACK EDGE = ORANGE



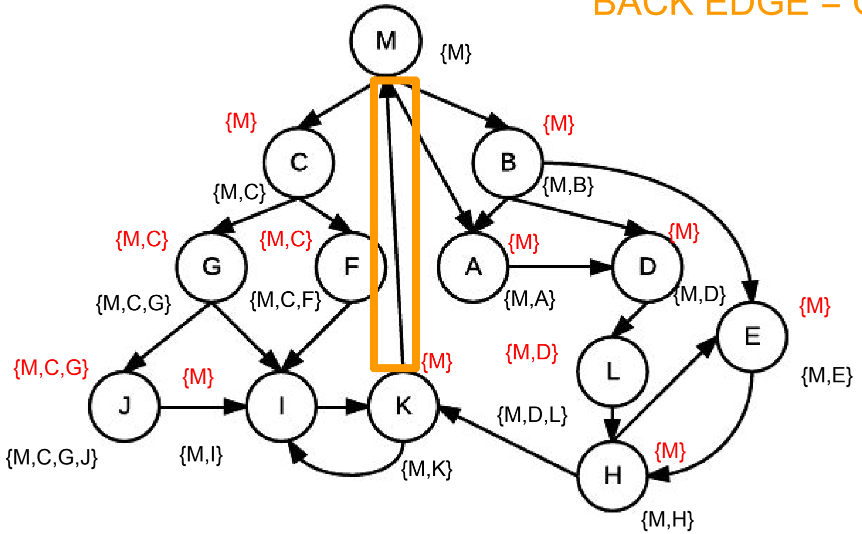
Dominator Tree



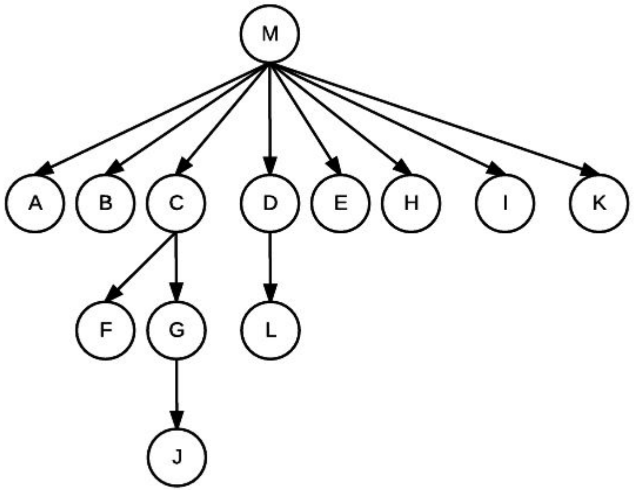
Find the back edges and natural loops in this graph.

Example

IN = RED
OUT = BLACK
BACK EDGE = ORANGE



Dominator Tree

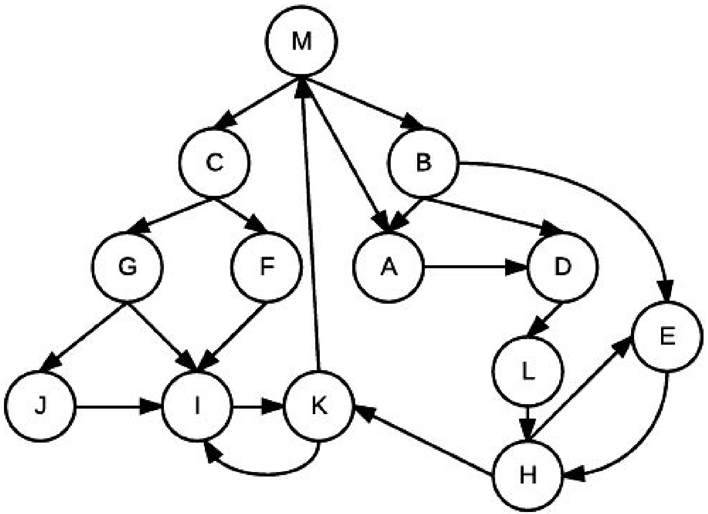


Find the back edges and natural loops in this graph.

Natural loop for back edge K → M: all nodes

* All nodes can reach K without passing through M

Post-dominators



How would we compute the post-dominators for this graph?

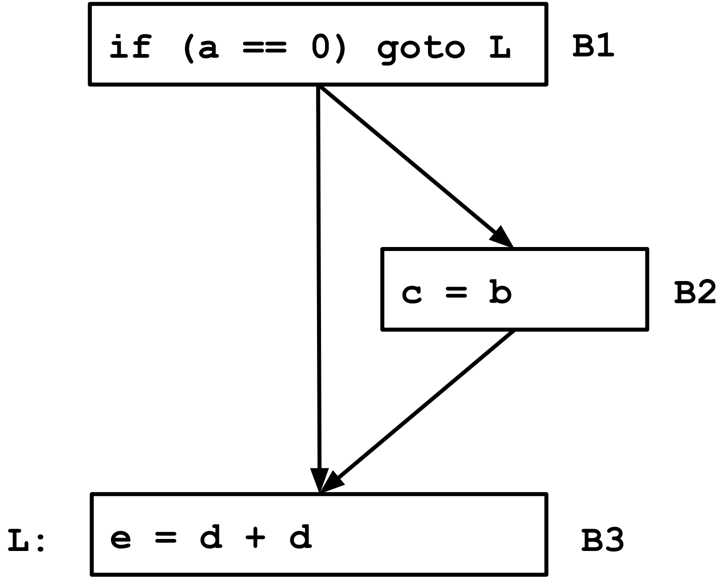
Definitions

A block B dominates block B’ if every path from the entry to B’ goes through B

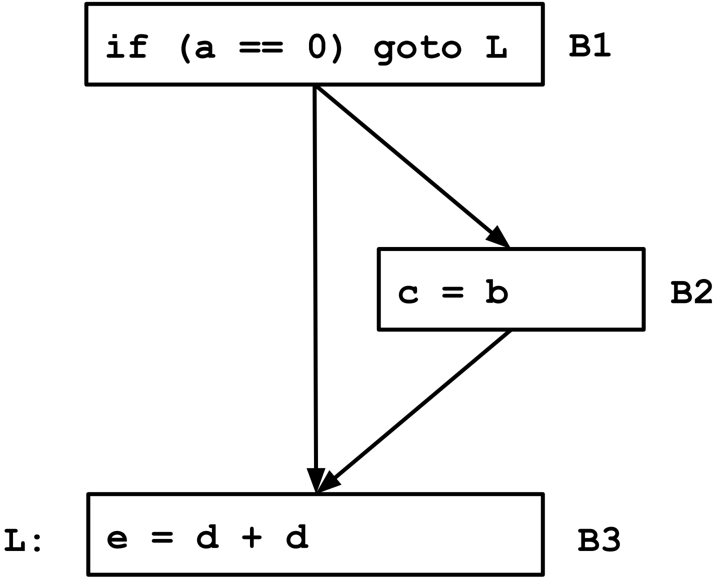
A block B postdominates block B’ if every path from B’ to the exit of the graph goes through B

If B dominates B’ and B’ postdominates B, B and B’ are control equivalent
* One is executed when and only when the other is

Example



Example



1. B1 and B3 are control equivalent.
2. B1 dominates B2, but B2 does not postdominate B1.
3. B2 does not dominate, B3 but B3 posdominates B2.

Code Motion

If two blocks are control-equivalent, you may move instructions between the two (upward/downward code motion) assuming there are no conflicting data dependences

More to come next week: instruction scheduling lecture

SSA

Construction of the static single assignment form (SSA) requires dominance frontier information.

The dominance frontier of a node ***d*** is the set of all nodes ***n*** such that ***d*** dominates an immediate predecessor of ***n***, but ***d*** does not strictly dominate ***n***.

More to come in Homework 3: converting to SSA form