

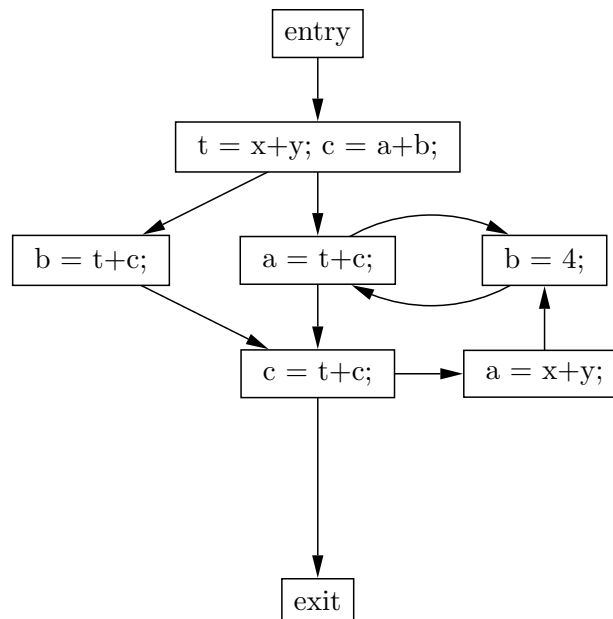
# Assignment 1

## Dataflow Analysis

Due: January 25, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on January 25. SCPD students may submit their homework by e-mail via [scpd-distribution@lists.stanford.edu](mailto:scpd-distribution@lists.stanford.edu) or give your homework to the courier.

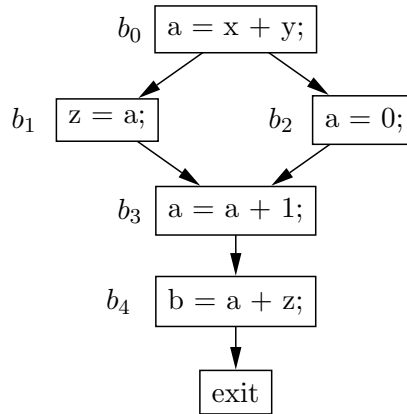
1. Compute the available expressions (Chapter 9.2.6 in ALSU) on entry and exit for each basic block in the flow graph below.



2. Is the following a meet operator? Please answer yes or no. No explanation is necessary.

- (a) Maximum value (on integers)
- (b) Product (on integers)
- (c) Addition (on integers)
- (d) Product mod 2 (on the set  $\{0, 1\}$ )
- (e) Addition mod 2 (on the set  $\{0, 1\}$ )
- (f) The GCD function (on integers)

3. We say that a program point  $p$  belongs to a live range of a definition  $d$   $u=x+y$  iff the value assigned to  $u$  in definition  $d$  may be accessed by using variable  $u$  at point  $p$ , for some path in the flow graph starting at  $p$ . Consider for example the code fragment below:



The live range of the definition ‘ $a = x + y$ ’ at point  $\text{entrypoint}(b_0)$  is  $\{\text{exitpoint}(b_0), \text{entrypoint}(b_1), \text{exitpoint}(b_1), \text{entrypoint}(b_3)\}$ . Similarly, the live range of the definition ‘ $a = a + 1$ ’ at point  $\text{entrypoint}(b_3)$  is  $\{\text{exitpoint}(b_3), \text{entrypoint}(b_4)\}$ .

This concept of live ranges can be used to increase flexibility in register assignment. For example, the live ranges of the definitions of  $a$  in  $b_0$  and  $b_3$  do not intersect at any points, so it may be possible to use same registers to hold these two values.

Describe an algorithm to find the live range of every definition in a program.

4. Define a compiler analysis that generates two kinds of warnings to help programmers detect uninitialized variables in their programs.
- (a) Warning I is issued on each use of a variable that may potentially be uninitialized.
  - (b) Warning II is issued on each use of a variable that is definitely uninitialized.

You can define one or more data flow problems to solve this problem. State clearly how you generate the warnings. For each data flow analysis defined, fill in the following table:

Direction (forward or backward)	
Lattice values	
Meaning of lattice values	
Meet operator	
Top	
Bottom	
Boundary conditions	
Initial interior values	
Transfer functions	

5. This question asks you to think about how changes to the initial values in a data flow analysis can affect the result. Recall that an answer to a data flow problem is considered “safe” if it is no bigger than the ideal solution.

Suppose you have defined a forward dataflow algorithm that is distributive and has finite descending chains. You accidentally initialized  $\text{OUT}[B]$  to  $\perp$  for all nodes other than  $\text{ENTRY}$ .

- (a) Will your algorithm give a safe answer for all flow graphs?
- (b) If not, will it give a safe answer for some flow graphs? If it will, give an example.
- (c) Will your algorithm give the MOP solution for all flow graphs?
- (d) If not, will it give the MOP solution for some flow graphs? If it will, give an example.