

Assignment 3

Dataflow analysis and Register allocation

Due: February 8, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on February 8th. SCPD students may submit their homework by e-mail via scpd-distribution@lists.stanford.edu or give your homework to the courier.

1. Several program optimizations depend on knowing the signs of variables or whether variables can be zero. For example, in C the expression $x / 2$ can be simplified to $x \gg 1$ if x is non-negative, while in Java a divide-by-zero runtime check on y / z can be eliminated if z is non-zero.
 - a. Describe an algorithm to find the signs and relation to zero of variables in a program. Your algorithm must determine, for each basic block, if each variable in the program is positive, non-positive, equal to zero, negative, non-negative, or unknown. The instructions in the program are: assignment, addition, subtraction, and multiplication. Instruction operands are either constants or variables.

Direction (forward or backward)	
Lattice Values	
Meaning of lattice values	
Meet Operator	
Top	
Bottom	
Boundary Conditions	
Initial Interior Values	
Transfer Function	

- b. The conditional branches in a program can provide important information to a dataflow analysis. For example, in the following program fragment x cannot be zero at the division z / x , even if we know nothing about its value at the start of the fragment.

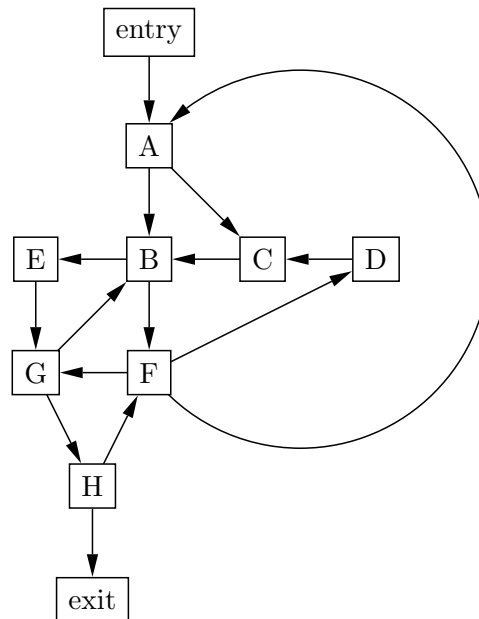
```

if (x > 0)
  y = z / x;
else
  y = 0;

```

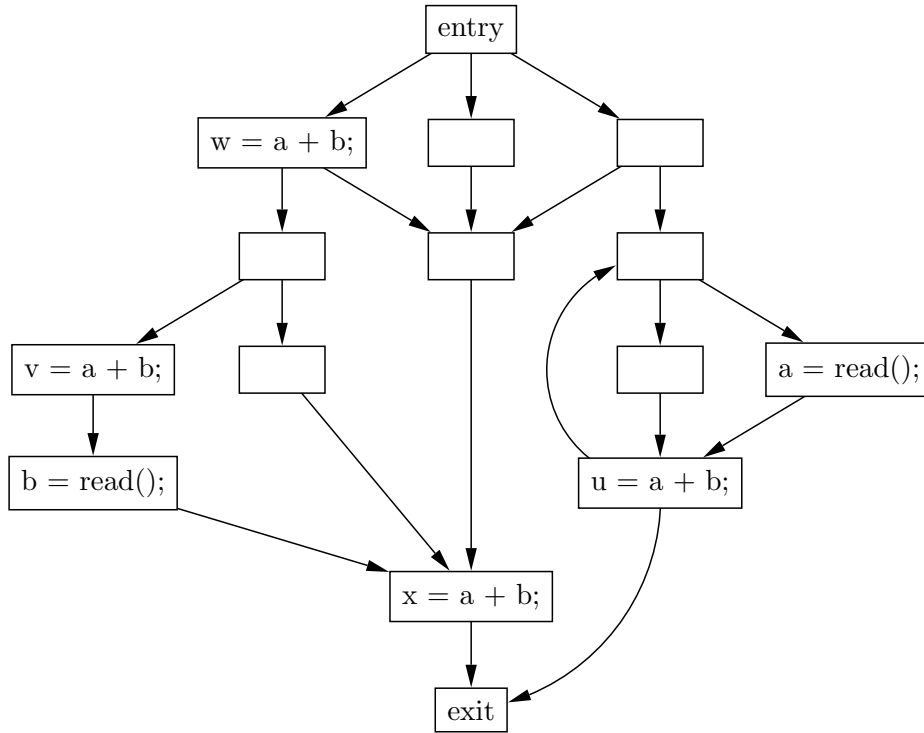
One way of incorporating branch conditions into a dataflow analysis is to insert into the CFG **assume** instructions, which do not change the values of any variables but specify a condition that definitely holds at that program point. The above program fragment can be represented with the following CFG:

3. Consider the following flow graph



- Give a reverse post-ordering for this flow graph.
- Draw the dominator tree for this graph.
- Find the back edges and natural loops in this graph.
- Is this graph reducible? Explain.
- Give a minimal set of edges that should be added or removed to change your answer to question d.

4. Apply lazy code motion to the following program. You do not need to show the intermediate steps, just show the optimized code. You may add basic blocks to the flow graph, but only show those that are not empty in your solution.



5. Consider a simple machine with only 3 registers R1, R2 and R3. Is it possible to find a register allocation for the following interference graph such that no spilling is necessary? Please specify a register assignment if the answer is yes, and explain why if the answer is no.

