Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.035 Fall 2016

Test II Solutions

I Register Allocation

In this problem, you will perform register allocation for the following code. Each instruction is labeled with a number. Assume that you do not perform any other optimizations, and none of the variables is subsequently used.

```
    int a, b, c, d;
1:  a = read_int();
2:  b = read_int();
3:  while (a > b) {
4:    c = a - 5;
5:    if (c > b) {
6:      d = c - 1;
7:      a = d + a;
      } else {
8:      d = c + 1;
9:      a = d + a;
      }
10:   b = a - c;
    }
11: print(a);
```

1. [5 points]: Write the set of def-use chains for each variable in the program. Write each def-use chain as number pair (d, u) where d is the label of an instruction that defines the variable and u is the label of an instruction that uses that definition.

Solution:

`a`: (1, 3) (1, 4) (1, 7) (1, 9) (1, 11) (7, 3) (7, 4) (7, 10) (7, 11) (9, 3) (9, 4) (9, 10) (9, 11)

`b`: (2, 3) (2, 5) (10, 3) (10, 5)

`c`: (4, 5) (4, 6) (4, 8) (4, 10)

`d`: (6, 7) (8, 9)

2. [5 points]: Write the set of webs in the program. Write each web as the set of instructions that belong to the web. We have given you names `w1-w7` for the webs, use only as many names as you need.

Solution:

`w1`: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}

`w2`: {2, 3, 4, 5, 10}

`w3`: {4, 5, 6, 7, 8, 9, 10}

`w4`: {6, 7}

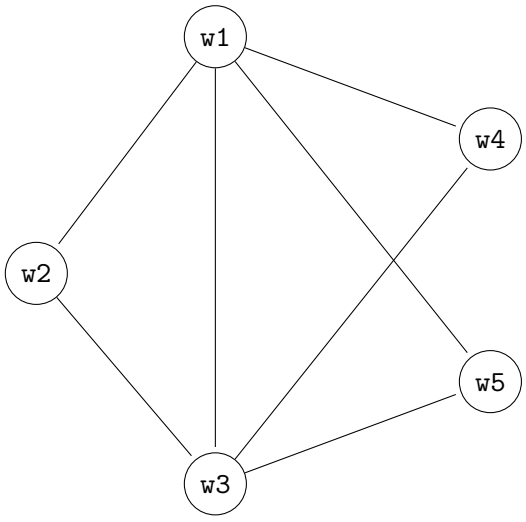
`w5`: {8, 9}

`w6`:

`w7`:

3. [5 points]: Draw the interference graph for the webs. Each node in the interference graph should represent one web. There should be an edge between two nodes if the two webs interfere. Label each node with the name (`w1-w7`) of the corresponding web.

Solution:



4. [5 points]: Suppose that the architecture we are targeting for compilation has three general purpose registers. Can we place all the variables in this code in registers (ignore any constraint due to calling convention)? If yes, describe an assignment of variables to registers. If no, explain the reason using your interference graph as part of your justification.

Solution: Yes. The web of `b`, which is `w2`, does not interfere with the webs of `d`, which are `w4` and `w5`, so `b` and `d` can be put in the same register, and the other two variables can occupy one register each.

`a`: `w1` → register 1

`b`: `w2` → register 2

`c`: `w3` → register 3

`d`: `w4` and `w5` → register 2

II Parallelization

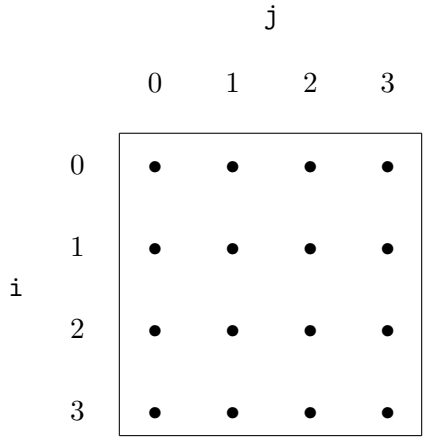
Consider the following program:

```
for (i = 0; i < n; i += 1) {
  for (j = 0; j < n; j += 1) {
    A[i,j] = A[i - 1, j - 2] + 3;
  }
}
```

`A[i, j]` means the element at the `i`-th row and `j`-th column in the 2-dimensional array `A`. Ignore out-of-bound array access.

5. [3 points]: Assume that `n = 4`. In the grid below, circle the dots that represent the iteration space for this loop. Ignore out-of-range cases. Each dot represents the values of `i` and `j` for an iteration.

Solution:



6. [5 points]: What is the distance vector for these loops?

Solution:

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

7. [5 points]: Without any other optimizations or transformations, is either loop fully parallelizable as a "FORALL" loop? If so, is it the outer loop (i), the inner loop (j), or both, that can be parallelized?

Solution: Only the inner loop

For programs with a sequential portion T_s and a parallelizable portion T_p running on a machine with n processors, recall Amdahl's law and the definition of speedup:

$$T(n) = T_s + \frac{T_p}{n}$$

$$\text{speedup} = \frac{T(1)}{T(n)}$$

Consider a program where 20% of the operations are sequential and 80% are parallelizable.

8. [4 points]: Assume the program runs on a machine with 8 processors, what is the speedup?

Solution:

$$\text{speedup} = \frac{T(1)}{T(8)} = \frac{1}{0.2 + \frac{0.8}{8}} = \frac{10}{3}$$

9. [6 points]: Regardless of the number of processors in the machine, what is the maximum possible speedup for this computation?

Solution:

$$\text{speedup} = \frac{T(1)}{T(\infty)} = \frac{1}{0.2} = 5$$

III Dataflow Analysis

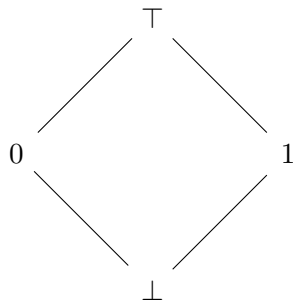
Your task in this problem is to design a dataflow analysis that will determine whether each variable $v_1, ..., v_k$ is odd or even at each point in the program. The program itself will be represented as a control flow graph with two kinds of assignment statements:

- $v = c$: sets a variable v to a constant c.
- $v_1 = v_2 + v_3$: sets a variable v_1 to the sum of variables v_2 and v_3.

10. [5 points]: Design the lattice for this dataflow analysis problem. You should specify the set S of lattice elements and the least upper bound operator $\vee$ over the set of lattice elements. We are expecting each lattice element to record an abstract value for each variable $v_1, ..., v_k$. As part of your definition you should define the set of abstract values.

Solution:

We define the base lattice $B = \{\bot, 0, 1, \top\}$, where 0 represents all even values, and 1 represents all odd values. Below is the Hasse diagram defining $\vee$.



For clarity, let $b \in B$, we have:

$$\top \vee b = \top$$
$$0 \vee 0 = 0$$
$$0 \vee 1 = \top$$
$$1 \vee 0 = \top$$
$$1 \vee 1 = 1$$
$$\bot \vee b = b$$

For S we have:

$$S = \{[v_1 \to b_1, \ldots, v_k \to b_k] : b_1, \ldots, b_k \in B\}$$

and

$$[v_1 \to b_1, \ldots, v_k \to b_k] \vee [v_1 \to b'_1 \ldots, v_k \to b'_k] = [v_1 \to b_1 \vee b'_1, \ldots, v_k \to b_k \vee b'_k].$$

11. [5 points]: Specify the transfer function for each kind of assignment statement. Specifically, specify the transfer function f_n when the control flow graph node n is of the form $n : v = c$ and when n is of the form $n : v_1 = v_2 + v_3$. As part of the solution to this problem we are expecting you to define a version of + that operates on the abstract values.

Solution:

$$f_{v=c}(e) = e[v \to c \,\%\, 2]$$
$$f_{v_1=v_2+v_3}(e) = e[v_1 \to e[v_2] + e[v_3]]$$

The operator "+" is defined by the following table.

+	$\top$	0	1	$\bot$
$\top$	$\top$	$\top$	$\top$	$\bot$
0	$\top$	0	1	$\bot$
1	$\top$	1	0	$\bot$
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

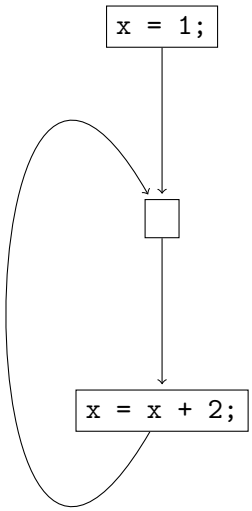
12. [5 points]: Give the abstraction function $AF(x)$ (here x is a program state of the form $[v_1 \to c_1, ..., v_k \to c_k]$ that specifies a value c_i for each variable v_i).

Solution:

$$AF([v_1 \to b_1, \ldots, v_k \to b_k]) = [v_1 \to b_1 \,\%\, 2, \ldots, v_k \to b_k \,\%\, 2]$$

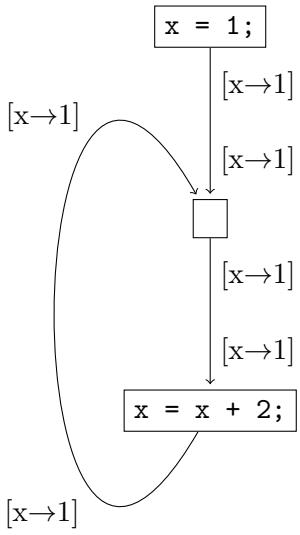
13. [5 points]: Give a program, in the form of a control-flow graph, for which the **meet over all paths** solution to the odd/even analysis problem does not equal the solution that the dataflow analysis algorithm produces. If no such program exists, explain why it does not exist.

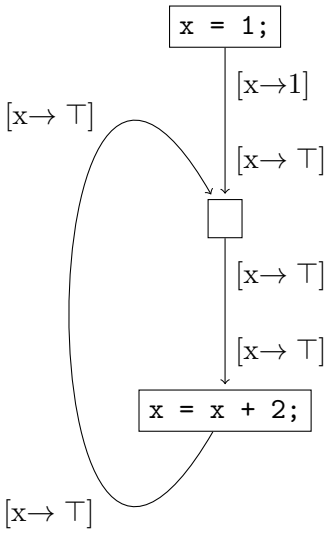
Solution:



14. [5 points]: Give a program, in the form of a control-flow graph, that has **multiple fixed-point** solutions to the dataflow equations that your analysis generates. For this program, provide at least two of the multiple fixed-point solutions. If no such program exists, explain why it does exist.

Solution:





IV Lattices

You are working with the reaching definitions lattice. Recall that the elements of this lattice are sets of definitions, with each set represented as a bit vector with one position for each definition. Also recall that the order is $x \leq y$ if $x \subseteq y$ and the least upper bound operator is $x \vee y = x \cup y$.

Your teammate comes to you with a set of transfer functions that includes the transfer function $f(x) =$ bitwise not x. In other words, $f(x)$ flips all of the bits in x so that $f(1011) = 0100$ (for example).

15. [6 points]: Is $f(x)$ monotone? Why or why not?

Solution:

No. $0000 \leq 1111$, but $f(0000) = 1111 \not\leq f(1111) = 0000$.

16. [6 points]: Is $f(x)$ distributive? Why or why not?

Solution:

No. $f(01) \vee f(10) = 10 \vee 01 = 11$, which is not euqal to $f(01 \vee 10) = f(11) = 00$.

V Loop Optimizations

In the following program, `j` is a derived induction variable in the family of the base induction variable `i`.

```
i = 0;
while (i < 7) {
  j = i * 4 + 10;
  sum = sum + j;
  i = i + 1;
}
```

17. [**10 points**]**:** Rewrite the program after induction variable recognition and induction variable strength reduction (and no other optimizations):

Solution:

```
i = 0;
j = 10;
while (i < 7) {
  sum = sum + j;
  i = i + 1;
  j = j + 4;
}
```

18. [**10 points**]**:** Rewrite the program after induction variable recognition, induction variable strength reduction, and induction variable elimination (and no other optimizations):

Solution:

```
j = 10;
while (j < 38) {
  sum = sum + j;
  j = j + 4;
}
```