CS243 Midterm
11AM – 12:15PM
February 10, 2014

The exam is open book/notes/laptop. We do not guarantee power or Internet access, however.

Answer all 5 questions on the exam paper itself.

Write your name here: _______________________________________

I acknowledge and accept the honor code.

(signed) _____________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Max</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
QUESTION 1 (Flow graphs)

The figure above is a flow graph with entry 1. Answer the following questions about it.

a) Fill in the table of dominators. The trivial dominators (i.e. the entry dominates every node, and every node dominates itself) have already been filled in.

<table>
<thead>
<tr>
<th>Dominator</th>
<th>Dominates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

b) If we build a depth-first spanning tree for the flow graph, there are several different trees that can result, depending on which successor we visit first, when there is a choice. However, in all these trees, node 1 is the root, four of the other nodes have the same parent, and one node can have two different parents. Which node can have different parents?
(the flow graph from the previous page)

c) List each back edge and its corresponding natural loop.
(Note that the number of rows in the table isn’t necessarily representative of the number of back edges in the graph.)

<table>
<thead>
<tr>
<th>Back edge</th>
<th>Natural loop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>


d) What is the depth of the flow graph?

______
QUESTION 2 (DFA Frameworks)

A useful analysis in some applications is a framework that determines when two program variables must have the same value. In this framework,

- V is set of all possible partitions.
- A partition is a set of equivalence classes.
- Variables in an equivalence class have the same value.

For example, if the program has only two variables, x and y, then,

\[ V = \{ \{x,y\}, \{x\}, \{y\} \} \]

The meet operation \( \wedge \) for this framework is the partition that places variables x and y in the same equivalence class if and only if both argument partitions place them in the same equivalence class. For example, \( \{a,b,c\}, \{d,e\}, \{f\} \wedge \{a,b\}, \{c,d\}, \{e,f\} \wedge \{a,b\}, \{c\}, \{d\}, \{e\}, \{f\} \). Intuitively, as a confluence operator, this meet says that we know two variables have the same value coming into a block if and only if they had the same value coming out of all the predecessors of that block. Answer the following questions:

a) If there are three variables in the program, list the elements of V (the set of possible partitions):

\[ \{\{x,y\}\}, \{\{x\}, \{y\}\} \]

b) What are the top and bottom elements for this semilattice of three variable program?

Top element = __________________________

Bottom element = __________________________

c) Let p and q be partitions in V. When is \( p \leq q \)? Note, we are looking for an answer in terms of the equivalence classes in the two partitions, not just “if \( p \wedge q = p \).”

______________________________________________________________________________

______________________________________________________________________________

______________________________________________________________________________
d) Describe the transfer function for a block that consists of only the assignment statement $x=y$.


e) Describe the transfer function for a block that consists of only the assignment statement $x=y+1$.


QUESTION 3 (Dataflow solver)

Ben Bitdiddle is implementing a forward dataflow solver. The pseudocode for his solver is below.

```
procedure MakeIterationOrder(G):
1:    let I be a list
2:    call DFS(G,G.entry(),I)
3:    return I

procedure DFS(G,v,I):
4:    label v as discovered
5:    for all edges from v to w in G.successors(v) do
6:       if w is not discovered
7:          recursively call DFS(G,w,I)
8:    append v to list I

procedure ForwardSolver(G,V,∧,F):
9:    for all blocks B in G do
10:       IN[B] = OUT[B] = V
11:    OUT[G.entry()] = V
12:    I = MakeIterationOrder(G)
13:    while changes to any OUT do
14:       for each block B in I do
15:          IN[B] = V
16:          for each block P in G.predecessors(B)
17:             IN[B] = ∧(IN[B],OUT[P])
18:             OUT[B] = F_B(IN[B])

The parameters to ForwardSolver are:
   G: control flow graph
   V: domain (possible values for IN/OUT)
   ∧: meet operator
   F: transfer functions
```

Ben’s good friend Alyssa P. Hacker notices a bug in Ben’s code that makes his forward dataflow solver take a long time to converge.

a) Assume the input graph G is reducible and has N nodes, E edges, and maximum depth D, and the dataflow problem is reaching definitions. What are the maximum number of iterations around the while loop at line 13 before Ben’s dataflow solver will converge and detect there are no changes?
b) Alyssa points out an easy, one-line fix to Ben’s code that will make it converge much more quickly. What is the fix?

______________________________________________________________________________

______________________________________________________________________________

c) Again assume the input graph $G$ is reducible and has $N$ nodes, $E$ edges, and maximum depth $D$, and the dataflow problem is reaching definitions. After Alyssa’s fix, what are the maximum number of iterations around the while loop at line 13 before the fixed dataflow solver will converge and detect there are no changes?

______________________________________________________________________________

______________________________________________________________________________

d) Does your answer to the previous question change if we drop the assumption that the input graph is reducible? If so, how?

______________________________________________________________________________

______________________________________________________________________________

e) Does your answer to the previous question change if we also drop the assumption that the dataflow problem is reaching definitions, and instead assume the dataflow problem is monotone with lattice height $H$? If so, how?

______________________________________________________________________________

______________________________________________________________________________
QUESTION 4 (Register Allocation)

a) Allocate registers using the register coloring algorithm we discussed in class on a machine with 3 registers. Ignore the impact of “...”. Show the interference graph and each step of the algorithm.
b) Can you allocate with two registers? If not, can you rearrange the code within one or more blocks so that you can? If you need to rearrange the code, show the rearrangement and the new register assignment.
QUESTION 5 (Instruction Scheduling)

Consider a processor with two functional units Load/Store and Everything_else. The result of a load is not available for an ALU operation until two cycles later; i.e. if the load is in cycle i, a use of the loaded data can not issue until cycle i+2. Loaded data can be stored in cycle i+1. All other results are available in the next cycle.

Consider the following program

1: load a8 <- 2(a4)  # load a8 from *(2+a4)
2: load a9 <- 0(a4)  # load a9 from *(a4)
3: load a10 <- 4(a4) # load a10 from *(4+a4)
4: add a8, a8, a9    # a8 = a8 + a9
5: addi a10, a10, 2  # a10 = a10 + 2
6: store a8 -> 0(a4) # store a8 into *(a4)
7: store a10 -> 2(a4) # store a10 into *(2+a4)

a. Draw a data dependency graph of the code above. Denote true dependencies (RaW) as solid lines and label them by their latencies. Denote anti- and output dependencies (WaR and WaW) as dashed lines.
b) Schedule the code using the list scheduling algorithm from class.

<table>
<thead>
<tr>
<th>Cycle</th>
<th>ALU</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c) Give the basic global scheduling algorithm from class and the following code. Assume that the load is a load of a global variable that is always safe to load. Will the algorithm speculate the load? Is it always the right thing to speculate the load?

```
add a8, a8, a9
add a10, a10, a9
load a5 <- 0(a5)
```