Final Exam
CS6013

Maximum marks = 50, Time: 3hrs

23-Nov-2012

1. [10] **Dominance Frontiers**
   Define dominance frontier and write an algorithm to compute the Dominance Frontiers for all the nodes in the CFG.

   Input: `Set of Nodes N, Set of Edges E, root node r`

   Output: $\forall v \in$ `N, DF(v)`

   Assume that you are given the following maps:

   - `Succ` and `Pred` functions, that return the successor and predecessors respectively.
   - `DOM`, `SDOM`, `IDOM` functions return the dominators, strict dominators, and immediate dominators for a given node.

   What is the complexity of the algorithm?
   [*Bonus* 5] Can you implement a faster algorithm?

2. [15] **Register allocation**
   a) Briefly describe the iterated register coalescing algorithm.
   b) Intra-procedural register allocation suffers from many drawbacks: the values present in different registers before a call are copied to argument registers / swap area; the caller and callee save registers have to appropriately saved. Extend the iterated register coalescing algorithm to design an inter-procedural register allocation scheme that avoids these copy related issues. Assume that standard intra-procedural liveness information is available. Assume that the globals are live throughout the program.

```
b=1;                                  r1 = 1;
c=2;                                  r2 = 2;
a = foo(b, c).                        call foo
print a;                              print r3;
p = c;                                r1 = r2;
q = 3;                          ⟶     r2 = 3;
x = foo (p, q);                       call foo
print x;                              print r3
int foo(int f, int g){                foo:
  k = f op g                          r3 = r1 op r2
  return k;
}                                     return
```

3. [15] **Copy propagation**
Given an assignment `x := y`, the *copy propagation* optimization, replaces each later use of `x` with `y`, as long as there is no redefinition of `x` or `y` in between. Write an iterative algorithm to do intra-procedural copy propagation. Apply your algorithm on the following code:

```
     c = a + b
     d = c
     e = d * d
L1:  f = a + c
     g = e
     a = g + d
     if (a < c) {
           h = g + 1
       L2: b = g * a
           if (h < f) goto L1
     }
     else {
       f = d - g
       if (f > a)  goto L2
       else
         c = 2
     }
```

4. [10] **Dependence analysis**
Answer the below mentioned queries for the given sample code:

```
for (i = 1; i <= n; ++i) do
  for (j = n; j >= 1; --j) do
    for (k = 1; k <= n+1; ++k) do
      S1: A[i,j,k] = A[i,j-1,k-1] + A[i-1,j,k]
      S2: B[i,j-1,k] = A[i,j-1,k-1] * 2.0
      S3: A[i,j,k+1] = B[i,j,k] + 1.0
    endfor
  endfor
endfor
```

   (a) Draw the iteration space for the loop nest.

   (b) Draw the execution order relationships between the three labeled statements.

   (c) Write the dependence relations (flow, anti and output) between the three labeled statements.

   (d) Restate the dependence relations in terms of distance vectors, direction vectors and dependence vectors.

   (e) For the different references to A and B, use the GCD test to check if/when there exists any same iteration or different iteration dependence.