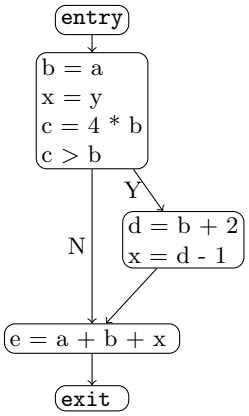
Midterm Exam
CS6013

14-Mar-2015

Read all the instructions and questions carefully. You can make any reasonable assumptions that you think are necessary; but state them clearly. There are total four questions, totalling 40 marks. You will need approximately 30 minutes for answering a 10 marks question (plan your time accordingly). For questions with sub-parts, the division for the sub-parts are given in square brackets.

You will get an answer sheet with 12 pages (if you get a answer sheet with fewer pages then ask for a replacement sheet). Leave the first page empty. Start each question on a new page. Think about the question before you start writing and write briefly. **The answer for any 12 mark question (including all the sub-parts) should NOT cross more than three pages.** If the answer for any 12 mark question is spanning more than three pages, we will strictly ignore the spill-over text. If you scratch/cross some part of the answer, you can use space from the next page.

1. [12] **Control Flow Analysis and Copy propagation**
   Given an assignment `x = y`, the *copy propagation* optimization, replaces each latter use of `x` with `y`, as long as there is no redefinition of `x` or `y` in between. Design an intra-procedural copy propagation algorithm based on structural analysis, by defining the flow functions (for pass one) and propagation functions (pass two) [4.5+4.5]. Assume that the function consists of a sequence of statements and we only admit the following types of statements: copy statements (of the form x = y), computation statements (of the form x = y op z), conditional statements (of the form 'if (x) S'), and while loops (of the form 'while (x) S'). Apply your algorithm on the following code and show a trace [3].



2. [12] **Dependence analysis**: Answer the queries for the given sample code:

```
for (i = 1; i <= n; i+=1)
  for (j = 1; j <= i; j+=1)
    for (k = 1; k <= j; k+=1) {
      Sl: A[i,j,k] = A[i-1,j-1,k-1] + A[i-1,j,k]
      S2: B[i,j-1,k] = A[2*i+1,j-1,k-1] * 2.0
      S3: A[3*i-2,j,k+1] = B[i,j,k] + 1.0
    }
```

   (a) Draw the iteration space for the loop nest. [1.5]

   (b) Draw the execution order relationships between the 3 labeled stmts. [1.5]

   (c) Write the dependence relations (flow, anti and output) between the S1, S2 and S3. [1.5]

   (d) Compute the distance vectors, and direction vectors. [3]

   (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. [4.5]

3. [4] **SSA**: How to eliminating SSA ϕ nodes, in a semantics preserving way? Illustrate using examples.

4. [12] **Scheduling**: Design a scheme to schedule a given a basic block of instructions, to improve the performance, by reducing the wait time [9]. Assume:

   (a) Each instruction I has at most two operands, returned by $I.useOps$.

   (b) Each instruction I has a def, returned by $I.defOp$.

   (c) Each instruction I has an associated execution time (returned by $XTime(I)$) that gives the number of cycles I takes to produce the value in $I.defOp$.

   (d) If an instruction i_1 uses an operand o_1 that is produced by another instruction i_0, then i_1 may have to wait till o_1 is produced. For i_1 to not wait for any additional cycles, i_1 must be at a distance of $XTime(i_0) - 1$ or more from i_0.

   (e) In the worst case, the time to execute the basic block is given by $\sum_i XTime(i)$.

   (f) Your input would be a sequence of instructions. And the output of your algorithm is expected to be a sequence of instructions, such that the total execution time is reduced.

   (g) An instruction can be issued in each cycle.

   Example: Say each instruction has an $execTime$ of two cycles, except a memory instruction. For a memory instruction $execTime$ is 4 cycles. Consider the BB:

```
I1. r1 = r2 * r3
I2. r4 = r2 + r1
I3. r5 = r4[0] // memory op
I4. r6 = r3 - r2;
I5. r7 = r6 / r1
I6. r8 = r7 + 1
```

   The given BB will take – 6 × 1 cycle per instruction + delay for I2 (1) + delay for I3 (1) + delay for I5 (1) + delay for I6 (1) + delay after I6(1) = 11 cycles. The BB: I1, I4, I2, I5, I3, I6 will take 6 × 1 cycle per instruction + delay due to I3 after I6 has finished (1) + delay after I6 (1) = 8.

   Apply your algorithm on this example (I1, I2, I3, I4, I5, I6) and show a trace. [3]