Final Exam
CS3300

Maximum marks = 60, Time: 3hrs

21-Nov-2014

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 six questions, each 10 marks worth. You will need approximately 30 minutes for answering an 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 question (including all the sub-parts) should NOT cross more than two pages.** If the answer for any question is spanning more than two 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. [10] **IR Generation**:

Recall the semantic actions discussed in the class to generate IR for booleans, conditional statements and loops. Write similar rules to translate statements generated by the grammar shown on the right hand side [7]. Note I: `repeat S1 until(B)` keeps repeating `S1` ($\geq$ 1 times), till `B` is *true*. Note II: `^` is the xor operator.

```
S -> if (B) S1
S -> repeat S1 until (B);
B -> B1 ^ B2
B -> true
B -> false
B -> Identifier
S -> ;
```

Use your stated rules to generate IR for the code shown in the right hand side [3].

```
repeat
  if (x ^ false) ;
until (false ^ y);
```

2. [10] **Control flow**:

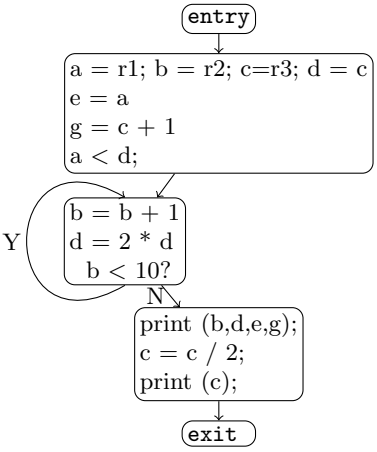
For the code shown in the right hand side, identify the basic blocks [2] and draw the control flow graph [2]. For the first statement in each of the basic blocks, compute the dominator information [3]. Identify the backedges [2], and the loops (set of basic blocks and edges) [1].

```
void foo(){
   x = 1; y = n;
   L1: x = x + 1;
   if (cond) goto L2;
   y = y + 1;
   L2: if !cond1 goto L3
   x = bar(x);
   cond1 = cond1 && x < 10
   goto L2;
   L3: x = x - y;
   if (cond2) goto L1;
   print (x, y);
}
```

3. [10] **Low level optimizations**:

Show two unique example (pattern) codes where you can apply the following optimizations: i) load-store optimizations, ii) Eliminating useless branch instructions, iii) instruction scheduling to reduce data hazards, iv) instruction scheduling to reduce control hazards, v) instruction scheduling to reduce structural hazards. [2+2+2+2+2]

4. [10] **Register Allocation**:
Compute the liveness information [2], interference graph [2], and do register allocation using Kempe's heuristic [1.5], assuming that the machine has only three registers (plus two special registers for spill instructions), and show the code after register allocation (temporaries replaced with registers, and insertion of spill code, wherever necessary) [1.5]. Re-do the register allocation assuming that there are no special registers for spilling [3].
Note I: Load V1 R1; loads the value of variable V1 into the register R1. Note II: Store R1 V1; store R1 to the variable V1.



5. [10] **Optimizations**:
Optimize the code in the right hand side in a step by step manner, using machine independent optimizations. At each step, indicate the optimization applied and the resulting code (or at least the difference).

```
void foo(double z, int a, int A[10]){
  int n = 4;
  double x = n * 4;
  while (z < x) {
    print (A[z]);
    y = a * n;
    p = c;
    q = p + A[z]
    p = c + q;
    z = z + 1;
  }
  printf (p, y);
  goto L1;
    a = a + 1;
  L1: n = n - 2;
  if (n > 4) goto L2;
     print a/n;
  L2: print (n);
}
```

6. [10] **Translating Exceptions**: Translate the following code to MiniJava. Note: MiniJava does not support exceptions.

```
class A{
public void foo(){
      try {
            if (cond) { e1 = new E1(); }
    else { e1 = new E2(); }
            throw e1;
      } catch (E2 e){ fbar2(e); }
}
public void bar(){
      try { foo(); }
      catch (E1 e){ fbar(e); }
} } /* end of class A */
```