Final Exam, CS6013

Maximum marks = 60, Time: 3hrs

08-May-2024

Read all the instructions and questions carefully. You can make any reasonable assumptions that you think are necessary; but state them clearly. It is your responsibility to write legibly. There are total six questions; answer any five. If you answer more than five, we will only correct the first five. Each question will approximately require around 30 minutes to answer. For questions that have sub-parts, the division for the sub-parts is mentioned in square brackets.

Start each question on a new page (and write your roll number on each additional sheet). Think about the question before you start writing and write briefly.

1. [3+3+3+3] **Loop Optimization and Dependence Analysis**

In each of the examples below, make sure that the variables in the loop-body are either arrays or loop index variables.

(a) Explain using a C example code snippet, how a loop can be distributed, and the advantage of distributing that loop. [3 marks]

(b) Explain using a C example code snippet, when a loop cannot be distributed, and show the dependences (in terms of distance vectors, and direction vectors) because of which that loop cannot be distributed. [3 marks]

(c) Explain using a C example snippet, how a rectangular (non-square) loop nest can be tiled, and the advantage of tiling that loop. [3 marks]

(d) Explain using a C example snippet, when a loop nest cannot be tiled and show the dependences (in terms of distance vectors, and direction vectors) because of which that loop cannot tiled. [3 marks]

2. [3+4+5] **Liveness Analysis and Register Allocation**

(a) Write a C function with 4 local variables, such that each variable is live in exactly one line. [3 marks]

(b) Write C function with four local variables, such that the live range of each variable interferes with every other variable. [4 marks]

(c) Draw an interference graph with coalescing edges such that after a pass of simplification and coalescing (using Brigg's criteria) the graph can be colored in three colors (without needing spilling). The graph should not be two colorable. Further, the coalescing phase must remove at least one node from the graph. [5 marks]

3. [4+4+4] **Alias/Points-to/Escape Analysis**

(a) For the intra-procedural escape analysis studied in the class, state the transfer functions for different types of Java statements (remember: four important statements will do). [4 marks]

(b) Assume that for a given Java program, you are given the flow-sensitive may points-to analysis results. State how will you compute the may alias analysis results and must alias analysis results. [2+2 marks]

(c) Explain using a Java function how points-to analysis can help two different optimizations. [4 marks]

4. [8+4] **Control Flow Analysis**

(a) State how you will compute the CFG for any given Java function. [2 marks]. State how you will extend the same in the presence of Java Exceptions. Handle the following three Java constructs: `try`, `catch`, and `throw`. Assume that you can add a new type of control-flow edge called exception edges. Assume that points-to information is already available to you. [6 marks]

(b) Write a C function, where the algorithm discussed in the class to compute the CFG, will lead to the creation of a CFG containing a chain of exactly five basic-blocks. Show the CFG. [4 marks]

5. [4+4+4] **Call Graph Construction and Inter-procedural Analysis**

(a) Consider a subset of Java language with only static method members, which can only be invoked by referencing the class name (for example, `A.foo(...)`, where `A` is a class). Give an algorithm to build the call graph. [4 marks]

(b) Consider the following C code. In the context of flow-sensitive value-context-sensitive constant propagation, list the value contexts in which the function `foo` will be analyzed. [4 marks]

```
main(){
  x = bar (2, 3);
  y = foo (x-2, 2);
  x = foo(x, y);
  print ("%d", x);
}
int bar(int a, int b){
  return foo (a, b);
}
int foo(int p, int q){
  return p + q;
}
```

(c) Write a C code, which shows the precision improvement due to flow-insensitive context-sensitive analysis, compared to flow-insensitive context-insensitive analysis. [4 marks]

6. [4+4+4] **Potpourri**

(a) Many researchers convert input programs to SSA form and use flow-insensitive analysis (for scalability). Write a Java code which shows that despite the input code being in SSA form, flow-sensitive points-to analysis can lead to improved precision (compared to flow-insensitive analysis). [4 marks]

(b) Explain using a Java code the importance of weak-updates and strong-updates (why we should study both). [4 marks].

(c) Write the three-address code for the following C code, using the three-address-code IR discussed in the class. [4 marks]

```
x = 10;
y = 0;
do {
   x = x - y * 2;
   if (x == 6) continue;
   y = y + x - 8;
} while (x + y >= 10);
```