Final Exam 2008 (Advanced Compilers)

6/4 (Wed) 10:45 ~ 11:45 AM

Answer the problems briefly. Unnecessarily long (even correct) answers can get a penalty.

(1) Explain the phase ordering problem between the register allocation phase and instruction scheduling phase. More specifically, what are the advantages and disadvantages when we perform (a) instruction scheduling first, followed by register allocation, and (b) register allocation first followed by instruction scheduling.

(2) Even for while loops or for loops, we often locate the loop exit branch at the bottom of the loop body, not at the top, as in do-while loops. What could be the reason for this code layout?

(3) For a given initiation interval II, S(x) is defined as a cycle where x is scheduled in a modulo schedule. For a data dependence edge from a node u to v whose iteration difference is p and whose delay cycle is d, explain what the following inequality means and why it is correct: $S(v) - S(u) \geq d - II \times p$

(4) What is the cross-iteration anti-dependence problem in software pipelining? How is it handled in modulo scheduling?

(5) Some instruction scheduling techniques try to schedule branch instructions as early as possible, although such a code motion duplicate instructions on the way of code motion. What would be the impact of branch code motion, in terms of the usefulness of code motions performed after the branch code motion is performed? (when a branch is available for code motion, compare the following two scenarios: (a) we never schedule the branch, leaving it at its original location, but schedule other instructions first including those located after the branch, (b) we schedule the branch first and then schedule other instructions)

(6) Explain what data structures and object layouts are needed to accelerate the *virtual method calls* in object-oriented languages and how such a call is compiled.

(7) What is the main difference between the Chaitin's graph-coloring register allocator and Briggs' optimistic coloring register allocator?