Code Improvement

1. Describe several levels of "aggressiveness" in code improvement.

2. Give three examples of code improvements that must be performed in a particular order. Give two examples of code improvements that should probably be performed more than once (with other improvements in between).

3. What is a peephole optimization? Describe at least four different ways in which a peephole optimizer might transform a program.

4. What is a constant folding? Constant propagation? Copy propagation? Strength reduction?

5. What does it mean for a value in a register to be live?

6. What is a control flow graph? Why is it central to so many forms of global code improvement? How does it accomodate subroutine calls?

7. What is value numbering? What purpose does it serve?

8. Explain the connection between common subexpressions and expression rearrangement.

9. Why is it not practical in general for the programmer to eliminate common subexpressions a the source level?

10. What is static single assignment (SSA) form? Why is SSA form needed for global value numbering, but not for local value numbering?

11. What are merge functions in context of SSA form?

12. Give three distinct examples of data flow analysis. Explain the difference between forward and backward flow. Explain the difference between all-paths and any-path flow.

13. Explain the role of the In, Out, Gen, and Kill sets common to many examples of data flow analysis.

14. What is a partially redundant computation? Why might an algorithm to eliminate partial redundancies need to split an edge in a control flow graph?

15. What is an available expression?

16. What is forward substitution?

17. What is live variable analysis? What purpose does it serve?

18. Describe at least three instances in which code improvement algorithms must consider the possibility of aliases.

19. What is a loop invariant? A reaching definition?

20. Why might it sometimes be unsafe to hoist an invariant out of a loop?

21. What are induction variables? What is strength reduction?

22. What is control flow analysis? Why is it less important than it used to be?

23. What isregister pressure? Register spilling?

24. Is instruction scheduling a machine-independent code improvement technique? Explain.

25. Describe the creation and use of a dependence DAG. Explain the distinctions among flow, anti-, and output dependences.

26. Explain the tension between instruction scheduling and register allocation.

27. List several heuristics that might be used to prioeritize instructions to be scehduled.

28. What is the difference between loop unrolling and software pipelining? Explain why the latter may increase register pressure.

29. What is the purpose of loop interchange? Loop tiling (blocking)?

30. What are the potential benefits of loop distribution? Loop fusion? What is loop peeling?

31. What does it mean for loops to be perfectly nested? Why are perfect nests important?

32. What is a loop-carried dependence? Describe three loop transformations that may serve in some cases to eliminate such a dependence.

33. Describe the fundamental difference between the parellelization strategy for multicore machines and the parallelization strategy for vector machines.

34. What is self scheduling? When is it desirable?

35. What is the live range of a register? Why might it not be a contiguous range of instructions?

36. What is a register interference graph? What is its significance? Why do production compilers depend on heuristics (rather than precise solutions) for register allocation?

37. List three reasons why it might not be possible to treat architectural registers uniformaly for purposes of register allocation.