Note: This exam is a practice exam. The actual exam will have different questions.

1. Short Answers

Briefly define each of the following terms:

a. Extended basic block
b. Useless code
c. Dope vector
d. Translation lookaside buffer

2. Time

In the lectures and in the reading, we discussed many events. Some of those occur at compile-time (while the compiler is running) and some that occur at run-time (while the compiled code is running).

For each of the following events, does it occur at compile time or at runtime?

a. Spill code is created
b. Access link is traversed
c. Activation record is instantiated
d. Display is referenced
e. Static coordinate is created
f. Dependence graph is created
g. False zero is computed (A₀ is the “false zero” of A)
h. Dope vector is used
i. New lexical level is added to the symbol table
j. Tree pattern is matched
3. **Brief (but Deep) Answers**

*For each part, please limit your answer to a single page.*

a. The compiler must lay out, for each procedure, an activation record (AR). What information should it store in the AR? How does the compiler access data in the AR? How does it know the sizes of the various fields in the AR?

b. The compiler must choose between numerical representation of a Boolean value and positional representation of a Boolean value. For each, define how the representation works. Give two examples: one in which a numerical representation is better than a positional one, and another in which a numerical representation is worse than a positional one.

c. In an OOL, method invocations (function calls) that are resolved at runtime (e.g., virtual functions in C++) incur different kinds of costs than method invocations that are resolved at compile time. This difference is sometimes characterized as static dispatch versus dynamic dispatch. Explain the difference between these two and the impact that the difference has on the cost of a method invocation.

4. **Code Optimization**

One technique for eliminating redundancies in a basic block is called *Value Numbering*.

a. State the algorithm for value numbering a single basic block. Be sure to state any assumptions that you make about whether or not values are live before and after the block.

b. What is the complexity of your algorithm? On what does it depend?

   Further refinements to this algorithm have been proposed. In particular:

c. Explain how the basic algorithm can be extended to discover constant-valued expressions and to evaluate those expressions at compile time.

d. Explain how the basic algorithm can be extended to capitalize on algebraic identities. What additional costs are entailed in handling algebraic identities?

5. **Instruction Selection**

An instruction selector built around peephole matching uses three distinct phases: a code expander, a simplifier, and a matcher.

Explain the role of each phase and its operation.

Explain how a peephole instruction selector can improve the code in ways that a tree-pattern matching selector cannot.