COMP 412, Fall 2018
Practice Final Exam

This exam has questions drawn from several years. It is intended to show you the kinds of questions that I have asked in past years. It is not an actual final exam, but, instead, is a composite of other years' exams.

1. Short Answers

Briefly define the following terms:

a. Data-flow analysis

b. Basic block

c. Useless code

d. Dope vector

2. General Knowledge

a. For each pair of design decisions, explain the issue and the difference between the two alternatives. Give strengths and weaknesses of each. Please keep your answer brief; write no more than a paragraph on each alternative.

b. Null-termination versus explicit length in string representations

c. Heap-allocated activation records versus stack-allocated activation records

d. Numerical representation versus positional representation for Boolean values

e. Access links versus a global display

3. Brief (but Deep) Answers

For each part, please limit your answer to no more than two paragraphs. In most cases, one paragraph will suffice.

a. One step in a Chaitin-Briggs graph-coloring register allocator coalesces register-to-register copy operations. How does the allocator determine which copies it can coalesce and which ones must remain in the code?

b. In a compiler that uses list scheduling over extended basic blocks, some actions by the scheduler can require the insertion of compensation code. Describe two scenarios that require compensation code and explain what kind of code the scheduler must insert.

c. 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? What issues arise in providing storage for local data in the AR?

4. General Compilation

The BCPL language, developed at Cambridge University in the late 1960's and early 1970's, had the world's simplest type system. All values were of a single type.

a. What effects would you expect this decision about the type system to have on a BCPL compiler and its implementation?
b. Are there any features of modern microprocessors that would make the implementation of this single-type language more complex today?
c. The BCPL compiler used indirection vectors to implement array addressing. What were the advantages and disadvantages of this scheme in 1972? What are they today?

5. Code Shape

Assume that you are building a compiler for a very simple language. Array subscripts are indicated by square brackets. Arrays are stored in row-major order. Integers occupy four bytes. **A** is declared as an integer array of three dimensions, with bounds **A[1..m,2..n,3..p].**

Your compiler must generate code for the ILOC subset used in labs 1 and 3. You have as many registers as the compiler needs.

d. What code would you generate for an array reference, on the right-hand side of an assignment, for **A[i,j,k]**?
e. How would the code change if **A** was passed into the current procedure as a call-by-reference formal parameter?