Computer Science 320: Final Examination

May 14, 2003

You have up to 3 hours to answer the following questions. This final exam is closed book/closed notes. For partial credit, show all work. State all assumptions. All work must be done in your exam booklet. Write out and sign the Honor Code pledge before turning in the test.

"I pledge my honor that I have not violated the Honor Code during this examination."

1 Lexical Analysis

Do the following:

1. Write a regular expression that describes all strings of alternating a's and b's where the number of a's is even (including 0).

2. Build a *Deterministic Finite Automaton* (DFA) that recognizes this regular expression.

2 Instruction Selection

Intel has come up with a new, super-fast arithmetic chip instruction set. It is a CISC chip in which the operations have widely differing costs. The table below lists the important arithmetic instructions and their costs. In the table below, r_1, r_2, r_3, etc. are registers whereas c is a constant.

instruction:	cost:
$r_1 = r_2 + r_3$	1
$r_1 = r_2 + r_3 + r_4$	3
$r_1 = r_2 + c$	3
$r_1 = r_2 * r_3$	5
$r_1 = r_2 * r_3 * r_4$	8
$r_1 = r_2 + r_3 * r_4$	5
$r_1 = c$	1

Answer the following questions. Assume you are doing code generation starting with a tree-like intermediate language with binary (two-argument) arithmetic operations, as in the textbook (or the Tiger compiler intermediate representation).

1. Write down the "tree patterns" that match each of the instructions listed above.

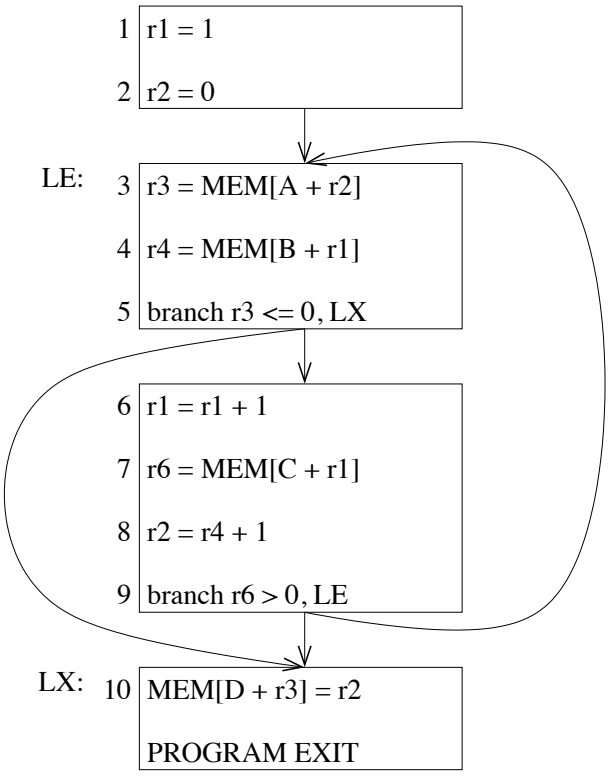
2. Find the lowest cost sequence of instructions possible (the "optimum" sequence) to compute the following arithmetic expression, assuming that all variables are already in registers.

$$(x * ((3 + y) * 17)) + ((y * x) + 14)$$

3. Explain how to extend the algorithm for finding the "optimum" cost sequence of instructions so that it does common sub-expression elimination efficiently at the same time as instruction selection and generation.

4. Create an example arithmetic expression with some common subexpressions and show how your new algorithm works on that expression.

3 Live Variable Analysis/Register Allocation



Do the following:

1. Compute the LIVE IN and LIVE OUT sets for instructions $1 \rightarrow 10$ in the code above. Show each step of the dataflow analysis.
2. Draw the interference graph.
3. Show how to allocate registers by graph coloring this example.
4. Suggest a change in the insruction order which would reduce the number of registers necessary.

4 Optimization

You're the professor for a compiler construction class, and you have to write a final exam question on code optimization. Create a small code example for the exam which reduces to:

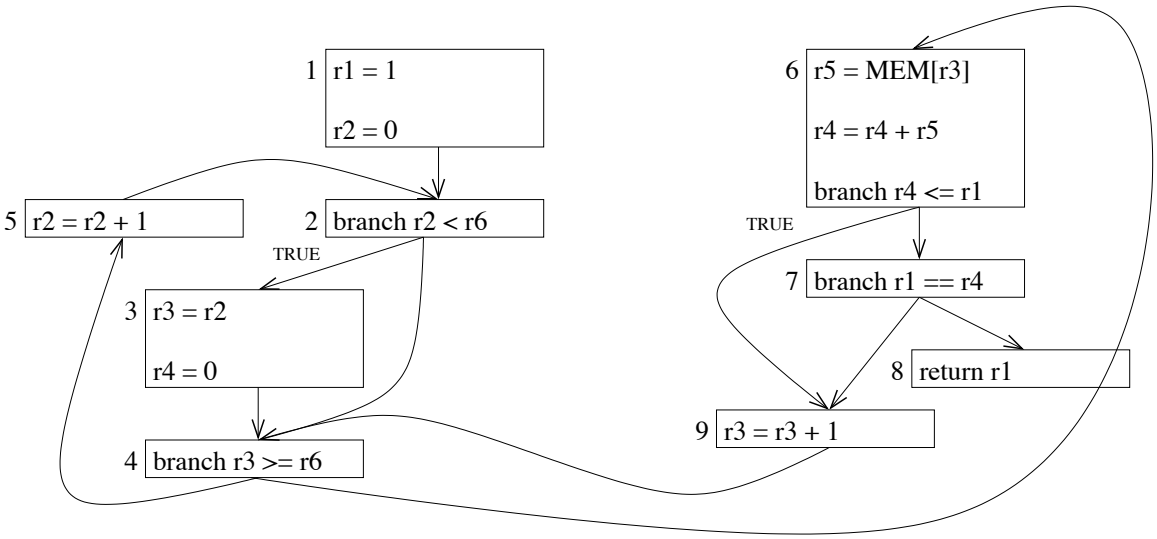
```
return 100
```

after the following optimizations are applied (only once) in this order:

1. Common Subexpression Elimination
2. Copy Propagation
3. Dead Code Elimination
4. Constant Folding
5. Constant Propagation
6. Constant Folding
7. Constant Propagation
8. Dead Code Elimination

Create the answer key by showing the code before and after each optimization step. Each optimization should have some real impact (ie: your initial program should not just be `return 100`!!) Remember, the smaller the original code example, the easier it is for you to grade it. Hint: work backwards.

5 Static Single Assignment



Convert this program to SSA form. Show your work after each stage.

1. Add a start node containing initializations of all registers.
2. Draw the dominator tree.
3. Insert ϕ-functions and rename temporaries to create SSA.
4. SSA form is somewhat complicated when compared with ordinary control-flow graph representations of programs. Briefly explain why a compiler writer would want to use SSA rather than other common representations of programs. For full credit, give an example program that illustrates one of the benefits of SSA as opposed to other representation techniques.

6 Garbage Collection

1. At run time, when does a value such as a record or an array become garbage?

2. Explain why no one has developed the perfect garbage collector yet.

3. Describe the steps in mark and sweep garbage collection. (Use at most 2 paragraphs)

4. In Java, every object supports a `hashCode` method that returns an integer with the following properties.

   (a) Different objects may return the same integer, but two objects selected at random should have only a small chance of having the same hash code.

   (b) Each object must return the same hash code every time the function is called.

   The Java language specification says that "This is typically implemented by converting the address of the object to an integer..." What implications does this implementation technique have on the construction of a garbage collector for the language?

7 Object-Oriented Language Implementation

1. What is the difference between a multiple-inheritence language and a single-inheritence language?

2. In three or four sentences, explain the impact that multiple-inheritence has on compiling OO-languages (make at least two different points in your discussion).

3. Consider the partial class definitions given below.

```
class A extends Object {var a := 0; var b := 0}

class B extends A {var c := 0}

class C extends Object {var d := 0}

class D extends B,C {var e := 0; var f := 0}

class E extends A {var g := 0}

class F extends C {var h := 0}
```

   Draw pictures that illustrate how objects of each class can be represented. Your representation should be efficient (assume that in general, there will be many objects for each class).

4. Explain the steps (instructions) involved in extracting the value in any given field from an object given your object representation.