EECS 583 – Winter 05 – Midterm Exam

Monday, April 4, 2005

2 hours: open book, open notes

Name: _____

Please sign indicating that you have upheld the Engineering Honor Code at the University of Michigan.

"I have neither given nor received aid on this examination."

Signature: _____

There are 14 questions divided into 2 sections. The point value for each question is specified with that question. Please show your work unless the answer is obvious. If you need more space, you can use the back side of the exam sheets or there are 2 blank pages at the end of the exam.

Part I: Short Answer

7 questions, 35 pts total Score:_____

Part II: Problems

7 questions, 115 pts total Score:_____

Total (150 possible): _____

Part I. Short Answer (Questions 1-7)

1. A common warning message in compilers is the use of a possibly uninitialized variable. Explain briefly how such variables are detected using standard dataflow analysis. (5 pts)

2. Register renaming is a crucial optimization for speculating operations across branches, why is this? (5 pts)

3. In trace selection, the lengths of traces are often artificially limited by compiler heuristics that examine branch probabilities. What is the purpose of limiting the length of traces? (5 pts)

4. Backedge coalescing is often heuristically performed in loops to enable more branches to be removed via if-conversion. However, it can result in a performance loss if applied blindly. Briefly describe a situation in which a performance loss is likely to occur. (5 pts)

5. Using loop invariant code motion, a load instruction cannot be hoisted out of a loop unless some form of architectural support is provided for speculative loads (i.e., general, sentinel, or some other speculation model). Is this statement True or False, briefly explain. (5 pts)

6. The bottom-up greedy (BUG) algorithm for multicluster partitioning uses two passes to perform cluster assignment. Briefly explain why a single pass algorithm is likely not to work as well. (5 pts)

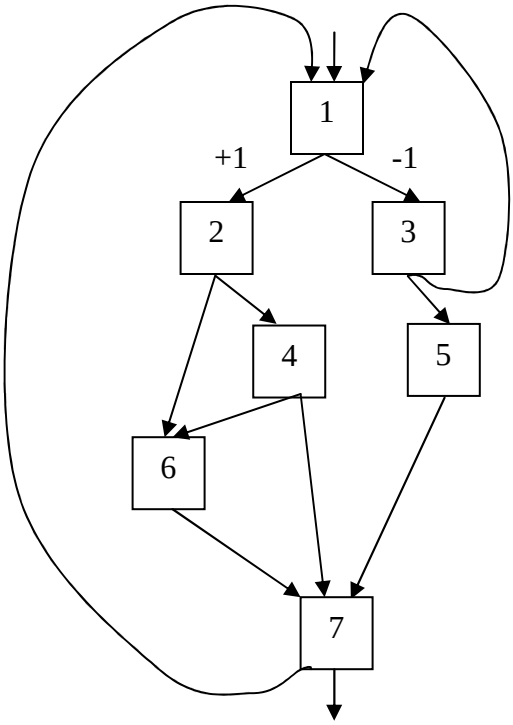
7. One of the benefits of superblock formation is that it exposes more optimization opportunities via tail duplication. Give an example that shows a classical optimization that is made possible by superblock formation (thus it should not possible prior to tail duplication). (5 pts)

Part II. Problems (Questions 8-14)

Question 8 has 2 parts that use the control flow graph given below. (15 pts)

8a. Compute the control dependence set (CD) set for each basic block. Out of each basic block, the leftmost edge is considered +BBid and the rightmost edge is –BBid. (5 pts)

8b. If-convert the control flow graph removing as much control flow as can be eliminated (note: do not apply backedge coalescing). Create a linear listing of pseudo code that shows: the CMPP operations, just label each condition using the block number, i.e., p1,p2 = CMPP.UN.UC(C1) if p3; and, create a representative instruction for each basic block to show the predicate assigned to each block, i.e., bb3 if p4. Remember to indicate whether each predicate is OR-type or Unconditional-type. (10 pts)



CD(1) =
CD(2) =
CD(3) =
CD(4) =
CD(5) =
CD(6) =
CD(7) =

9. Suppose that you were given a new dataflow problem: at a point *p* in a program, the set *X* is defined as the set of expressions (right hand side of an operation) that are computed and not killed along <u>at least one path</u> from *p* to EXIT. An expression is killed when any of its operands are redefined. Define the GEN, KILL, IN, and OUT sets to compute the set *X* at all entry/exit edges of basic blocks. (12 pts)

10. In the code segment below, the compiler is able to determine the facts to the right using predicate analysis about possible legal transformations to the code segment. Working backwards from these facts, list all the predicate relationship information that you can deduce. (8 pts)

```
1: r1 = r2 if p1
2: r3 = r1 if p2
3: r4 = r1 + 1 if p3
4: r5 = r4 + r3 if p4
```

* Forward copy propagation can be applied between ops 1 and 2
* Forward copy propagation can be applied between ops 2 and 4
* Ops 3 and 4 can be freely reordered during scheduling

11. You are compiler writer developing a new optimizer and you are experiencing some strange bugs when your optimizations are repeatedly applied. You decide to create a manual assembly testcase to debug your optimizer. Create a small pseudo assembly function which after optimization is complete, reduces to:

```
r1 = r2 + r3
store (r1, 0)
```

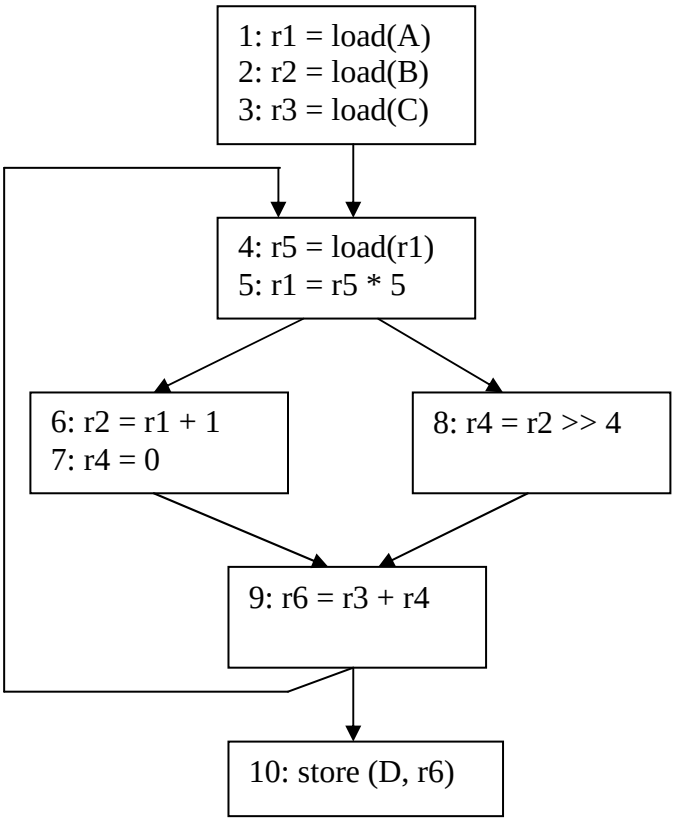
after the following optimizations are applied in this order:

1. Common subexpression elimination
2. Copy propagation
3. Common subexpression elimination
4. Copy propagation
5. Dead code elimination

Note, you should assume that each invocation of your optimizer exhaustively applies the optimization in question before moving onto the next optimization. Show a snapshot of the code before/after each optimization. (15 pts)

12. Consider applying register allocation to the following program segment. Note that A B, C, and D are symbolic addresses that do not require registers. (15 pts)

12a. Compute the merged liveranges. Express each as a set of operations. (8 pts)
12b. What is the fewest number of physical registers needed for this program segment such that no spills will be generated? (7 pts)



13. You are a highly sought after compiler consultant who has been hired by a new startup company that is building a VLIW processor. They don't want to tell you the details of their processor, so they provide you with an abstract description of their machine (see table below). You are further told that each resource is fully pipelined. Their most critical application is the assembly below on the right. (20 pts)

Opcode	Latency	Resources
Add	1	2 X, 1 Y
Mpy	2	1 X, 3 Y
Div	3	2 X, 3 Y
Load	2	1 X, 2 Z

Total resources: 3 X, 3 Y, 3 Z

```
1: r1 = load (_A)
2: r2 = load (_B)
3: r3 = load (_C)
4: r4 = load (_D)
5: r5 = r1 / r2
6: r6 = r3 * r4
7: r7 = r2 + 12
8: r8 = r5 + r6
9: r9 = r7 + r8
```

13a. Show the resulting schedule for the company's current VLIW processor. You can assume the priority is given by the linear order of the operations. (5 pts)
13b. What is the fewest cycles that this application could be completed in if the number of resources was unconstrained? (5 pts)
13c. Is the current machine over-provisioned in any way? If so, what resource do you recommend cutting back to reduce cost that does not affect performance? (5 pts)
13d. They have enough spare area to increase the number of one of the resources (X, Y, or Z) by one. What is your recommendation to maximize their performance gain? (5 pts)

14. Given the following loop in DSA form and the associated machine model, answer the following questions related to modulo scheduling. (30 pts)

```
1: r2[-1] = load(r1[0])
2: r3[-1] = load(r2[1])
3: r4[-1] = r4[2] * r3[0]
4: store (r1[0], r4[-1])
5: r1[-1] = r1[0] + 4
6: brlc Loop
```

Processor model
4 fully pipelined function units
1 ALU, 2 MEM, 1 BR

Latencies (all srcs read at time 0)
add: 1
load: 2
mpy: 4

14a. Draw the dependence graph using the notation <delay,dist> on each edge. (5 pts)
Note: There are <0,0> anti dependences between ops 1-5 and 4-5. Also, the first operand of the store is the address and it is known there are no aliases between op2 and op4.

14b. Calculate ResMII, RecMII, and the MII. (5 pts)

14c. Generate the modulo schedule. Assume the priority of the operations is the linear order. Show both the unrolled and rolled schedules, along with the final assembly code with staging predicates assigned. (15 pts)

14d. Assuming the loop has 100 iterations, how many cycles does the loop run in total? (5 pts)

Extra space

Extra space (2)