EECS 583 – Fall 2020 – Midterm Exam

Open: Monday, November 2, 2020 10:30AM EST
Close: Monday, November 3, 2020 10:30AM EST
Time constraint: 3 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 12 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, just use additional sheets of paper. Just be sure to clearly mark additional sheets with the problem number.

Part I: Short Answer
6 questions, 30 pts total Score:_____

Part II: Medium Problems
6 questions, 70 pts total Score:_____

Total (100 possible): _______

Part I. Short Answer (Questions 1-6) (30 pts)

1) Profiling offers many advantages including identification of hot code or frequent behaviors in an application. But, it also suffers from some potential *pitfalls*, name one of those pitfalls. (5 pts).

2) It is often possible to improve the performance of a loop limited by *RecMII* by adding resources to the processor. Is the preceding statement True or False? Justify your answer. (5 pts)

3) Suppose that you have defined the GEN, KILL, IN, and OUT sets for a *top-down any-path* dataflow analysis. What is the primary change required to convert this analysis to a *top-down all-path*? (5 pts)

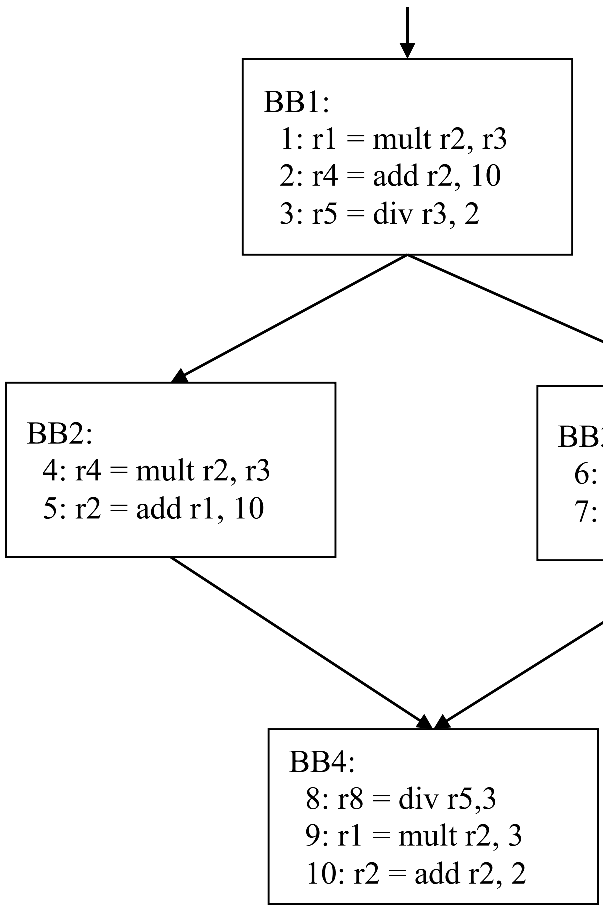
4) Name one performance advantage of loop unrolling. (5 pts)

5) Can an instruction with a smaller Estart be scheduled after an instruction with a larger Estart? Justify your answer. (5 pts)

6) A backedge is defined as a control flow edge where the target basic block dominates the source. Can backedge be equivalently defined as a control flow edge where the source basic block post dominates the target? Briefly explain your answer. (5 pts)

Part II. Medium Problems (Questions 7-12) (70 pts)

7) Compute the Available Expression GEN/KILL/IN/OUT sets at BB4. Assume r2, r3, r6 are properly initialized before entering BB1. (10 pts)

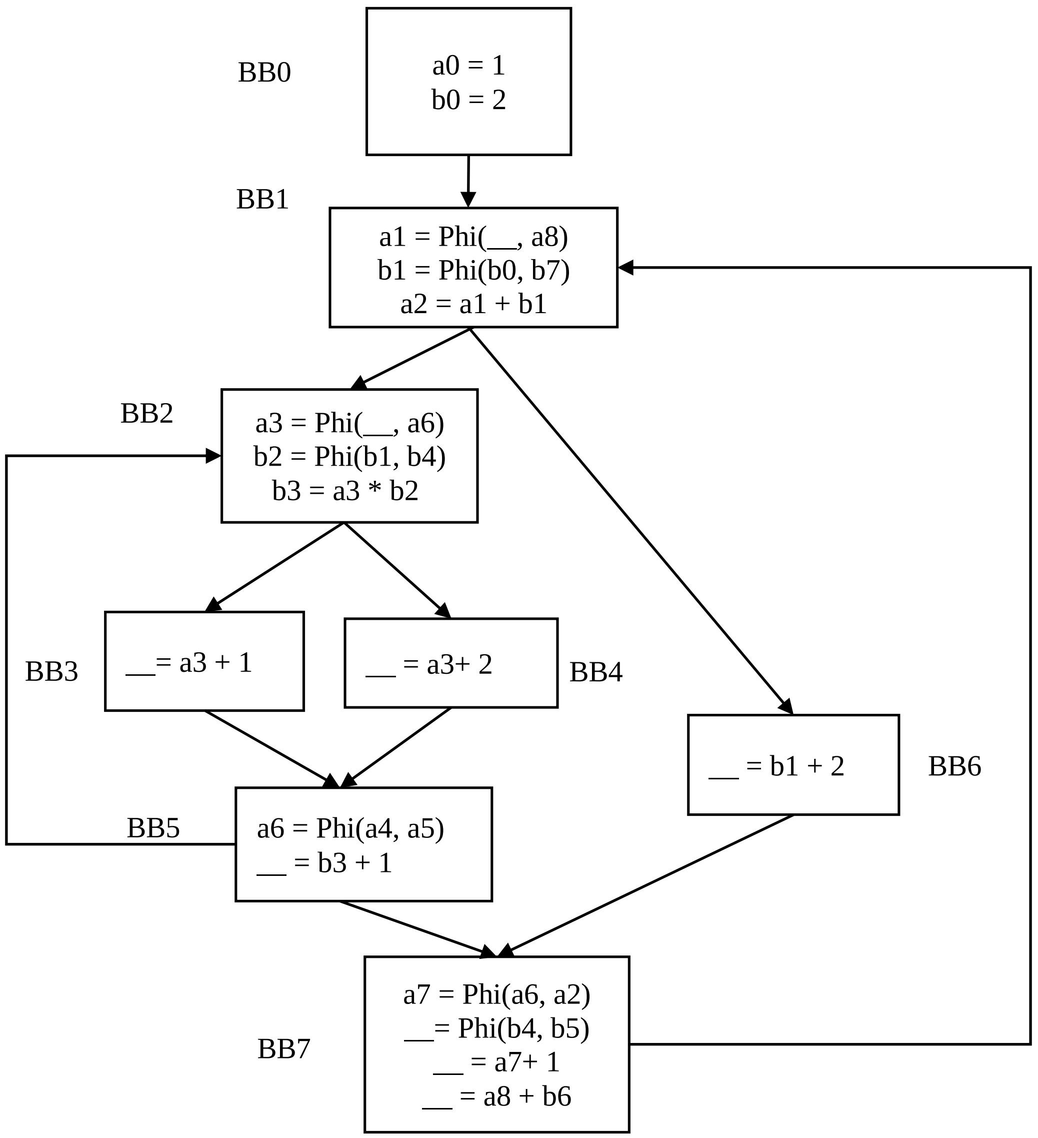


8) Draw the control flow graph (CFG) and determine the *minimum* number of predicates required to if-convert the following code. Justify your answers. (10 pts)

```
do{
   if (a>0 && b>0){
     a -= 1;
     if(c>0){
       b -= a;
     }
     c -= 1;
   }else{
     a += 1
   }
   b += a;
}while(b>10);
```

9) Satisfy static single assignment (SSA) form by **filling in the blanks with variables from the below pool**. The result and arguments of a Phi node must be all the same variable (i.e., a1 = Phi(a0, a4)). Solving by inspection is fine. (15 pts)

Variable pool: a0, a2, a4, a5, a8; b3, b4, b5, b6, b7 (*All variables can only be used once*)

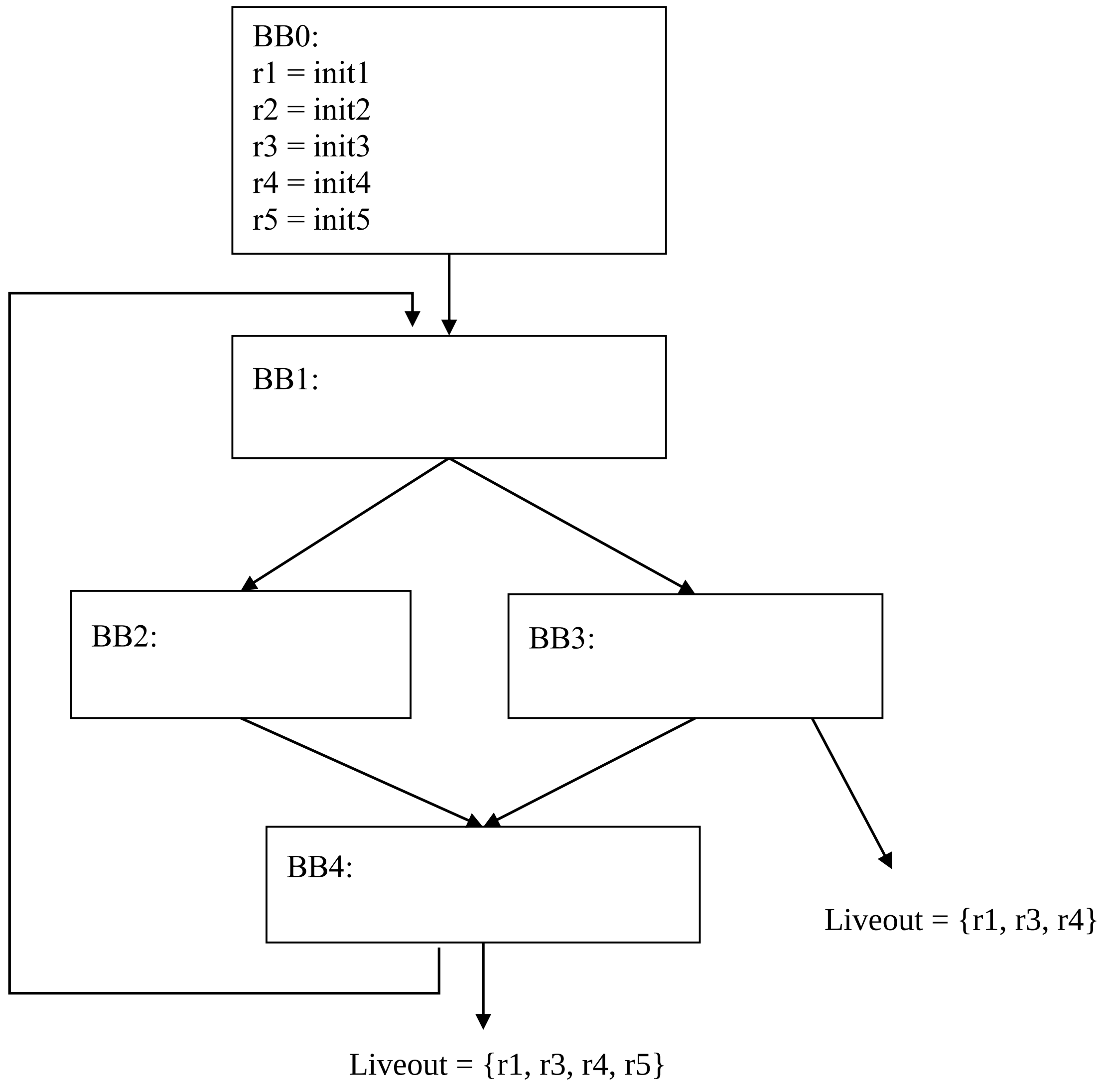


10) You developed a new LICM optimization in your compiler and want to create a testcase to validate your optimization. For your testcase, you will use the loop below with 4 additional instructions that you will insert such that 2 of the instructions can be hoisted to BB0 as invariant and the other 2 cannot be hoisted because they violate one of the LICM conditions.

The new instructions are:

a. r1 = add(r1, 1)
b. r3 = add(r2, 1)
c. r4 = add(r2, 1)
d. r5 = add(r3, 1)

Each new instruction must go to a *separate BB* (i.e., one instruction in BB1, one in BB2, one in BB3 and one in BB4). Show where they should be inserted on the diagram below such that the code before and after LICM yields the same results. Hint: Remember to pay attention to live-out registers. (10 pts.)



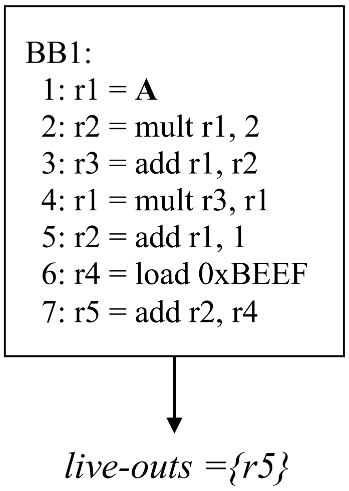
11) You are building a compiler where the goal is to extend conventional constant propagation and folding by speculatively applying them using profiling data similar to Homework 2. Speculative constant propagation propagates statistically likely constants to subsequent uses and enables speculative simplification using constant folding. (10 pts)

Your primary testcase is the program segment below that initializes r1 with a constant value, A. The profiling data shows that *A=10 with 90% probability*.

a. How should the compiler transform the code below to apply speculative constant propagation/folding? Show the optimized assembly and repair code. (7 pts)

b. What is the expected instruction savings for this example? (3 pts)

Note, *when counting saved instructions, DO NOT consider operation type but be sure to include any checking/repair instructions.* For example, if you could save 10 loads and 5 add instructions by inserting 1 comparison followed by 1 branch, your expected saving would be 10+5-1-1=13.



```
BB1:
  1: r1 = A
  2: r2 = mult r1, 2
  3: r3 = add r1, r2
  4: r1 = mult r3, r1
  5: r2 = add r1, 1
  6: r4 = load 0xBEEF
  7: r5 = add r2, r4
```

live-outs ={r5}

12) Given the modulo scheduled loop on the left that achieved II=MII, answer the following questions. Note that the scheduled code is just listed in linear order even though multiple instructions may execute each cycle. (15 pts)

```
Loop:
  1: r4[-1] = load(r2[0]) if p1[0]
  2: r6[-1] = load(r3[0]) if p1[0]
  3: r2[-1] = add r2[0], 4 if p1[0]
  4: r3[-1] = add r3[0], 4 if p1[0]
  5: r5[-1] = mult r4[-1], 30  if p1[0]
  6: r7[-1] = add r5[-1], r6[-1] if p1[1]
  7: store (r2[-1], r7[-1]) if p1[1]
  remap r1,r2,r3,…
  8: brlc Loop
```

Processor model

<u>Resources</u>
4 issue, 4 fully pipelined units: 2 ALUs, 2 MEMs. **Note, branch (Instr 8) uses ALUs.**

<u>Delays</u>
mult =3 cycles
add = 1 cycle
load = 1 cycles
store = 1 cycle
br = 1 cycle

* store(*X,Y*) saves value of *Y* to memory location *X*.

(a) Determine MII and explain whether the given code is resource constrained or recurrence constrained. (10 pts)

(b) How many cycles would it take to finish the loop? Assume LC = 99 (i.e., 100 iterations). (5 pts)