EECS 583 – Fall 2019 – Midterm Exam

Wednesday, November 13, 2019; 10:40am-12:20pm
Friday, November 15, 2019; 10:40am-12:20pm
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 11 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, use the back side of the exam sheets.

Part I: Short Answer
5 questions, 25 pts total Score:_____

Part II: Medium Problems
6 questions, 75 pts total Score:_____

Total (100 possible): _____

Part I. Short Answer (Questions 1-5) (25 pts)

1) Name a way that a compiler can use profiling information to improve performance. (5 pts)

2) When is a control flow edge from basic block A to B (A→B) a backedge? (5 pts)

3) Why is it necessary to prove that an instruction will not cause an exception (Restriction 2) when performing upward code motion? (5 pts)

4) If any instruction in a basic block is scheduled later than its Lstart, then the schedule length for the basic block will be longer than the critical path length. Is the preceding statement True or False? Explain your answer. (5 pts)

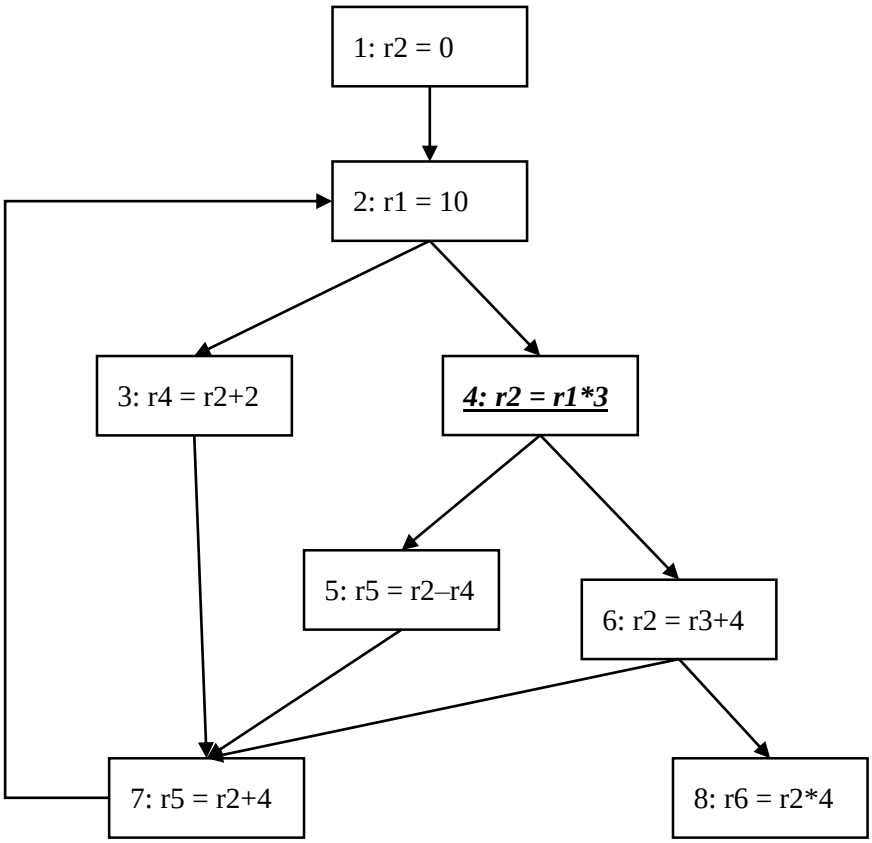
5) A basic block can only be control dependent on an immediate predecessor block. Is the preceding statement True or False? Explain why True or give counter example if False (5 pts)

Part II. Medium Problems (Questions 6-11) (75 pts)

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

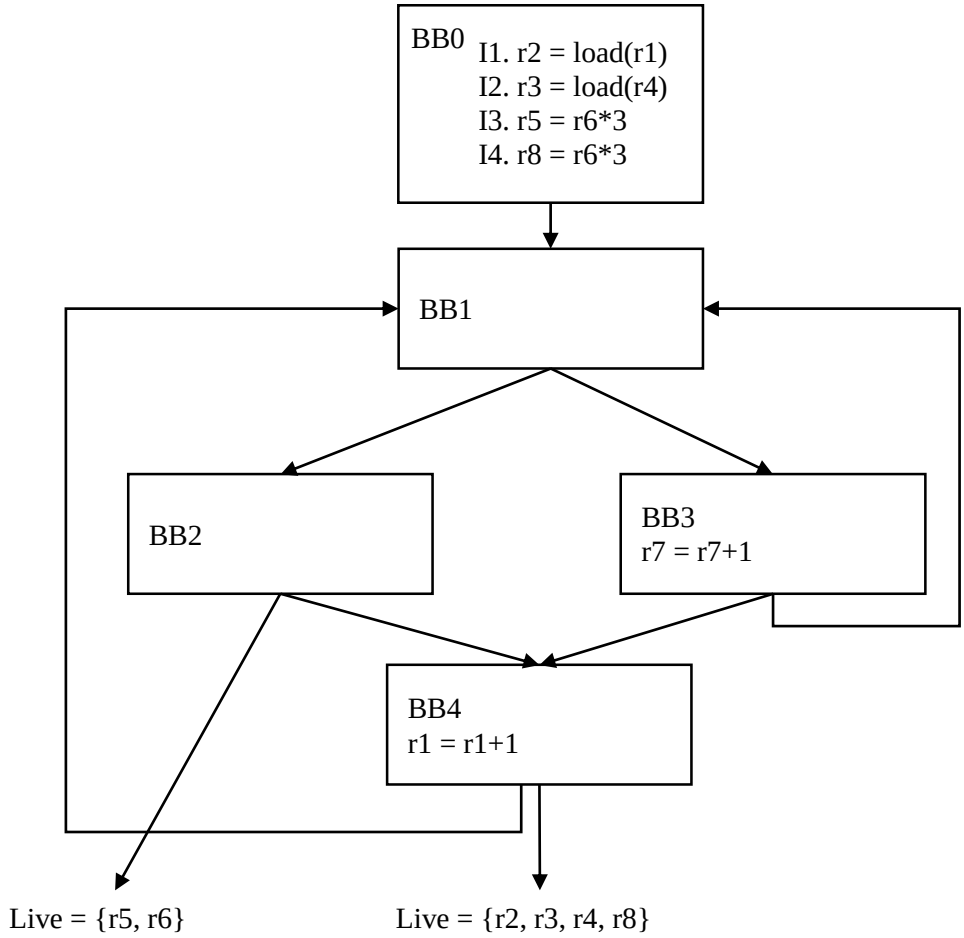
```
do{
   if (a>0 && b>0){
      if (c>0)
         x+=1;
      else
         x+=2;
      z=x/3;
   }else{
      y+=1;
   }
}while(z<100)
```

7) Find *all uses* of r2 defined by instruction 4 (i.e., DU chain for instruction 4). You may solve by inspection. (10 pts)



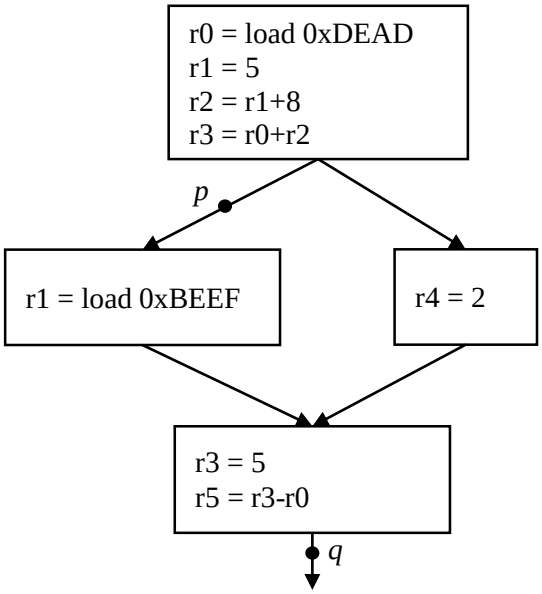
8) You are trying to reverse engineer some optimized assembly code to determine the original locations of instructions before optimization. In the following loop consisting of 4 basic blocks (BB1-BB4), the preheader (BB0) contains 4 instructions (I1, I2, I3, I4) that were possibly removed from the loop using LICM. For each instruction, determine whether LICM could have been legally applied and if so, which basic block(s) the instruction could have originally resided. Circle the correct answers. (10 pts)

I1	LICM legal?	Yes	No	Possible original blocks:	BB1	BB2	BB3	BB4
I2	LICM legal?	Yes	No	Possible original blocks:	BB1	BB2	BB3	BB4
I3	LICM legal?	Yes	No	Possible original blocks:	BB1	BB2	BB3	BB4
I4	LICM legal?	Yes	No	Possible original blocks:	BB1	BB2	BB3	BB4



9) You are building a new dataflow analysis that can identify trusted variables. To be conservative, we assume all values loaded from memory to be untrusted and any variables consumed in a block that are defined in another basic block to also be untrusted. Constants are assumed trusted. For other types of instructions, the destination register is trusted if all of its source operands contain known trusted values. *For your compiler, build a dataflow analysis pass that identifies trusted variables at a certain point p.* (15 pts)

For example,



At point *p*, {r1,r2} are trusted while {r2,r3} are trusted at point *q*.

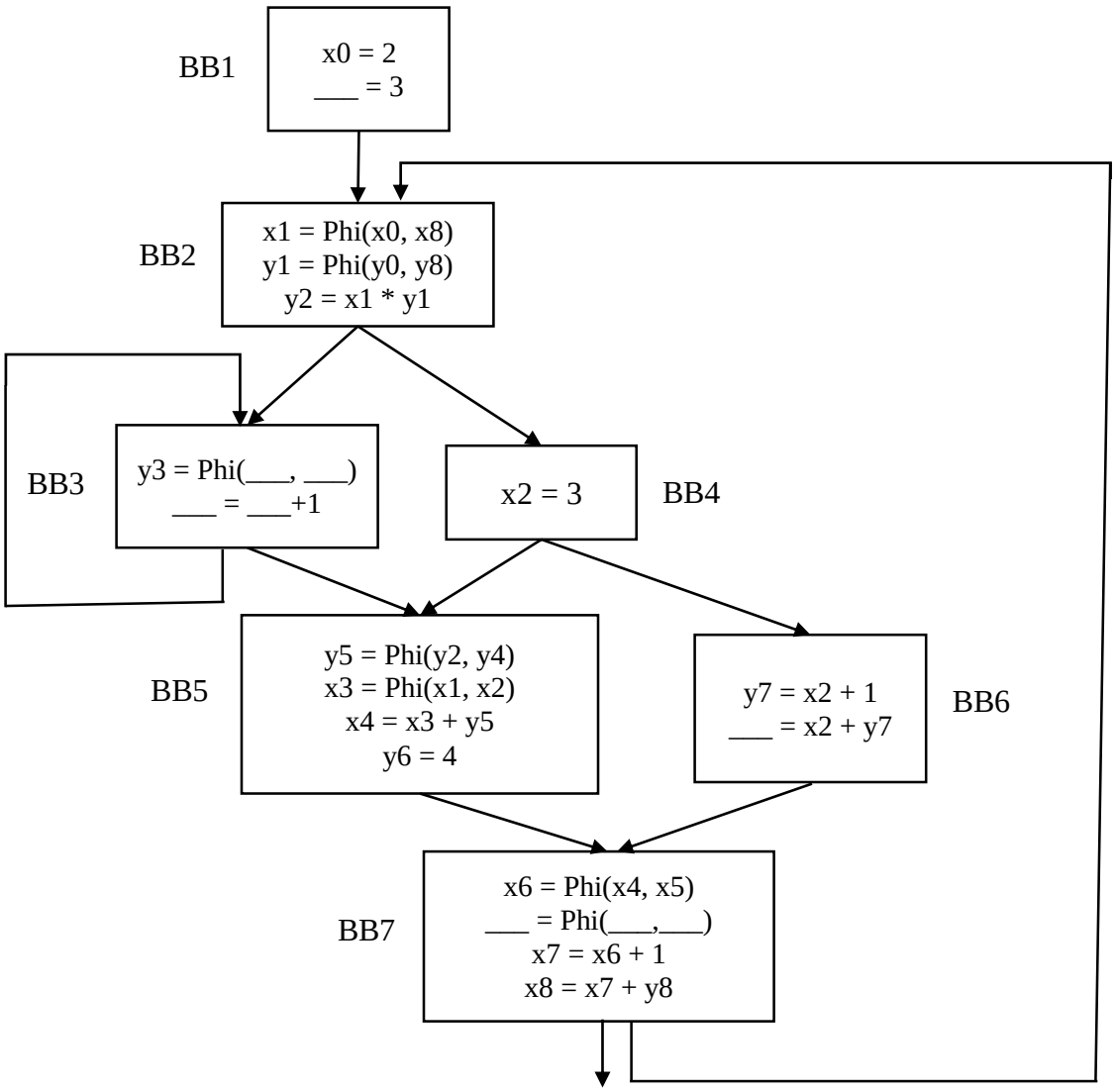
(a) Is this a forward or backward data flow analysis problem?

(b) Is this an all-path or any-path dataflow analysis problem?

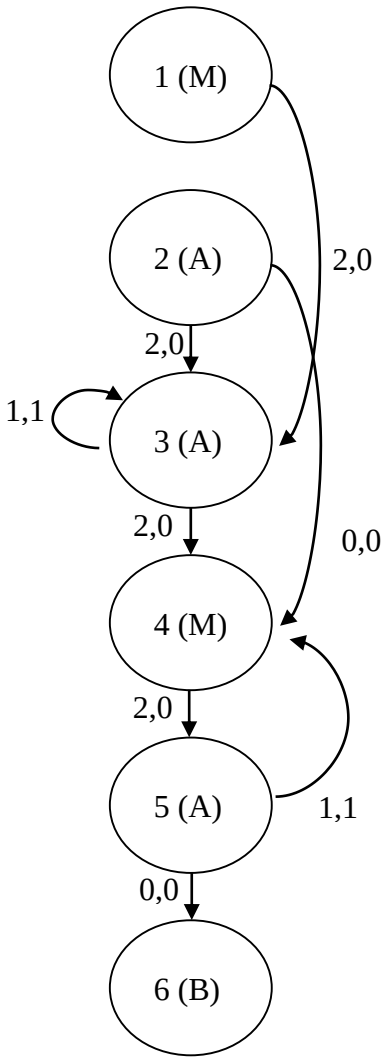
(c) Define GEN and KILL sets to identify trusted variables.

10) 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 either be all x's or all y's (i.e. x5 = Phi(x0, x4)). Solving by inspection is fine. (15 pts)

Variable Pool (each can be used more than once):
x5, y0, y2, y3, y4, y6, y7, y8



11) Compute the ResMII, RecMII, and MII for following dependence graph and processor model. Then, **generate the MII modulo schedule. Show the unrolled and rolled schedules for your answer.** You can assume that instruction 1 is the highest priority, 2 is second, etc. You do not need to assign staging predicates. **Also, calculate how many cycles it takes to execute 100 iterations of the loop.** (15 pts)



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

Instructions 1 and 4 are memory
Instructions 2, 3, 5, and 6 use the ALU
Instruction 6 is a branch