EECS 583 – Fall 2018 – Midterm Exam

Wednesday, November 14, 2018; 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, 20 pts total Score:_____

Part II: Medium Problems
6 questions, 80 pts total Score:_____

Total (100 possible): _____

Part I. Short Answer (Questions 1-5) (20 pts)

1) What is the main difference between any-path and all-path dataflow analysis? (4 pts)

2) Name a *forward* dataflow analysis discussed in class and an *optimization* that uses those analysis results to identify optimization opportunities. (4 pts)

3) When a compiler scheduler wants to speculate an instruction, name one issue that it must consider to preserve correctness of the resulting code. (4 pts)

4) A compiler can schedule an instruction before its Estart time if there are sufficient resources. Is this statement True or False? Briefly explain. (4 pts)

5) In a register interference graph, the node with the largest number of neighbors always determines the number of registers required to color the graph without spilling. Is this statement True or False? Briefly explain. (4 pts)

Part II. Medium Problems (Questions 6-11) (80 pts)

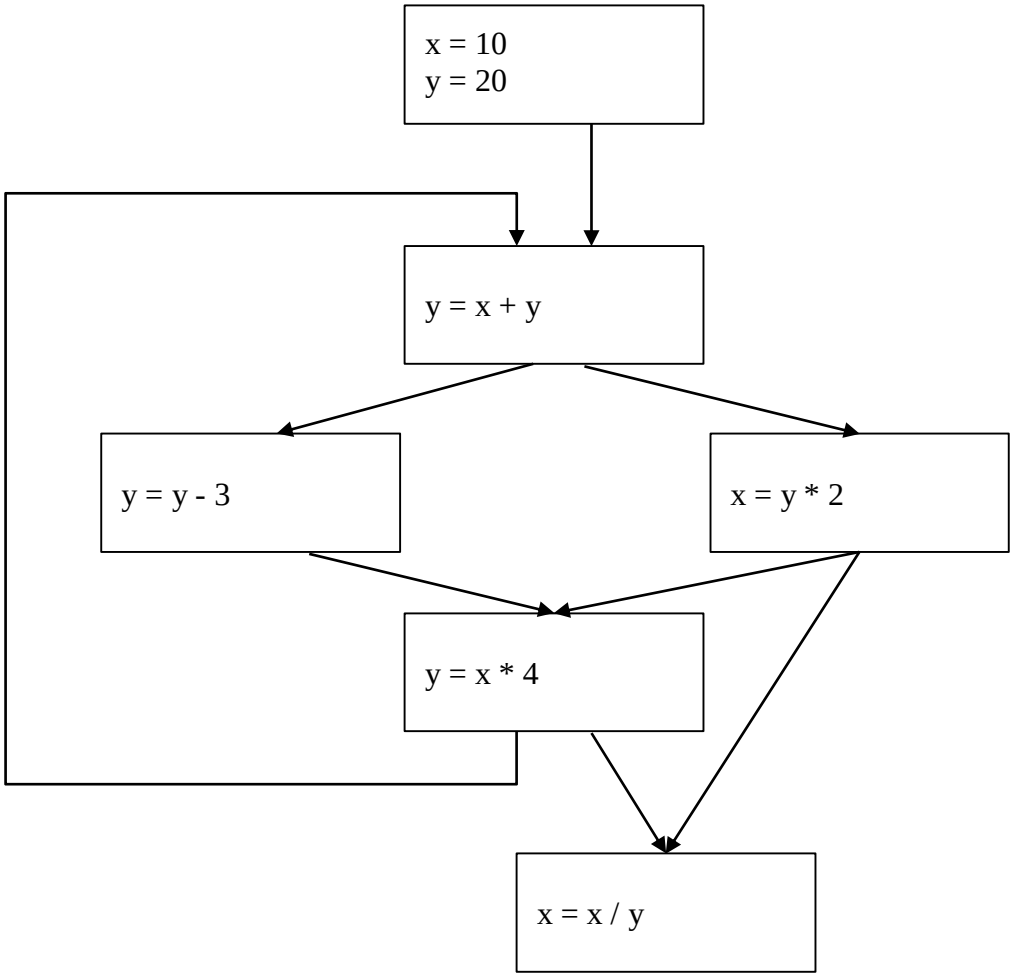
6) Draw a control flow graph (CFG) consisting of 5 nodes (A, B, C, D, and E) that satisfies the 4 properties below. (10 pts)

I. A dominates all nodes.
II. B only dominates C and itself.
III. D only post dominates C and itself.
IV. E post dominates all nodes

7) 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 {
    a = load(x++);
    if (a > 0)
        x += 2;
    else if ((b>0) && (c>0))
        c--;
    else
        b--;
} while (x < 100);
```

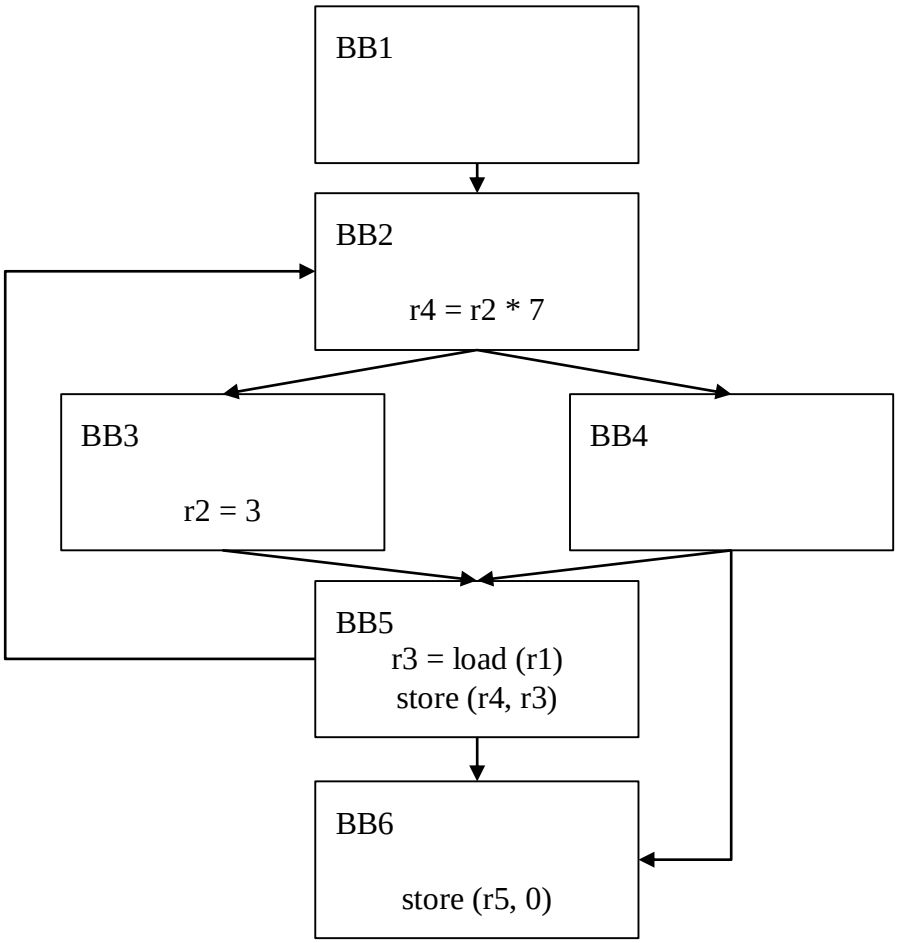
8) Convert the following program segment into static single assignment (SSA) form. You should perform the necessary renames and show the Phi nodes. Solving by inspection is fine and you can put your solution directly on the diagram. (15 pts)



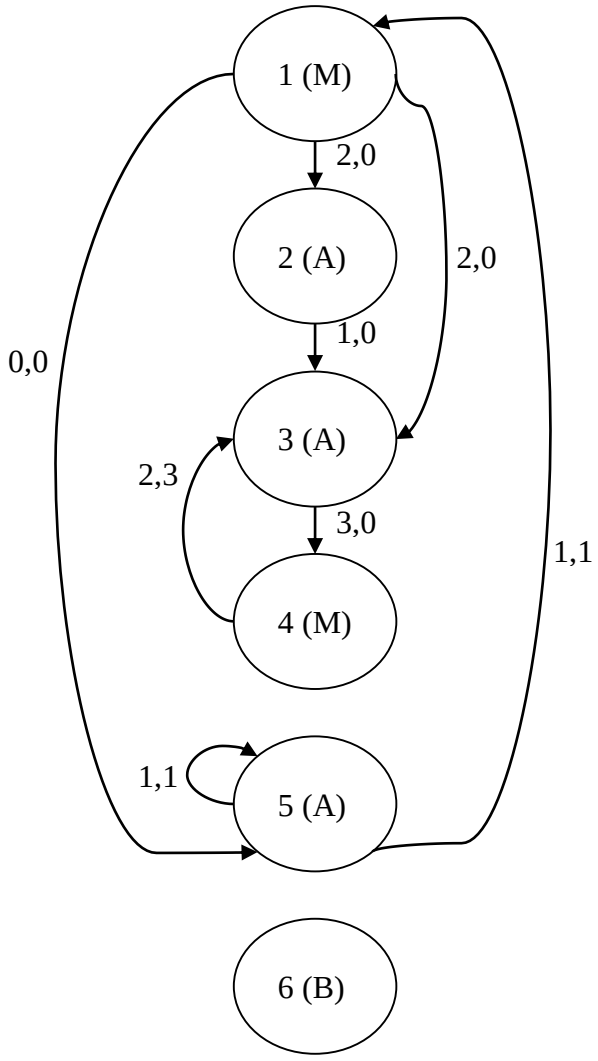
9) In the following control flow graph (CFG), place the 4 instructions (I-IV) such that the following conditions are met: at most 1 instruction is added to each basic block (BB); each instruction is placed at the beginning of a BB; after correctly placing all the instructions, the following optimizations should be applicable to the resulting code at least one time each: a) Forward Copy Propagation, b) Common Subexpression Elimination, c) Loop-Invariant Code Motion, and d) Dead Code Elimination. Do not worry about the impact of where you place the instructions on the execution results of the code segment. You may assume any relevant registers are properly initialized and the loads/stores are known to go to different addresses. (15 pts)

Note: Store instructions use the syntax: store(address, data)

I. r1 = r1 + 1
II. r5 = r8 * 7
III. r8 = r2
IV. r2 = r4 + 1



10) 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. (15 pts)

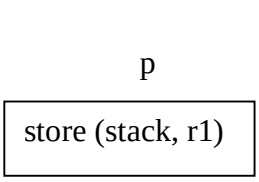


Processor model
6 fully pipelined function units
3 ALU, 2 MEM, 1 BR

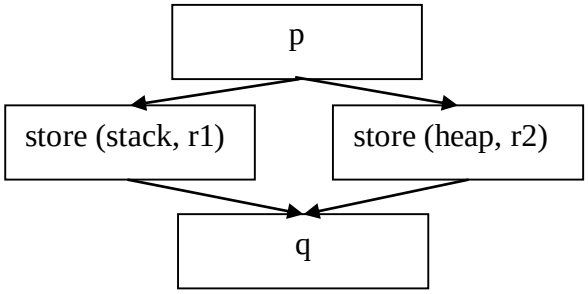
Instructions 1 and 4 are memory
Instructions 2, 3, and 5 use the ALU
Instruction 6 is the branch

11) You are building a new compiler where you will have no memory dependence analysis capabilities. The only intelligence that you have is that you can differentiate stack/heap accesses. To enable some optimizations of loads (i.e., CSE or LICM), you decide to build a dataflow analysis pass to summarize the presence of heap/stack store instructions to avoid scanning basic blocks repeatedly. Your dataflow is defined as follows: *A heap (stack) store is present if there exists at least one store to the heap (stack) starting from p and ending at q*. When no heap (stack) store is present, you will be able to freely optimize loads between p and q. (15 pts)

For example,



Stack store is present



Stack store is present
Heap store is present

Answer the following questions about this dataflow problem:

(a) Is this a forward or backward dataflow analysis problem?

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

(c) Define GEN and KILL sets to compute store presence. *Hint: Consider the use of special variables: heap and stack.*