EECS 583 – Fall 2021 – Midterm Exam

Wednesday, November 3, 2021
Time constraint: 1hr 45min
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
6 questions, 30 pts total Score:_____

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

Total (100 possible): _____

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

1) What is the main difference between reaching definitions and available definitions? (5 pts)

2) Does the order you process instructions in a basic block to compute GEN/KILL matter for a given dataflow analysis? Yes/No and briefly explain. (5 pts)

3) Is it possible to unroll a loop with a statically (compile-time) unknown number of iterations? Yes/No and briefly explain. (5 pts)

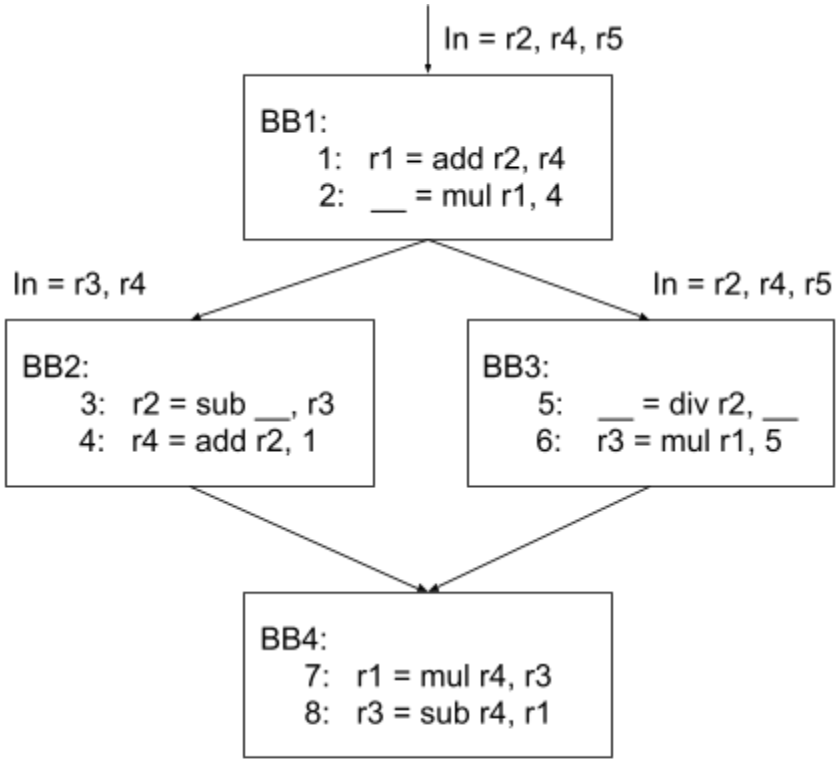
4) Can performing LICM on an instruction ever hurt performance? Yes/No and briefly explain. (5 pts)

5) The 90/10 rule says that 90% of an application's execution is spent in 10% of the code. Give an example of how a compiler can exploit this rule to improve performance. (5 pts)

6) Can the Estart for an instruction in a basic block ever be larger than its Lstart? Yes/No and briefly explain. (5 pts)

Part II. Medium Problems (Questions 7-11) (70 pts)

7) Given the following control flow graph and liveness information for BB1, BB2, and BB3, compute the Liveness IN/OUT sets for BB4, and fill in the missing operands to satisfy the Liveness analysis result. You should use each register r1, r2, r3, r4, or r5 **at most** once for specifying the missing source/destination operands. (15 pts)



BB4:

In = _____

Out = _____

8) Given the following if-converted code, draw the original CFG graph indicating the home location of all arithmetic/load/store instructions. Hint: the original CFG should have 8 BBs. (10 pts)

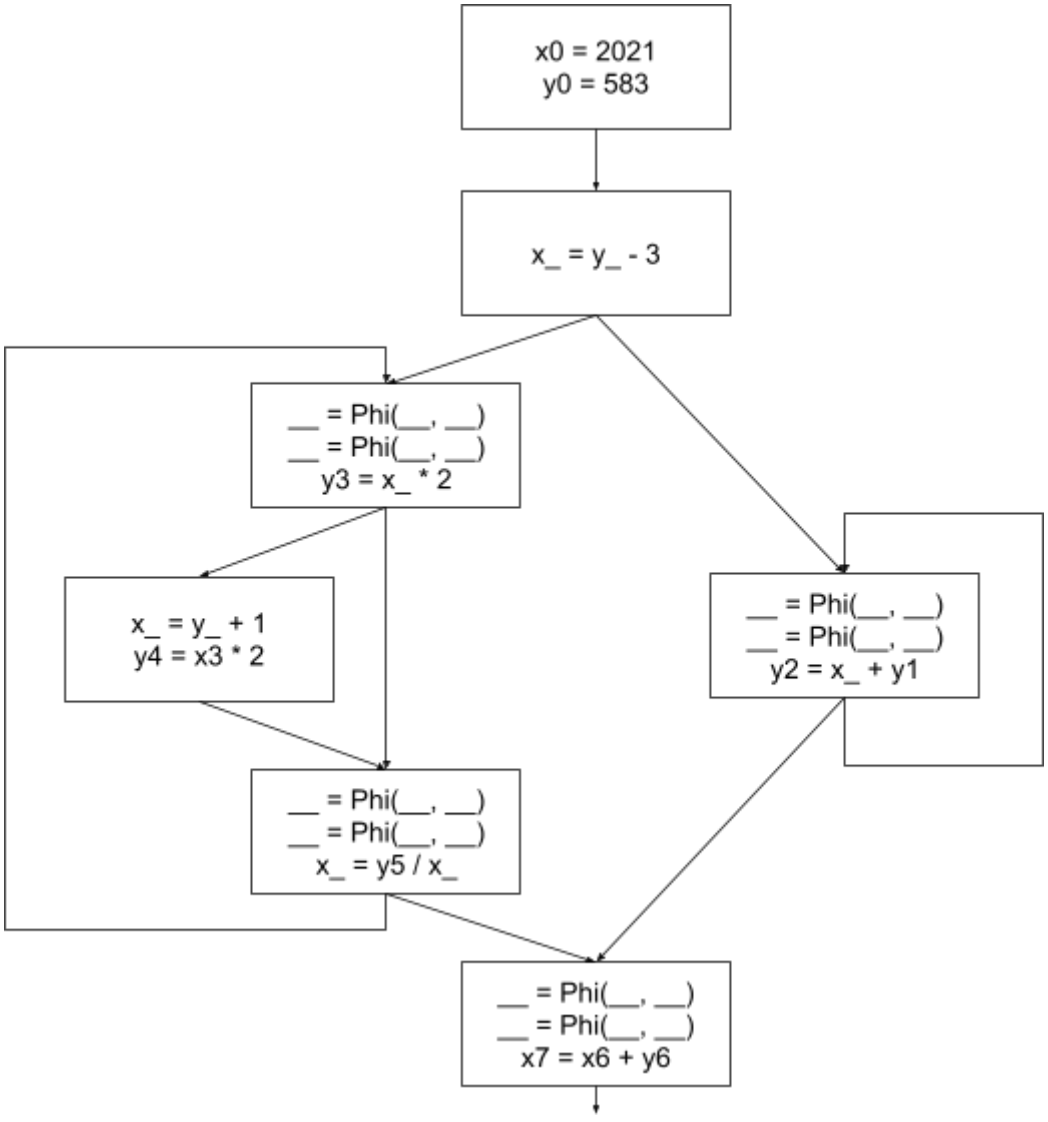
Recall that the format for cmpp instruction is as follows:

p1, p2 = CMPP.D1a.D2a(cond) if p3, where

p1 = first destination predicate
p2 = second destination predicate
D1a = action specifier for first destination
D2a = action specifier for second destination
cond = compare condition
p3 = guarding predicate

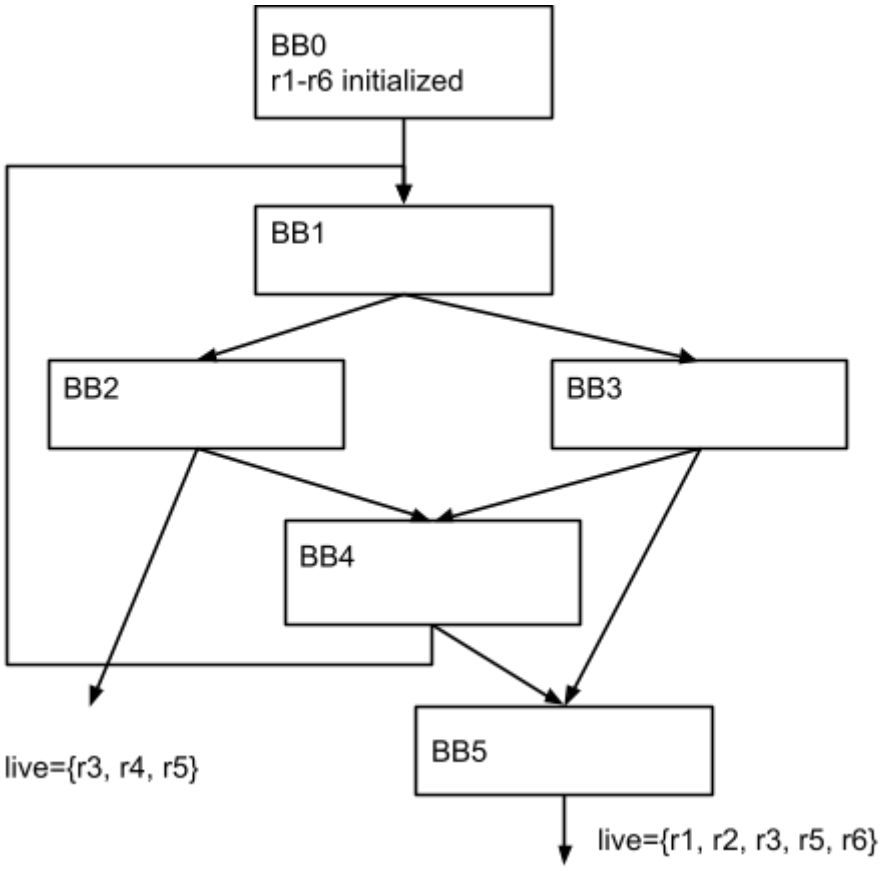
```
x = load(addr)
p1, p2 = cmpp.UN.UC(a<0) if T
p3 = cmpp.UN(b<0) if p1
x = x-a if p1
x = x+a if p2
b = b*2 if p3
c = c+b if p1
p4,p5 = cmpp.UN, UC(c<0) if p1
c = c*2 if p5
c = c+1 if p4
store(x, addr) if T
```

9) Satisfy static single assignment (SSA) form by filling in the blanks in the code segment below. Remember, the result and arguments of a Phi node must be different instances of the same variable (i.e., x1 = Phi(x2, x3)). Note that some Phi nodes may be unnecessary and should be left empty. For your answers, choose from x1 to x6 and y1 to y6. (15 pts)



10) You want to apply Loop Invariant Code Motion (LICM) to the CFG below. Insert the following instructions I1-I4 into BB1-BB4 with a maximum of 1 instruction added to each BB (i.e., one instruction in BB1, one in BB2, etc.) so that LICM can hoist as many instructions as possible. Just mark on the CFG below where the instructions should be placed and indicate whether they can be hoisted to the preheader. For those instructions that could not be hoisted, specify a reason. (15 pts)

I1: r6 = r6 + 1
I2: r5 = r3 * r1
I3: r4 = r2 + 2
I4: r3 = r1 + r2

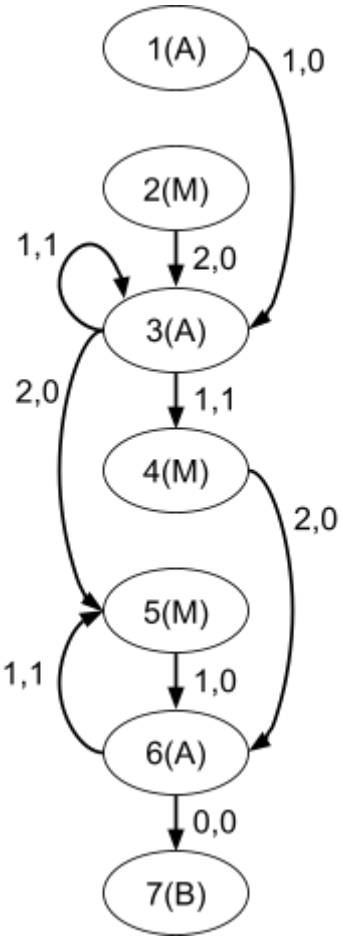


11) Given the dependence graph and the processor model below, answer the following questions related to modulo scheduling. (15 pts)

(a) Is the graph resource or recurrence constrained? Justify your answer. (5 pts)

(b) Generate both unrolled and rolled schedules for MII = 3. (10 pts)

For scheduling, you can assume instruction 1 is the highest priority, 2 is the second highest priority, etc. You do not need to assign staging predicates.



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

Instructions 2, 4, and 5 are memory
Instructions 1, 3, 6, and 7 use the ALU
Instruction 7 is a branch

Unrolled Schedule (may contain extra rows):

	ALU1	ALU2	MEM
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			

Rolled Schedule:

	ALU0	ALU1	MEM
0			
1			
2			