CS243 Homework 5 Solutions

Winter 2017

Problem 1. Instruction Scheduling (12 points)

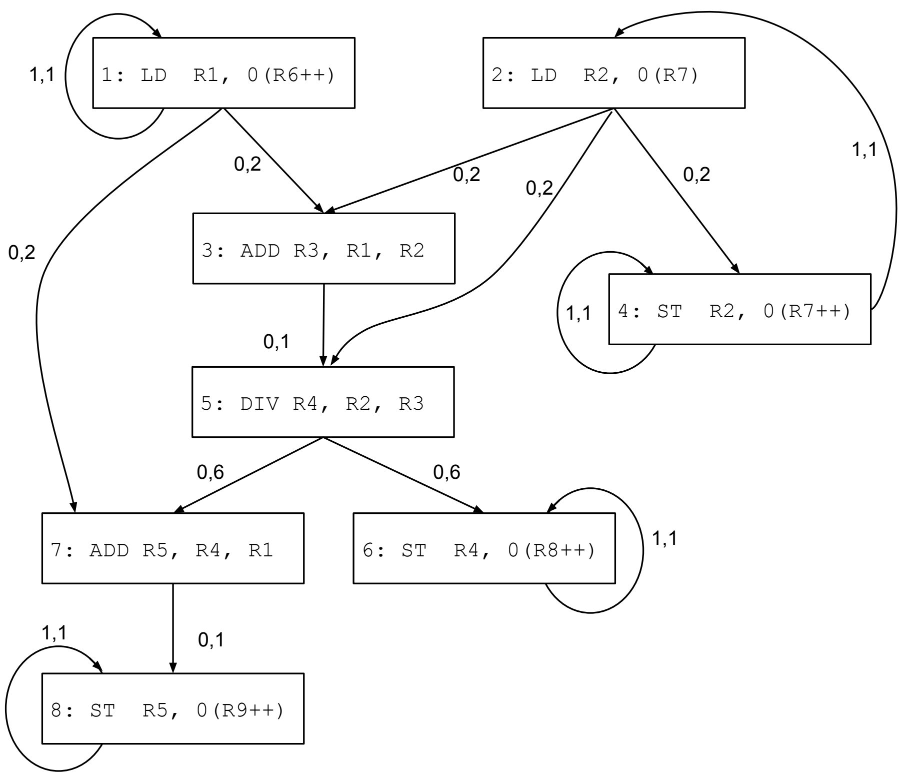
Example 1. This transformation is possible because it respects data dependencies, but requires speculation. It wouldn't improve the average case execution because it adds a cycle to the frequently executed path.

Example 2. The transformation is not allowed because it requires speculation of a instruction (`div`) which may trap. If it were allowed, then it would improve execution time.

Example 3. This transformation violates data dependencies via memory accesses in the "if" block. If it were applied, then the execution time would improve whenever the branch block is executed. (It is incorrect to say the execution time would remain the same.)

Problem 2. Software Pipelining (20 points)

1. (4 points) Dependence graph



2. (3 points) Schedule if executing the basic block just once (there are other possible answers too, but total number of cycles required should be 12):

Memory	ALU
1	
2	
4	3
	5
6	7
8	

3. (2 points) Unroll once. Need additional register for `R1`, and `R4`. Additional register for `R4` is not required if you mentioned that result of `DIV` becomes available at the end of 6 cycles, from when it starts, and so the live ranges for `R4` don't conflict.

4. (2 points) Lower bound from memory resource constraints is 5, from ALU constraints is 3, and from precedence constraints is 3. So, initiation interval ≥ 5.

5. (1 points) 1 iteration every 5 cycles

6. (4 points) Schedule is given below. Note that instructions in this table will be from different iterations, so that dependency constraints are satisfied.

Memory	ALU
2	7
1	
4	
6	3
8	5

Other schedules were accepted, as long as the initiation interval stayed at 5 cycles, the SCC (instructions 2 and 4) were scheduled tightly, and the number of times the loop needed to be unrolled stayed 1. Partial credit was given for partially correct answers.

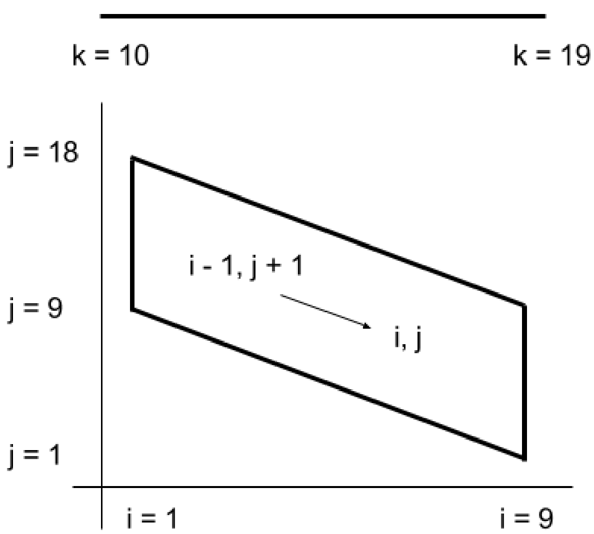
7. (4 points) Generated code:

```
PROLOGUE:     LD R2 <- 0(R7)
              LD R1 <- 0(R6++)
              ST 0(R7++) <- R2      ADD R3 <- R1, R2
                                    DIV R4 <- R2, R3
              LD R2 <- 0(R7)
              LD R11 <- 0(R6++)
              ST 0(R7++) <- R2
                                    ADD R3 <- R11, R2
                                    DIV R41 <- R2, R3
L:            LD R2 <- 0(R7)        ADD R5 <- R4, R1
              LD R1 <- 0(R6++)
              ST 0(R7++) <- R2
              ST 0(R8++) <- R4      ADD R3 <- R1, R2
              ST 0(R9++) <- R5      DIV R4 <- R2, R3
              LD R2 <- 0(R7)        ADD R5 <- R41, R11
              LD R11 <- 0(R6++)
              ST 0(R7++) <- R2
              ST 0(R8++) <- R41     ADD R3 <- R11, R2
              ST 0(R9++) <- R5      DIV R41 <- R2, R3  (BL L)
EPILOGUE:                           ADD R5 <- R4, R1
              NOP
              NOP
              ST 0(R8++) <- R4
              ST 0(R9++) <- R5
                                    ADD R5 <- R41, R11
              NOP
              NOP
              ST 0(R8++) <- R41
              ST 0(R9++) <- R5
```

R11 and R41 are additional registers for the odd iterations. If you got the schedule incorrect for 2.6, you probably got this part incorrect. Points were deducted if you did not clearly mark the prologue, epilogue or the steady state, or if you did not unroll the loop.

Problem 3. Dependence Analysis and Affine Transformations (20 points)

1. (2 points) The iteration spaces for these two loops is shown below:



2. (5 points) We write down the integer linear programs to be solved to find the data dependences in this program, for the specified two pairs.

   - For write access A[i, j] with itself, we have the following:

$$i = i'$$
$$j = j'$$
$$i \neq i' \vee j \neq j'$$
$$1 \leq i, i' \leq 9$$
$$10 - i \leq j < 20 - i$$
$$10 - i' \leq j' < 20 - i'$$

   - For write access A[i, j] with read access A[i - 1, j + 1], we have the following:

$$i = i' - 1$$
$$j = j' + 1$$
$$1 \leq i, i' \leq 9$$
$$10 - i \leq j < 20 - i$$
$$10 - i' \leq j' < 20 - i'$$

3. (2 points) We have the following pairs of operations with data dependences:

- Write access A[i, j] with read access A[i - 1, j + 1]
- Write access A[0, k] with read access A[i - 1, j + 1]

4. (1 points) There is one degree of communication-free parallelism.
5. (2 points) The affine partition mapping for statement s1 is $p = k$.
6. (4 points) The affine partition mapping for statement s2 is $p = i + j$.
7. (4 points) The SPMD code for this parallelized program is given below:

```
On each processor:

    A[0, p] = B[p]
    for (i = 1; i < 10; i++) {
        A[i, p - 1] = A[i - 1, p - i + 1]
    }
```