CS243 Homework 5

Winter 2017

Due: Feb 28, 2017

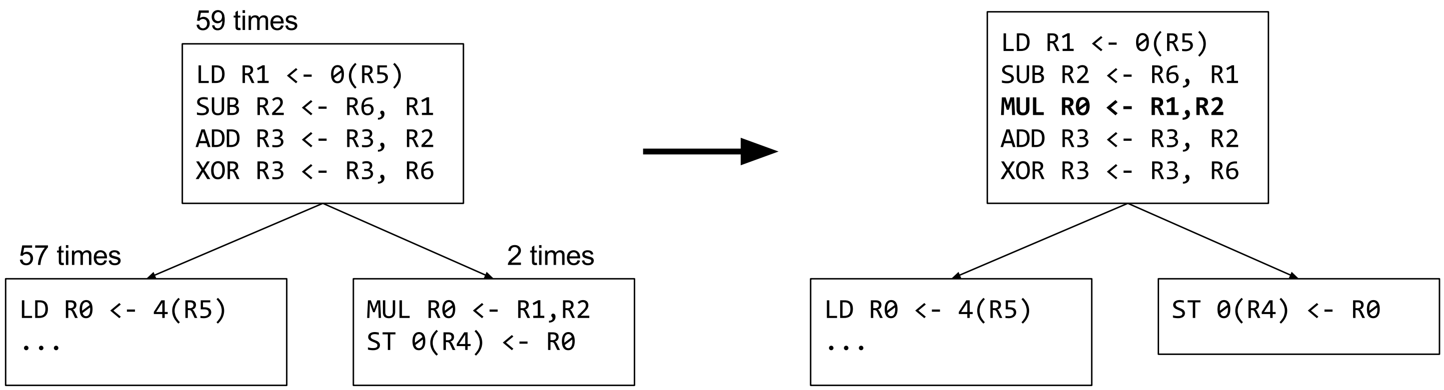
Directions:

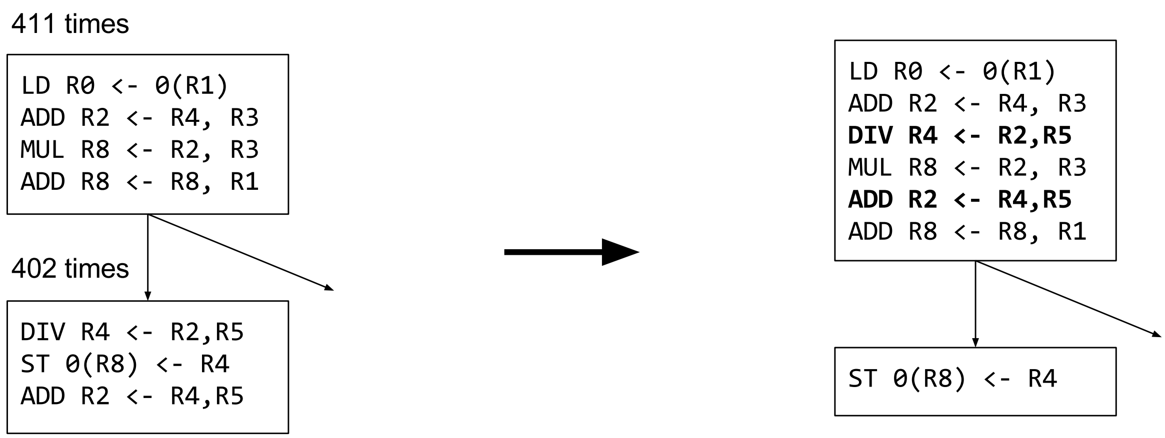
- Complete the following problems and hand them in at the beginning of class with your **name and SUID at the top**. Please do not forget your SUID.
- Remember to complete the corresponding Gradiance quizzes by the start of class on the due date. **There are no late days for Gradiance.**
- If you need to use one late day, hand in your assignment by 3pm Wednesday in the assignment drop-box (not the unlocked handout box) on the first floor of Gates. If you need two days, drop it in the drop-box by 2:50pm or hand it by 3pm at the beginning of class on Thursday.

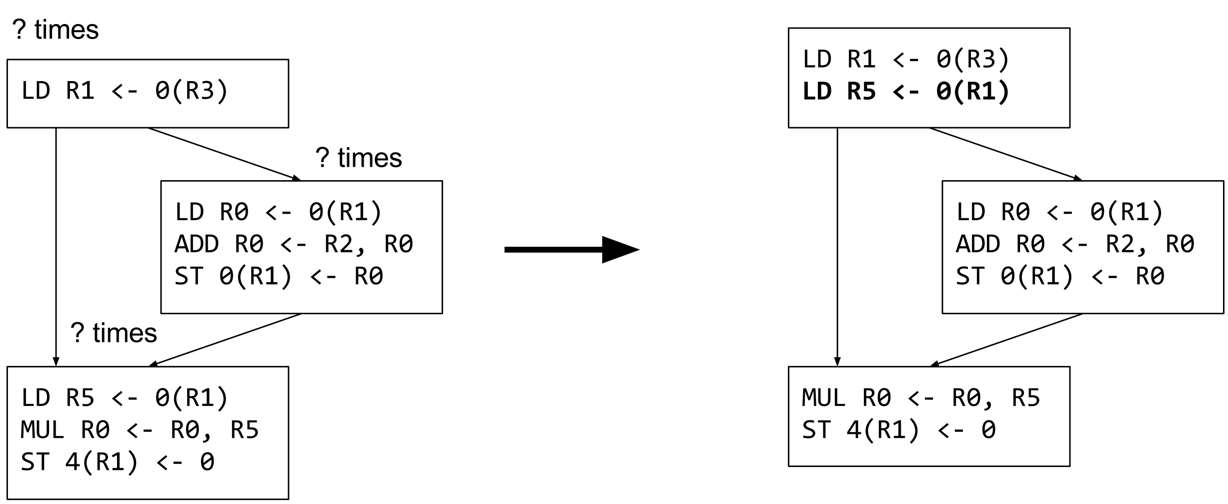
Problem 1. Instruction Scheduling

Consider the global instruction scheduling algorithm for regions presented in class. Assume LD, ST, and DIV may generate exceptions, and LD, MUL, and DIV have 10 cycle latencies, with all other operations. Also assume no registers are live upon exiting the regions. For each of the following pairs of regions, answer the following:

1. Would the global scheduling algorithm given in class perform this transformation for some choice of heuristic? If not, why not? If yes, is speculation required for this transformation?
2. Will this transformation improve execution time in the average case, assuming the basic block frequencies from profiling if indicated?


59 times
LD R1 <- 0(R5)
SUB R2 <- R6, R1
ADD R3 <- R3, R2
XOR R3 <- R3, R6
57 times
LD R0 <- 4(R5)
...
2 times
MUL R0 <- R1,R2
ST 0(R4) <- R0
LD R1 <- 0(R5)
SUB R2 <- R6, R1
MUL R0 <- R1,R2
ADD R3 <- R3, R2
XOR R3 <- R3, R6
LD R0 <- 4(R5)
...
ST 0(R4) <- R0


411 times
LD R0 <- 0(R1)
ADD R2 <- R4, R3
MUL R8 <- R2, R3
ADD R8 <- R8, R1
402 times
DIV R4 <- R2,R5
ST 0(R8) <- R4
ADD R2 <- R4,R5
LD R0 <- 0(R1)
ADD R2 <- R4, R3
DIV R4 <- R2,R5
MUL R8 <- R2, R3
ADD R2 <- R4,R5
ADD R8 <- R8, R1
ST 0(R8) <- R4


? times
LD R1 <- 0(R3)
? times
LD R0 <- 0(R1)
ADD R0 <- R2, R0
ST 0(R1) <- R0
? times
LD R5 <- 0(R1)
MUL R0 <- R0, R5
ST 4(R1) <- 0
LD R1 <- 0(R3)
LD R5 <- 0(R1)
LD R0 <- 0(R1)
ADD R0 <- R2, R0
ST 0(R1) <- R0
MUL R0 <- R0, R5
ST 4(R1) <- 0

Problem 2. Software Pipelining

Consider a machine that can initiate 1 arithmetic operation such as add, multiply, subtract or divide and 1 memory operation such as a load or a store operation every clock cycle. The result of `DIV` is available after 6 clocks, while that of `LD` after 2 clocks. Results for `ADD` and `ST` are available at the next clock. Consider the following code that is executed in one iteration of a loop, and answer the following questions.

```
LOOP: 1: LD  R1 <- 0(R6++)
      2: LD  R2 <- 0(R7)
      3: ADD R3 <- R1, R2
      4: ST  0(R7++) <- R2
      5: DIV R4 <- R2, R3
      6: ST   0(R8++) <- R4
      7: ADD R5 <- R4, R1
      8: ST  0(R9++) <- R5
```

1. Show the data dependence graph, with a node for each of the 8 instructions and edges between true dependences. Clearly label the edges with iteration difference and delay (δ, d).
2. What is the best schedule of the basic block, if we were executing the above code just once? Show the schedule as a resource reservation table.
3. How many times do we have to unroll the loop to remove unnecessary dependencies caused due to output dependences and anti dependences? How many additional registers do we need for unrolling?
4. What is the lower bound on initiation interval from precedence and resource constraints, if we unroll the loop?
5. What is the achievable throughput if we unroll the loop and use software pipelining?
6. Write down the schedule for software pipelined code as a modulo resource reservation table.
7. Show the generated code for the software pipelined, unrolled code.

Problem 3. Dependence Analysis and Affine Transformations

Consider the following program:

```
int A[100,100];
int B[100];

/* Initialization for A and B */

for (k = 10; k < 20; k++) {
    A[0,k] = B[k]; /* s1 */
}
for (i = 1; i < 10; i++) {
    for (j = 10 - i; j < 20 - i; j++){
        A[i,j] = A[i - 1, j + 1]; /* s2 */
    }
}
```

1. Draw the iteration spaces for these two loops, and show the dependences.
2. Write down the integer linear programs to be solved to find the data dependences in this program, for the following two pairs only.
    - Write access A[i, j] with itself.
    - Write access A[i, j] with read access A[i - 1, j + 1]
3. What are the pairs of operations that have data dependence?
4. How many degrees of communication-free parallelism are there?
5. What is the affine partition mapping for statement s1?
6. What is the affine partition mapping for statement s2?
7. Write the SPMD code for this parallelized program.