CSC 462/562 Computer Architecture
Homework #4: Due Wednesday, February 27

This assignment covers chapter 3.1-3.5. Answer any five of the six problems. Word process all answers.

1. Given the following RISC-V FP code and assuming the latencies from figure C.28 (page C-47) and assuming a 1 cycle branch penalty but assuming the fsd and the fadd.d can enter the MEM/WB stages at the same time
   a. describe the stalls that arise
   b. schedule the code to remove as many stalls as possible
   c. unroll the loop a sufficient number of times and schedule the code to remove all stalls
   d. compute the speedup of the unrolled and scheduled code over the original (compute the CPI of both versions, CPI is # of total clock cycles / # of instructions, or (# of instructions + stalls) / # of instructions)

```
Loop:   fld      f1, 0(x1)
        fmult.d  f3, f1, f2     // f2 and f3 are scalar values
        fadd.d   f5, f4, f3     // we are computing a[i]=a[i]*b+c
        fsd      f5, 0(x1)
        addiw    x1, x1, 8
        bne      x1, x2, Loop
```

2. Repeat #1 on the following code.

```
Loop:   fld      f1, 0(x1)
        fld      f2, 0(x2)
        fadd.d   f3, f1, f2
        fld      f4, 0(x3)
        fmult.d  f5, f4, f3
        fsd      f5, 0(x4)
        addiw    x1, x1, 8
        addiw    x2, x2, 8
        addiw    x3, x3, 8
        addiw    x4, x4, 8
        bne      x4, x5, Loop
```

3. Assume a Tomasulo-style architecture has a 2-cycle fetch stage preceding the issue stage. During these 2 cycles, branch prediction is handled as follows:
   a branch target buffer is accessed to fetch a target location and branch prediction
   if a buffer hit and predict taken, the next instruction fetched is at the target location
   if predict not taken, the next instruction fetched is the next sequential instruction
   if a buffer miss, nothing immediately happens
   Branches are computed in an integer unit during the execute stage
   If a miss-prediction is determined, all instructions fetched after the branch are flushed and the buffer is updated – assume this is a 2-cycle penalty

The actual impact on the performance of this style architecture is harder to compute than on a pipeline because the pipeline simply flushes all earlier instructions, but here we are unsure of the time it took between issuing the branch instruction and its execution completing (because the integer unit may not execute immediately if it is waiting on a result being forwarding over the CDB)

Let's assume for 40% of all branches that the value is already available, for 30% the value becomes available 1 cycle later, for 20% the value becomes available 2 cycles later, and for 10% the value becomes available 3 cycles later. We make a further assumption that an integer functional unit will always be available.

Assuming access to the buffer is a hit 90% of the time and predictions are 80% accurate for conditional branches and 100% accurate for unconditional branches, what is the CPI for the Tomasulo-style architecture for a benchmark of 19% conditional branches and 4% unconditional branches and assuming no other sources of stalls?

4. The Scoreboard approach dates back to the late 1960s and so it did not benefit from more modern concepts that were implemented with the Tomasulo-style arcthiecture. One restriction of the Scoreboard is the need to wait until both operands become available before an instruction, waiting at a functional unit, can read the operands and begin executing. Another restriction is that instructions cannot be issued if a functional unit is busy even if it is not currently computing (because it is waiting to read registers).
    a. There are at least two drawbacks to waiting to read both operands at the same time rather than obtaining operands immediately upon their availability. What problems do you see? Describe at least two.
    b. Why does the Scoreboard need to make a restriction whereby a functional unit reads both operands in the same cycle and no other functional unit is able to read operands in that same cycle? That is, why not have the functional unit read a register as soon as its value is ready?
    c. How did the Tomasulo approach get around this restriction?

5. Given the code from problem #1, show how it will execute using a Scoreboard-style architecture and a Tomasulo-style architecture (see sample problems 8 and 9 from the chapter 3 sample problems part 1). In both cases, there are 2 load/store units, 1 FP multiplier (not pipelined), 1 FP adder (not pipelined) and 2 integer units. Include the fetch stage in your table, similar to those of the sample problems, and assume one instruction is fetched every cycle. Show two complete iterations of the loop. NOTE: assume the instruction fetch stage continues even if the issue stage stalls.

6. Repeat #5 on the following code. Note that unlike #5, there is no store here. Instead, this code is computing a summation, like sum = sum + a[i]*b[i];

```
Loop:  fld      f1, 0(x1)
       fld      f2, 0(x2)
       fmult.d  f3, f1, f2
       fadd.d   f4, f4, f3
       addiw    x1, x1, 8
       addiw    x2, x2, 8
       bne      x1, x3, Loop
```