CSC 462/562 Computer Architecture
Homework #3: Due Monday, February 18

This assignment covers appendix C.4-C.6. Undergraduate students answer five of the six problems. Graduate students answer all six. All answers are to be word processed. Submit by hardcopy if possible.

1. The following RISC-V loop includes FP operations (use the latencies shown in figure C.29). Assume forwarding is available and branches are handled in the ID stage. NOTE: two instructions are not allowed to enter the MEM or WB stage simultaneously.

```
loop:  fld     f1, 0(x2)
       fld     f2, 8(x2)
       fmul.d  f3, f1, f2
       fld     f4, 0(x3)
       fadd.d  f5, f3, f4
       fsd     f5, 0(x3)
       addiw   x2, x2, 16
       addiw   x3, x3, 8
       subiw   x4, x4, 1
       bne     x4, x0, loop
```

   a. Provide the timing diagram for this loop plus the first instruction of the second iteration, and any extra instructions fetched after the branch instruction.
   b. Schedule the code to remove as many of the stalls/branch delay as possible. How much improvement did scheduling give you?

2. Repeat #1 with the following code.

```
loop:  fld     f0, 0(x1)
       fadd.d  f1, f0, f2
       fld     f3, 0(x2)
       fdiv.d  f4, f1, f3
       fsd     f4, 0(x1)
       addiw   x1, x1, 8
       addiw   x2, x2, 8
       bne     x1, x3, loop
```

3. Provide the pipeline timing diagram for the following RISC-V code run on the MIPS R4000 pipeline assuming forwarding is available and taken branches accrue a 3-cycle branch penalty (not taken branches accrue no penalty). Show two versions, one where the beq is taken and one where it is not. Assume in both cases the bne is taken. Show one full iteration and the first instruction of the next iteration.

```
loop:  lw      x1, 0(x2)
       lw      x3, 0(x4)
       slt     x5, x1, x3
       beq     x5, x0, else
       addiw   x6, x6, 1
       j       next
```

```
else:   subiw  x6, x6, 1
next:   addiw  x2, x2, 4
        addiw  x4, x4, 4
        bne    x2, x7, loop
```

4. Assume we have a pipeline where branch conditions are determined in stage 5 and branch target locations are determined in stage 3. If our average benchmark has 24% conditional branches, 5% unconditional branches and 70% of all conditional branches are taken, should we implement assume taken or assume not taken?

5. We improved the 5-stage pipeline by moving branch decisions from EX (with the MUX changing the PC in MEM) to ID. This allowed us to reduce the branch penalty from 3 cycles to 1 and then we try to let the compiler schedule the branch delay slot to remove all penalties. Why not do something similar for the MIPS R4000?
    a. Currently, branches are determined in the EX stage (4^{th} stage) leading to a 3 cycle penalty. We can move the branch mechanisms to the 3^{rd} stage (RF) to reduce the penalty to 2 cycles, but not earlier in the pipeline. Why not?
    b. If we move the branch decision to the RF stage, it reduces branch penalties but yields other complications. What drawbacks can you see to making this change?
    c. Let's assume a compiler can fill one branch delay slot 70% of the time but does not attempt to fill a second branch delay. Further, the MIPS R4000 assumes all branches are not taken so that if the branch is not taken, no instructions need to be flushed. Given a benchmark of 20% conditional branches and 4% unconditional branches where 75% of all conditional branches are taken, and moving branch decisions to the RF stage, what does the ideal CPI become assuming branches are the only source of penalty?

6. Consider figures C.26 and C.27 to answer this question.
    a. In which stage would a breakpoint cause an exception to be inserted into the pipeline?
    b. In which stage would an I/O device request cause an exception to be inserted into the pipeline?
    c. Aside from any answers you provided in parts a and b, for each of the stages of the pipeline that can cause an exception, explain why an exception might arise in that stage. Offer an example RISC-V instruction if appropriate.