CSC 462/562 Computer Architecture
Homework #5: Due Monday, March 18

This assignment covers chapter 3.6-3.12. Answer any 5 of the 6 problems. Word process all answers.

1. Use the following RISC-V code where loads/stores and int operations take 1 cycle to execute, and FP add/subtract takes 2 cycles. Unroll and schedule the code on a 2-issue superscalar as described below so that every cycle at least one operation is issued (no empty cycles). Fill the branch delay slot. Compute the CPI for each.
   a. One pipeline handles all non-FP operations, one handles all FP operations.
   b. One pipeline handles loads/stores, the other handles all FP/int/branch operations.
   c. No restrictions.

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

2. Do we get any benefit in using Tomasulo's architecture when running integer programs? We have a 2-issue superscalar implemented using two parallel 5-stage pipelines in one processor and a dual-issue Tomasulo-style architecture in another processor. For both processors, assume 2 load/store units, 2 int units and 2 register file ports so that any pair of instructions can be issued at a time, any pair of instructions can execute at a time and in the pipeline, any pair of instructions can write to registers at the same time (as long as they are writing to different registers). In Tomasulo, there is only one CDB and only one instruction commits at a time. Assume forwarding is available in the 5-stage pipeline and an unlimited ROB size for Tomasulo. Show how the following RISC-V code executes on both and compare the results (which finished in fewer cycles)?

```
Loop:   lw      x1, 0(x2)
        lw      x3, 0(x4)
        addw    x5, x1, x3
        subiw   x5, x5, 1
        sw      x5, 0(x6)
        addiw   x2, x2, 4
        addiw   x4, x4, 4
        addiw   x6, x6, 4
        subiw   x7, x7, 1
        bne     x7, x0, Loop
```

3. Use the following RISC-V code where loads/stores and int operations take 1 cycle to execute, FP add/subtract takes 2 cycles and FP multiply takes 3 cycles. Both architectures are dual issue superscalars where any pair of instructions can be issued in the same cycle as long as there is an available functional unit.
    a. Show how the code executes on a Scoreboard-style architecture with 1 integer unit, 2 load/store units, 2 FP adders and 1 FP multiplier where these functional units are not pipelined. Assume two instructions can read their operands in the same cycle if they are reading different register values.
    b. Repeat part a using a Tomasulo-style architecture with the same number of functional units. Include the fetch and issue stages in your table but not commit (there are no branches here so we can largely forget about the commit stage).

```
fld     f1, 0(x1)
fld     f2, 8(x1)
fmul.d  f3, f1, f2
fld     f4, 0(x2)
fld     f5, 8(x2)
fadd.d  f6, f4, f5
fsub.d  f7, f3, f6
fsd     f7, 0(x1)
addiw   x1, x1, 16
addiw   x2, x2, 16
```

4. Using the VLIW architecture as described in section 3.7, and assuming the fmult.d takes 3 cycles to compute, unroll and schedule the following code on the VLIW to have as few empty slots as possible.

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

   Compute the code's CPI.

5. It is critical to use branch speculation to support a multi-issue superscalar. Why? Briefly explain these types of branch speculation mechanisms/methods and then select the type you would use to support a 2-issue, dynamically scheduled superscalar: assume taken, assume not taken, branch prediction buffer, branch target buffer, branch unfolding. Would you select a different approach for VLIW?

6. In examining the ARM Cortex pipeline, answer the following questions.
    a. what stage(s) are branch predictions made? what stage(s) are branch results determined?
    b. what stage(s) will data hazards be determined and how many stalls would a data hazard incur?

c. how many instructions can be issued to functional units at a time? what happens if consecutive instructions in the instruction stream share the same functional unit?
d. what are the sources of stalls for the ARM Cortex? rank them from most severe to least