# CMPE 200 HW#4

Due: Monday, Nov. 4, 11:59pm
Total Score: /100

Instructor: Hyeran Jeon

Computer Engineering Department, San Jose State University

---

**Static Pipelining (10pts + 20pts)**

1. In the static pipelining with superscalar architecture, explain how loop unrolling can help improving the performance.

2. Consider the following code.

```
                lw   $t0, 0($t1)          // I1
LOOP:           lw   $t2, -8($t1)         // I2
                add  $t0, $t2, $t0        // I3
                sw   $t0, -8($t1)         // I4
                subi $t1, $t1, #8         // I5
                bne  $t1, $t2, LOOP       // I6
```

Assume that we know that bne (I6) is "taken" in the first iteration. To reduce the control hazard overhead, a programmer decided to unroll the loop code. Show a code that unrolled the above loop code for two iterations. If register renaming is needed, use any temporal register name from $t3 (i.e. $t3, $t4, $t5…).

**Tomasulo Algorithm (20 + 20 + 30pts)**

For the problem 3 to 4, use the following assumption:

- One instruction can issue per cycle
- Latencies (cycles):
  - add : 1
  - ld : 4
  - mul : 8
  - div : 20
- There is one adder, one multiply/divide unit, and one load port
- Functional units are not pipelined
- WB and EXE occur in the same cycle (i.e. an instruction writes to the CDB in cycle n, and its dependent instruction can start execution in cycle n as well)
- There is a single CDB
- # entries in reservation stations:
  - add : 3 (call these add1, add2, add3)
  - ld : 2 (call these ld1, ld2)
  - mul/div: 2 (call these md1, md2)
- An instruction stalls at issue if an appropriate reservation station (RS) is not available. It may issue in the cycle after the previous instruction writes back (i.e., if add x is stalled because all add RSs are occupied, and then one of the add's writes back on cycle n, then the add x can issue on cycle n+1).
- If more than one reservation station is available (e.g., add1 and add3), issue the instruction to the RS with the lower number (e.g., add1).

3. For each of the following instructions, write the RS identifier that the instruction issues to, write the sources (e.g., $t2 if read from the register file, md1 if waiting on md1's result), and the cycle in which the instruction issues, executes, and writes-back its result.

| Instruction | RS | Source(s) | ISSUE | EXE | WB |
|---|---|---|---|---|---|
| div $t1, $t2, $t3 | md1 | $t2, $t3 | 1 | | |
| ld $t4, 0($t5) | | | | | |
| mul $t2, $t1, $t6 | | | | | |
| add $t3, $t5, $t6 | | | | | |
| div $t6, $t7, $t3 | | | | | |
| add $t1, $t4, $t3 | | | | | |
| ld $t4, 4($t5) | | | | | |
| add $t1, $t1, $t2 | | | | | |
| add $t1, $t1, $t5 | | | | | |
| ld $t3, 0($t5) | | | | | |

4. Repeat the previous exercise using the same assumptions except for the following differences:
   - There is a re-order buffer with six entries (call these rob1 – rob6).
   - Rename instructions based on the ROB identifier.
   - Instructions commit in program order, at a rate of at most one instruction per cycle.
   - Similar to the deallocation of RS entries, if an instruction in rob*i* commits in cycle n, a new instruction cannot issue into rob*i* until cycle n+1.

| Instruction | RS | ROB | Source(s) | ISSUE | EXE | WB | COMMIT |
|---|---|---|---|---|---|---|---|
| div $t1, $t2, $t3 | md1 | rob1 | $t2, $t3 | 1 | | | |
| ld $t4, 0($t5) | | | | | | | |
| mul $t2, $t1, $t6 | | | | | | | |
| add $t3, $t5, $t6 | | | | | | | |
| div $t6, $t7, $t3 | | | | | | | |
| add $t1, $t4, $t3 | | | | | | | |
| ld $t4, 4($t5) | | | | | | | |
| add $t1, $t1, $t2 | | | | | | | |
| add $t1, $t1, $t5 | | | | | | | |
| ld $t3, 0($t5) | | | | | | | |

5. Repeat the previous exercise using the same assumptions except for the following differences:
   - There is an unified set of six reservation stations (call these rs1 – rs6).
   - You don't need to show the renaming for this problem.
   - Two different type instructions can issue per cycle.
   - Two instructions can commit per cycle.
   - Still only one CDB.
   - Separate mul and div units. (One of each can execute in parallel.)

| Instruction | ISSUE | EXE | WB | COMMIT |
|---|---|---|---|---|
| div $t1, $t2, $t3 | 1 | | | |
| ld $t4, 0($t5) | | | | |
| mul $t2, $t1, $t6 | | | | |
| add $t3, $t5, $t6 | | | | |
| div $t6, $t7, $t3 | | | | |
| add $t1, $t4, $t3 | | | | |
| ld $t4, 4($t5) | | | | |
| add $t1, $t1, $t2 | | | | |
| add $t1, $t1, $t5 | | | | |
| ld $t3, 0($t5) | | | | |