CMSC411 Fall 2009 Midterm 1 Solutions

You have 70 minutes to complete this midterm. There are 100 points. For “essay” questions 1-2 sentence answers are preferred. You do not need to provide a number if you can show the appropriate fraction. E.g., 1/13 is acceptable in place of .0769. **You must show your work!**

1. (12 pts) Architectures & Measurement
   a. Historically, what has been the relationship between latency and bandwidth? Explain.
      Bandwidth has improved faster than latency, due to both architectural and commercial reasons.
   b. Give one reason CPU implementation is becoming as important (if not more) as instruction set design in influencing CPU performance.
      Instruction set architectures (ISA) have converged, while implementation options (e.g., pipelining, superscalar, multicore) for the same ISA have a major impact on CPU performance.
   c. Give one reason computer architects use benchmarks instead of real programs to measure computer performance.
      Benchmarks provide reproducible results that are easy to compare across computing platforms, and are easier to use than real programs.
   d. Why do computer architects prefer using geometric mean instead of arithmetic or harmonic mean when presenting benchmark results?
      Results are consistent regardless of choice of reference platform.

2. (18 pts) Pipelining
   a. What is a structural hazard?
      Hardware limitation preventing pipeline instruction from executing (e.g., single-ported memory).
   b. Why are RAW hazards fundamentally different than WAR and WAW hazards?
      RAW hazards represent actual flow of values, whereas WAR and WAW hazards are name dependences caused by reuse of storage (and can be eliminated using additional storage).
   c. Explain why exceptions are problematic for pipelined processors.
      Because when an exception occurs, both previous & succeeding instructions may be still executing in the pipeline, and need to be handled before the exception can be processed.
   d. Describe the difference between synchronous and asynchronous exceptions.
      Synchronous exceptions occur when a particular instruction is executed. Asynchronous exceptions may occur any time.
e. Explain why pipelined processors face additional problems when certain instructions (e.g., mult) require more time (i.e., clock cycles) to complete.

Instructions may complete out of order, requiring additional checks to avoid data hazards.

f. Explain why filling the branch delay slot with an instruction from before the branch is preferred to the alternatives.

If permitted by language semantics, putting an instruction from before the branch will not introduce extra execution costs regardless of whether branch is actually taken.

3. (20 pts) Reliability, Performance, and Amdahl’s Law

a. (6 pts) The 2009-10 NBA season begins next week. You are suddenly hired as the general manager of the DC basketball team, the Washington Wizards. Your starting players consist of 2 guards, 2 forwards, and 1 center. Suppose your guards (e.g., Gilbert Arenas) are usually injured once every 3 months (MTTF=3), forwards (e.g., Antawn Jaimison) are injured every 2 months (MTTF=2), and centers are injured every 3 months (MTTF=3). When do you expect your first starting player to be injured?

Failure rate = (2*1/3)+(2*1/2)+(1*1/3) = 2/3+1+1/3 = 2
MTTF = 1/Failure rate = ½ months = 15 days

b. (6 pts) Your guards can score at a rate of (on a 48 minute basis) 20 points/game at home and 15 points/game on the road, while your forwards can score 25 points/game at home and 12 points/game on the road (your center is so bad he does not score at all). Using geometric mean, what is the speedup for points/game scored when comparing guards to forwards (assuming equal number of home and road games)?

Geometric mean = sqrt( 20/25 * 15/12) = sqrt(4/5 * 5/4) = sqrt (1) = 1

c. (8 pts) Another NBA team, the Houston Rockets, have noticed your team needs a better center and offers to trade you an All-Star center (Yao Ming) for your draft picks for the next 10 years. Having Yao Ming on your team would improve your defense. Currently opposing teams score against your team as follows: 60% from mid-range jump shots, 25% from 3-point shots, and 15% from interior shots. If Yao Min can reduce interior shots by a factor of 3, how will your opponents scoring be affected? If Yao can eliminate interior shots completely, how will your opponents score be affected?

Using Amdahl’s Law we find:

for 3x reduction = 1/(.85+.15/3) = 1/(.85+.05) = 1/.9 = 1.11
for infinite reduction = \frac{1}{(.85+.15/\text{inf})} = \frac{1}{.85} = 1.176
4. (24 pts) Basic pipelining. Use the following code fragment:

```
Loop:    LD    R2,4(R1) ; load R2 from address 4+R1
       LD    R3,0(R2) ; load R3 from address 0+R2
       DADD  R3,R1,R2 ; R3=R1+R2
       BNEZ  R3,Loop ; branch to loop if R3 != 0
```

Use the classic MIPS five-stage integer pipeline, show the timing of this instruction sequence for two loop iterations. Assume all memory accesses take 1 clock cycle, conditional branch is handled by predicted to be taken, and a register may be read and written in the same clock cycle.

a. (6 pts) Assume there is no forwarding or bypassing hardware.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>IF</td>
<td>ST</td>
<td>ST</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DADD</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNEZ</td>
<td>IF</td>
<td>ST</td>
<td>ST</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. (6 pts) Assume normal forwarding and bypassing hardware.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>LD</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD</td>
<td>IF</td>
<td>ID</td>
<td>ST</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DADD</td>
<td>IF</td>
<td>ST</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BNEZ</td>
<td>IF</td>
<td>ST</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c. (6 pts) List and classify all the potential data hazards in the code fragment.

i. RAW: LD R2, 4(R1) to LD R3, 0(R2)
ii. RAW: LD R2, 4(R1) to DADD R3, R1, R2
iii. WAW: LD R3, 0(R2) to DADD R3, R1, R2
iv. RAW: DADD R3, R1, R2 to BNEZ R3, Loop

d. (3 pts) Which of the potential RAW data hazards are eliminated because of forwarding?

No RAW hazards are eliminated completely, but forwarding reduces the stalls for the RAW hazards (i) and (iv). The hazard (ii) does not cause stalls even without forwarding, since potential stalls are hidden by the stalls caused by hazard (i).

e. (3 pts) Are there any control hazards in the code fragment? Explain.

There is a control hazard from BNEZ to the other instructions in the loop, except on the first iteration of the loop, since whether the other instructions are executed additional times is dependent on the result of the branch instruction BNEZ.
5. (10 pts) **Pipeline hazards.** Consider the following MIPS floating point pipeline

![Pipeline Diagram]

a. (6 pts) The MIPS processor will need some checks for **WAW** hazards to insert stalls. For each of the following checks, explain why it is or is not needed:

<table>
<thead>
<tr>
<th>Opcode field</th>
<th>Opcode field of IF/ID(IF/ID.IR₀₋₅)</th>
<th>Check needed? Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD.D (A1/A2.IR)</strong></td>
<td><strong>LD.D</strong></td>
<td>Needed, since the LD.D instruction issued will complete before an ADD.D instruction still in the A1/A2 stage of the adder pipeline (since LD takes fewer cycles than ADD).</td>
</tr>
<tr>
<td><strong>MUL.D (M1/M2.IR)</strong></td>
<td><strong>MUL.D</strong></td>
<td>Not needed, since any new MUL.D instruction issued will write its result after all MUL.D instructions currently in the multiply pipeline (since all MUL take the same number of cycles).</td>
</tr>
</tbody>
</table>

b. (4 pts) The MIPS processor will also need some checks for **RAW** hazards to insert stalls. For each of the following checks, explain why it is or is not needed:

<table>
<thead>
<tr>
<th>Opcode field</th>
<th>Opcode field of IF/ID(IF/ID.IR₀₋₅)</th>
<th>Check needed? Explain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MUL.D (M1/M2.IR)</strong></td>
<td><strong>MUL.D</strong></td>
<td>Needed, since a new MUL.D instruction issued may use a value being computed by the MUL.D instruction currently in the multiply pipeline (and MUL instructions take the same number of cycles).</td>
</tr>
</tbody>
</table>
6. (16 pts) *Pipeline performance.* Suppose processor X executes instructions in the following 4 stages (no pipeline), where each stage could run this fast:

<table>
<thead>
<tr>
<th>Stage</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF&amp;ID</td>
<td>10ns</td>
</tr>
<tr>
<td>EX</td>
<td>5ns</td>
</tr>
<tr>
<td>MEM</td>
<td>20ns</td>
</tr>
<tr>
<td>WB</td>
<td>5ns</td>
</tr>
</tbody>
</table>

a. (6 pts) Compare the performance (average time/instruction) of a pipelined vs. unpipelined implementation of processor X.
   - Unpipelined = 10+5+20+5 = 40ns/instruction
   - Pipelined = 20ns/instruction (assuming no stalls), stages take 20ns each

b. (4 pts) Now assume that 20% of the instructions are load/save instructions. The remaining instructions do not need to execute the MEM stage. Compare the performance of a pipelined vs. unpipelined implementation of processor X.
   - Unpipelined = .80 * 20ns + .20 * 40ns = 16+8 = 24ns/instruction (avg)
   - Pipelined = 20ns/instruction, same as before

c. (6 pts) Now assume that 20% of the instructions are branch instructions. 50% of branches are mispredicted, with a 2 cycle penalty. What is the cycles per instruction (CPI) of the pipelined implementation of processor X? What is the performance (avg time / instr.) of the pipeline implementation of processor X?
   - CPI = .2 (.5 * 1 + .5 * 2) + .8 (1) = .2 (1.5) + .8 = .3 + .8 = 1.1
   - Pipelined performance = 20ns * 1.1 = 22ns/instruction