CMSC411 Fall 2009 Quiz 3 Solution

Name: ________________________________

You have 20 minutes to complete this quiz. **You must show your work!**

You are evaluating a new processor microprocessor, with instructions that have the following latencies. Consider the instruction schedule for the following code fragment. Ignore cycle times for front-end fetch and decode.

I1: LD F1, 0(Rx)
I2: LD F2, 4(Rx)
I3: ADDD F3, F1, F2
I4: LD F1, 0(Ry)
I5: MULTD F4, F2, F1
I6: LD F2, 8(Rx)
I7: ADDD F1, F3, F2

1. (10 pts) List all hazards (RAW, WAR, WAW) found in the code. You may ignore transitive hazards (i.e., if I1 -> I2 and I2 -> I3, ignore I1 -> I3).

   I1 -> I3 RAW
   I2 -> I3 RAW
   I3 -> I4 WAR
   I4 -> I5 RAW
   I5 -> I6 WAR
   I6 -> I7 RAW

2. (4 pts) What would be the baseline performance of the code sequence if no new instruction execution could be initiated until the previous instruction execution had completed? Assume that execution does not stall for lack of the next instruction, but only one instruction/cycle may be issued. Show when each instruction is executed (i.e., instruction schedule).

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory LD</td>
<td>+3</td>
</tr>
<tr>
<td>Memory SD</td>
<td>+1</td>
</tr>
<tr>
<td>Integer ADD, SUB</td>
<td>+0</td>
</tr>
<tr>
<td>Branches</td>
<td>+1</td>
</tr>
<tr>
<td>ADDD</td>
<td>+2</td>
</tr>
<tr>
<td>MULTD</td>
<td>+4</td>
</tr>
</tbody>
</table>

   I1: LD F1, 0(Rx)   // 1 + 3 stall
   I2: LD F2, 4(Rx)    // 1 + 3 stall
   I3: ADDD F3, F1, F2   // 1 + 2 stall
   I4: LD F1, 0(Ry)    // 1 + 2 stall
   I5: MULTD F4, F2, F1   // 1 + 4 stall
   I6: LD F2, 8(Rx)    // 1 + 3 stall
   I7: ADDD F1, F3, F2   // 1 + 2 stall
   // Total = 27 cycles

3. (4 pts) How many cycles would be needed to execute the example code if the pipeline stalled only on true dependences (RAW hazards) found in the code? Show the instruction schedule.

   I1: LD F1, 0(Rx)   // 1 + 3 stall
   I2: LD F2, 4(Rx)    // 1 + 3 stall
   I3: ADDD F3, F1, F2   // 1 + 2 stall
   I4: LD F1, 0(Ry)    // 1 + 2 stall
   I5: MULTD F4, F2, F1   // 1 + 4 stall
   I6: LD F2, 8(Rx)    // 1 + 3 stall
   I7: ADDD F1, F3, F2   // 1 + 2 stall
   // Total = 27 cycles
I1: LD F1, 0(Rx)  // 1
I2: LD F2, 4(Rx)  // 1 + 3 stall
I3: ADDD F3, F1, F2  // 1
I4: LD F1, 0(Ry)  // 1 + 3 stall
I5: MULTD F4, F2, F1  // 1
I6: LD F2, 8(Rx)  // 1 + 3 stall
I7: ADDD F1, F3, F2  // 1 + 2 stall  
// Total = 18 cycles

4. (4 pts) Now consider a 2-way multiple-issue processor (with infinite IF/ID bandwidth and forwarding). How many cycles would be needed to execute the example code if the pipeline stalled only on true dependences (RAW hazards) found in the code? Show the instruction schedule.

I1: LD F1, 0(Rx)  I2: LD F2, 4(Rx)  // 1 + 3 stall
I3: ADDD F3, F1, F2  I4: LD F1, 0(Ry)  // 1 + 3 stall
I5: MULTD F4, F2, F1  I6: LD F2, 8(Rx)  // 1 + 3 stall
I7: ADDD F1, F3, F2  // 1 + 2 stall  
// Total = 15 cycles

5. (2 pts) What instructions in the code sequence could be reordered to improve performance? You do not need to show the new instruction schedule.

Dependences for I4 and I6 are name dependences and may be eliminated with register renaming, after which I4 and I6 may be moved earlier to reduce stalls.

6. (4 pts) Assume your hardware has a collection of temporary registers (T0 to T63). Rename the registers in the code above to eliminate as many dependences (hazards) as possible, using the algorithm from homework 3 (i.e., substitute destination registers with temporary registers starting with T9, then modify code to preserve data dependences).

I1: LD T9, 0(Rx)
I2: LD T10, 4(Rx)
I3: ADDD T11, T9, T10
I4: LD T12, 0(Ry)
I5: MULTD T13, T10, T12
I6: LD T14, 8(Rx)
I7: ADDD T15, T11, T14

7. (2 pts) What is the relationship between renaming registers and the previous assumption that that processor pipeline stalled only on true dependences (RAW hazards)?

Renaming registers eliminates name dependences, leaving only RAW hazards.  
So register renaming makes the previous assumption true.