CIS 662 Makeup Midterm
Due 12/5

Name:_________________________
Points:___________/90

1. (40 points)
   A processor has the following stages in its pipeline: IF ID MEM ALU WB. IF, ID, MEM, ALU and WB stages have the same functionality as in 5-stage MIPS pipeline. ID stage is also used for effective address calculation for memory accesses. ALU stage is used for branch target calculation and branch condition calculation. LOAD, STORE and ALU instructions can access memory. Supported addressing modes are immediate, register direct and displacement addressing. All stages take 1 clock cycle.

   a) (10 points) Find all dependencies in the following code segment and list them by category (data dependence, output dependence, antidependence or control dependence).

   loop:
   ADD R3, R1, 100(R4)
   SUB R6, R5, R3
   STORE R6, 50(R2)
   SUB R2, R2, #8
   ADD R1, R1, 100(R4)
   BNEZ R2, Loop
b) **(10 points)** Assume that there is no forwarding. How many cycles does it take to execute one iteration of the loop? Indicate the total number of stall cycles. Assume that we are using flush pipeline technique for handling branches. Count all the cycles before IF of the next iteration as belonging to the current iteration.
c) **(10 points)** Now apply forwarding to reduce number of stalls wherever possible. Indicate the source and destination stages for forwarding. How many cycles does it take now to execute one iteration of the loop and how many stalls we have? Assume that we are using flush pipeline technique for handling branches. Count all the cycles before IF of the next iteration as belonging to the current iteration.
d) **(10 points)** Can you rearrange the code, just by shuffling commands and adjusting displacements, so that it takes fewer cycles? Assume that we can use delay branches and try to schedule instructions in delay slots (Hint: there may be more than one delay slot in this architecture so you may need to find more than one instruction to go into delay slots). How many cycles does it take now to execute one iteration and how many stalls are left? Count all the cycles before IF of the next iteration as belonging to the current iteration.
2. (20 points)
For the processor from question 1 we are considering an improvement that moves the
**condition calculation only** to ID stage.

a) (2 points) How large is the branch penalty in the original processor, for taken
   and for not taken branches?

b) (2 points) How large is the branch penalty in the modified processor, for
taken and for not taken branches?
c) **(16 points)** The CPU spends 60% of time handling branches, before the improvement. How big is the overall speedup after the improvement, assuming that 70% of branches are not taken? What percentage of time does the CPU spend handling branches after the improvement. If the average CPI for non-branch instructions is 1.5 calculate the percentage of branch instructions in the program mix.
3. (30 points)
Assume the following fragment of code:

\[
\begin{align*}
L.D & \ F10, 100(R1) \\
MUL.D & \ F12, F10, F8 \\
SUB.D & \ F4, F12, F14 \\
SUB.D & \ F14, F18, F0 \\
ADD.D & \ F12, F2, F8 \\
S.D & \ F12, 100(R1)
\end{align*}
\]

Let time 0 be the time when this fragment starts being scheduled and let MUL.D take 7 cycles of execution while ADD.D and SUB.D take 4 cycles of execution. L.D and S.D take 2 cycles of execution each, one for address calculation and one for memory access.

a) (10 points) Fill the table below that shows when each instruction is issued, reads operands, executes and writes result, using scoreboard. Assume there are no structural hazards. If an instruction has to be stalled for some reason, write this reason in the comment field.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Issue</th>
<th>Read operands</th>
<th>Execution starts</th>
<th>Execution ends</th>
<th>Write result</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.D F10, 100(R1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL.D F12, F10, F8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB.D F4, F12, F14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB.D F14, F18, F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD.D F12, F2, F8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.D F12, 100(R1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
b) **(10 points)** Fill the table below that shows when each instruction is issued, executes and writes CDB, using Tomasulo’s algorithm. Assume there are no structural hazards for reservation stations and there is one CDB. If an instruction has to be stalled for some reason, write this reason in the comment field.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Issue</th>
<th>Execution starts</th>
<th>Execution ends</th>
<th>Write CDB</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>L.D F10, 100(R1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL.D F12, F10, F8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB.D F4, F12, F14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB.D F14, F18, F0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD.D F12, F2, F8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S.D F12, 100(R1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
c) **(10 points)** Using the code segment from the problem statement to illustrate your discussion, discuss how Tomasulo’s algorithm handles RAW, WAW and WAR hazards and compare this with a way in which scoreboard handles these same hazards. How much faster did the code segment complete with Tomasulo’s algorithm when compared to scoreboard? What was the reason for speedup? What limited the speedup we could achieve?