Advanced Computer Architecture - FIRST EXAM
Fall 2019

Name: _______________________________

Time: Take Home. There are 9 questions and 5 pages to this test.

1. (20 pts) Your company, Upstart Computers, has designed a non-pipelined version of the RISC-V processor. It currently uses the five stage datapath as described in the textbook (and therefore has a base CPI of 5). One of your employees has just found a way to optimize the hardware so that the CPU clock speed can be doubled, and the MEM phase can be eliminated in those instructions that don’t need it. However, memory access speed has not changed, so at the new CPU clock speed, all memory accesses now take twice as long in terms of clock cycles. The employee claims a two-fold increase in performance on a benchmark with the instruction mix shown below. Calculate the speedup of the new design and determine if the claim is correct.

<table>
<thead>
<tr>
<th>Instruction Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loads</td>
<td>20%</td>
</tr>
<tr>
<td>Stores</td>
<td>10%</td>
</tr>
<tr>
<td>ALU Ops</td>
<td>45%</td>
</tr>
<tr>
<td>Branches</td>
<td>20%</td>
</tr>
<tr>
<td>Other</td>
<td>5%</td>
</tr>
</tbody>
</table>

\[
\text{NEW VERSION (CPI)} = \begin{cases} 
1 & \text{IF+1} \\
1 & \text{MEM+1} \\
1 & \text{BR} \\
1 & \text{MEM} \\
1 & \text{MEM} \\
\end{cases}
\]

\[
\begin{align*}
\text{OLD VERSION (CPI)} &= 5 \\
\text{NEW VERSION (CPI)} &= 1.4 \times 2 = 2.8 \times 2 = 5.6 \\
\text{SPEED UP} &= \frac{5}{5.6} \approx 1.75
\end{align*}
\]

However, clock speed is doubled, so adjusted CPI is:

\[
\frac{5.6}{2} = 2.85 \times \frac{2}{5} = 0.77
\]

Not quite 2, claim is not correct.

2. (10 pts) What type of hazard does a branch delay slot try to minimize? Explain how it works.

- Control hazard.

- Architecture is modified so inst following branch (the BR delay slot) is executed unconditionally, thus avoiding the stall. If an instruction that is independent or can be modified, which logically should execute before the branch, can be found, then it can be moved into the branch delay slot.
3. (12 pts) Hazard types. Below are several modifications that could be made to a processor implementation. Name the type(s) of pipeline hazard that the modification might influence, and state whether the effect would be to increase or decrease the likelihood of the hazard(s). If the modification would not affect a particular hazard type, say 'NONE.'

- Increasing the clock speed by 20%.
  
  **NONE**
  
  **MAYBE:** If memory speed isn't also increased, may cause data hazards

- Increasing the depth (i.e., number of stages) of the execution pipeline.
  
  **INCREASE DATA HAZARDS**
  
  **MAYBE:** control hazards

- Fully pipelining an ALU that is currently not pipelined.
  
  **DECREASE DATA HAZARD OR THE SAME EFFECT**
  
  **MAYBE:** DECREASE STRUCTURAL HAZARD

- Creating an "economy" version of the pipeline that reduces complexity by eliminating all data forwarding.
  
  **INCREASE DATA HAZARDS**

4. (5 pts) Data forwarding is a technique that can eliminate most, but not all, data hazards. For the five stage pipeline we studied in class, give an example where data forwarding cannot completely eliminate the hazard, and note where the hazard occurs.

  \[ \text{LD } R1, 0(R2) \]
  
  \[ \text{ADD } R3, R1, R4 \]
  
  **LOAD CANNOT PROVIDE VALUE OF R1 IN TIME TO PREVENT A DATA HAZARD**
5. (10 pts) What is a structural hazard? Provide an example.

SITUATION WHERE TWO INSTRUCTIONS TRY TO USE SAME HARDWARE IN THE SAME CYCLE (AT THE SAME TIME)

6. (5 pts) In the five stage pipeline, we assumed that we were able to read and write the register file in the same cycle. Describe the assumption we made to allow this to happen.

THE WRITE BACK OF A VALUE INTO THE REGISTER FILE HAPPENS FIRST HALF ON THE RISING EDGE OF THE CLOCK CYCLE, WHILE THE READING OF THAT REGISTER VALUE HAPPENS ON THE FALLING EDGE OF THE CLOCK. SECOND HALF

7. (10 pts) What is speculative execution? Briefly explain. How is speculation implemented in a Tomasulo machine?

a) SITUATION WHERE INSTRUCTIONS FROM BOTH LEGS OF A BRANCH ARE EXECUTED UNTIL THE FINAL BRANCH DECISION IS KNOWN.

b) WITH TOMASULO, INSTRUCTIONS FROM BOTH LEGS ARE ALLOWED TO EXECUTE WITH ANY RESULTS STORED IN THE REORDER BUFFER. THESE RESULTS ARE NOT COMMITTED UNTIL THE BRANCH RESULT IS DETERMINED. AT THAT TIME, THE NON-TAKEN RESULTS ARE DISCARDED.

8. (5 pts) What is the purpose of a reservation station in a Tomasulo processor?

HOLDS AN INSTRUCTION THAT HAS BEEN ISSUED, BUT IS WAITING ON ONE OR MORE OPERANDS
9. (25 pts) a) Consider a processor that uses the “standard” five stage pipeline, that includes all possible forwarding paths, and a branch delay slot. Show the pipeline timing diagram for one iteration of the following loop, and determine the number of cycles it takes from the start of execution of the first load in the loop until the start of execution of the same load in the next iteration. Do not rearrange the code. The first instruction is done for you. NOTE: The **nop** instruction is included to fill the otherwise unused branch delay slot. (Add more columns to your table if needed.)

<table>
<thead>
<tr>
<th></th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>loop:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ld x1,0(x3)</code></td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
</tr>
<tr>
<td><code>addi x1,x1,1</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>ld x2,0(x4)</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>add x2, x2, x1</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>sd x2,0(x3)</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>addi x3,x3,8</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>addi x4,x4,8</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>bne x3,x5,loop</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>nop</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
<tr>
<td><code>ld</code></td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
<td>S</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td>IF</td>
<td>ID</td>
</tr>
</tbody>
</table>

**11 CYCLES**
b) Now repeat part a), except this time you can rearrange the code, eliminate any unnecessary instructions, and utilize the branch delay slot to make the code run as fast as possible. (The original code is replicated below for your convenience).

```
loop:  ld  x1, 0(x3)
       addi x1, x1, 1
       ld  x2, 0(x4)
       add x2, x2, x1
       sd  x2, 0(x3)
       addi x3, x3, 8
       addi x4, x4, 8
       bne x3, x5, loop
       pop
```

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>loop:</td>
<td>ld</td>
<td>x1, 0(x3)</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MM</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

c) Determine the speedup obtained in part b) compared with part a). (This speedup represents what an optimizing compiler might be able to achieve with this code.)

\[
\text{SPEEDUP} = \frac{11}{6} = 1.833
\]