CIS 662: Sample midterm w solutions

1. (40 points)
A processor has the following stages in its pipeline: IF ID ALU1 MEM1 MEM2 ALU2 WB. ALU1 stage is used for effective address calculation for loads, stores and branches. ALU2 stage is used for all other calculations and for branch resolution. The only instructions that can access memory are load and store. The only supported addressing mode is displacement addressing. Because we have a slow memory unit, access to memory is pipelined through two stages – MEM1 and MEM2.

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

LD R1, 50(R2)
ADD R3, R1, R4
LD R5, 100(R3)
MUL R6, R5, R7
STORE R6, 50(R2)
ADD R1, R1, #100
SUB R2, R2, #8

Solution:
Let’s number instructions:
1. LD R1, 50(R2)
2. ADD R3, R1, R4
3. LD R5, 100(R3)
4. MUL R6, R5, R7
5. STORE R6, 50(R2)
6. ADD R1, R1, #100
7. SUB R2, R2, #8

Data dependencies:
Instruction 2 depends on instruction 1 for value R1
Instruction 6 depends on instruction 1 for value R1
Instruction 3 depends on instruction 2 for value R3
Instruction 4 depends on instruction 3 for value R5
Instruction 5 depends on instruction 4 for value R6

Antidependencies:
Instruction 6 is antidependent on instruction 2 for access to R1
Instruction 7 is antidependent on instruction 1 for access to R2
Instruction 7 is antidependent on instruction 5 for access to R2

Output dependencies:
Instruction 6 has output dependence with instruction 1 for access to R1
Control dependencies:
None

b) **(10 points)** Assume that there is no forwarding. How many cycles does it take to execute the above code segment? Indicate the total number of stall cycles.

**Solution:**

<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>
</tr>
</thead>
<tbody>
<tr>
<td>LD R1, 50(R2)</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD R3, R1, R4</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td>LD R5, 100(R3)</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td>MUL R6, R5, R7</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORE R6, 50(R2)</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td>ADD R1, R1, #100</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td>SUB R2, R2, #8</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></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>ALU2</td>
<td></td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>s</td>
</tr>
</tbody>
</table>

LD R1, 50(R2)  IF at time 1, WB at time 7
ADD R3, R1, R4  IF at time 2, stalls 3-6, ID at time 7, WB at time 12
LD R5, 100(R3)  IF at time 7, stalls 8-11, ID at time 12, WB at time 17
MUL R6, R5, R7  IF at time 12, stalls 13-17, ID at time 17, WB at time 22
STORE R6, 50(R2)  IF at time 17, stalls 18-21, ID at time 22, WB at time 27
ADD R1, R1, #100  IF at time 23, WB at time 28
SUB R2, R2, #8  IF at time 24, WB at time 29

It takes 29 cycles. We stall for 16 cycles.

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 the above code segment and how many stalls we have?
Solution:
Stars denote stages when result is ready and when is needed

<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>
</tr>
</thead>
<tbody>
<tr>
<td>LD R1, 50(R2)</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2*</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD R3, R1, R4</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td><em>ALU2</em></td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD R5, 100(R3)</td>
<td>IF</td>
<td>ID</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>*ALU1</td>
<td>MEM1</td>
<td>MEM2*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL R6, R5, R7</td>
<td>IF</td>
<td>s</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORE R6, 50(R2)</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD R1, R1, #100</td>
<td>IF</td>
<td>ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB R2, R2, #8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALU2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>ALU2</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LD R1, 50(R2) R1 available at the end of MEM2 at time 5, ADD R3, R1, R4 R1 needed at the beginning of ALU2 at time 7 We can forward from MEM2 to MEM2 or from ALU2 to ALU2 ADD R3, R1, R4 R3 available at the end of ALU2 at time 7 LD R5, 100(R3) R3 needed at the beginning of ALU2 at time 5 We need 3 stalls and then we forward from ALU2 to ALU1 LD R5, 100(R3) R5 available at the end of MEM2 at time 10 MUL R6, R5, R7 R5 needed at the beginning of ALU2 at time 12 We can forward from MEM2 to MEM2 or from ALU2 to ALU2 MUL R6, R5, R7 R6 available at the end of ALU2 at time 12 STORE R6, 50(R2) R6 needed at the beginning of MEM1 at time 11 We need 2 stalls and then we forward from ALU2 to MEM1

It takes 18 cycles. We have 5 stalls.

d) **(10 points)** Can you rearrange the code, just by shuffling commands and adjusting displacements, so that it takes less cycles? How many cycles does it take now to execute the code segment and how many stalls are left?
Solution:
We can move last two instructions before the STORE, to eliminate two stall cycles. Then we adjust the offset in STORE.

<table>
<thead>
<tr>
<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>
</tr>
</thead>
<tbody>
<tr>
<td>LD R1, 50(R2)</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2*</td>
<td>ALU2</td>
<td>WB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD R3, R1, R4</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td><em>ALU2</em></td>
<td>WB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LD R5, 100(R3)</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td>MEM2</td>
<td><em>ALU2</em></td>
<td>WB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUL R6, R5, R7</td>
<td>IF</td>
<td>S</td>
<td>S</td>
<td>s</td>
<td>s</td>
<td>ID</td>
<td>ALU1</td>
<td>MEM1</td>
<td></td>
</tr>
<tr>
<td>ADD R1, R1, #100</td>
<td>IF</td>
<td>ID</td>
<td>ALU1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB R2, R2, #8</td>
<td>IF</td>
<td>ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STORE R6, 58(R2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It takes 16 cycles. We have 3 stalls left.

2. (33 points)
For the processor from question 1
   a) **(4 points)** How large is the branch penalty?

   Solution:
   Since branches are resolved in ALU2 stage, we can start IF only after that. We would like to start after IF stage. So the penalty is 5 cycles

   b) **(4 points)** Assume that we can introduce an optimization so that branches are resolved in ALU1 stage (along with effective address calculation). How large is the branch penalty now?

   Solution:
   2 cycles

   c) **(10 points)** The optimization we introduced can be used in 75% of branches. Branches represent 30% of all instructions in our usual workload. Ideal CPI is 1. What is the average CPI?

   Solution:
   
   \[
   \text{CPI} = 1 + 0.3 \times \text{branch\_penalty} = 1 + 0.3 \times (0.75 \times 2 + 0.25 \times 5) = 1.795
   \]
d) **(15 points)** Assume that we are choosing between flush pipeline, predict taken and predict not taken strategy for handling branches. On the average 80% of branches are conditional branches and 60% of conditional branches are taken. What is the average CPI for each branch handling approach and which approach is the best?

**Solution:**

\[
\text{CPI} = 1 + 0.3 \times \text{branch penalty} \\
= 1 + 0.3 \times (\%\text{conditional} \times (\%\text{taken} \times \text{penalty\_taken} + \%\text{not\_taken} \times \text{penalty\_not\_taken} + \%\text{jumps} \times \text{penalty\_jumps})
\]

For jumps, for each approach penalty is 2 cycles.
For predict not taken penalty is 0 cycles for not taken branches and 5 cycles for taken branches.
For predict taken, penalty is 2 cycles for taken branches and 5 cycles for not taken branches.
For flush pipeline penalty is 5 cycles.

\[
\text{CPI(flush pipeline)} = 1 + 0.3 \times (0.8 \times (0.6 \times 5 + 0.4 \times 5) + 0.2 \times 2) = 2.32 \\
\text{CPI(predict taken)} = 1 + 0.3 \times (0.8 \times (0.6 \times 2 + 0.4 \times 5) + 0.2 \times 2) = 1.888 \\
\text{CPI(predict not taken)} = 1 + 0.3 \times (0.8 \times (0.6 \times 5 + 0.4 \times 0) + 0.2 \times 2) = 1.84 \text{ this is the best}
\]

3. **(20 points)**
Assume the following MIPS code:

```
DADD R1, R0, R0
Loop:  BNEZ R1, If2
       DADDI R1, R0, #2
If2:   DADDI R1, R0, #-1
       J Loop
Done:
```

a) **(10 points)** Use a one-bit predictor to predict outcomes of this branch. How many misses you have? Assume that initial prediction is not-taken

First time: prediction NT, outcome NT, new prediction NT, R1 becomes 1
Second time: prediction NT, outcome T, new prediction T, R1 becomes 0 – miss
Third time: prediction T, outcome NT, new prediction NT, R1 becomes 1 – miss
Fourth time: prediction NT, outcome T, new prediction T, R1 becomes 0 – miss

Every time but the first one we have a miss
b) **(10 points)** Use a two-bit predictor to predict outcomes of this branch. How many misses you have? Assume that initial prediction is 00

First time: prediction 00, outcome NT, new prediction 00, R1 becomes 1
Second time: prediction 00, outcome T, new prediction 01, R1 becomes 0 – miss
Third time: prediction 01, outcome NT, new prediction 00, R1 becomes 1
Fourth time: prediction 00, outcome T, new prediction 01, R1 becomes 0 – miss

Every second time we have a miss