Question 1:
Assume that moving the logic for the evaluation of the branch condition and target address from the ALU stage of the 5-stage pipeline to the ID stage causes an increase in the clock cycle time by 5%. Assume also that 20% of the instructions executing on the pipeline are branch instructions and that 60% of the branches are taken.
(a) Assume that the CPI = 2.12 when the branch condition and target are evaluated in the ALU stage. Will resolving the branch condition/target in the ID stage result in a more efficient execution?

\[
CPI_{new} = CPI_{old} - CPI_{improvement} = 2.12 - 20\% \times 60\% \times 1 = 2
\]

\[
\frac{\text{ExecutionTime}_{new}}{\text{ExecutionTime}_{old}} = \frac{\#\text{instructions}_{new} \times CPI_{new} \times \text{ClockCycleTime}_{new}}{\#\text{instructions}_{old} \times CPI_{old} \times \text{ClockCycleTime}_{old}} = 1 \times \frac{2}{2.12} \times \frac{1.05}{1} = 0.99
\]

Yes, resolving the branch condition/target in the ID stage will be more efficient.

(b) The proposed change creates a new type of data. For example, the following sequence of instructions

\[
\text{add} \quad \$6, \$2, \$2 \\
\text{beq} \quad \$5, \$6, L
\]

will cause a hazard that cannot be resolved by forwarding from EX/MEM or MEM/WB to ID/EX. Explain why will the above sequence of instructions cause a problem and describe the new forwarding paths that are needed to resolve this new type of hazard.

A new type of data hazard happens when a branch instruction which reads a register, R, follows an instruction which writes into register R. The two forwarding paths discussed so far will replace the wrong value of R before that value is used in the EX stage. But that would be too late since the branch instruction uses that value (the wrong value) when it is still in the ID stage.

<table>
<thead>
<tr>
<th>Cycle 1</th>
<th>IF</th>
<th>ID</th>
<th>EXE</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>add</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle 2</td>
<td>beq</td>
<td>add</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle 3</td>
<td></td>
<td>beq (reads and uses $6)</td>
<td>add(computes $6)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle 4</td>
<td></td>
<td>beq</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cycle 5</td>
<td></td>
<td>beq</td>
<td></td>
<td></td>
<td>add</td>
</tr>
<tr>
<td>Cycle 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>beq</td>
</tr>
</tbody>
</table>
To solve this hazard, we need to extend the two forwarding paths (from MEM/WB and EX/MEM to ID/EX) so that we can overwrite the inputs to the unit which is added to the ID stage to check that the contents of the two registers are equal.

Questions 2:
Consider a five stage pipeline in which the branch target address is determined in the ID stage and the branch condition is determined in the MEM stage. Assume that the instruction mix contains 20% branch instructions, 70% of which are taken. Assume also that the CPI is 1.9 without accounting for the overhead of dealing with control hazards. Compute the CPI when each of the following techniques is used to avoid control hazards.

(a) Always predict “branch not taken” and take a corrective action if the prediction is wrong

\[
\text{CPI} = 1.9 + 0.2 \times 0.7 \times 3 = 1.9 + 0.42 = 2.32
\]

(a) Always predict “branch taken” and take a corrective action if the prediction is wrong

\[
\text{CPI} = 1.9 + 0.2 \times (0.7 \times 1 + 0.3 \times 2) = 1.9 + 0.2 \times 1.3 = 1.9 + 0.26 = 2.16
\]

(a) Assume a branch predictor with an 80% accuracy. That is, the predictor makes the right prediction 80% of the time and the wrong prediction 20% of the time (these ratios include the cases where the predictor cannot make a prediction).

Assuming Branch predictor also predicts the target address

\[
\text{CPI} = 1.9 + 0.2 \times (0.2 \times 3) = 1.9 + 0.12 = 2.02
\]
**Question 3:** Consider the following loop which computes $a[i]=b[i]+a[i]$, $i=1,2,...$:

```
L: lw $t0, 1000($s4) /* load $a[i] */
    lw $t1, 4000($s4) /* load $b[i] */
    addi $s4, $s4, 4
    add $t0, $t0, $t1 /* compute temp = $a[i] + $b[i] */
    sw $t0, 996($s4) /* store temp in $a[i] */
    beq $s4, $s6, L /* repeat until $i$ reaches a maximum value */
```

(a) Show the scheduling of one iteration of the loop on a superscalar (static) 2-issue architecture with two pipelines, one for ALU/branch instructions and the other for lw/sw instructions. Assume that the architecture has forwarding and data hazard hardware.

<table>
<thead>
<tr>
<th>ALU/branch pipeline</th>
<th>Load/store pipeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>lw</td>
</tr>
<tr>
<td></td>
<td>Cycle 1</td>
</tr>
<tr>
<td>addi</td>
<td>lw</td>
</tr>
<tr>
<td></td>
<td>Cycle 2</td>
</tr>
<tr>
<td>add</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cycle 3</td>
</tr>
<tr>
<td>beq</td>
<td>sw</td>
</tr>
<tr>
<td></td>
<td>Cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) How many cycles will it take to execute the above loop 1000 times on the 2-issue architecture assuming perfect branch prediction (always correct prediction)?

- One iteration $\rightarrow$ 6 instructions in 5 cycles
- 1000 iterations $\rightarrow$ 6000 instructions $\rightarrow$ 5000 cycles $\rightarrow$ CPI = 5/6

(c) If the branch is resolved when the instruction is in the ID stage (second stage), how many cycles will it take to execute the above loop 1000 times on the 2-issue architecture assuming no branch prediction?

Due to wrong predictions, there will be one bubble after each branch (except the last)

- One iteration $\rightarrow$ 6 instructions in 6 cycles
- 1000 iterations $\rightarrow$ 6000 instructions $\rightarrow$ 6000 cycles $\rightarrow$ CPI = 6/6 = 1

(d) How many cycles will it take to execute the same loop 1000 times on a single issue architecture (the usual 5-stage pipeline) assuming perfect branch prediction?

There will be no data hazard if the loop executes on a single pipeline:

- One iteration $\rightarrow$ 6 instructions in 6 cycles
- 1000 iterations $\rightarrow$ 6000 instructions $\rightarrow$ 6000 cycles $\rightarrow$ CPI = 1
(e) Assuming perfect branch prediction, how many cycles will it take to execute the same loop 1000 times if the loop is unrolled to:

L:  
\begin{align*}
    &lw \ $t0, 1000($s4) \\
    &lw \ $t1, 4000($s4) \\
    &lw \ $t2, 1004($s4) \\
    &lw \ $t3, 4004($s4) \\
    &add \ $s4, $s4, 8 \\
    &add \ $t0, $t0, $t1 \\
    &add \ $t2, $t2, $t3 \\
    &sw \ $t0, 992($s4) \\
    &sw \ $t2, 996($s4) \\
    &beq \ $s4, $s6, L
\end{align*}

Two iterations of original loop \(\rightarrow\) one iteration of the unrolled loop \(\rightarrow\) 7 cycles
1000 iterations of the original loop \(\rightarrow\) 3500 cycles