# Architecture Tutorial 4 – John Sinclair – 16325734

## Q1.

|  |
| --- |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |

## Q2.

1. ALU Forwarding is enabled
2. ALU forwarding is disabled with CPU data dependency interlocks enabled
3. ALU Forwarding and CPU data dependency interlocks disabled

|  |  |  |  |
| --- | --- | --- | --- |
|  | R1 | R2 | Clock Cycles |
|  | 2 | 0 | 10 |
|  | 2 | 0 | 16 |
|  | 2 | 2 | 10 |

When ALU forwarding is disabled, stalls are introduced to the pipeline in order to avoid hazards. These stalls eliminate the hazards, but they increase the clock cycle count. This is why the result remains the same, but the number of clock cycles is greater.

When both ALU forwarding and CPU data dependency interlocks are disabled, the system takes no measures to prevent hazards and so the clock cycle count is the same, but due to those hazards the values in R1 and R2 are incorrect.

## Q3.

Part 1

Instructions executed: 38

Clock cycles: 50

Answer (R1): 48

A total of 12 stalls are introduced to the pipeline. Stalls are introduced to the pipeline due to a few things:

* Initially the pipeline is empty and so requires 4 clock cycles to fill.
* The branch prediction adds stalls before the jump statement if the prediction is false. Due to the loops in the program this stall is repeated 4 times.
* In order to avoid hazards arising from the data dependency when executing the LD and SRLi instructions. Due to the loop in the program hazard avoidance stall is added to the pipeline 4 time.

These three causes account for all 12 of the stalls introduced to the pipeline.

Part 2

With delayed branches this program outputs 78, the incorrect answer.

In order to fix the program and make it a valid multiplication program I introduced an NOP instruction after every branch or jump instruction(the two BEQZ, and the J). This introduces stalls to the pipeline in the required areas.

Part 3

The data dependency in this program that results in the necessary stalling of the pipeline occurs when the LD instruction executes, the following instruction SRLi requires the register that is being loaded and so a necessary stall is introduced to the pipeline.

We can avoid this data dependency by simply swapping the order of the two shift instructions, that way a stall is no longer needed.

|  |
| --- |
| Diagram  Description automatically generated |

These numbers are different from the original clock cycle count as the program no longer needs to introduce stalls to the pipeline to avoid the data dependency hazard.