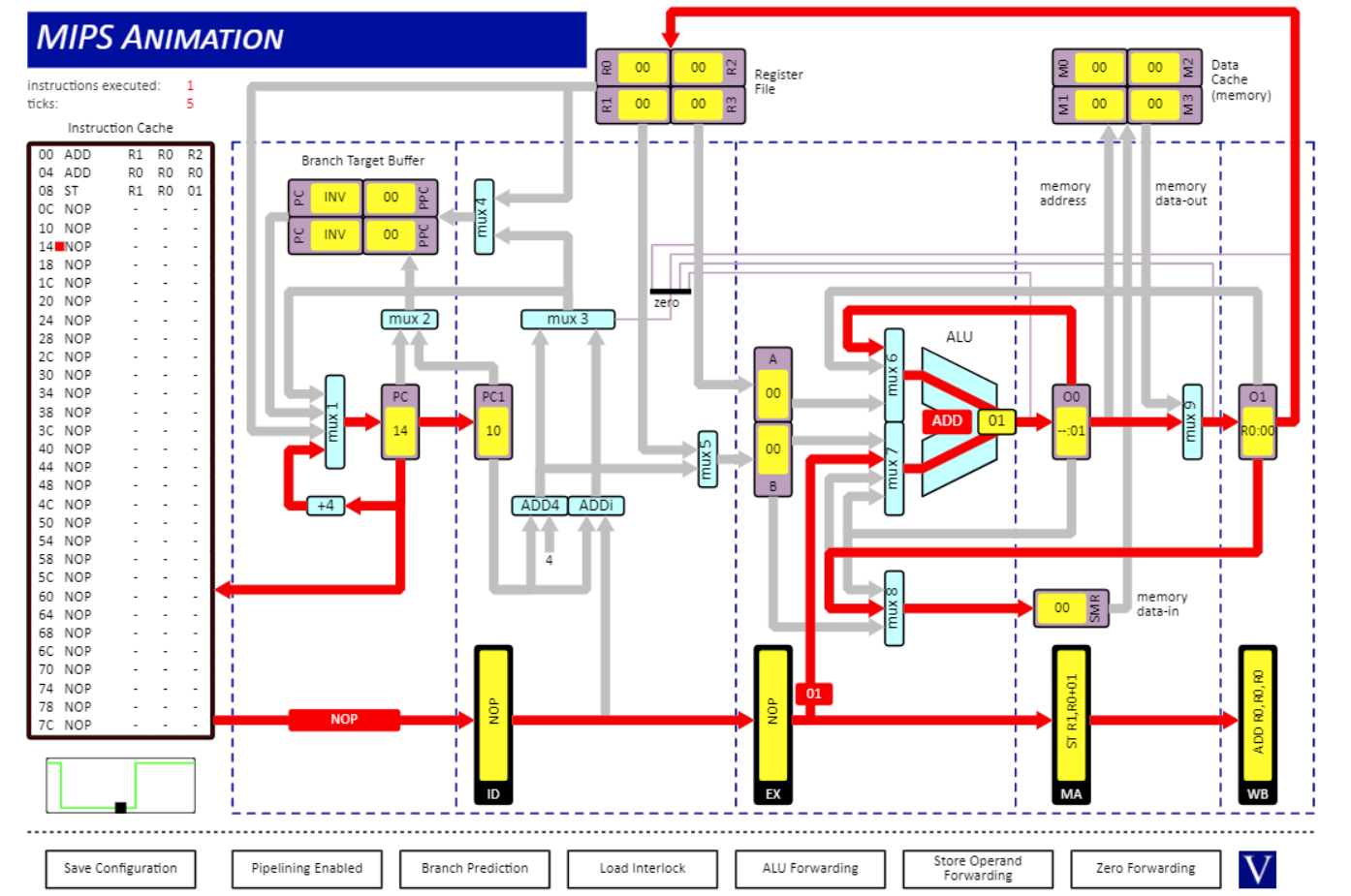
**Tutorial 4**

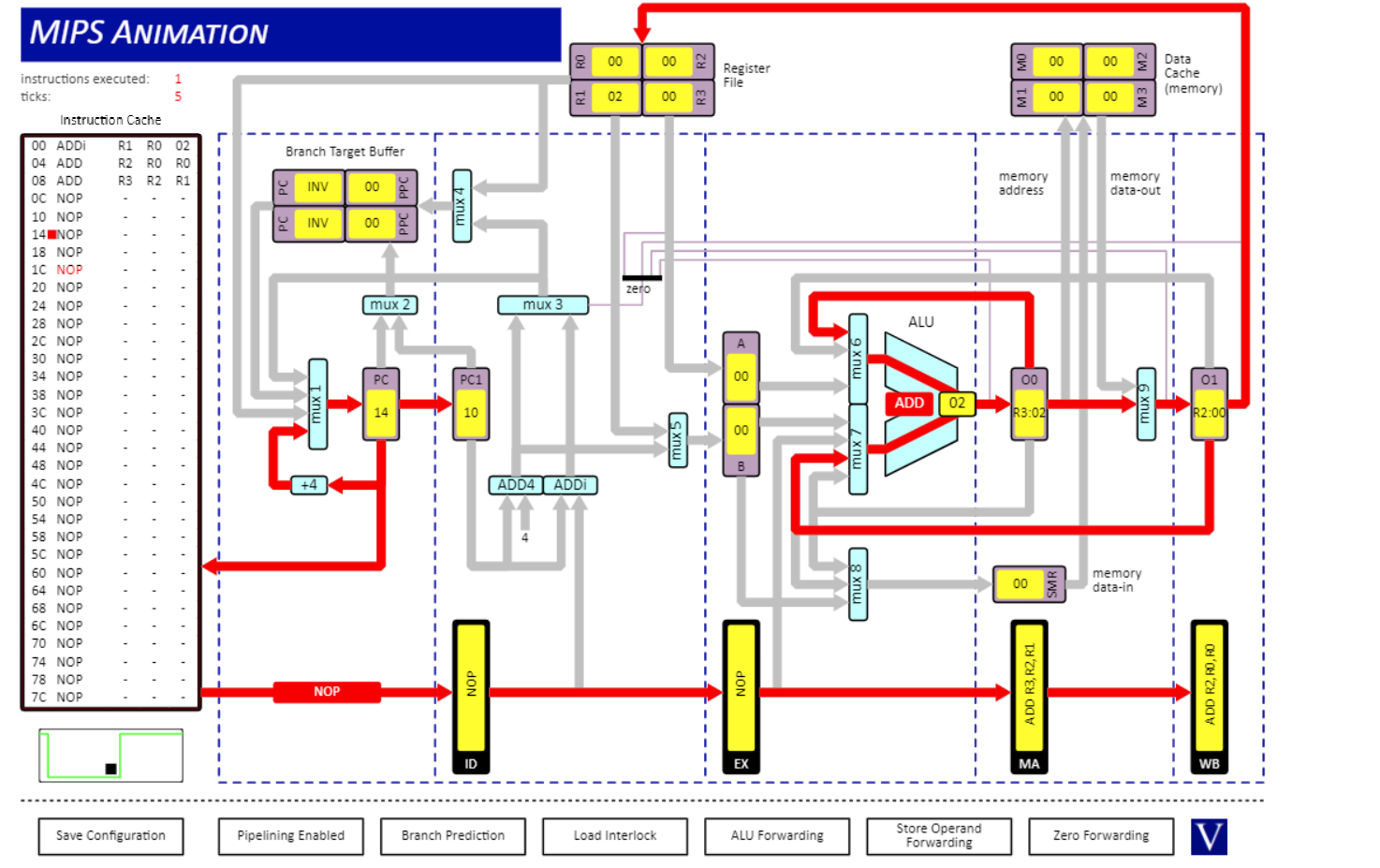
**Gregory Partridge- 17331009**

Q1

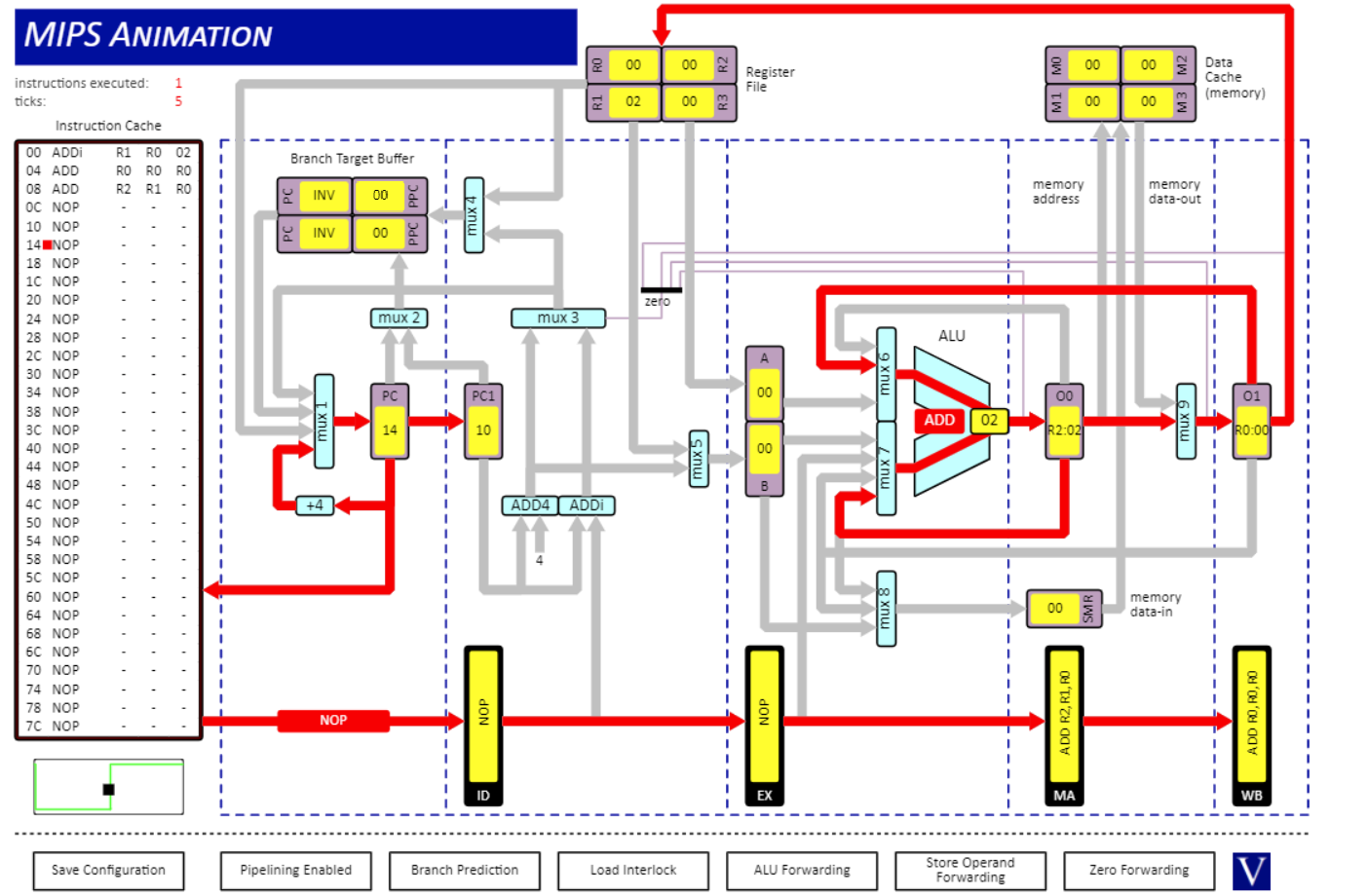
1.



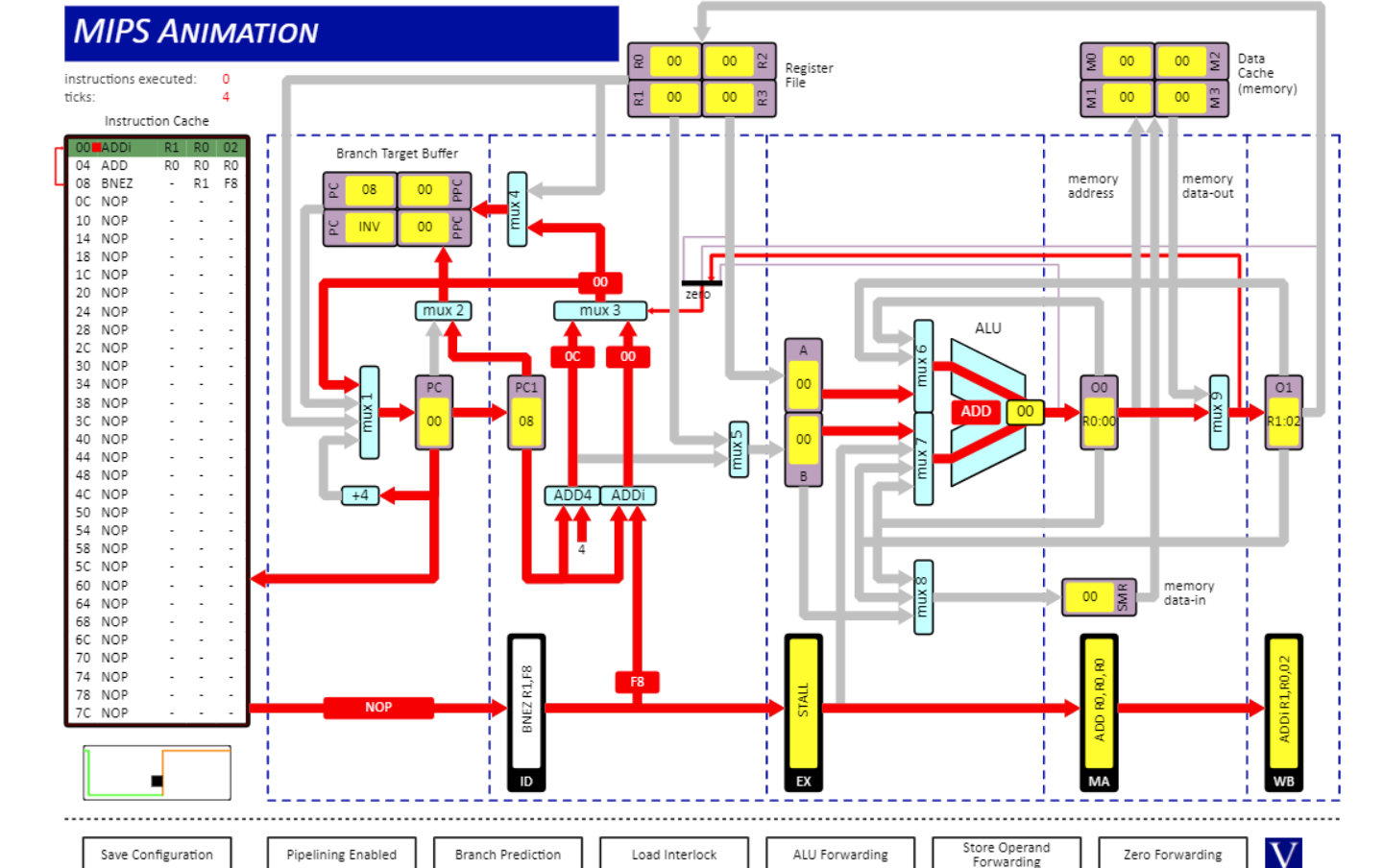
2.



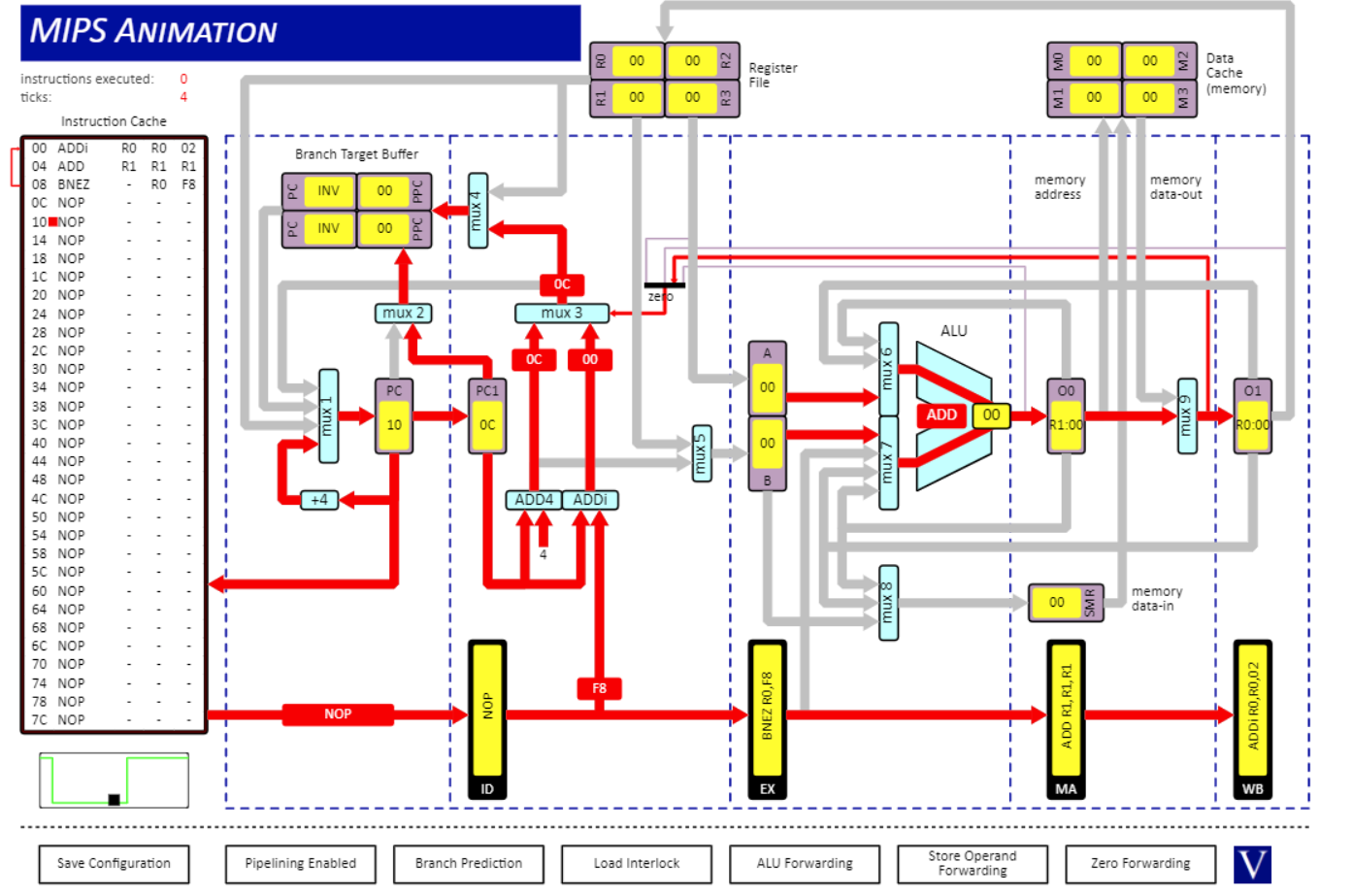
3.



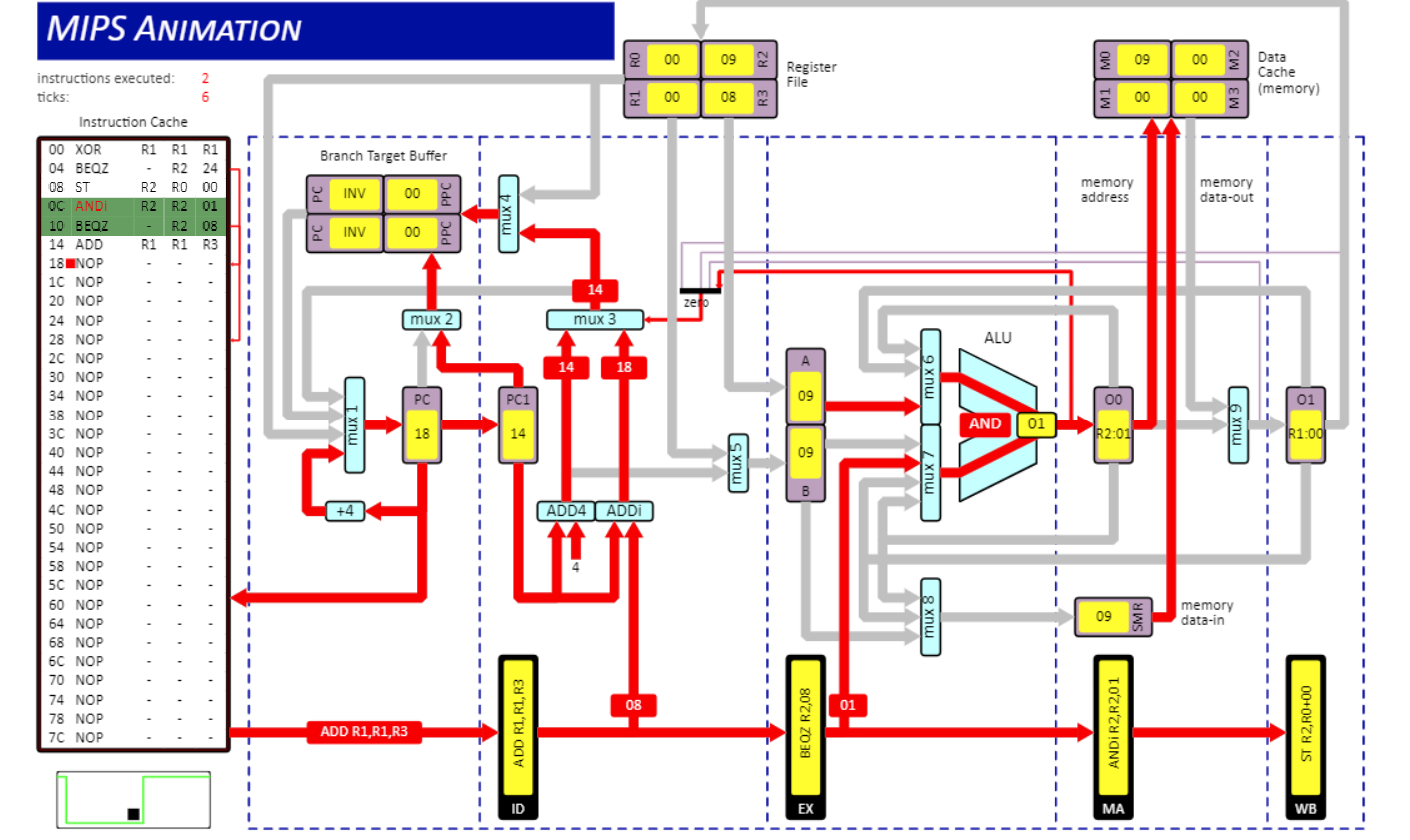
4.



5.

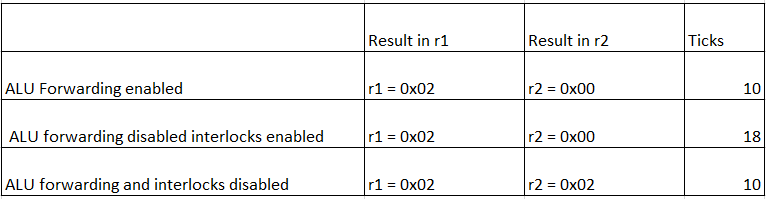


6.

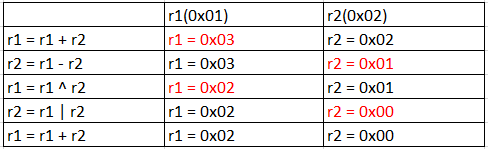


Q2)

The table below shows the results of the given code ran with ALU forwarding enabled, ALU forwarding disabled, interlocks enabled and ALU forwarding disabled, interlocks disabled. They include the register results and the tick count necessary to complete the code even after the HALT instruction.

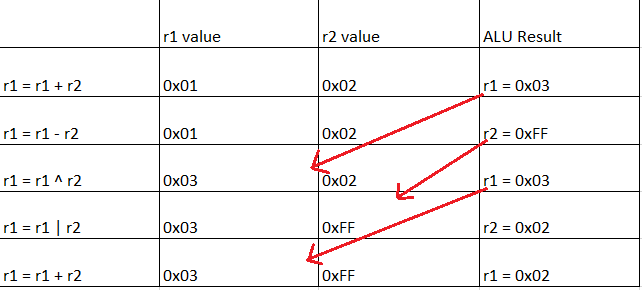


Both ALU forwarding and ALU interlocking produce the same register result of r1 = 0x02, r2 = 0x00. The result is acquired as seen by the table below.



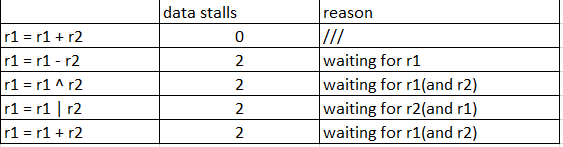
For ALU no interlock, there is no data stall so it may use values from the register before they have been written back into the register file. This is why the results from ALU no interlock differ from ALU forwarding and ALU interlock.

The results from the instructions takes 2 phases to be written back to the register file as described below. The final answers are the last two ALU results. As there is no data stall so the tick rate is identical to ALU forwarding.



ALU interlock has a notable 18 ticks compared to the ALU forwarding and ALU no interlock have both 10. The 10 is from 6 being from executing the instructions and 4 to fill the pipeline.

The added 8 for ALU interlock are due to the data stall. After the initial instruction the other four instructions wait 2 ticks for the results to reach the register file and only then do they continue to the next instruction.



Q3)

1. 38 instructions are executed while the tick count is 50. The tick count is higher to the amount of instructions due to two factors. 38 of the tick count comes from executing instructions. Another 4 come form filling the pipeline adding the tally to 42. The other 8 come from data stalls and control stalls. The control stalls primarily appear during the BEQZ instructions while all the data stalls appear at the SRLi instruction.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR | 1 |  |  |  |  |
| BEQZ | 2 | 11 | 19 | 27 | 36 +1 |
| ST | 3 | 12 | 20 | 28 |  |
| ANDi | 4 | 13 | 21 | 29 |  |
| BEQZ | 5 | 14 +1 | 22 | 30 +1 |  |
| ADD | 6 |  |  | 31 |  |
| LD | 7 | 15 | 23 | 32 |  |
| SRLi | 8 +1 | 16 +1 | 24 +1 | 33 +1 |  |
| SLLi | 9 | 17 | 25 | 34 |  |
| J | 10 +1 | 18 | 26 | 35 |  |
| ST |  |  |  |  | 37 |
| HALT |  |  |  |  | 38 |

1. 53 instructions are executed while the tick count is 57. The increase in instructions is from “Branch Prediction” to “Delayed Branches” which stems from the added NOP lines after any jump instruction such as BEQZ or J. This adds 15 extra instructions to the original program. The remaining 4 ticks can be attributed to filling the pipeline.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR | 1 |  |  |  |  |
| BEQZ | 2 | 14 | 26 | 38 | 50 |
| NOP | 3 | 15 | 27 | 39 | 51 |
| ST | 4 | 16 | 28 | 40 |  |
| ANDi | 5 | 17 | 29 | 41 |  |
| BEQZ | 6 | 18 | 30 | 42 |  |
| NOP | 7 | 19 | 31 | 43 |  |
| ADD | 8 | 20 | 32 | 44 |  |
| LD | 9 | 21 | 33 | 45 |  |
| SRLi | 10 | 22 | 34 | 46 |  |
| SLLi | 11 | 23 | 35 | 47 |  |
| J | 12 | 24 | 36 | 48 |  |
| NOP | 13 | 25 | 37 | 49 |  |
| ST |  |  |  |  | 52 |
| HALT |  |  |  |  | 53 |

1. By swapping the 2 shift instructions the new tick count is 46. This is so as SLLi uses the result of LD reducing the 4 data stalls. The result is the same 38 instructions being executed in a 46 tick count.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| XOR | 1 |  |  |  |  |
| BEQZ | 2 | 11 | 19 | 27 | 36 +1 |
| ST | 3 | 12 | 20 | 28 |  |
| ANDi | 4 | 13 | 21 | 29 |  |
| BEQZ | 5 | 14 +1 | 22 | 30 +1 |  |
| ADD | 6 |  |  | 31 |  |
| LD | 7 | 15 | 23 | 32 |  |
| SLLi | 8 | 16 | 24 | 33 |  |
| SLRi | 9 | 17 | 25 | 34 |  |
| J | 10 +1 | 18 | 26 | 35 |  |
| ST |  |  |  |  | 37 |
| HALT |  |  |  |  | 38 |