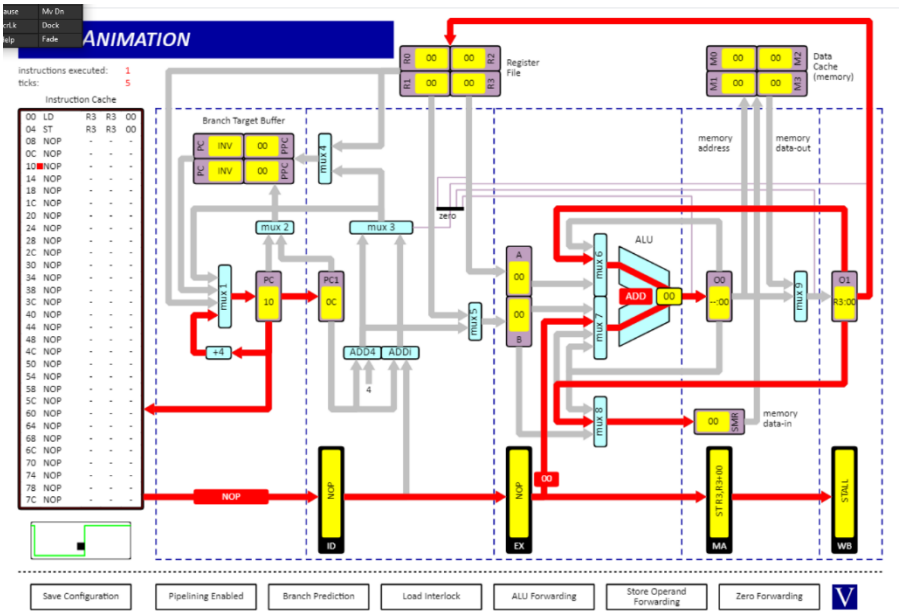
Computer Architecture Tutorial 4

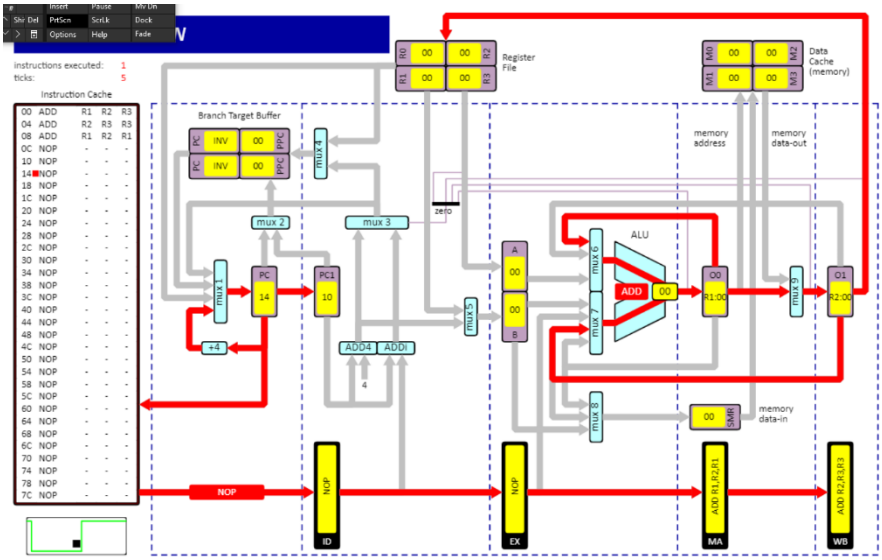
Daniel Nugent #18326304

Q1

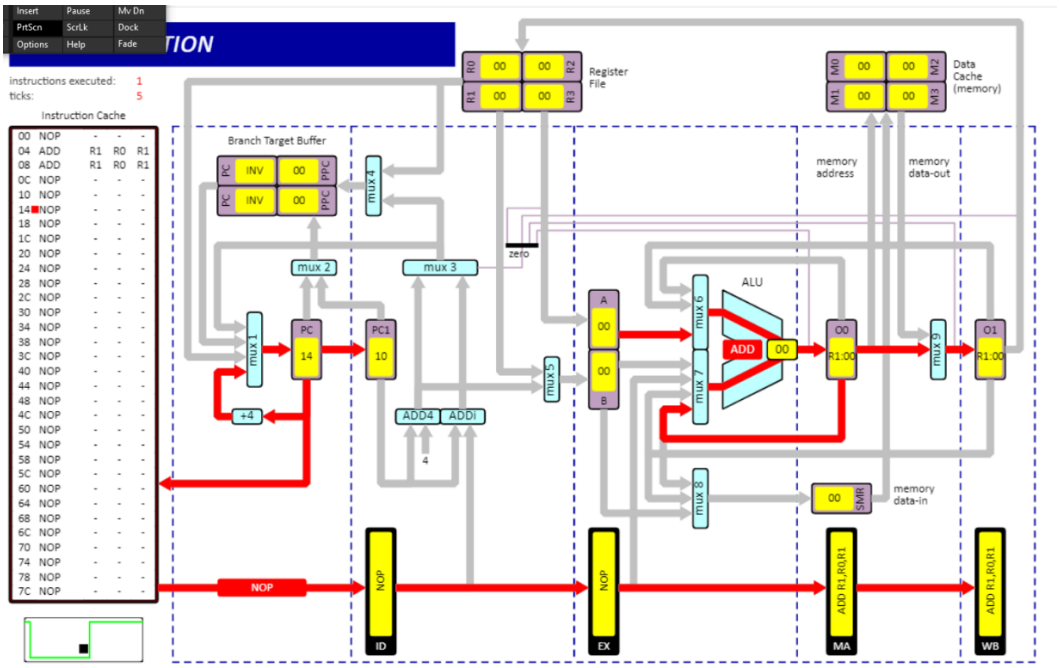
1. O1 to Mux8
   LD R3, R3
   ST R3, R3



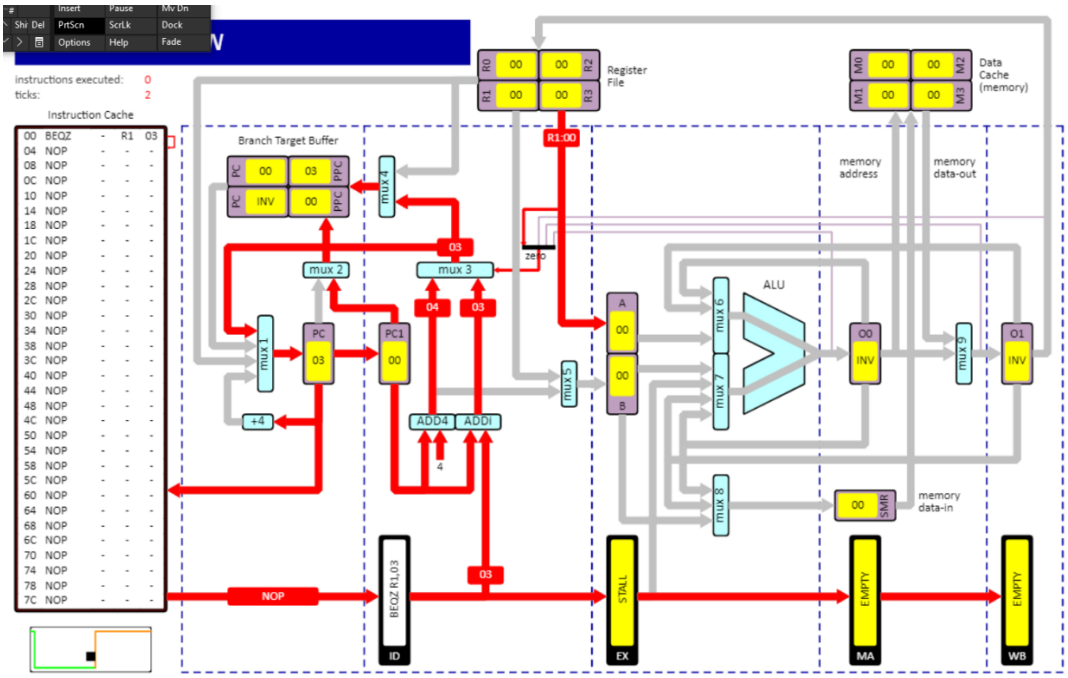
2. O0 to Mux6 and O1 to Mux7 (simultaneously)
   ADD R1, R2, R3
   ADD R2, R3, R3
   ADD R1, R2, R1



3. O0 to Mux7
   ADD R1, R0, R1
   ADD R1, R0, R1

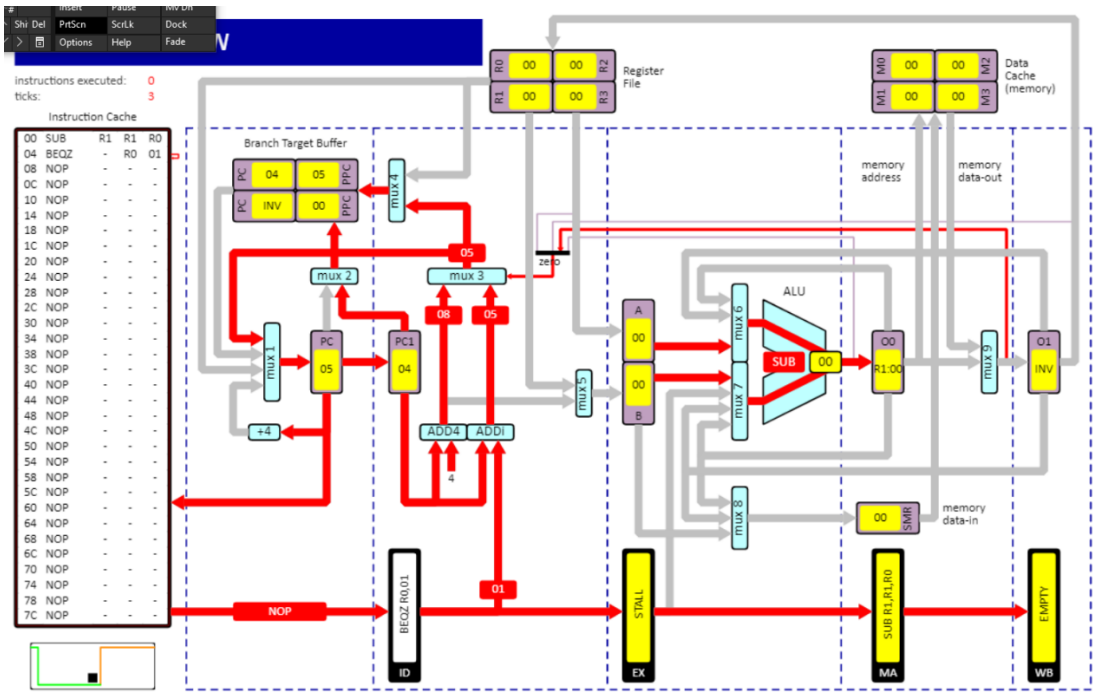


4. ID to Mux3 to Mux1/Mux4
   BEQZ _ R1 03

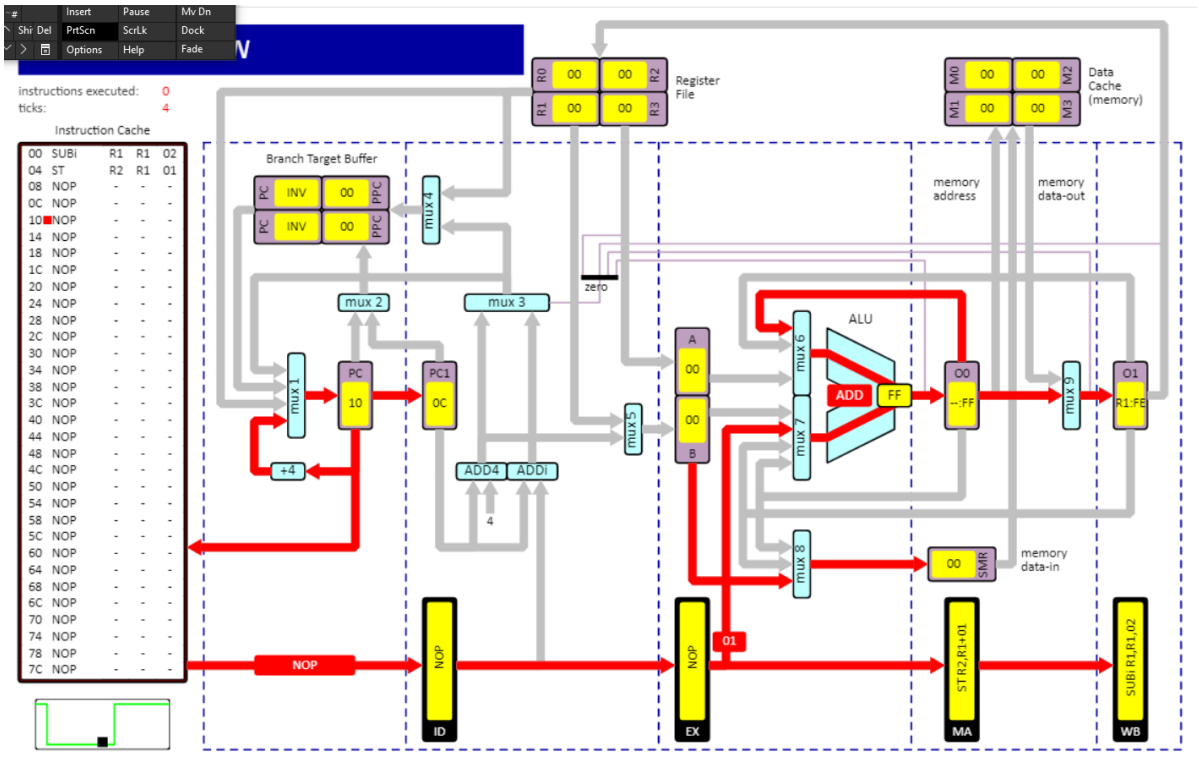


5. Mux9 out to Zero detector
   SUB R1, R1, R0

   BEQZ _ R0 00



6. EX to Mux7 and B to Mux8 simultaneously
   SUBi R1, R1, 02
   ST R2, R1, 01



Q2

i)
R1 = 2
R2 = 0
10 Clock cycles

This is the normal operation of the processor. The answers are correct.

ii)
R1 = 2
R2 = 0
18 Clock cycles

With ALU forwarding is disabled with CPU data dependency interlocks enabled the result is correct, but because the ALU isn't able to forward previous answers from O0 and O1 into the next clock cycle, so the CPU needs to wait for registers to be updated with the new values before continuing to the next instruction. If the CPU didn't wait, it would access the old values of the registers instead of the new updated ones, so it must wait for the ALU to pass the new data back into the registers. Pipeline is stalled.

iii)
R1 = 2
R2 = 2
10 Clock cycles

So, the answers are wrong because the CPU is blindly taking in the old values of the registers without waiting for them to be updated. So the clock cycles is the same for part i) but without ALU forwarding, like in part ii) the values aren't stored in O0 and 01 so the CPU must wait for the new values to be present in the registers before continuing, but it doesn't so hence the faster time to complete but wrong answer.

Q3

i) 50 clock cycles
38 instructions are executed
It takes more clock cycles than instructions executed because the pipeline must stall for jump instructions such as the loop. Then there are stalls for load instruction as the next instruction must wait for the value in memory to enter the register R2. There are 4 pipeline stalls at the beginning aswell.

LD and SRLi 4 times
Beginning 4 times
Jump loop 2 times
BEQZ two times

ii) We need to add in some NOP instructions to fix this program as there is no more branch predicting and we are delaying branching. Also NOP's are added after BEQZ to let the register value update

This results in

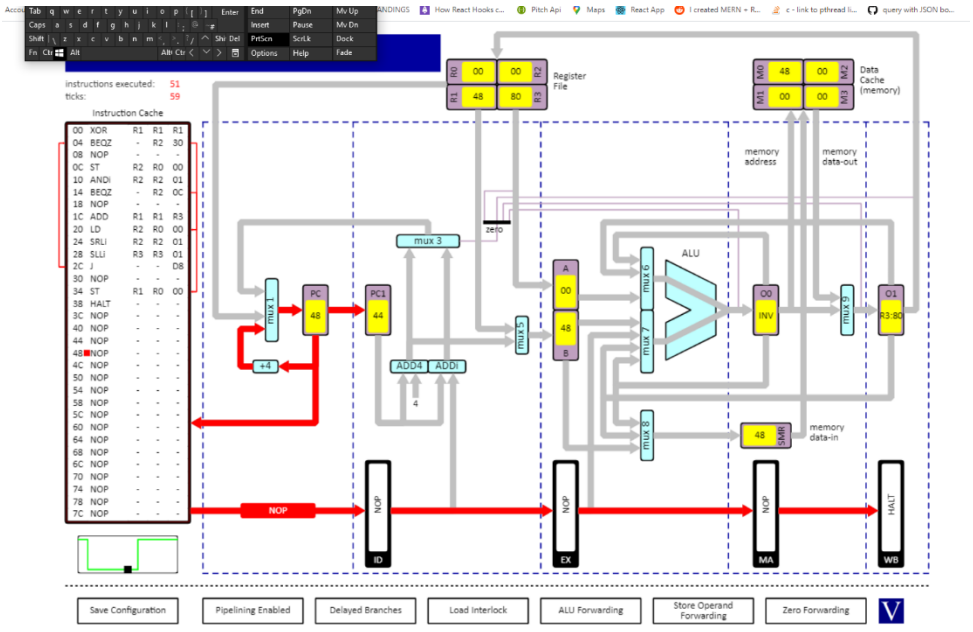
51 operations executed

59 clock cycles

It takes more clock cycles because of the NOP instructions. The NOP instructions take up an additional 13 in total. 38+13 = 51 so this is accounted for.

13 clock cycles in total. And as there are 4 pipeline stalls,

13-4 = 9. 50+9 = 59.



iii)

I optimized the program by switching the order of SRLi R2, R2, 01 and SLLi R3, R3, 01. The data dependency was R2 for SRLi R2, R2, 01 after doing the LD R2, R0, 00 instruction. I did this because the pipeline was stalling after loading the value of R0 into R2 and the new correct value of R2 wasn't immediately available to SRLi R2, R2, 01 so we can optimize the program by instead executing SLLi R3, R3, 01 first. We can see the correctness still stands.

Instructions executed : 38

Clock cycles : 46.

The reason for the less clock cycles is the 4 times in the program that the pipeline was stalled at the SRLi R2, R2, 01 beforehand. The instructions executed is the same as all we are doing is swapping the order of the two instructions.