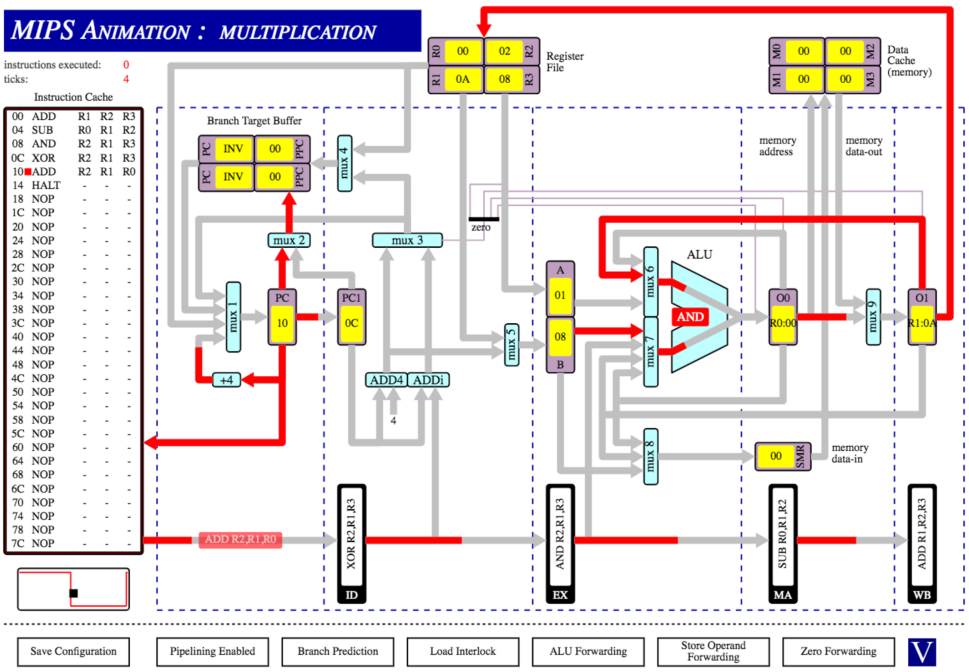
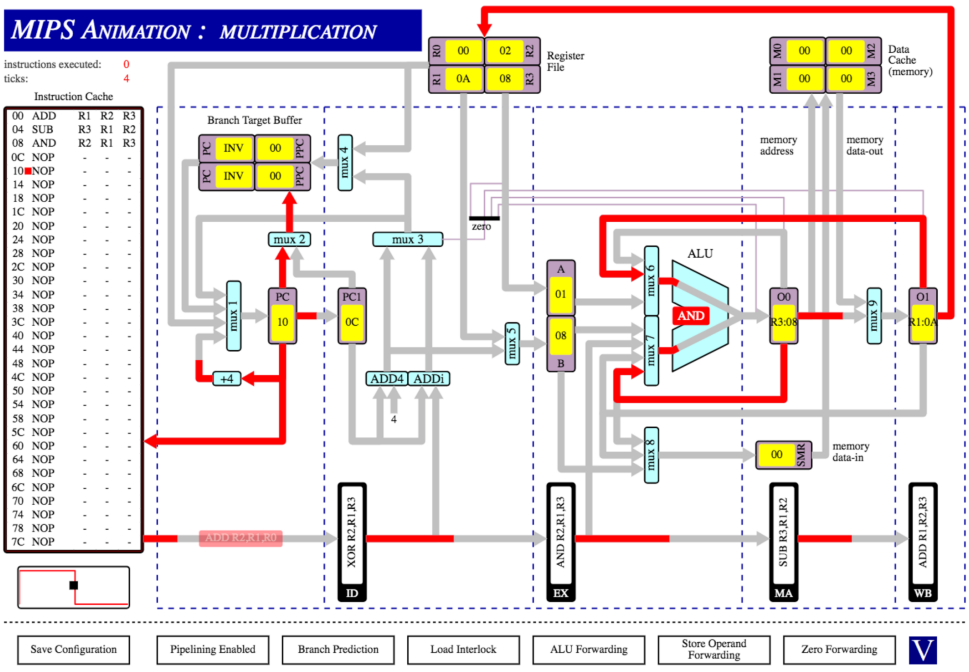
1. O1 to MUX6

ADD R1,R2,R3
SUB R0,R1,R2

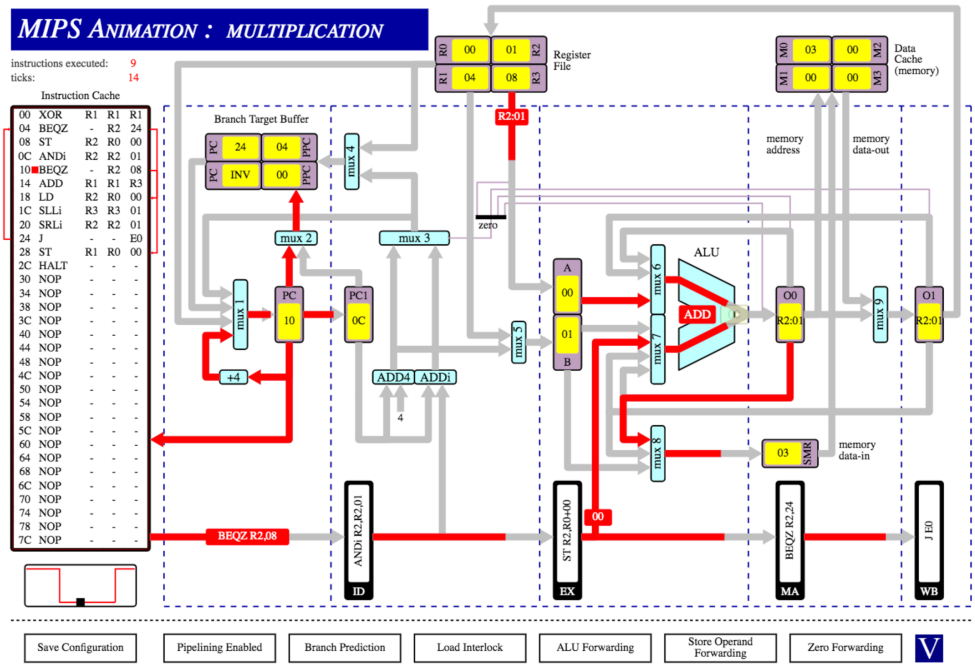


2. O0 to MUX7 and O1 to MUX6 (simultaneously)

ADD R1,R2,R3
SUB R3,R1,R2

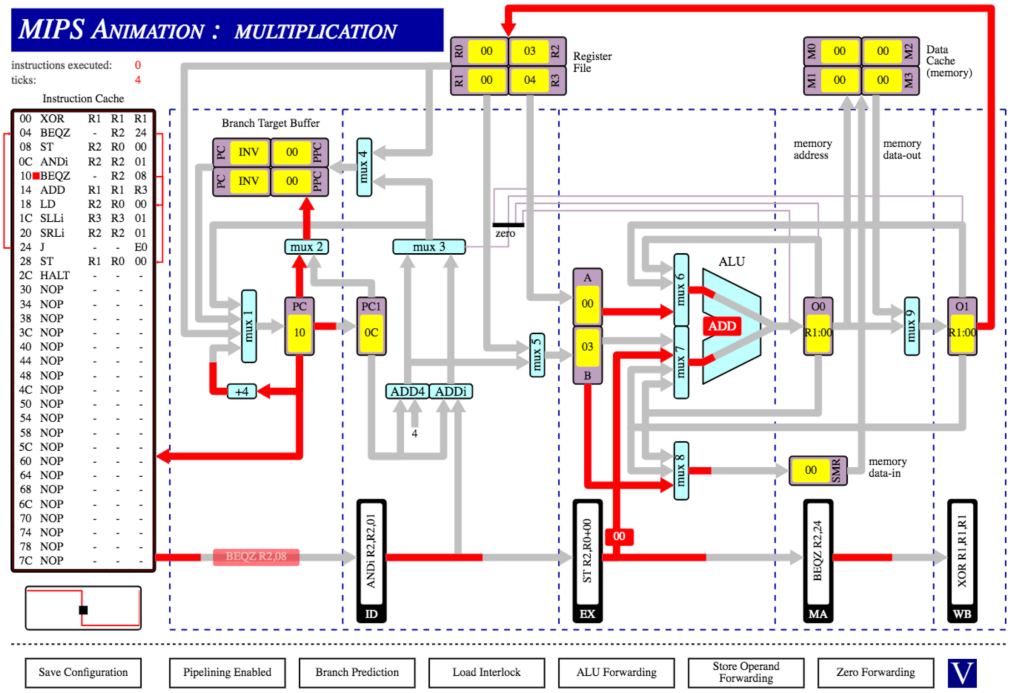


3. O0 to MUX8



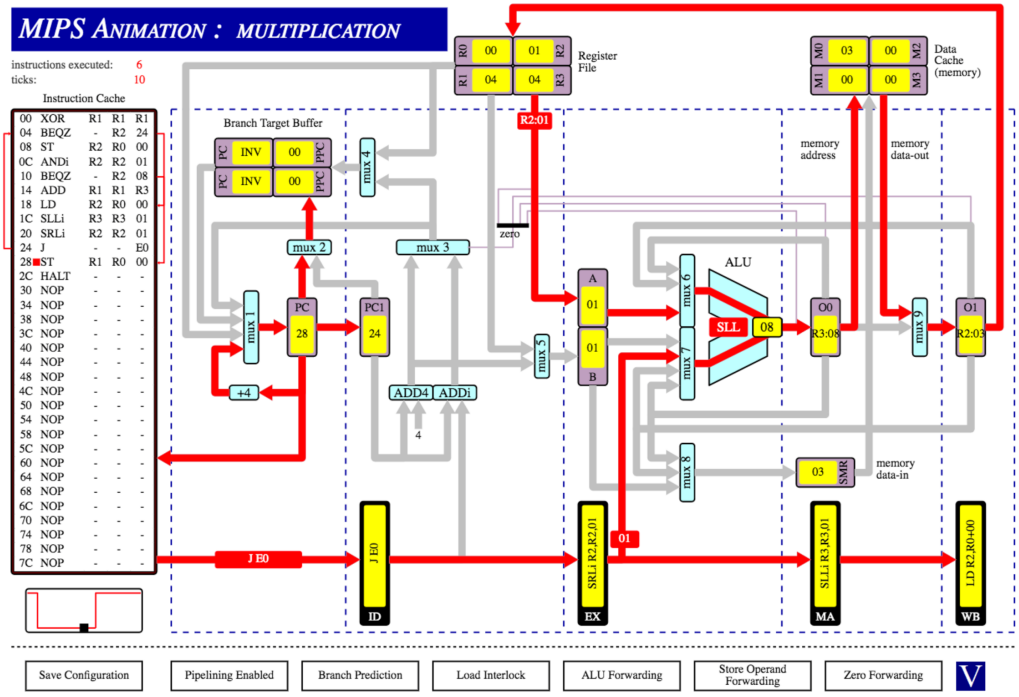
4. EX to MUX7

ST R2,R0



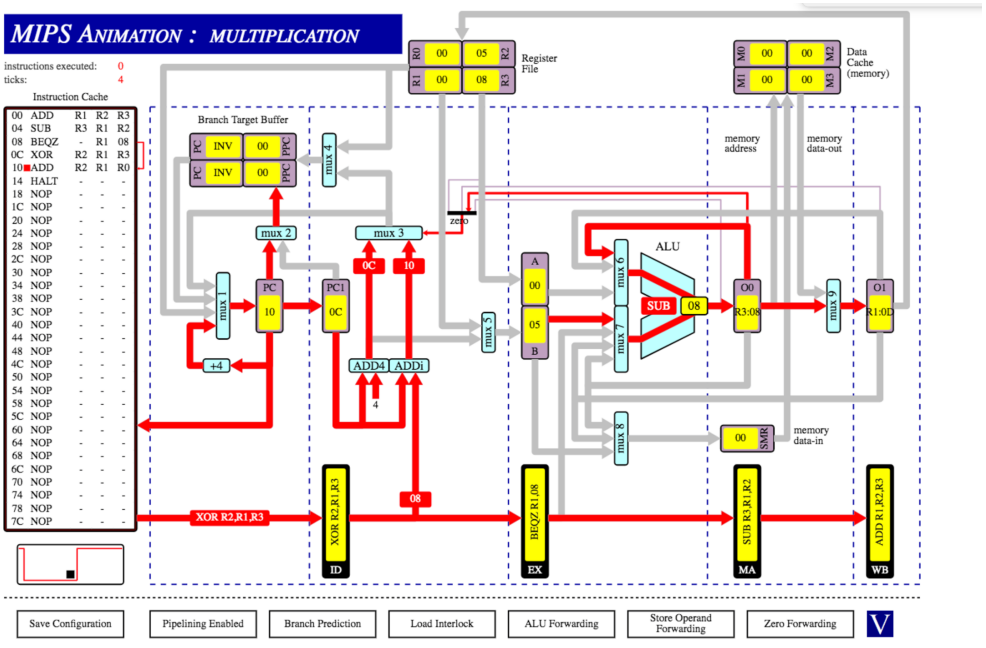
5. Data cache to MUX9 (memory data-out)

LD R2,R0,00



6. O0 to Zero detector

ADD R1,R2,R2
SUB R2,R1,R1
BEQZ - ,R1,08



Q2.

i)Result of R1 : 15
Clock Cycles : 10

The processor computes the result in r1 without stalling as ALU forwarding is enabled and no branch or return functions are present.

ii)Result of R1 : 15
Clock Cycles : 18

Without ALU forwarding, the processor cannot fetch the most recent value in a register for three ticks after it has been updated by the ALU. As data dependency interlocks are enabled we receive the same answer as in part one because the processor stalls until the newest value is available.

iii)Result of R1 : 06
Clock Cycles : 10

Without data dependency interlocks the processor does not worry about waiting for the newest values of registers coming from the ALU therefore we end up with the same number of ticks as the first run but the incorrect answer as the processor is fetching out of date values for R1 for some of the calculations.

3.

i) Instructions Executed: 22
Ticks: 28

Instructions and ticks are not equal because it takes 4 ticks for the pipeline to execute the first instruction. Adding to this there are also 2 stalls for each jump meaning that there is a total of 28.

ii) Ticks: 29

We introduced a one tick delay as the processor does not correctly predict whether or not the branch is taken as it does in the first case, thus it must stall for a tick.