***Question 1***

***PT1***

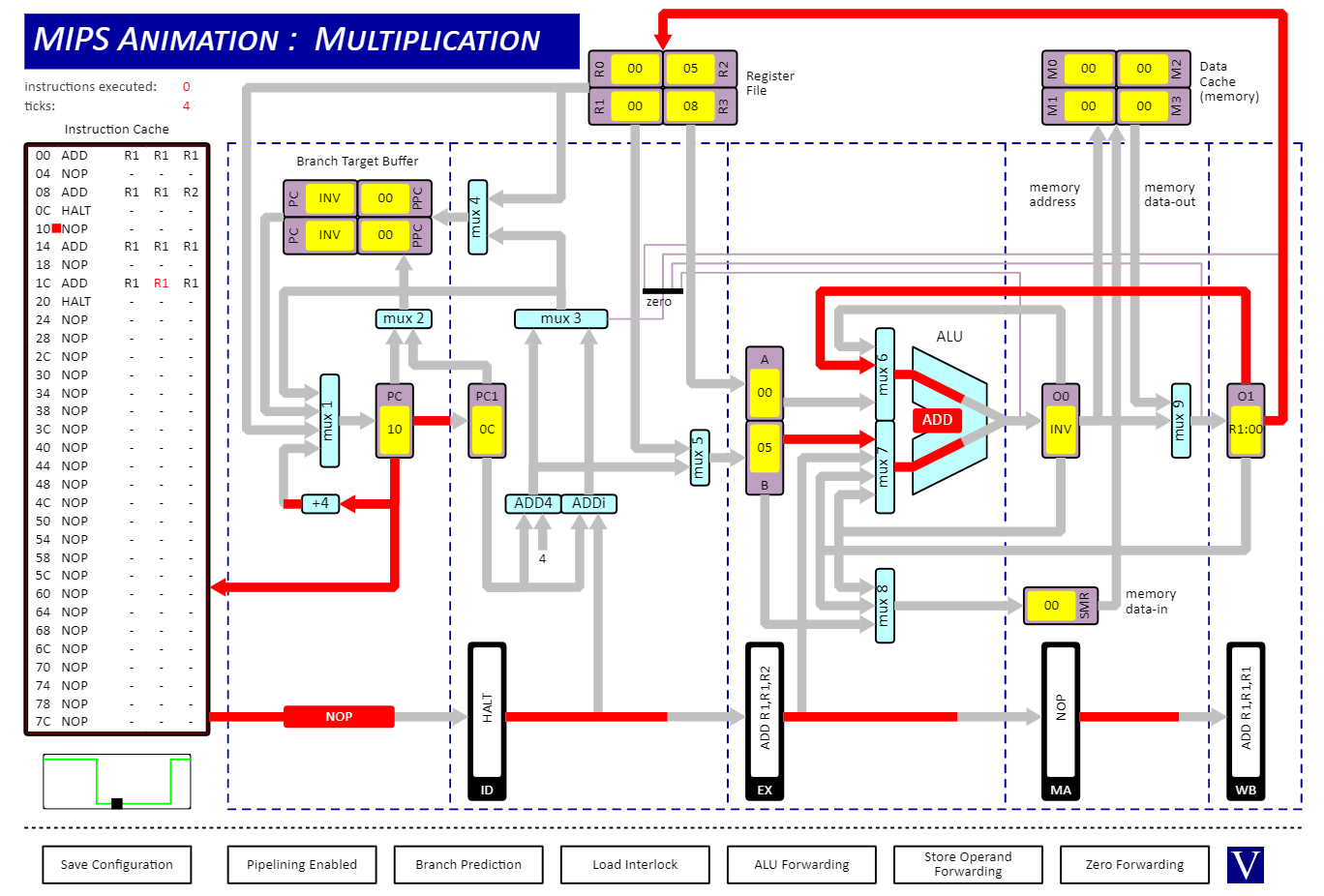
**Instructions**

ADD R1, R1, R1

NOP -, -, -

ADD R1, R1, R2

**Screenshot**



***PT2***

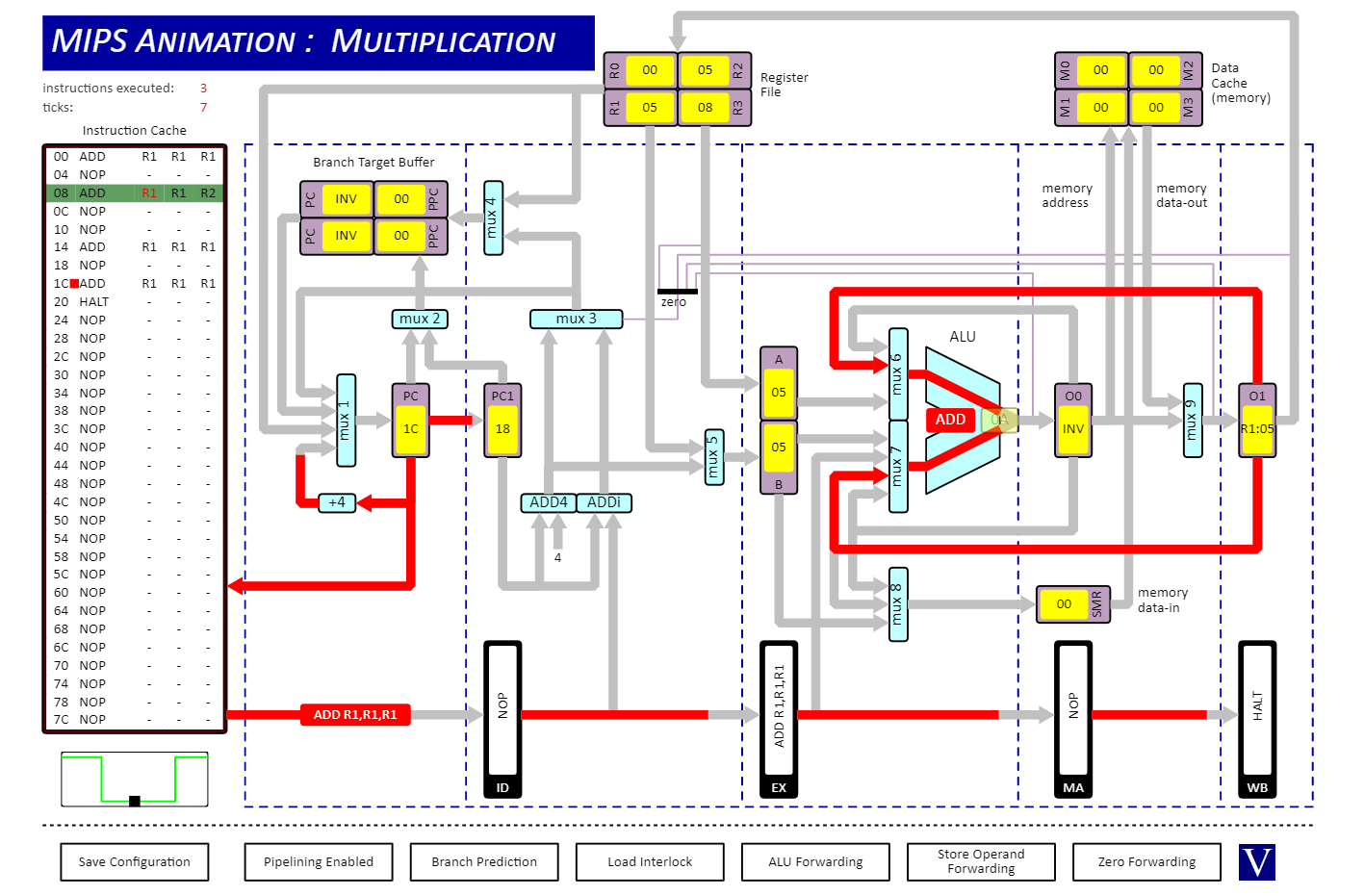
**Instructions**

ADD R1, R1, R1

NOP -, -, -

ADD R1, R1, R1

**Screenshot**

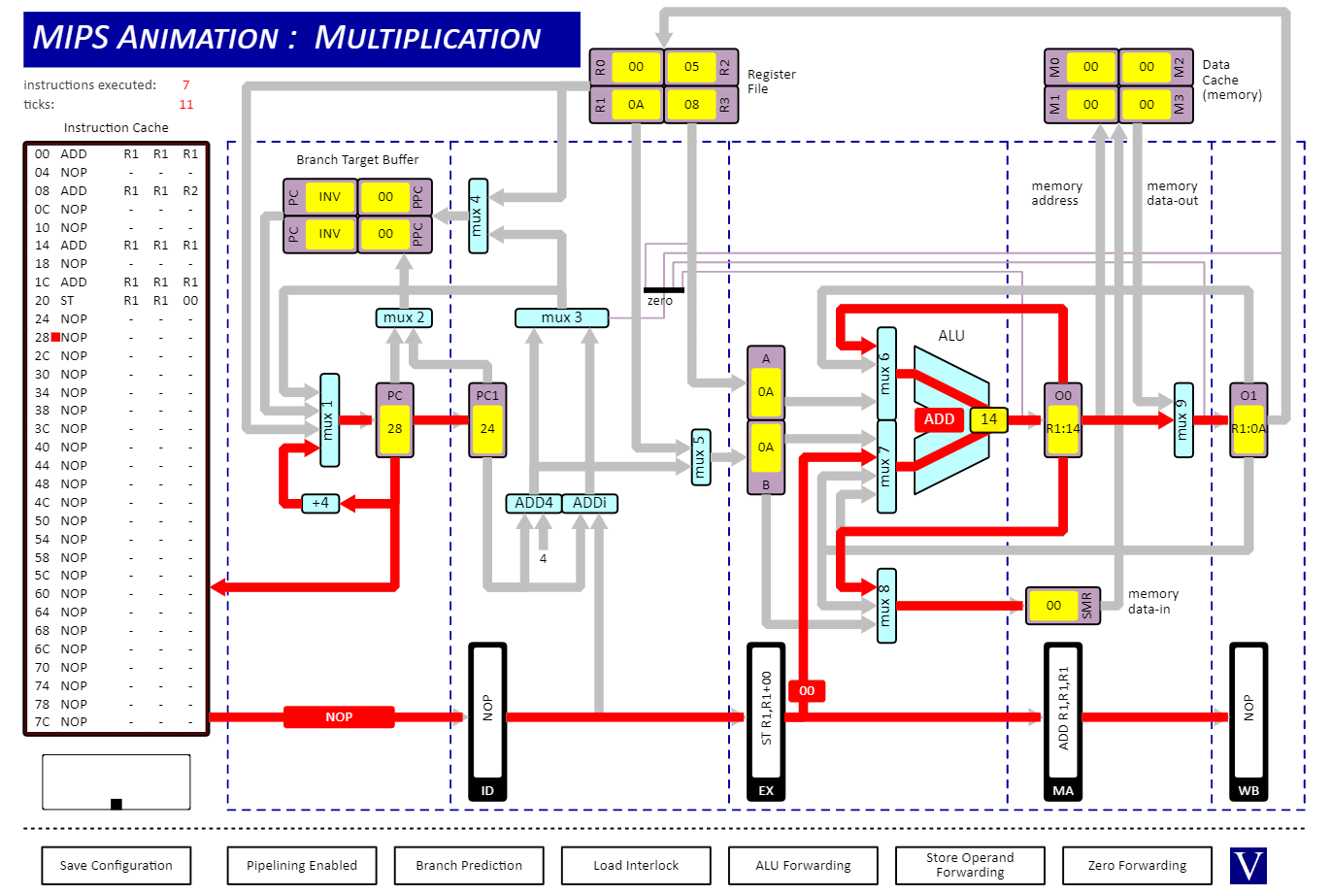


***PT3***

**Instruction**

ST R1, R1, 00

**Screenshot**

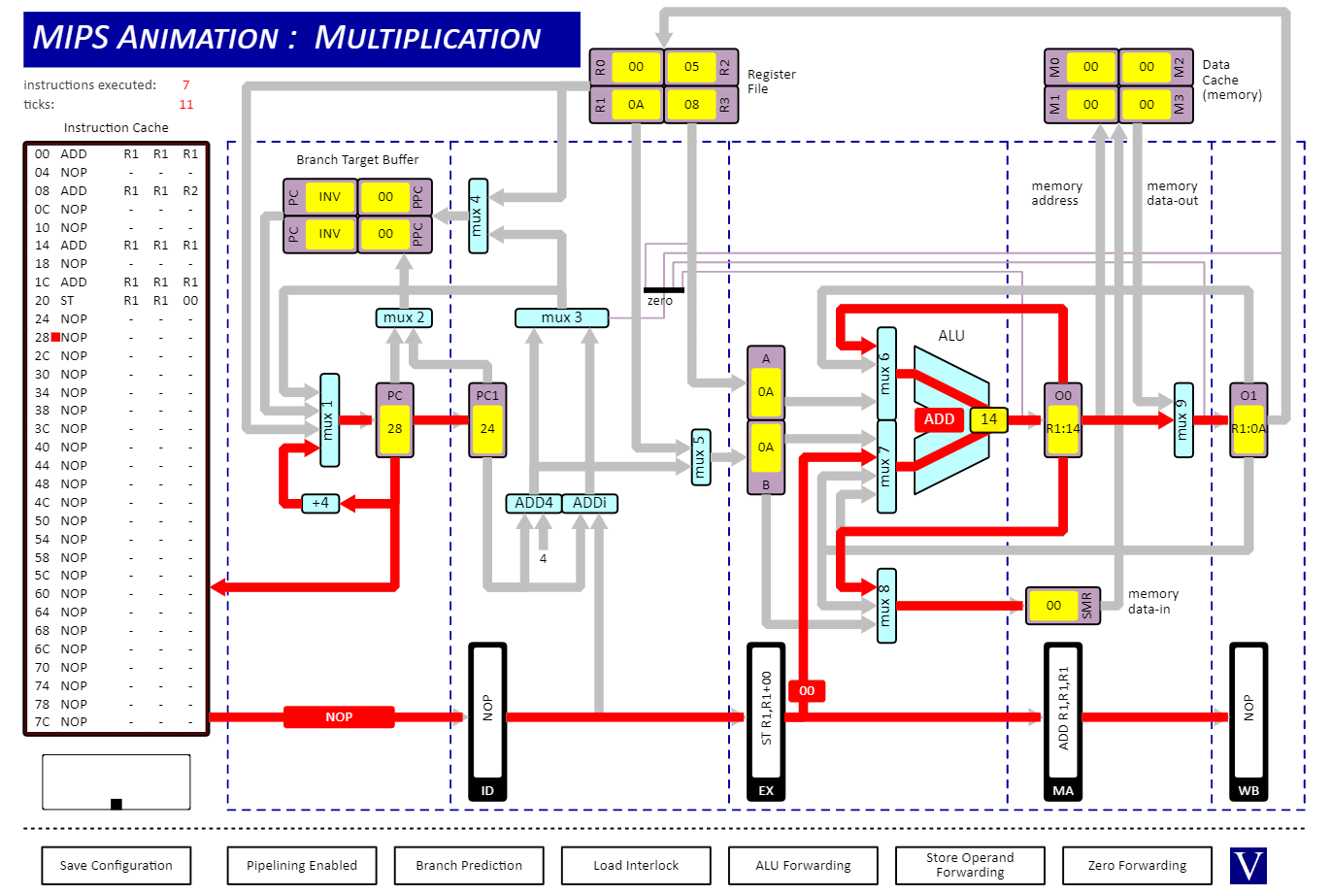


***PT4***

**Instruction**

ST R1, R1, 00

**Screenshot**

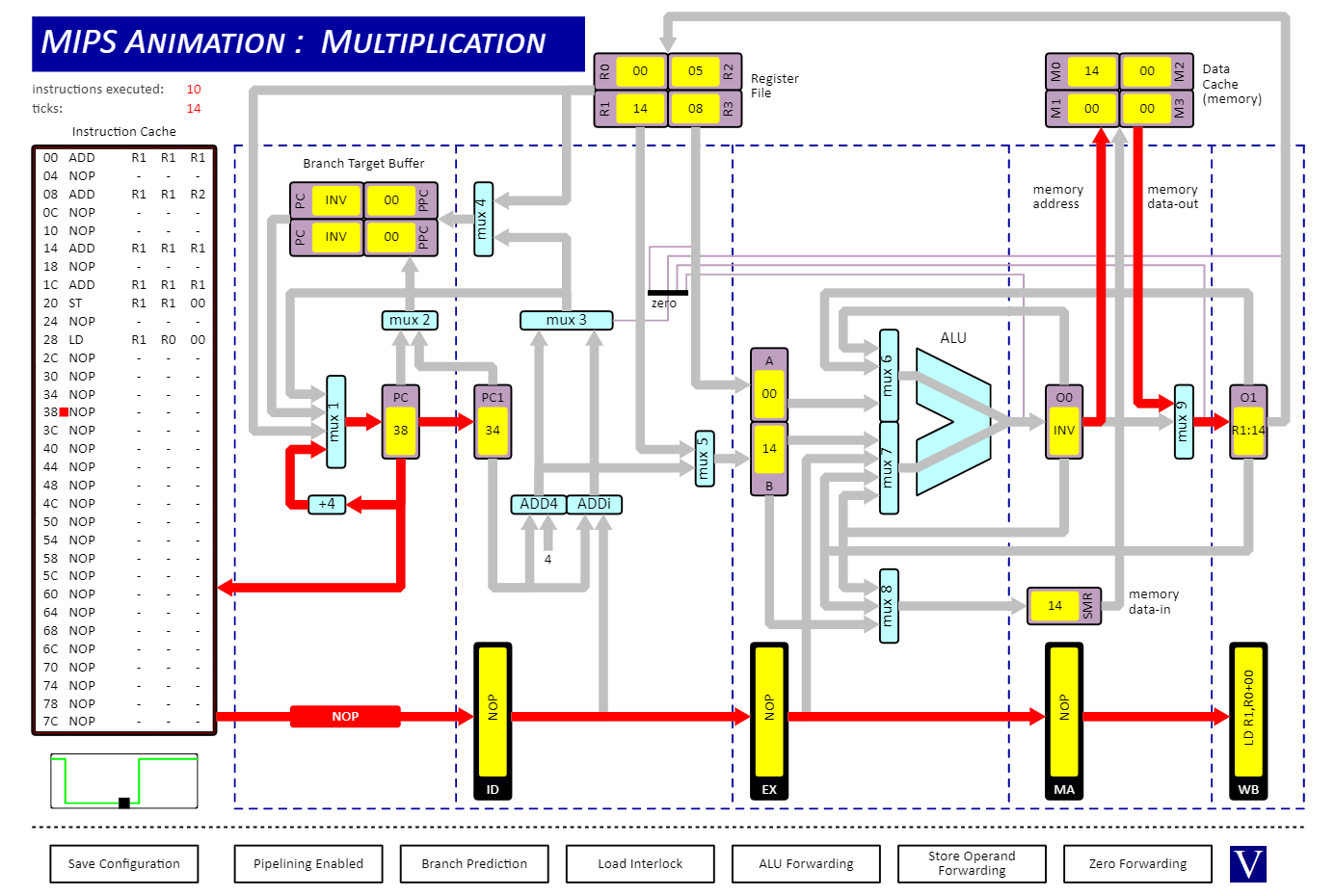


***PT5***

**Instruction**

LD R1, R0, 00

**Screenshot**



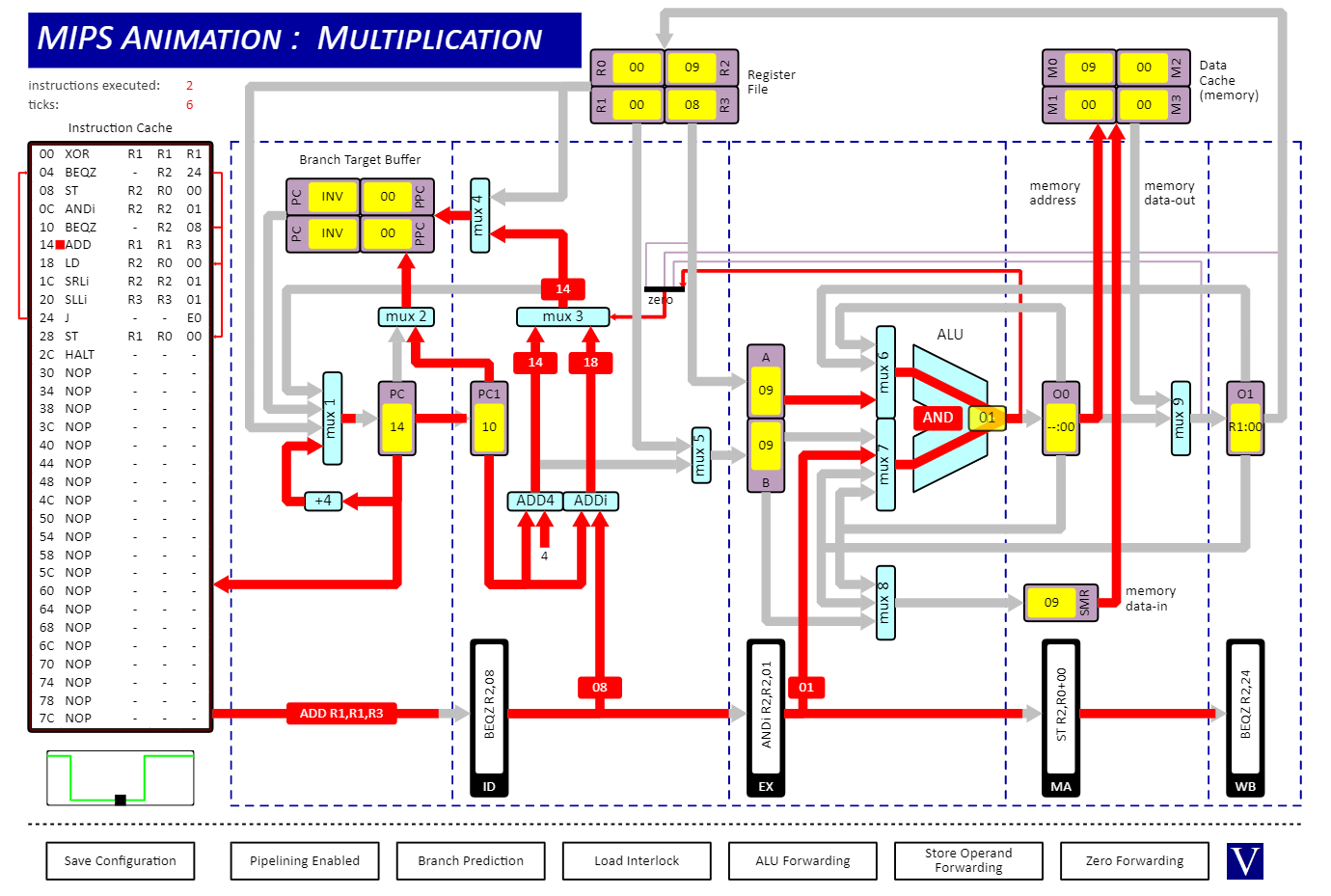
***PT6***

**Instructions**

ANDi R2, R2, 01

BEQZ -, R2, 08

**Screenshot**

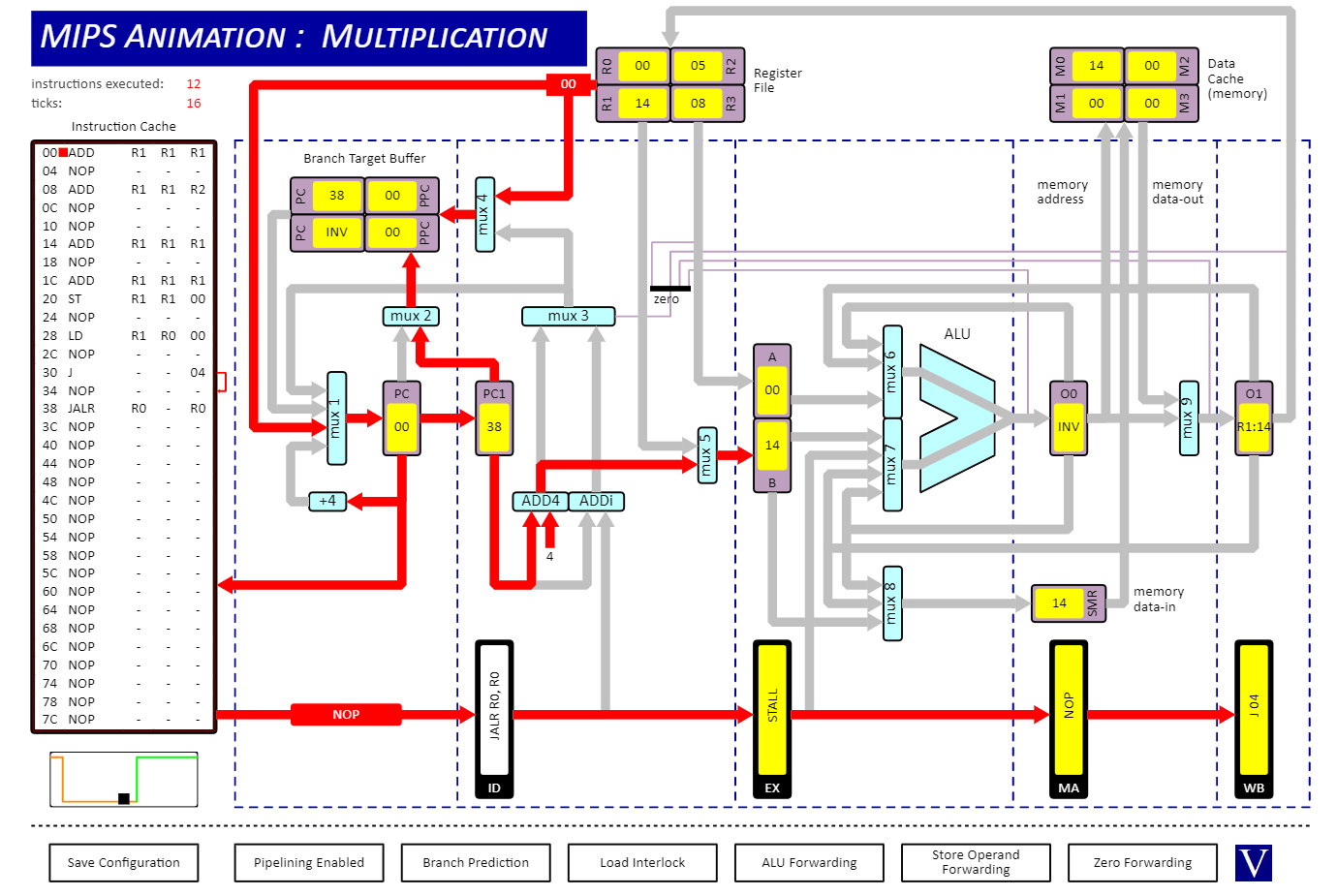


***PT7***

**Instruction**

JALR RO, -, R0

**Screenshot**

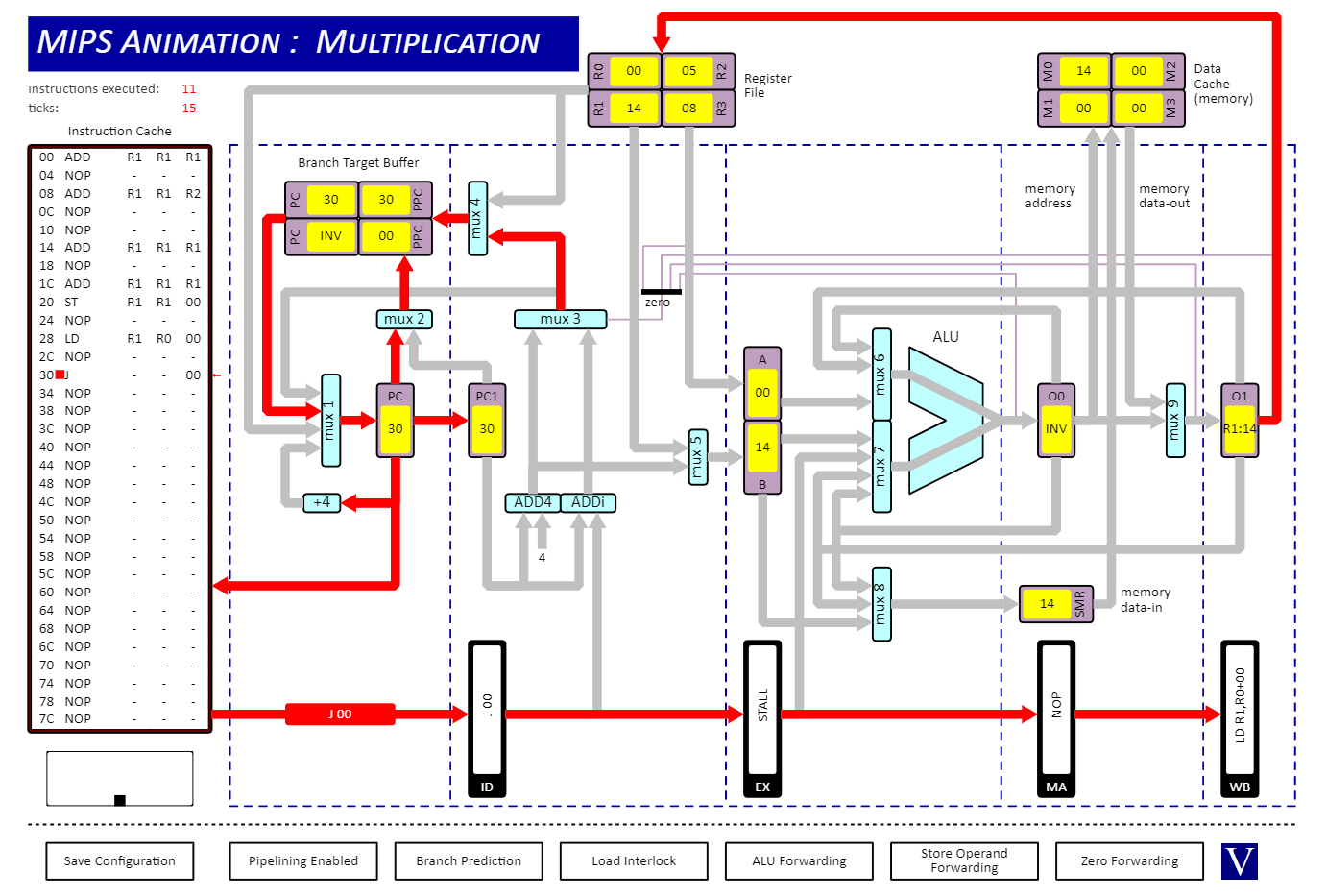


***PT8***

**Instruction**

J -, -, 00

**Screenshot**



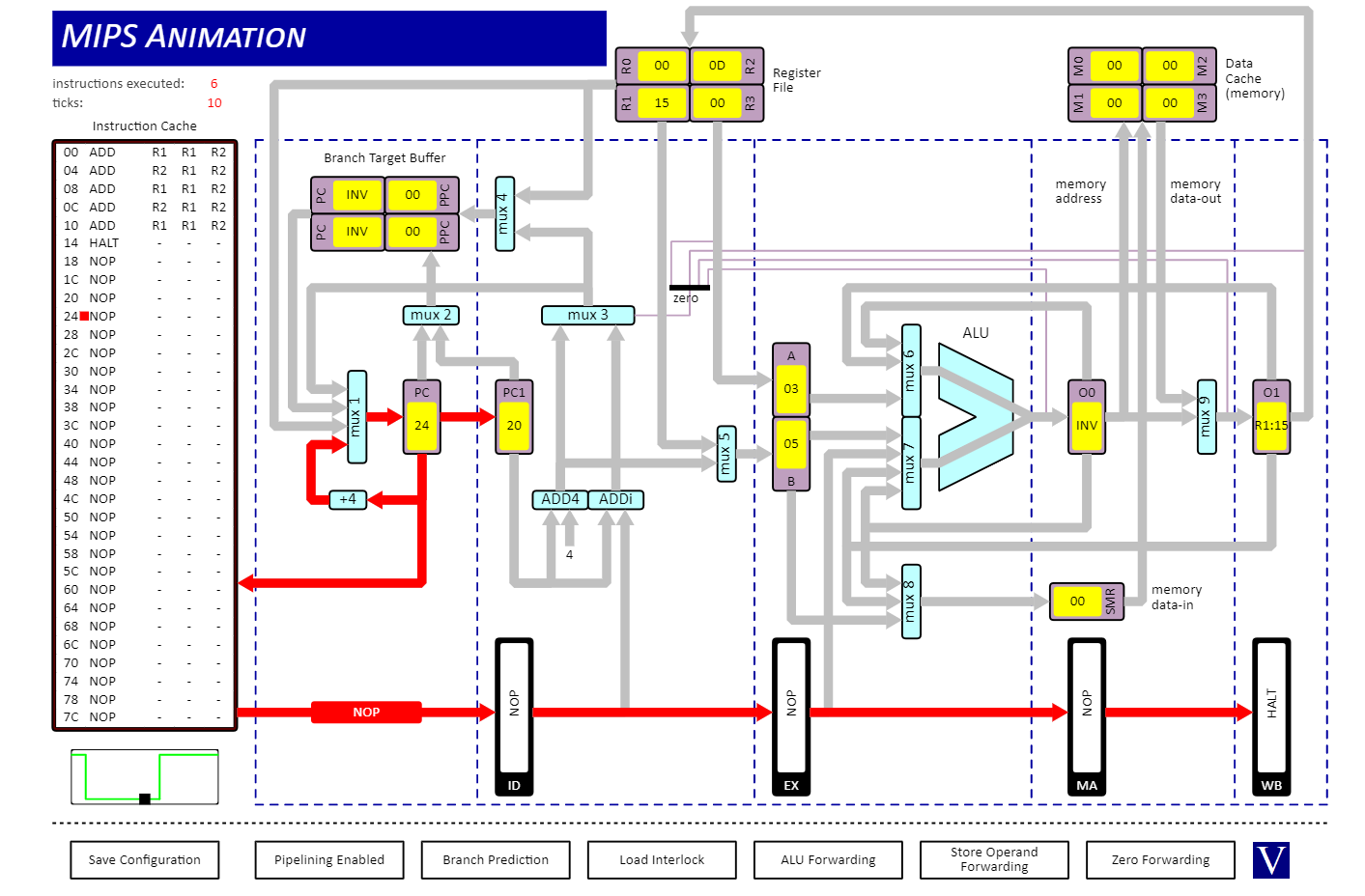
***Question 2***

**Part I**

Number of clock cycles needed to execute : 10

Resulting value of R1 : Hex 0x15 == 21

**Screenshot**

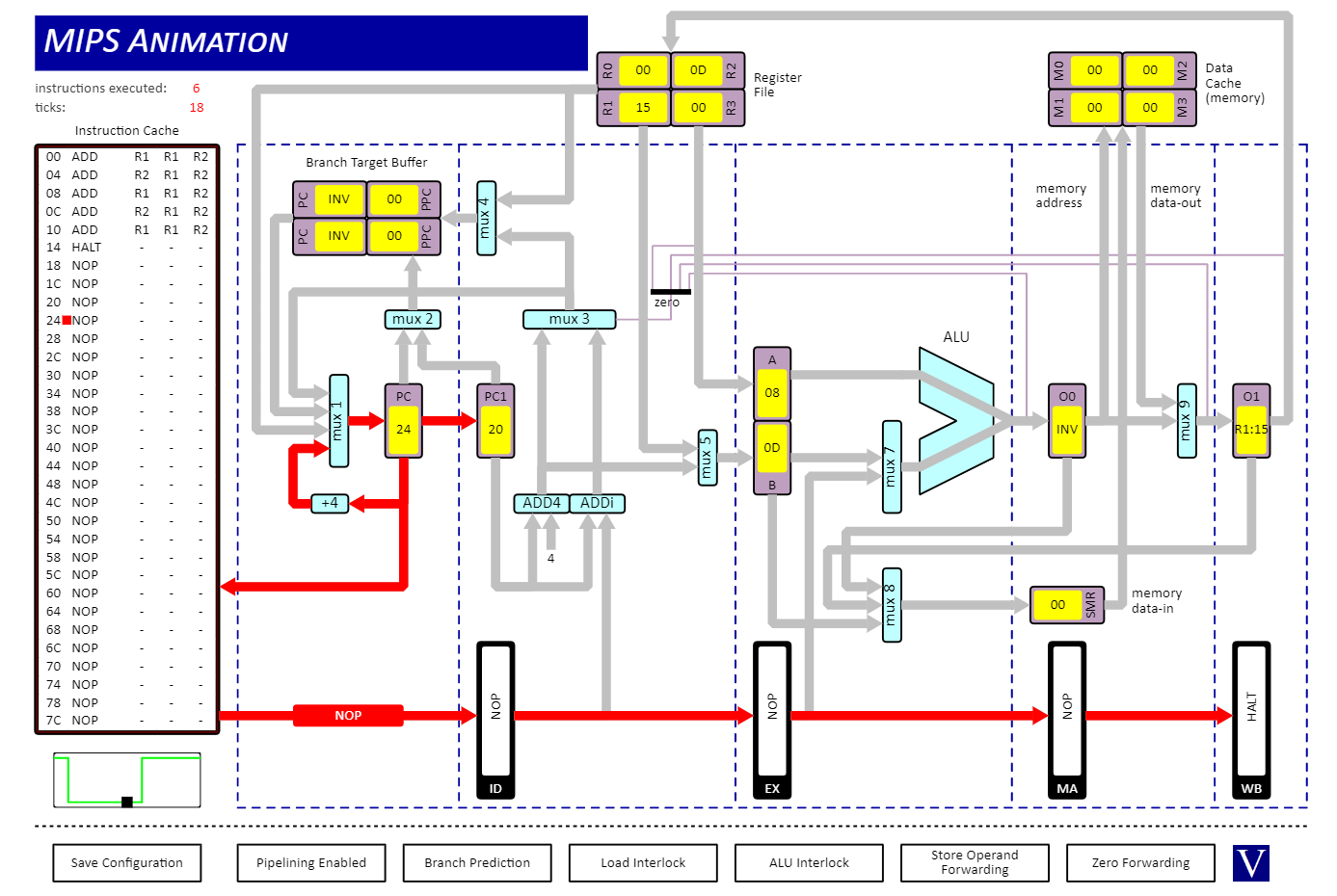


**Part II**

Number of clock cycles needed to execute : 18

Resulting value of R1 : Hex 0x15 == 21

**Screenshot**

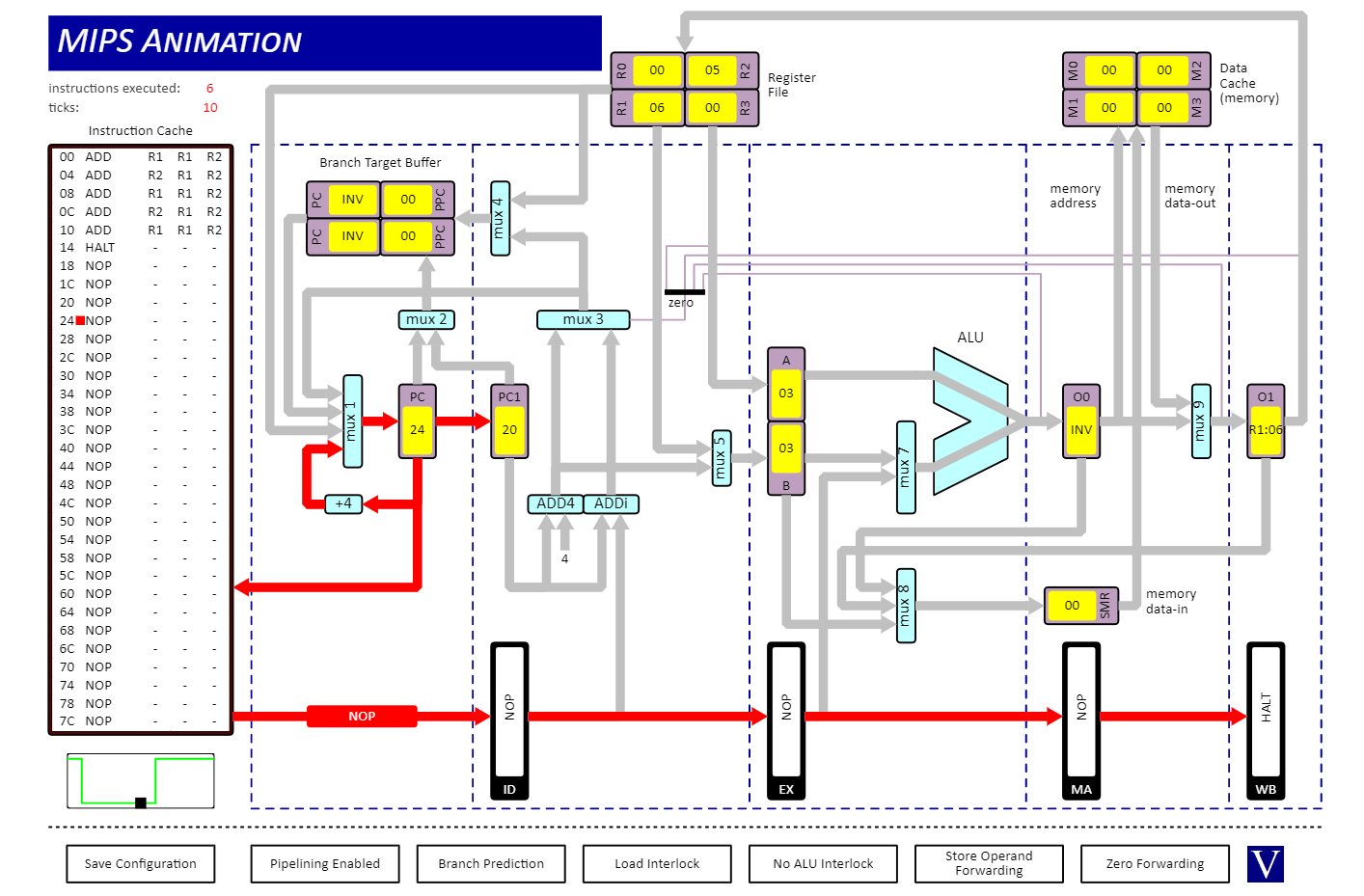


**Part III**

Number of clock cycles needed to execute : 10

Resulting value of R1 : Hex 0x06 == 6

**Screenshot**



**Explain in detail why the results and number of clock cycles are different?**

The number of clock cycles are different because in part 1 we have alu forwarding enabled. This means that we can immediately get access to the result of the alu operation after its is completed. This is facilitated by O0 and O1. This means that we can get the new values in the following instruction executions even before it has been written back thus preventing data hazard pipeline stalls.

In part 2 with alu interlock we have no such facility, and thus whenever an instruction is executed, we must wait for the alu operation to be written back (in that case where we have a data hazard) before we can continue with the next operation. This results in the same number of instructions taking more cycles to execute due to these pipeline stalls while waiting for the write back to be completed.

Regardless of execution time, we obtain the correct answer in both parts I and II. This is because we ensure that we use the most up to date version of each register and give adequate time for updates to propagate.

In part III, it takes the same amount of cycles as alu forwarding, but we do not care whether updates have propagated, thus we get an incorrect answer as we proceed ahead with alu operations even though the results of the last alu operation have not been given time to update the registers.

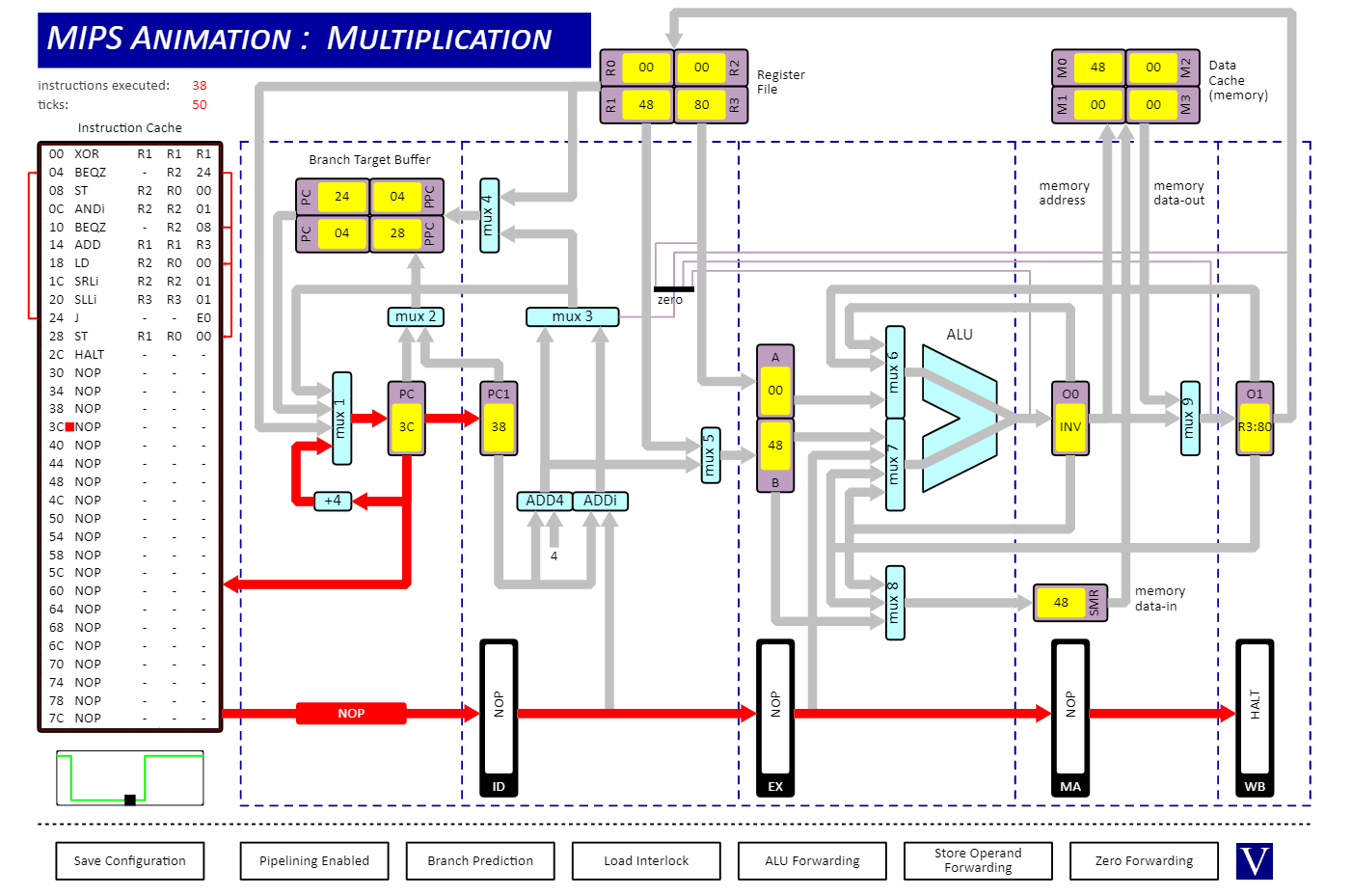
***Question 3***

**Part I**

Number of instructions : 38

Number of clock cycles : 50

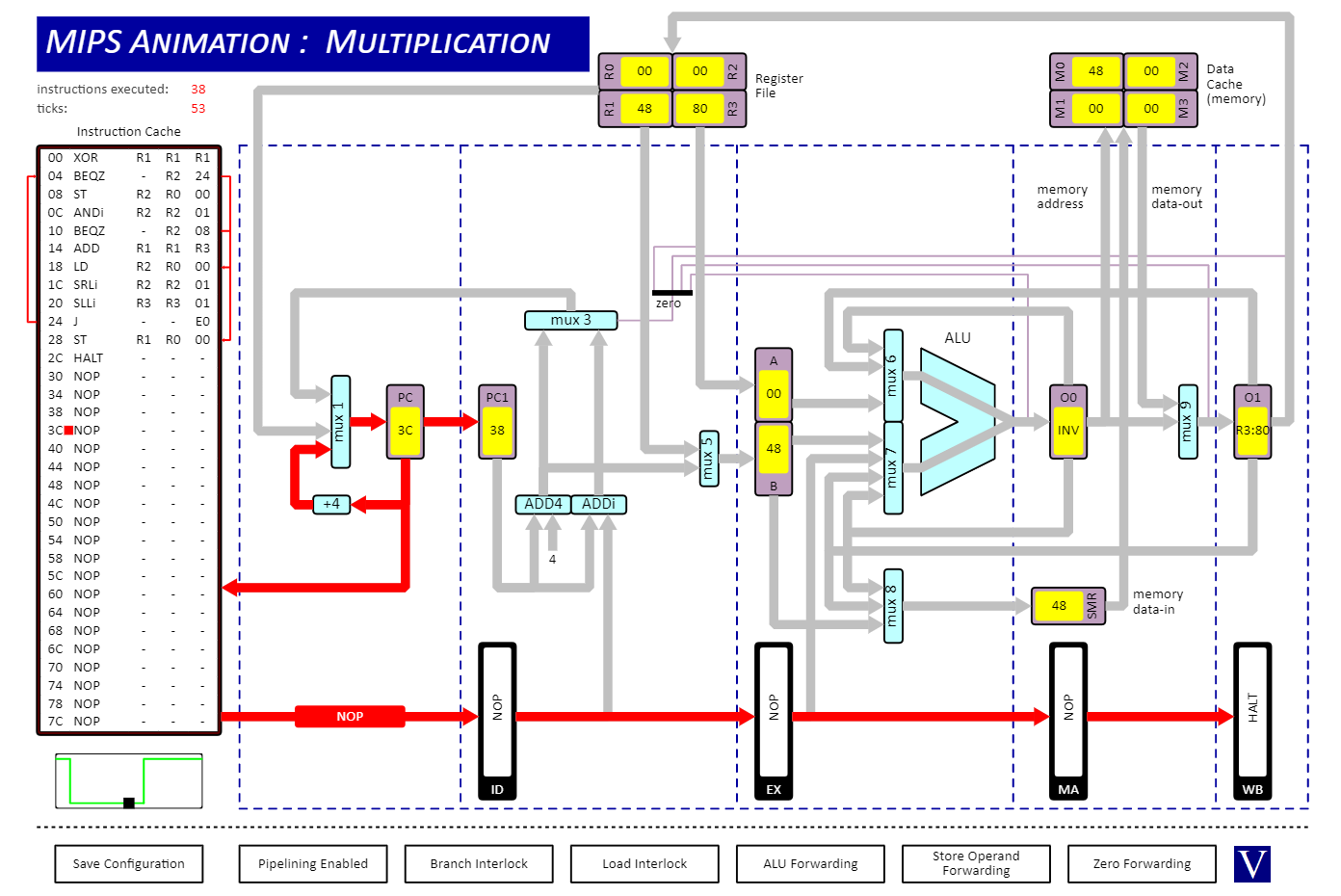
Why are these numbers not equal? Pipeline stalls



**Part II**

Number of instructions : 38

Number of clock cycles : 53



**Part III**

