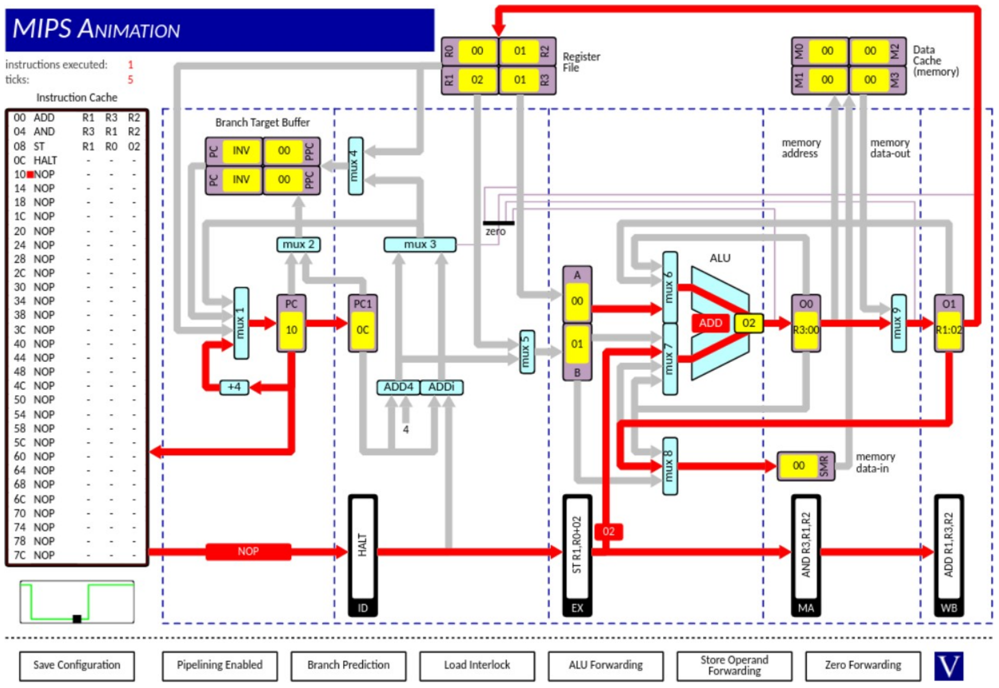
Question 1:

1. O1 to Mux8

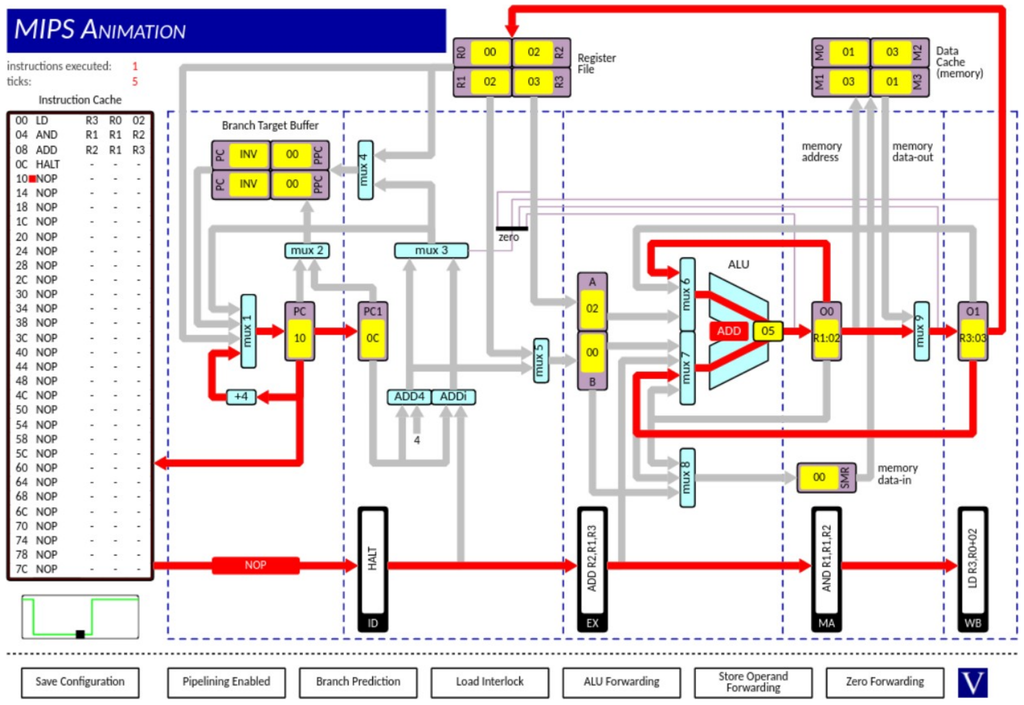


instructions executed: 1
ticks: 5

Instruction Cache

```
00  ADD   R1  R3  R2
04  AND   R3  R1  R2
08  ST    R1  R0  02
0C  HALT  -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
18  NOP   -   -   -
1C  NOP   -   -   -
20  NOP   -   -   -
24  NOP   -   -   -
28  NOP   -   -   -
2C  NOP   -   -   -
30  NOP   -   -   -
34  NOP   -   -   -
38  NOP   -   -   -
3C  NOP   -   -   -
40  NOP   -   -   -
44  NOP   -   -   -
48  NOP   -   -   -
4C  NOP   -   -   -
50  NOP   -   -   -
54  NOP   -   -   -
58  NOP   -   -   -
5C  NOP   -   -   -
60  NOP   -   -   -
64  NOP   -   -   -
68  NOP   -   -   -
6C  NOP   -   -   -
70  NOP   -   -   -
74  NOP   -   -   -
78  NOP   -   -   -
7C  NOP   -   -   -
```

2. O0 to Mux6 and O1 to Mux7 (simultaneously)

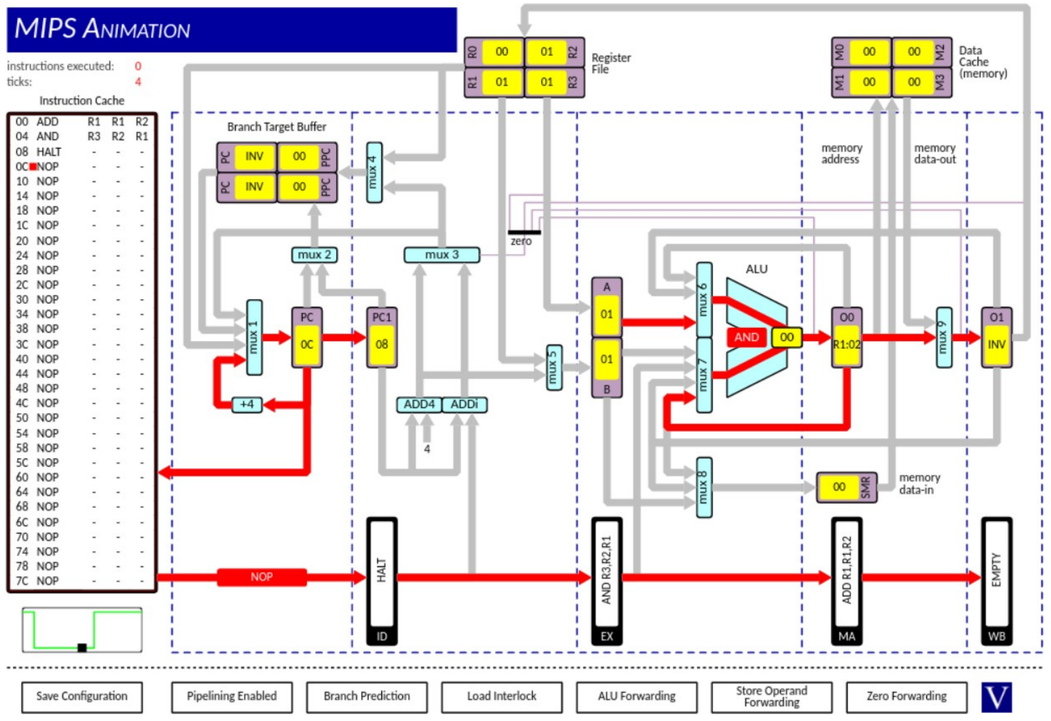


instructions executed: 1
ticks: 5

Instruction Cache

```
00  LD    R3  R0  02
04  AND   R1  R1  R2
08  ADD   R2  R1  R3
0C  HALT  -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
18  NOP   -   -   -
1C  NOP   -   -   -
20  NOP   -   -   -
24  NOP   -   -   -
28  NOP   -   -   -
2C  NOP   -   -   -
30  NOP   -   -   -
34  NOP   -   -   -
38  NOP   -   -   -
3C  NOP   -   -   -
40  NOP   -   -   -
44  NOP   -   -   -
48  NOP   -   -   -
4C  NOP   -   -   -
50  NOP   -   -   -
54  NOP   -   -   -
58  NOP   -   -   -
5C  NOP   -   -   -
60  NOP   -   -   -
64  NOP   -   -   -
68  NOP   -   -   -
6C  NOP   -   -   -
70  NOP   -   -   -
74  NOP   -   -   -
78  NOP   -   -   -
7C  NOP   -   -   -
```

3. O0 to Mux7



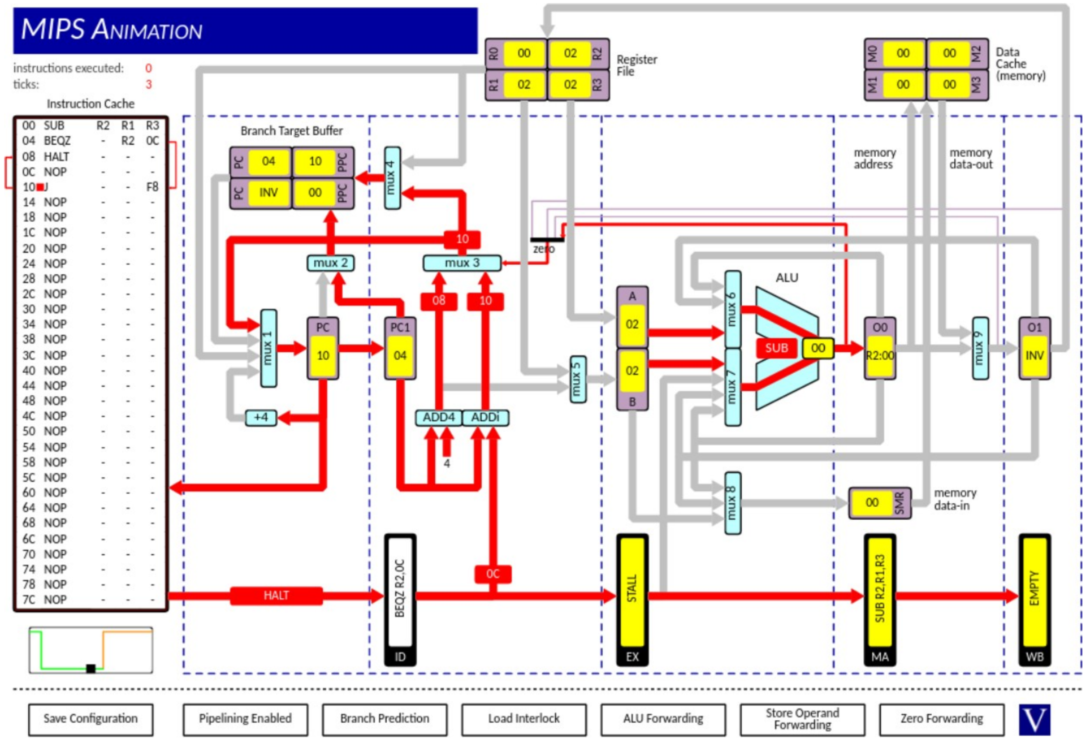
MIPS Animation

instructions executed: 0
ticks: 4

Instruction Cache

```
00  ADD    R1  R1  R2
04  AND    R3  R2  R1
08  HALT   -   -   -
0C  NOP    -   -   -
10  NOP    -   -   -
14  NOP    -   -   -
18  NOP    -   -   -
1C  NOP    -   -   -
20  NOP    -   -   -
24  NOP    -   -   -
28  NOP    -   -   -
2C  NOP    -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

4. ID to Mux3 to Mux1/Mux4



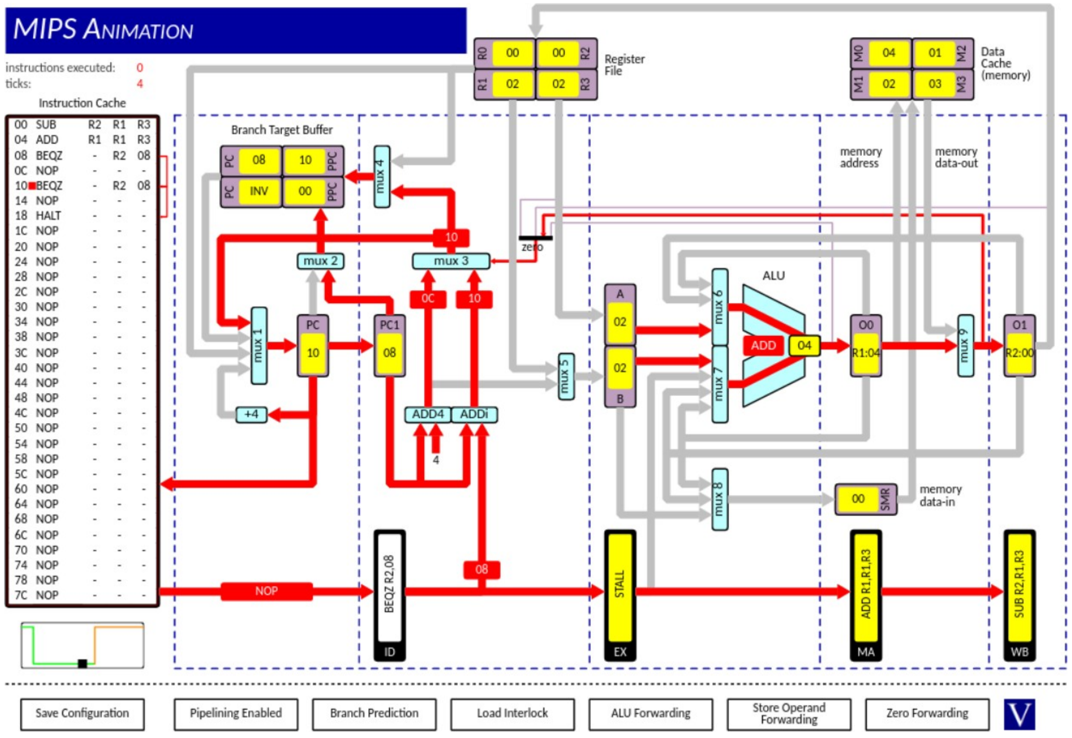
MIPS Animation

instructions executed: 0
ticks: 3

Instruction Cache

```
00  SUB    R2  R1  R3
04  BEQZ   -   R2  0C
08  HALT   -   -   -
0C  NOP    -   -   -
10  J      -   -   F8
14  NOP    -   -   -
18  NOP    -   -   -
1C  NOP    -   -   -
20  NOP    -   -   -
24  NOP    -   -   -
28  NOP    -   -   -
2C  NOP    -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

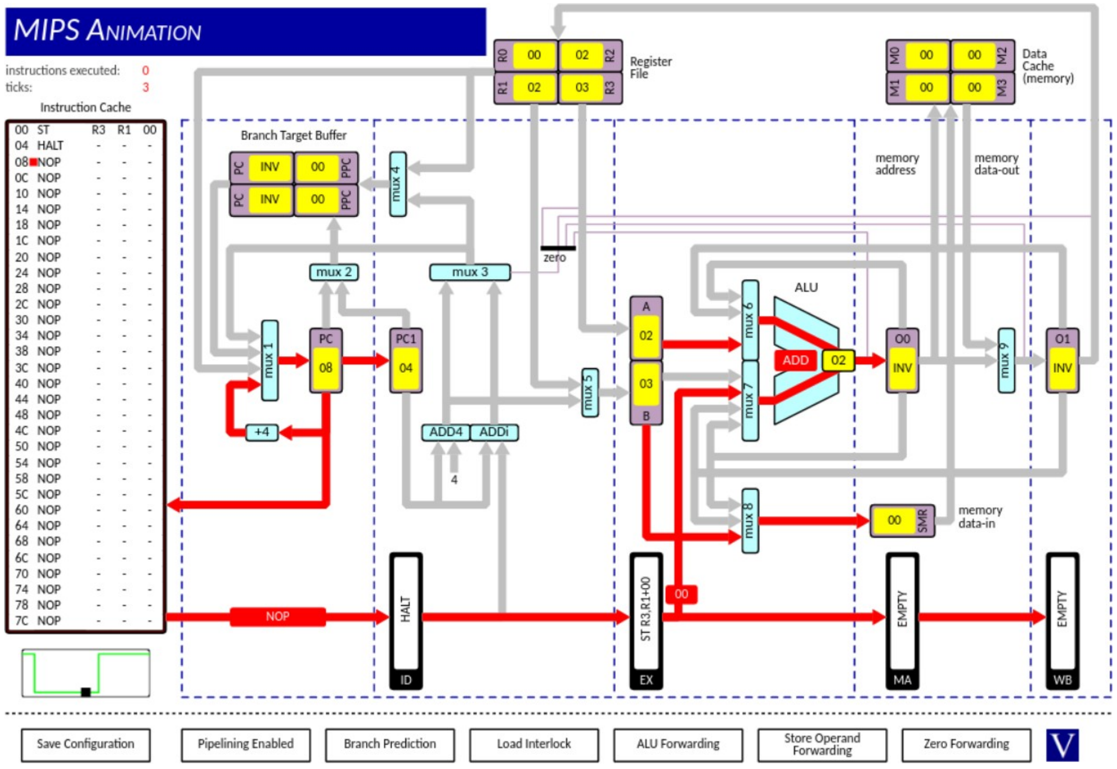
5. Mux9 out to Zero detector



```
instructions executed:   0
ticks:                   4

Instruction Cache
00  SUB    R2  R1  R3
04  ADD    R1  R1  R3
08  BEQZ   -   R2  08
0C  NOP    -   -   -
10  BEQZ   -   R2  08
14  NOP    -   -   -
18  HALT   -   -   -
1C  NOP    -   -   -
20  NOP    -   -   -
24  NOP    -   -   -
28  NOP    -   -   -
2C  NOP    -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

6. EX to Mux7 and B to Mux8 simultaneously



```
instructions executed:   0
ticks:                   3

Instruction Cache
00  ST     R3  R1  00
04  HALT   -   -   -
08  NOP    -   -   -
0C  NOP    -   -   -
10  NOP    -   -   -
14  NOP    -   -   -
18  NOP    -   -   -
1C  NOP    -   -   -
20  NOP    -   -   -
24  NOP    -   -   -
28  NOP    -   -   -
2C  NOP    -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

Question 2

(i) ALU Forwarding Enabled

- Final value of R1: 0x02
- Final value of R2: 0x00
- Total clock cycels: 10

(ii) ALU Forwarding Disabled with Interlocks Enabled

- Final value of R1: 0x02
- Final value of R2: 0x00
- Total clock cycles: 18

(iii) ALU Forwarding and Interlocking Disabled

- Final value of R1: 02
- Final value of R2: 02
- Total clock cycles: 10

Explanation

The number of clock cycles for case (i) and (iii) are same because in both cases there will be no pipelining stall. ALU forwarding allows data to be forwarded from one stage to another so that no stall cycles are needed to ensure correct result while in case (iii), stall will never happen because the pipeline is not set up for that.

For case (ii), it takes 18 clock cycles for the program to complete since there are data dependencies between consecutive instructions. In order to ensure correct result of the program, the pipeline had to be stalled for two clock cycles between consecutive instruction execution. With '5' instructions, the pipeline is stalled for 4x2=8 clock cycles, resulting in a total of 18 clock cycles.

Regarding the values, case (i) and (ii) give correct results. The result for case (iii) is not correct for both R1 and R2. The fact that R1 returns the same result is irrelevant as the data dependcies between instructions were not accounted for in the pipeline and the ALU read wrong data from the register file before the correct result was written there.

Question 3

1. Program Execution

- Total instructions executed: 38
- Total clock cycles: 50

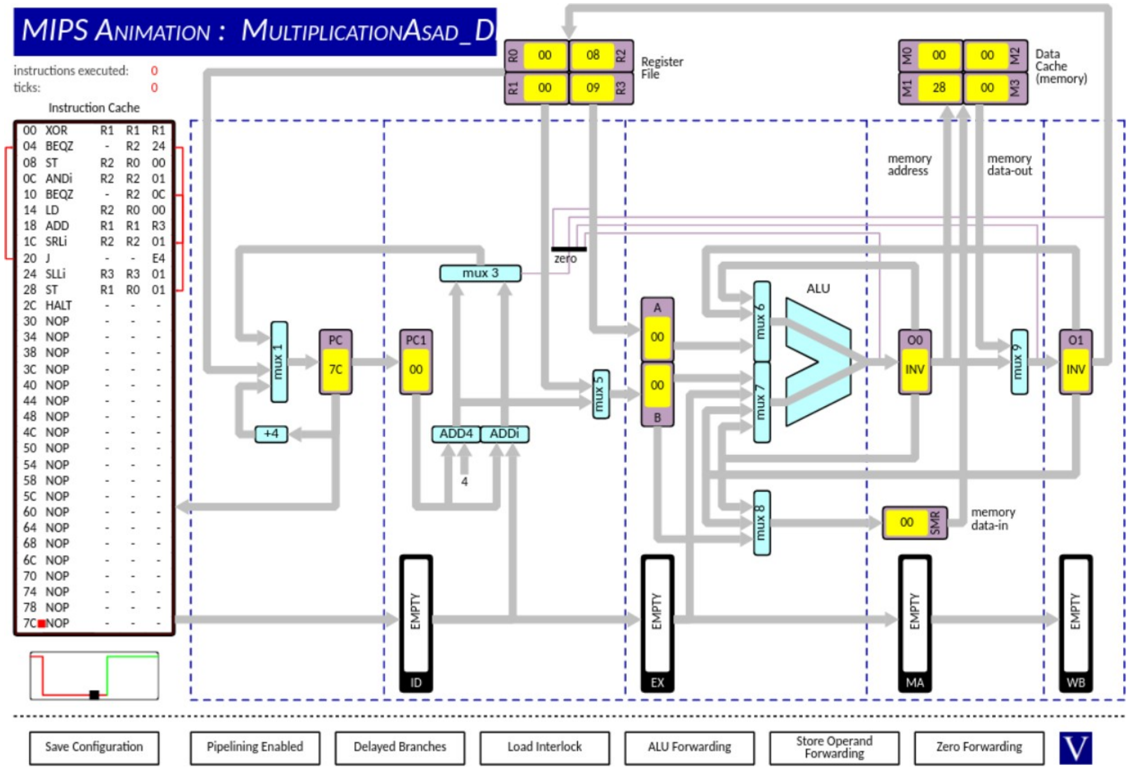
The difference in the two numbers is due to:

- The initial delay required to fill the pipeline (4 cycles).
- After every load instruction (LD R2, R0, 0) the pipeline is stalled for ‘1’ clock cycle as there is a data dependency between the load and the instruction following load (SRLi). The loop is executed 4 times, resulting in a total of 4 stall cycles.
- In the first loop, when the instruction J E0 is decoded, there are no valid entries in the branch target buffer (BTB). The pipeline is stalled for ‘1’ clock cycle to ensure the instruction after the jump is not decoded. For the remaining loops, there will no longer be a stall cycle because the entry in BTB will be updated.
- In the second loop, the instruction BEQZ R2, 0x08 is taken because R2 is zero. Since this BEQZ is taken for the first time, this results in a stall cycle. For the next loop, there is no stall until the last loop when this branch is not taken while the BTB had a valid entry, resulting in another stall. This results in a total of 2 stall cycles
- In the final loop, when the value of R2 is reduced to ‘0’, the first branch instruction (BEQZ R2, 24) is taken which results in a stall since it is the first time it is taken.
- So the total extra cycles needed as compared to instruction count: 4 + 4 + 1 + 2 + 1 = 12. With a total of 38 instructions, it takes ‘50’ clock cycles for the program to execute

2. Delayed Branches

- Total instructions executed: 39
- Total clock cycles: 45

The key to this question is to identify which useful instructions can be placed in the delay slot of every branch and jump. Only one instruction needs to be placed since the branches and jumps are decoded and target instruction known in the 2nd clock cycle. As compared to the original version, there is an increase in the number of instructions executed by ‘1’. This is because the instruction ‘ST R2, R0, 0’ is in the delay slot of ‘BEQZ R2, 24’ and is executed when finally this branch is taken. However, the execution of this store instruction does not effect the correctness of the program as the final result in ‘R1’ is now stored in a different location in the memory as compared to the original program. The snapshots are provided here, showing both the program at initialization point and when it is complete:



MIPS Animation : MultiplicationAsad_D

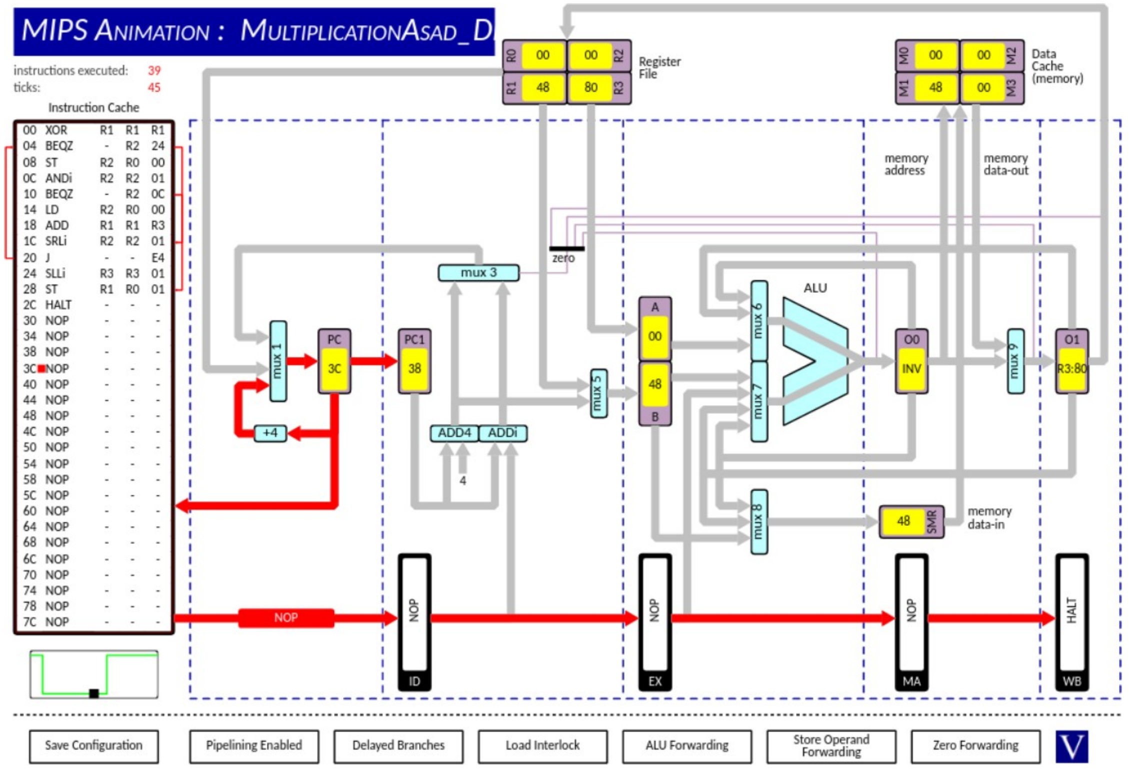
instructions executed: 0
ticks: 0

Instruction Cache

```
00  XOR    R1  R1  R1
04  BEQZ   -   R2  24
08  ST     R2  R0  00
0C  ANDi   R2  R2  01
10  BEQZ   -   R2  0C
14  LD     R2  R0  00
18  ADD    R1  R1  R3
1C  SRLi   R2  R2  01
20  J      -   -   E4
24  SLLi   R3  R3  01
28  ST     R1  R0  01
2C  HALT   -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

Register File: R0 00, R1 00, R2 08, R3 09

Data Cache (memory): M0 00, M1 28, M2 00, M3 00



MIPS Animation : MultiplicationAsad_D

instructions executed: 39
ticks: 45

Instruction Cache

```
00  XOR    R1  R1  R1
04  BEQZ   -   R2  24
08  ST     R2  R0  00
0C  ANDi   R2  R2  01
10  BEQZ   -   R2  0C
14  LD     R2  R0  00
18  ADD    R1  R1  R3
1C  SRLi   R2  R2  01
20  J      -   -   E4
24  SLLi   R3  R3  01
28  ST     R1  R0  01
2C  HALT   -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  NOP    -   -   -
3C  NOP    -   -   -
40  NOP    -   -   -
44  NOP    -   -   -
48  NOP    -   -   -
4C  NOP    -   -   -
50  NOP    -   -   -
54  NOP    -   -   -
58  NOP    -   -   -
5C  NOP    -   -   -
60  NOP    -   -   -
64  NOP    -   -   -
68  NOP    -   -   -
6C  NOP    -   -   -
70  NOP    -   -   -
74  NOP    -   -   -
78  NOP    -   -   -
7C  NOP    -   -   -
```

Register File: R0 00, R1 48, R2 00, R3 80

Data Cache (memory): M0 00, M1 48, M2 00, M3 00

3. Data Dependency

In the original program, the data dependency is between the following two instructions:

- LD R2, R0, R0
- SRLi R2, R2, 01

This data dependency can be easily removed by swapping the SRLi instruction with the following instruction of SLLi. Doing this will remove ‘4’ stall cycles from the overall execution, resulting in the program needing a total of ‘46’ clock cycles as compared to ‘50’ clock cycles for the original program.