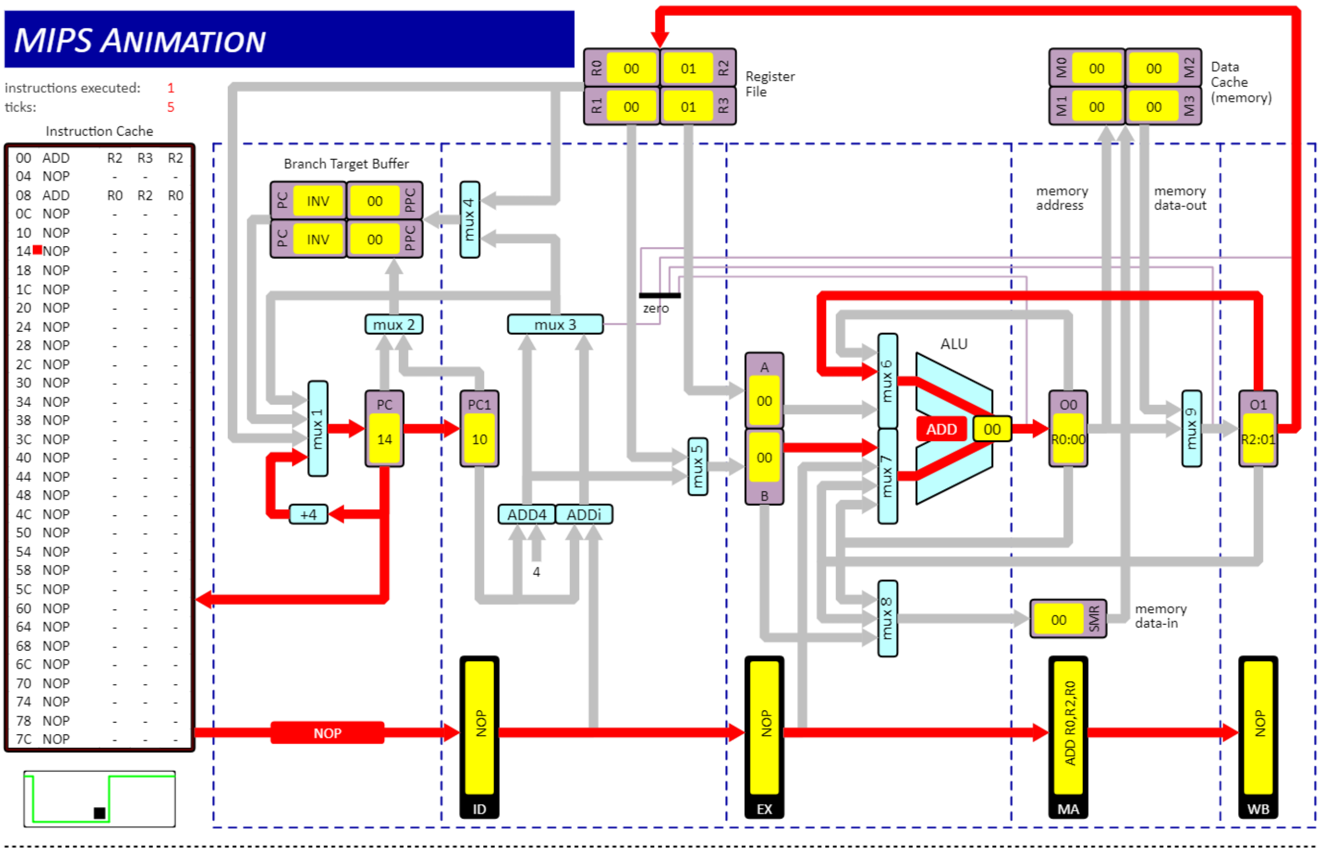
Q1

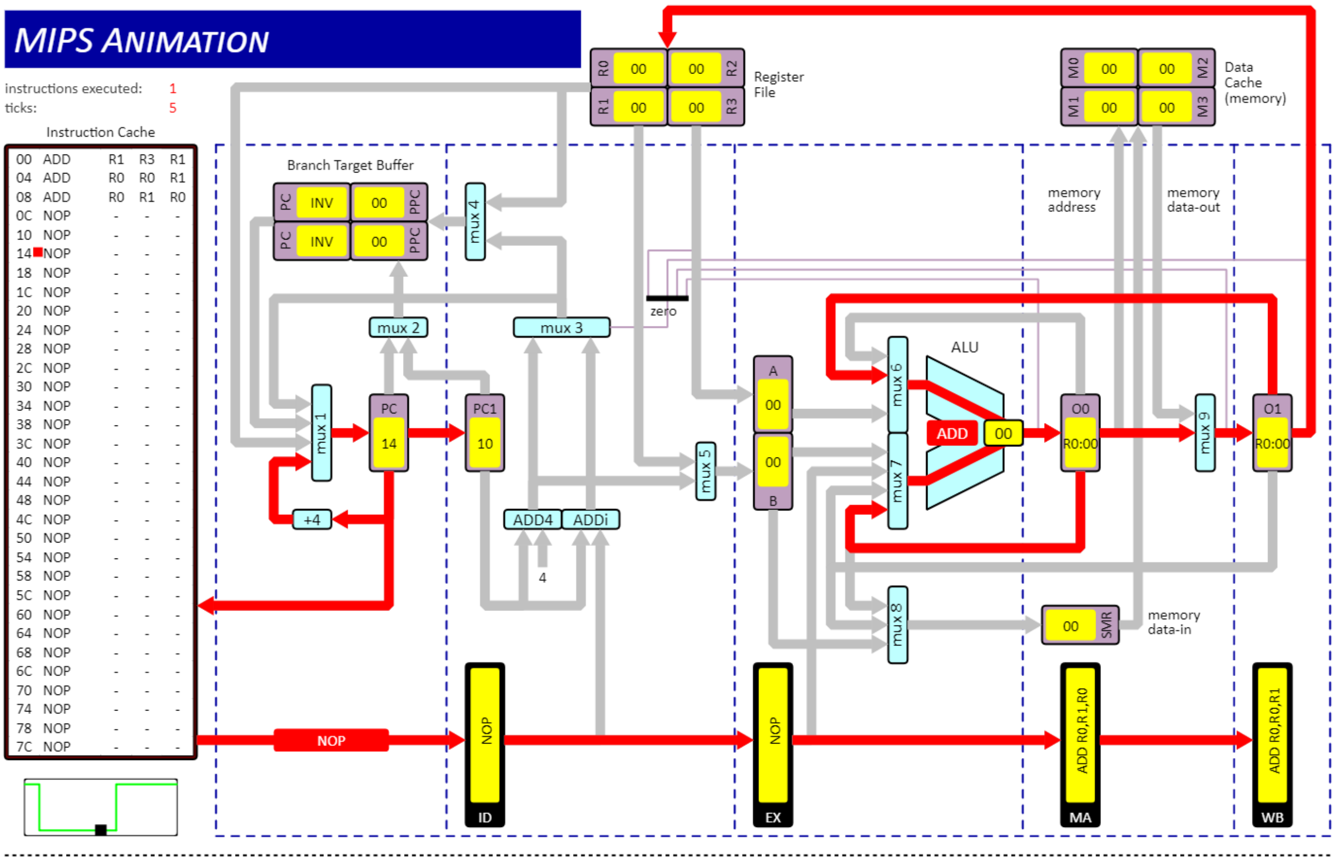
1. O1 to MUX6



```
00  ADD   R2  R3  R2
04  NOP   -   -   -
08  ADD   R0  R2  R0
0C  NOP   -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
18  NOP   -   -   -
1C  NOP   -   -   -
20  NOP   -   -   -
...
7C  NOP   -   -   -
```

```
R0 00   R2 01
R1 00   R3 01
M0 00   M2 00
M1 00   M3 00
```

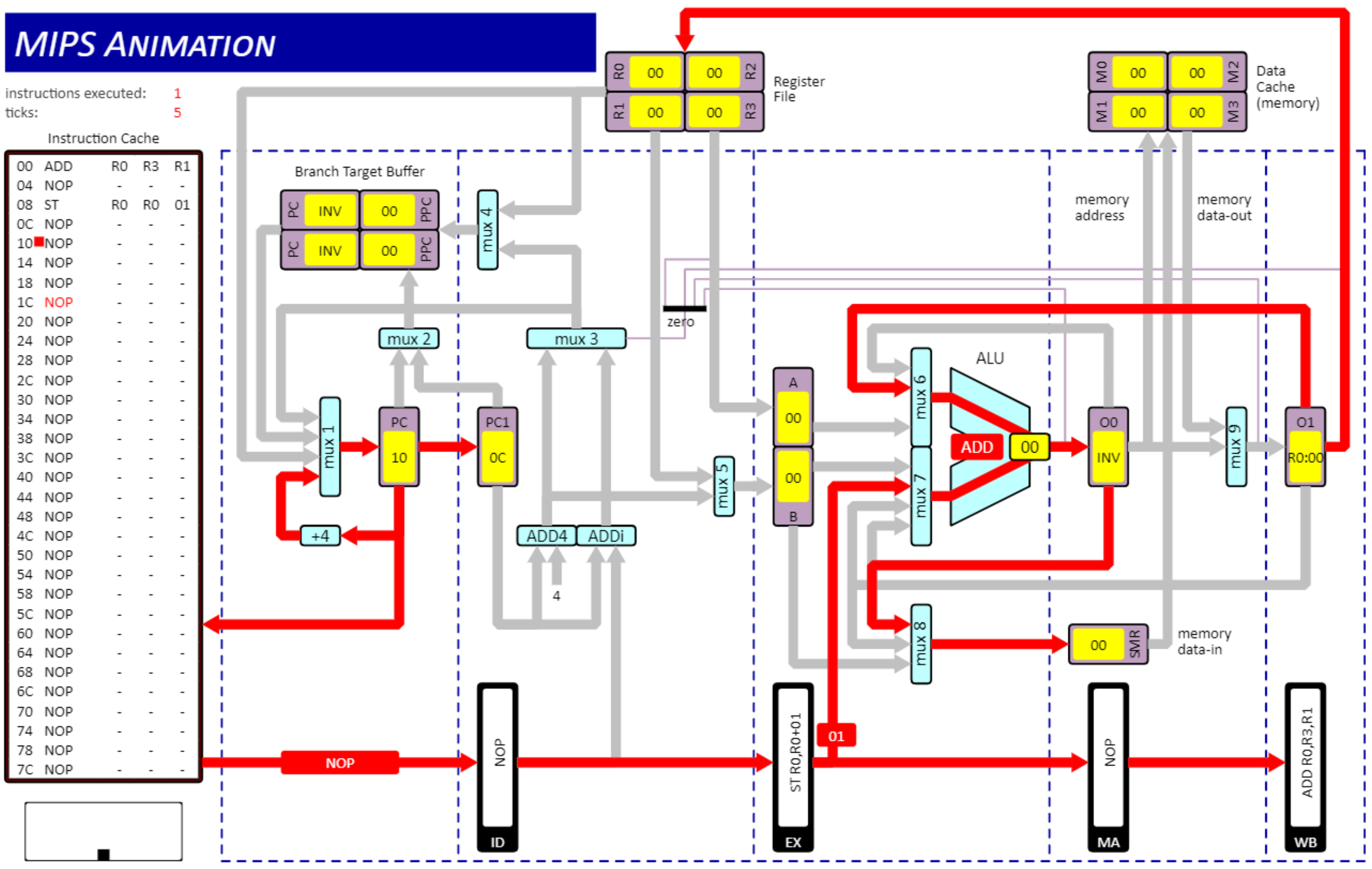
2. O0 to MUX7 and O1 to MUX6 (simultaneously)



```
00  ADD   R1  R3  R1
04  ADD   R0  R0  R1
08  ADD   R0  R1  R0
0C  NOP   -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
...
7C  NOP   -   -   -
```

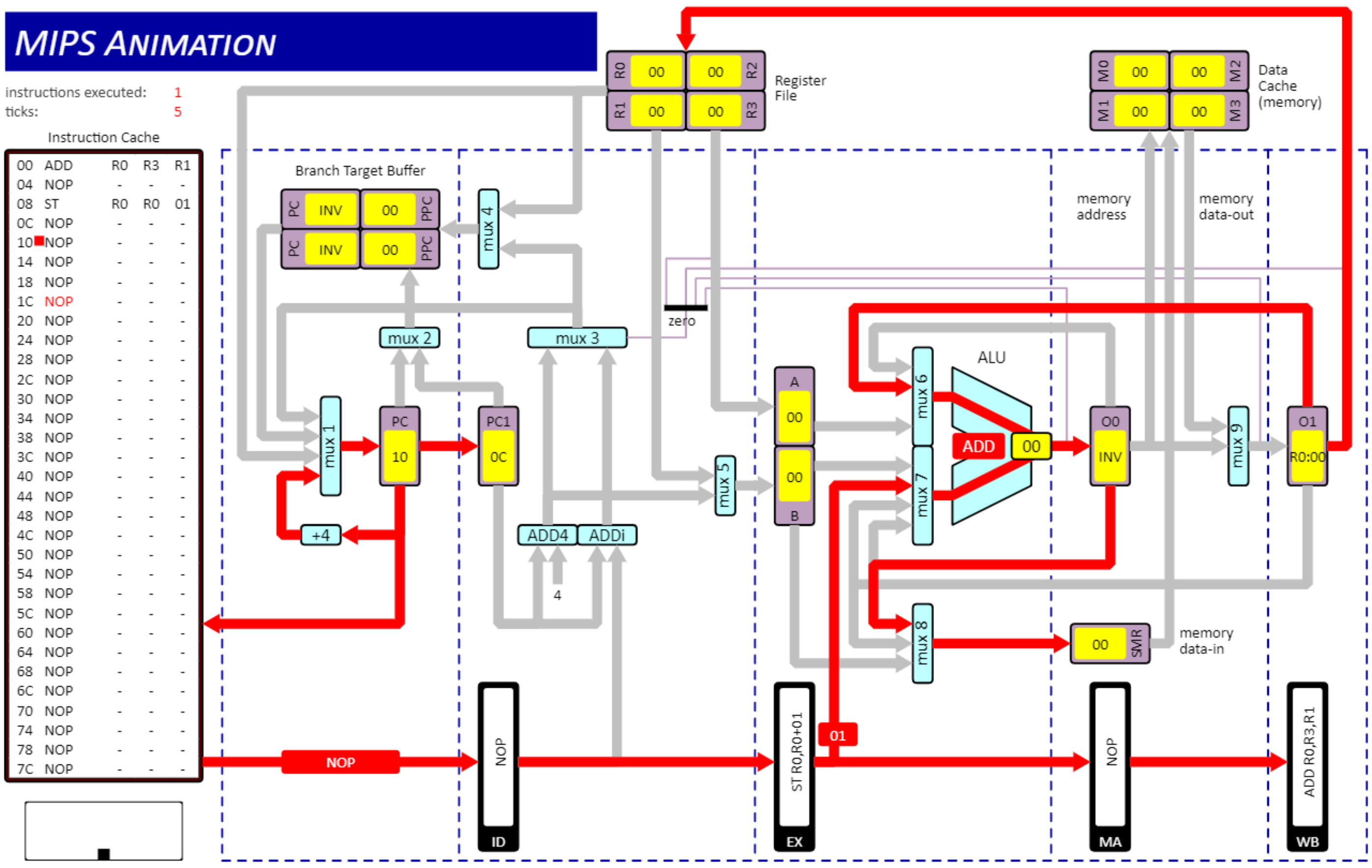
```
R0 00   R2 00
R1 00   R3 00
M0 00   M2 00
M1 00   M3 00
```

3. O0 to MUX8



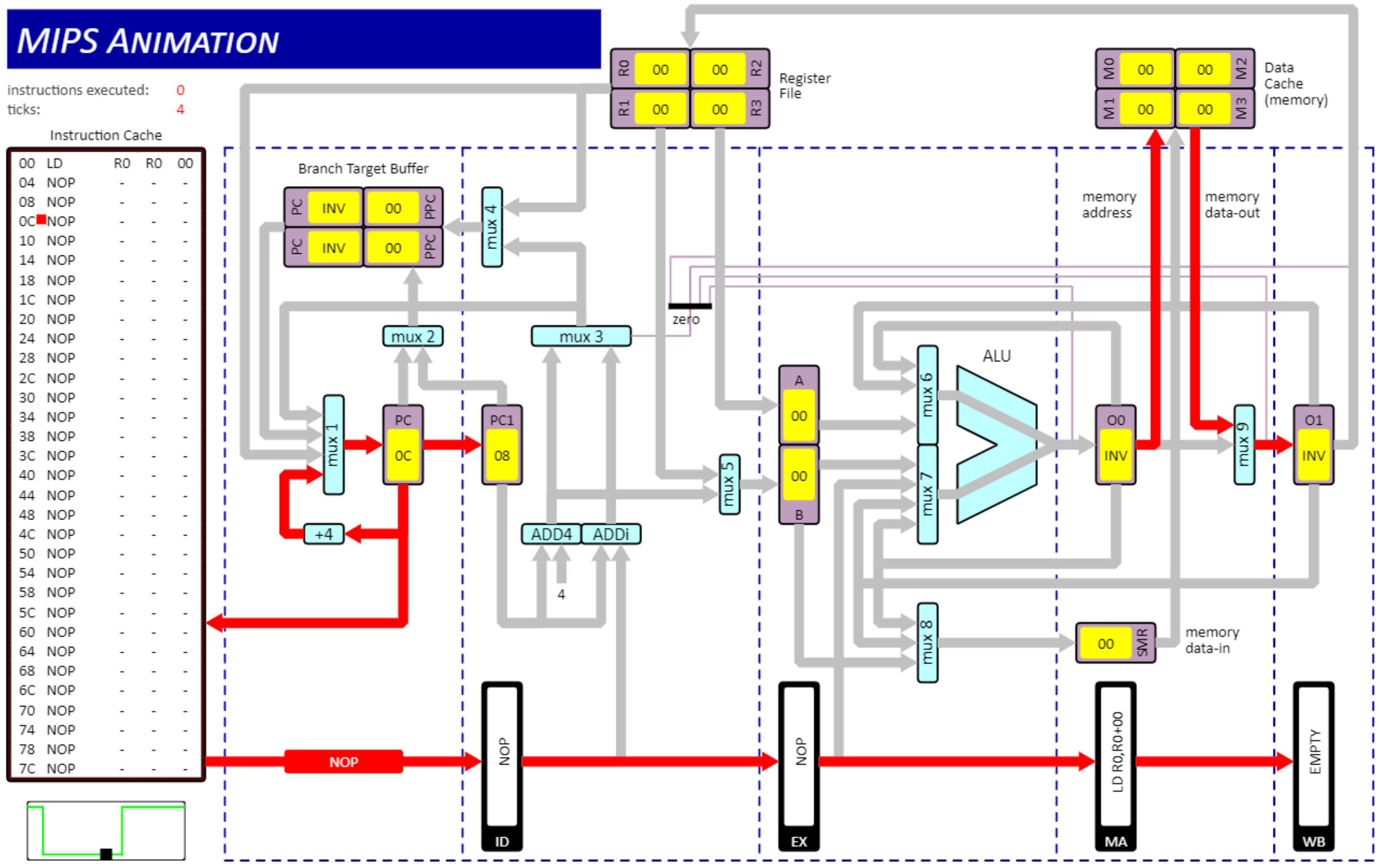
```
00  ADD   R0  R3  R1
04  NOP   -   -   -
08  ST    R0  R0  01
0C  NOP   -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
18  NOP   -   -   -
1C  NOP   -   -   -
20  NOP   -   -   -
24  NOP   -   -   -
28  NOP   -   -   -
2C  NOP   -   -   -
30  NOP   -   -   -
34  NOP   -   -   -
38  NOP   -   -   -
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. EX to MUX7 – Code and Image identical to part 3



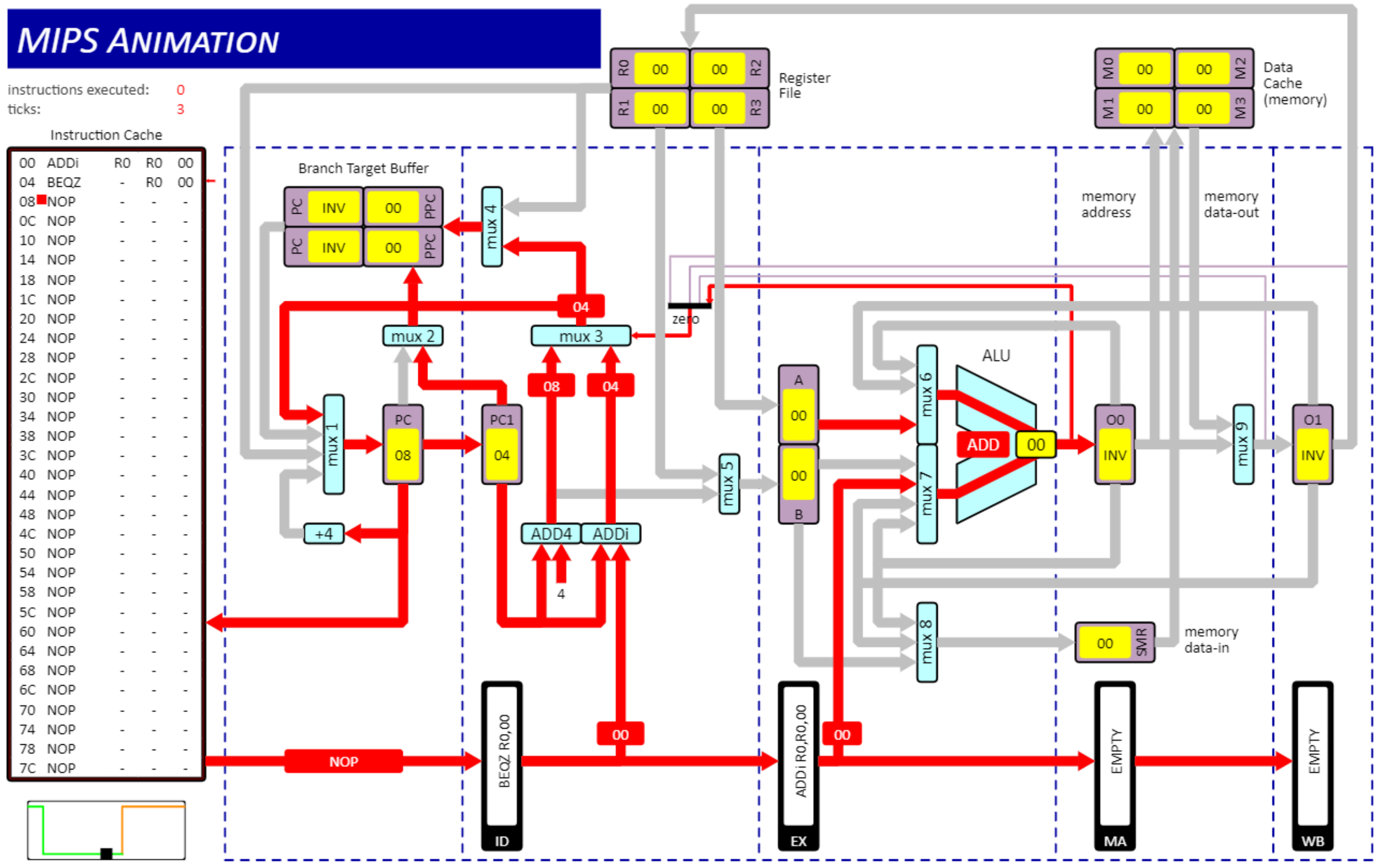
```
00  ADD   R0  R3  R1
04  NOP   -   -   -
08  ST    R0  R0  01
0C  NOP   -   -   -
10  NOP   -   -   -
14  NOP   -   -   -
18  NOP   -   -   -
1C  NOP   -   -   -
20  NOP   -   -   -
24  NOP   -   -   -
28  NOP   -   -   -
2C  NOP   -   -   -
30  NOP   -   -   -
34  NOP   -   -   -
38  NOP   -   -   -
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   -   -   -
```

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



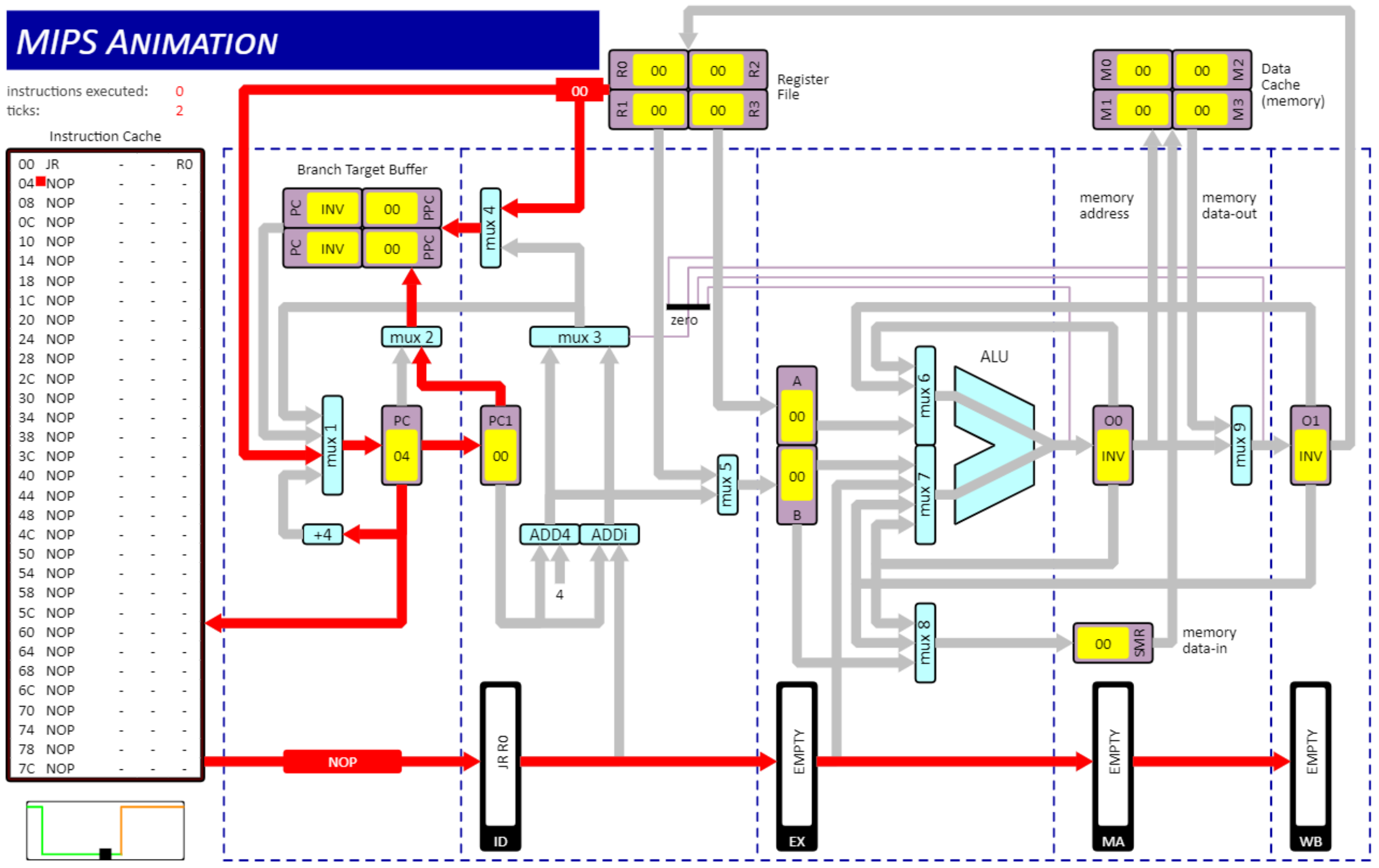
```
00  LD     R0  R0  00
04  NOP    -   -   -
08  NOP    -   -   -
0C  NOP    -   -   -
10  NOP    -   -   -
14  NOP    -   -   -
18  NOP    -   -   -
1C  NOP    -   -   -
20  NOP    -   -   -
24  NOP    -   -   -
28  NOP    -   -   -
2C  NOP    -   -   -
30  NOP    -   -   -
34  NOP    -   -   -
38  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    -   -   -
```

6. O0 to Zero detector



```
00  ADDi   R0  R0  00
04  BEQZ   -   R0  00
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    -   -   -
```

7. Register File to MUX1

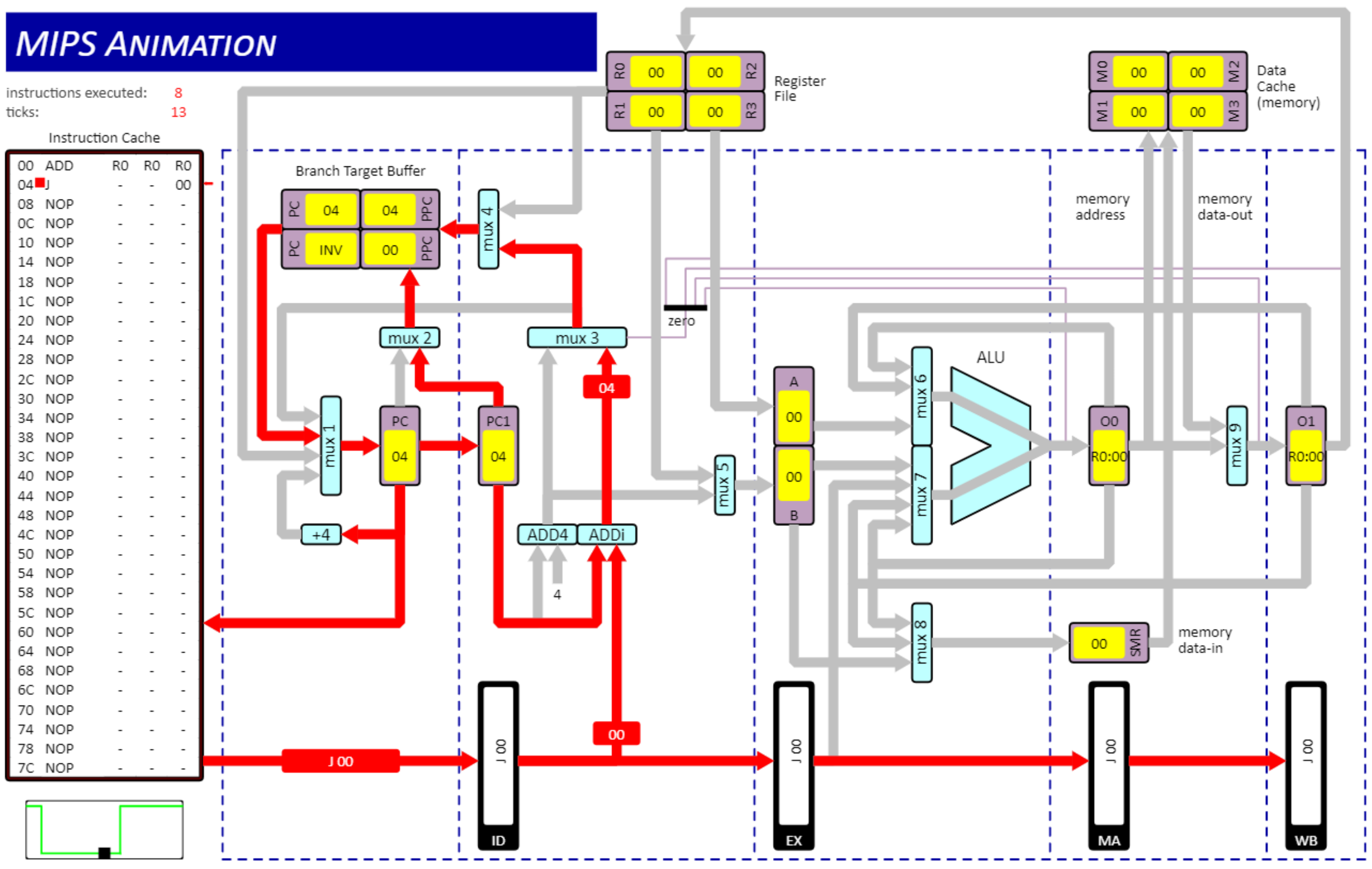

MIPS ANIMATION

instructions executed: 0
ticks: 2

Instruction Cache

```
00  JR    -   -   R0
04  NOP   -   -   -
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   -   -   -
```

8. Branch Target Buffer to MUX1


MIPS ANIMATION

instructions executed: 8
ticks: 13

Instruction Cache

```
00  ADD   R0  R0  R0
04  J     -   -   00
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   -   -   -
```

Q2

i)	ALU forwarding:	r1 = 15, Clock Cycles = 10
ii)	No ALU forwarding, CPU interlocks enabled:	r1 = 15, Clock Cycles = 18
iii)	ALU forwarding, CPU interlocks disabled:	r1 = 6, Clock Cycles = 10

The number of clock cycles is different with no ALU forwarding because the result from one instruction cannot be fed back into the ALU as an input. When ALU forwarding is enabled, the processor keeps the result stored in the immediate registers Out0 then Out1 for one clock cycle each. If one of the register operands in a subsequent instruction is the same as the register value of Out0 or Out1, this value will be sent to MUX6 or MUX7 depending on its positioning in the instruction. The benefit of this is we can avoid stalling due to the Memory Access and Write Back stages in the pipeline. When forwarding is absent, these stages must be carried out in full, leading to an increase in the amount of clock cycles required to carry out the same amount of instructions.

In the case of part iii) R1 = 6 because there are no interlocks in place to prevent data hazards from occurring. R1 and R2 are used in subsequent instructions, before these registers can be properly written back into memory.

Q3

i) Instructions = 39, Clock Cycles = 51

These numbers are not equal due to the design and nature of the pipeline itself. For the first 4 clock cycles no instruction is executed in full (despite having passed the “Execution stage”), as the result still needs to be written back into memory to be considered ‘complete’. An instruction cannot be executed if it hasn’t been fetched or decoded yet. Similarly, when the processor passes the Load or Store instructions, the pipeline is stalled for another clock cycle. This is to avoid a data hazard when storing and loading values to and from memory to registers that are subsequently used. Enabling interlocks prevents this hazard from occurring at the cost of lower throughput.

ii) Instructions = 39, Clock Cycles = 53

When branch prediction is exchanged for branch interlock, the processor takes more clock cycles due to the Jump instructions. Whenever the Jump instruction is taken the processor stalls for an additional clock cycle to prevent a control hazard from occurring. The next logical instruction stored in memory may not be the next instruction to be executed as the processor may branch to a different one.

iii) Instructions = 39, Clock Cycles = 47

When the Logical Shift instructions are exchanged the code takes shorter to execute because R2 is being shifted right one instruction later. This has a knock-on effect on the loop in the program as it terminates when r2 =0. The BEQZ is operating on R2 before it can be written into memory .