

MIPS ANIMATION

Instructions executed: 1
ticks: 5

Instruction Cache

00	ADD	R1	R2	R3
04	NOP	-	-	-
08	AND	R0	R1	R2
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	-	-	-

The diagram illustrates the MIPS CPU architecture and the execution of a NOP instruction. The Instruction Cache contains a list of instructions, with the current instruction (NOP) highlighted in red. The CPU components include the Branch Target Buffer, Register File, ALU, and Data Cache. The execution path for the NOP instruction is shown in red, indicating the flow of data and control signals. The PC is incremented by 4, and the next instruction is fetched from the cache. The ALU performs a zero test, and the Data Cache provides memory data-in and data-out.

NOP

AND R0, R1, R2

MIPS ANIMATION

instructions executed: 1
ticks: 5

Instruction Cache

00	ADD	R1	R2	R3
04	ADD	R2	R2	R1
08	AND	R0	R1	R2
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	-	-	-

Branch Target Buffer

PC	INV	00	PPC
PC	INV	00	PPC

Register File

R0	00	02	R2
R1	05	03	R3

Data Cache (memory)

M0	00	00	M2
M1	00	00	M3

memory address

memory data-out

memory data-in

ALU

AND

R0:00

R2:07

ADD4

ADDi

4

zero

mux 1

mux 2

mux 3

mux 4

mux 5

mux 6

mux 7

mux 8

mux 9

SMR

00

PC

14

PC1

10

+4

NOP

HALT

AND R0,R1,R2

ADD R2,R2,R1

ID

EX

MA

WB

ADD R2, R2, R1

AND R0, R1, R2

QUESTION 1.4 - EX to MUX7

[illegible]

ST R1, R0, 05

MIPS ANIMATION

instructions executed: 0
ticks: 4

Instruction Cache

	00	LD	R2	R1	05
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	-	-	-	-

Branch Target Buffer

	PC	INV	00	ppC
PC	INV	00	ppC	

Register File

	R0	00	00	R2
R1	00	00	00	R3

ALU

Data Cache (memory)

	M0	00	05	M2
M1	03	04	M3	

memory address

memory data-out

memory data-in

LD R2,R1+05

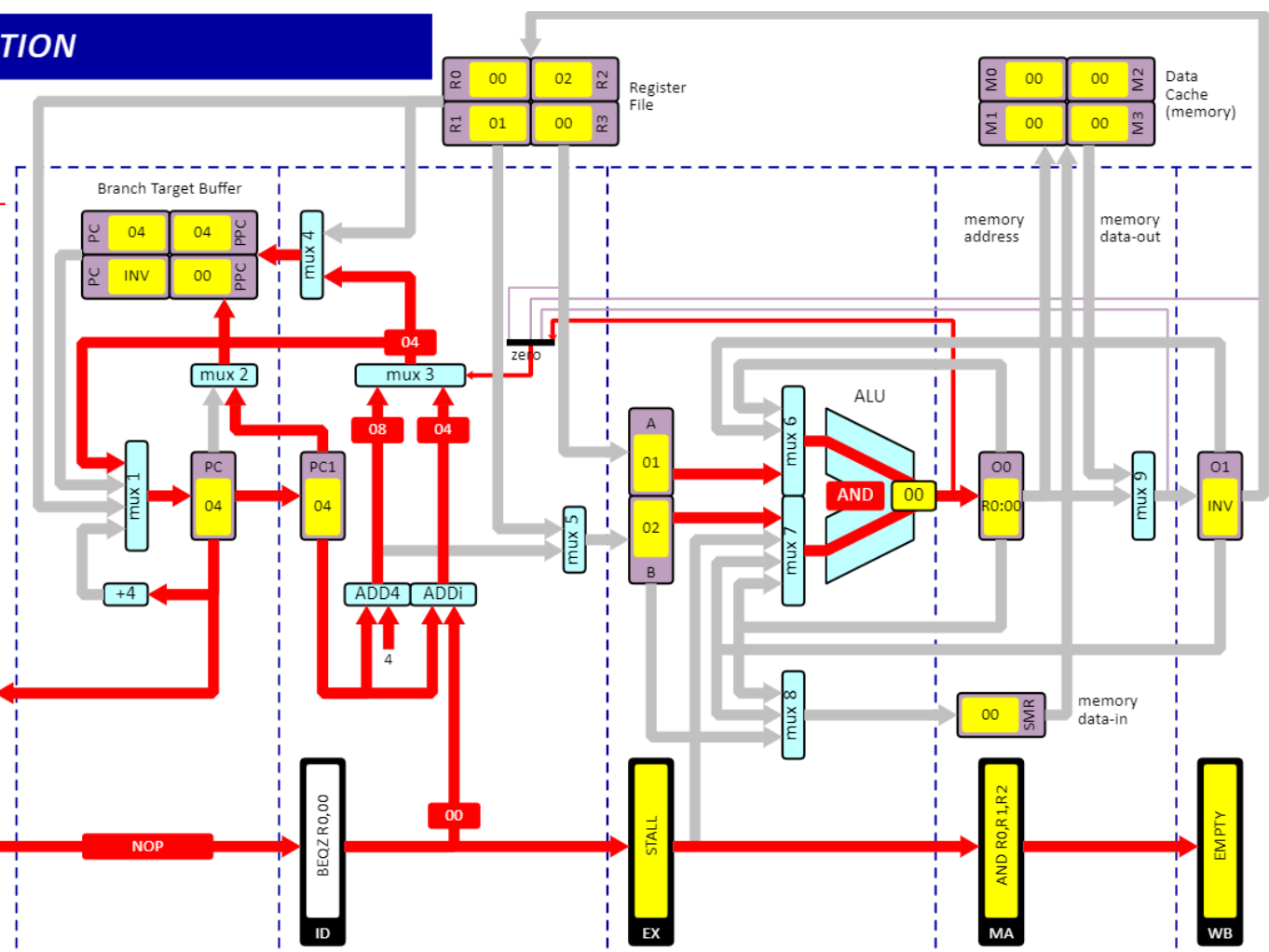
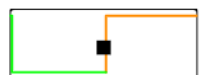
LD R2, R1, 05

MIPS ANIMATION

ticks:

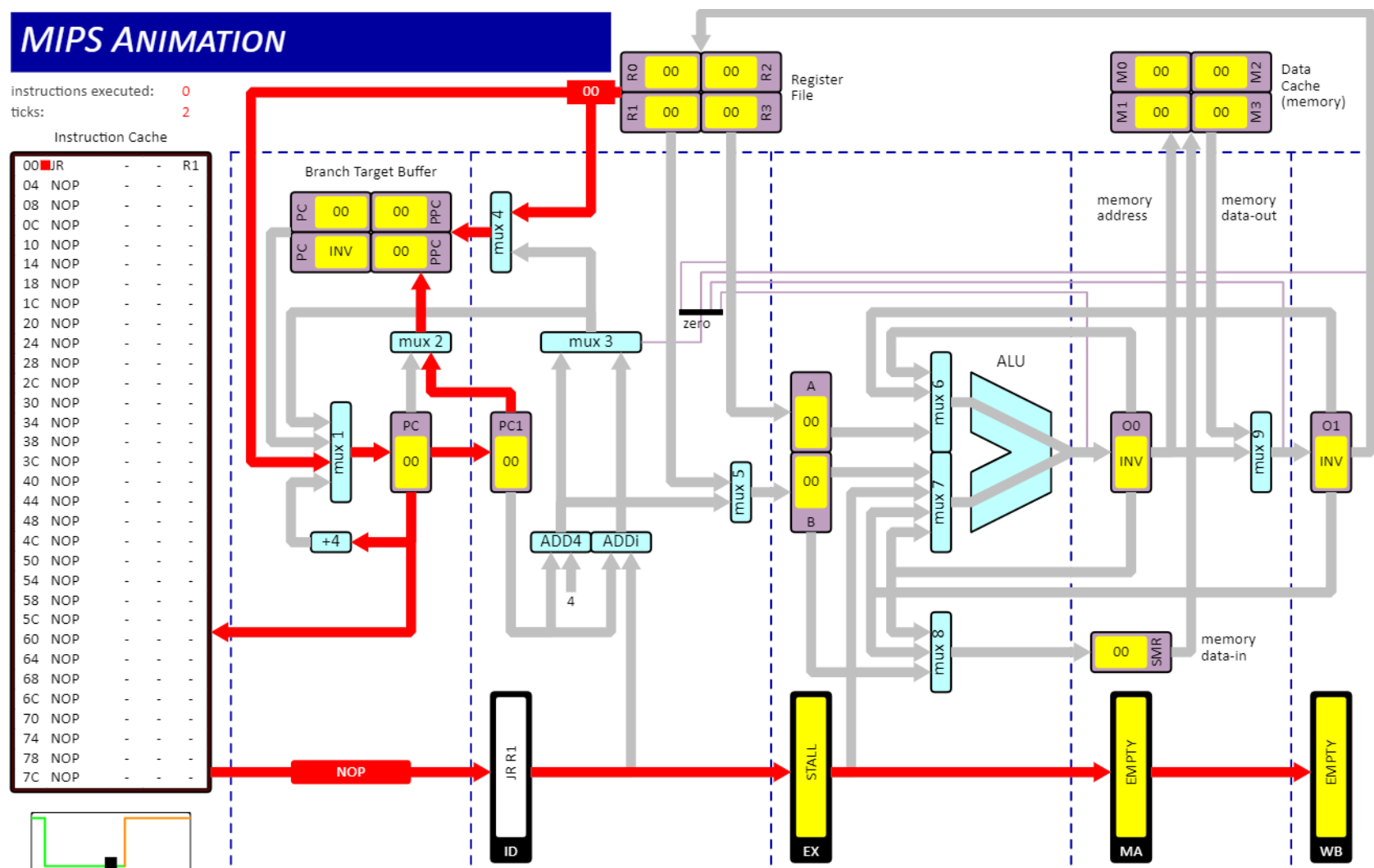
Instruction Cache

	00	AND	RO	R1	R2
04	BEQZ	-	RO	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	-	-	-	



BEQZ R0, 00

QUESTION 1.7 - Register File to MUX1



JR R1

MIPS ANIMATION

instructions executed: 0
ticks: 3

Instruction Cache

00	J	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	-

Branch Target Buffer

PC	00	00	ppC
PC	INV	00	ppC

Register File

R0	00	00	R2
R1	00	00	R3

Data Cache (memory)

M0	00	00	M2
M1	00	00	M3

memory address

memory data-out

memory data-in

ALU

mux 1, mux 2, mux 3, mux 4, mux 5, mux 6, mux 7, mux 8, mux 9

ADD4, ADDI, +4, zero

PC, PC1, INV, SWR

J 00, STALL, EMPTY

ID, EX, MA, WB

J 00

QUESTION 2

- (i) Ticks = 10
Result = 0x15
- (ii) Ticks = 18
Result = 0x15
- (iii) Ticks = 10
Result = 0x06

When ALU forwarding is enabled, the results from previous instruction can be stored in O0 and O1 before the MA or WB phase.

When ALU forwarding is disabled but CPU data dependency interlocks enabled, the processor has to stall between instructions so that the previous instructions can update their registers before the following instruction is executed. There are 4 instructions after the first one, with a 2 cycle stall per instruction ($4 \times 2 = 8$), this leads to an 8 tick delay.

When ALU forwarding is disabled and CPU data dependency interlocks are disabled too, the processor doesn't implement stalls and just executes the instructions "as is", leading to data hazards, which explains the different result.

QUESTION 3

- (i) Ticks = 50
Instructions = 38

The numbers aren't equal because of stalls. The beginning of the pipeline includes 4 stalls as its being loaded up to its capacity. There is a stall between LD and SRLi to allow for the value in R2 to be stored in an immediate register O1. Since this is instruction is performed 4 times, there are 4 ticks of stalls. The unconditional jump J causes a stall before the jump is executed, so that the correct instruction can be fetched (BEQZ rather than ST). This happens 2 times as there is branch prediction which predicts correctly twice. There are also two stalls due to the 2 different BEQZ statements once the conditions for the branch are met, as they each need to fetch the correct next instruction.

So ultimately $50 - 38 = 12$ ($4+4+2+2$) stalls.

(ii) Ticks = 53
Instructions = 38

11 of the stalls are the same as part (i), the other 4 come from the absence of branch prediction in the processor. The reason why it's only 11 is because only one of the BEQZ statements (BEQZ R2 24) is satisfied to branch, which means it requires a tick of stall to fetch the correct instruction. As there is no branch prediction, the unconditional jump J E0 will have a tick of stall 4 times so that the correct instruction can be fetched (BEQZ rather than ST). In total there is $53 - 38 = 15$ (11 + 4) ticks of stall.

(iii) Ticks = 46
Instructions = 38

By swapping the shift instructions we can get rid of the stall between LD and SRLi as SLLi doesn't use any operands from LD. As such we get an overall 4 stalls less, totalling in 8 (46-38) which are equivalent to the first 8 from part (i).