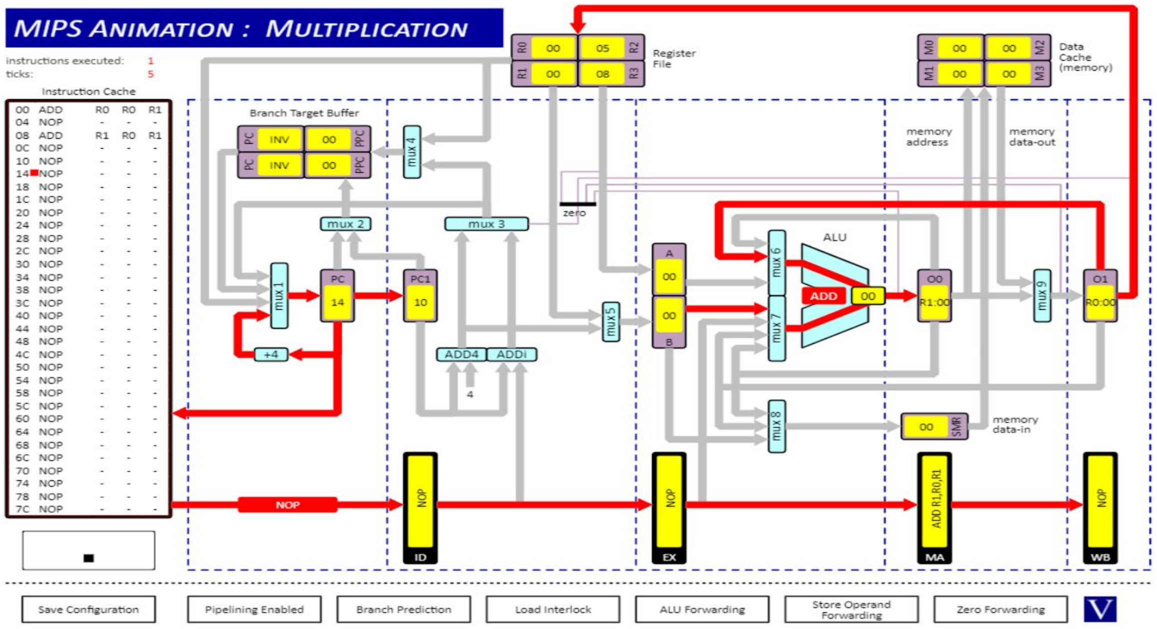
# CS3021

## Tutorial 4

## Q1)

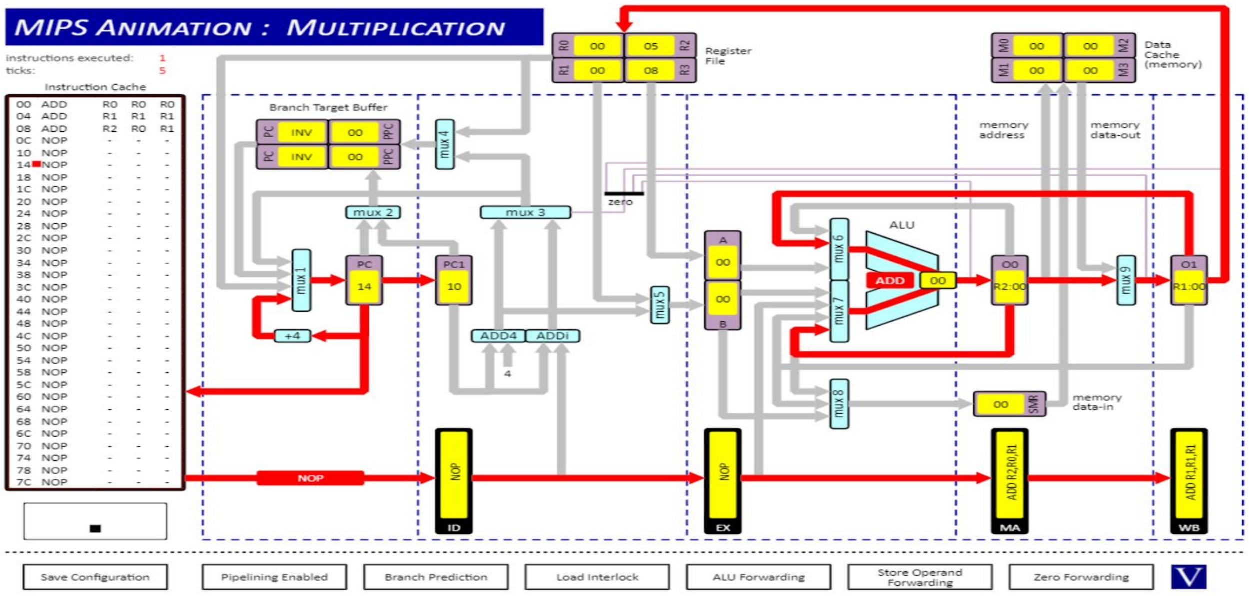
1)



```
00 ADD  R0 R0 R1
04 NOP  -  -  -
08 ADD  R1 R0 R1
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  -  -  -
```

ADD R0, R0, R1
NOP
ADD R1, R0, R1

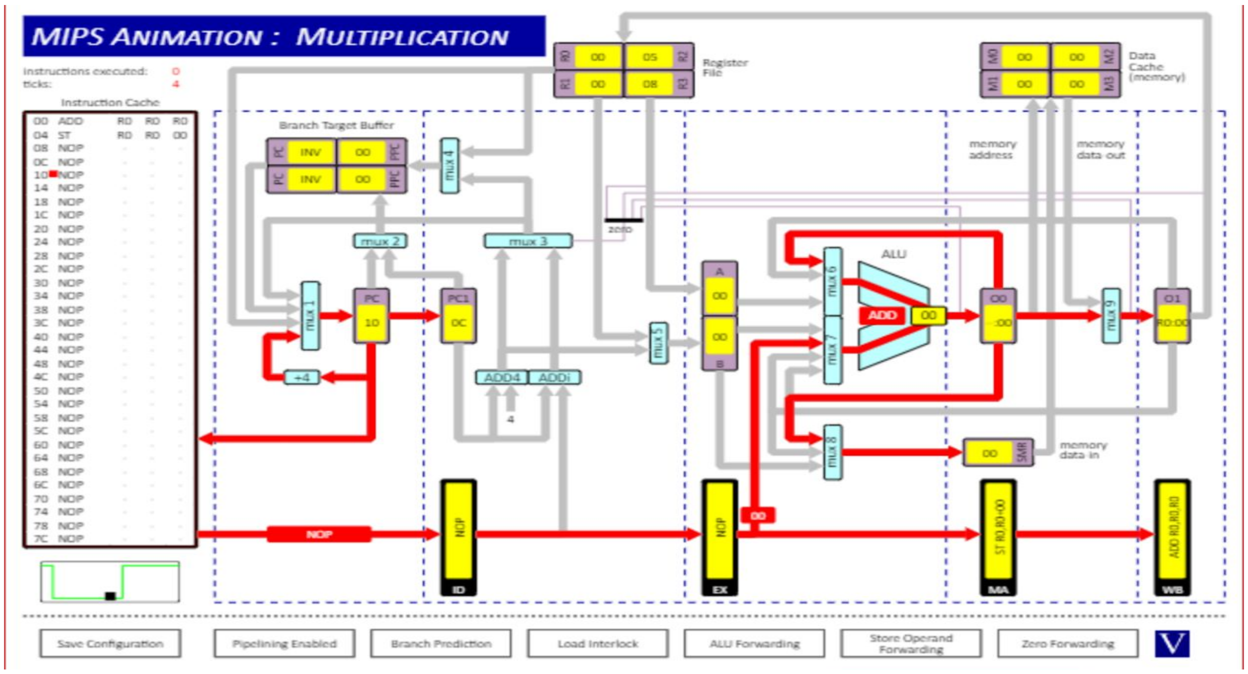
2)



```
instructions executed:   1
ticks:                   5

Instruction Cache
00  ADD    R0  R0  R0
04  ADD    R1  R1  R1
08  ADD    R2  R0  R1
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    -   -   -
```

ADD R0, R0, R0
ADD R1, R1, R1
ADD R2, R0, R1

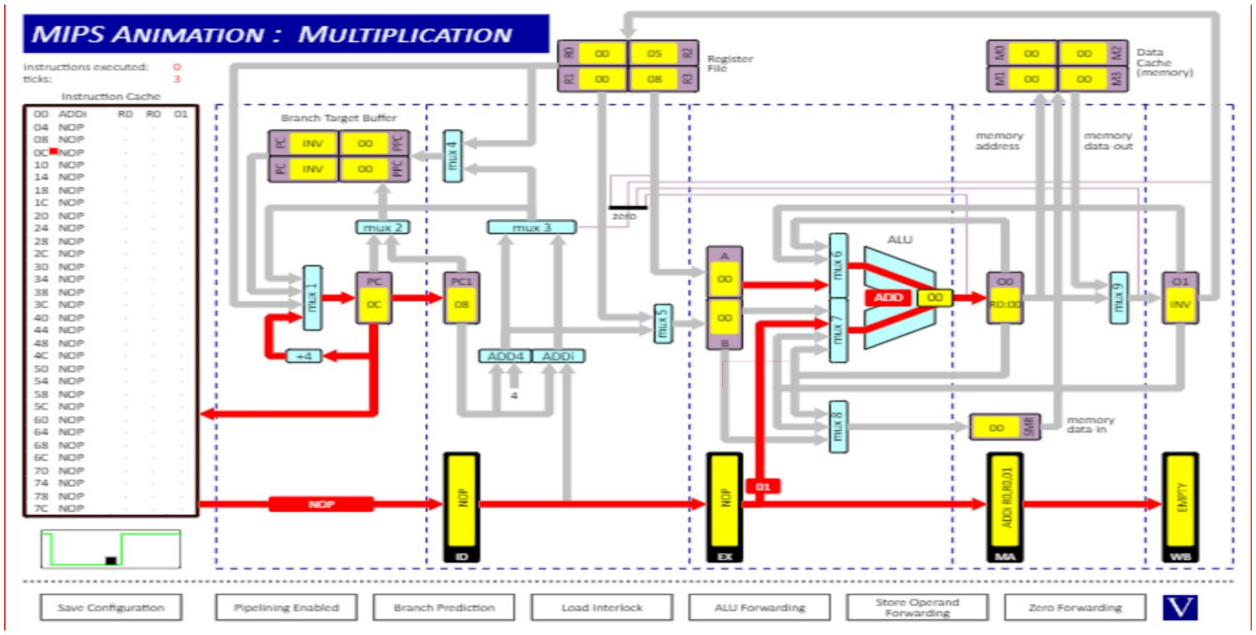
3)



```
instructions executed:   0
ticks:                   4

Instruction Cache
00  ADD    R0  R0  R0
04  ST     R0  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    -   -   -
```

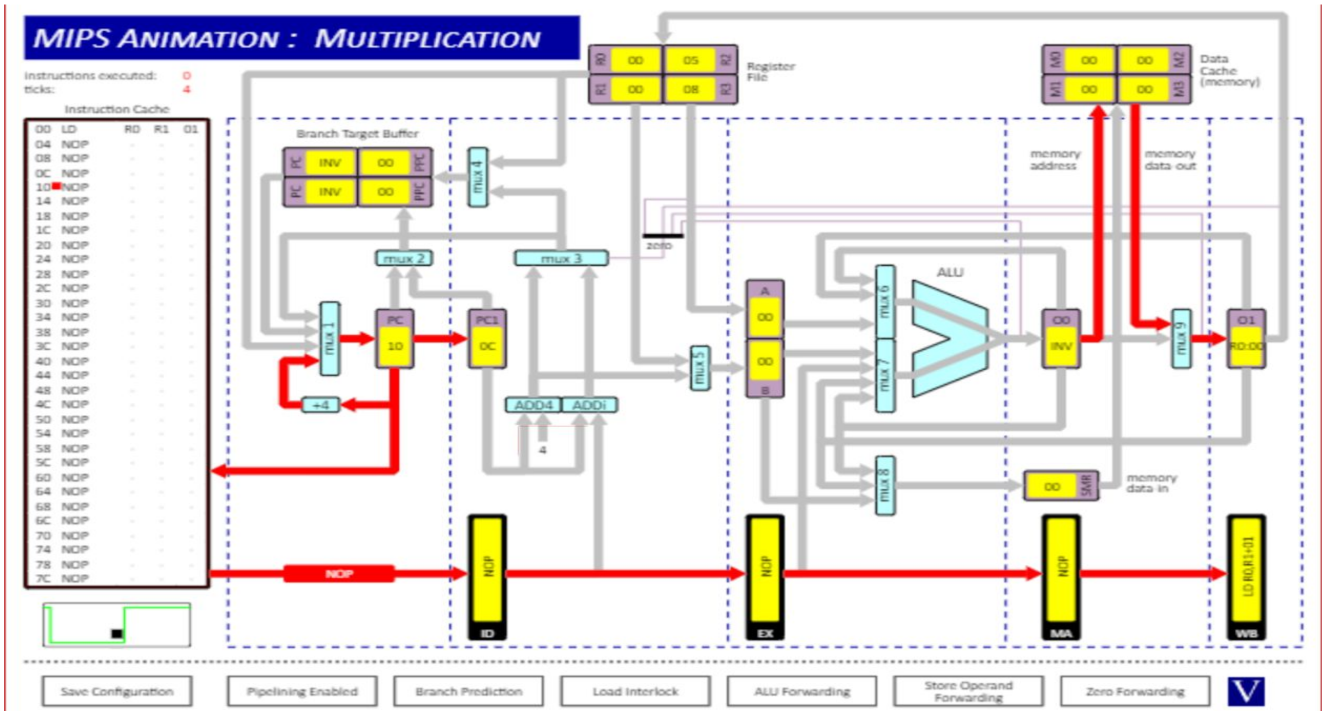
ADD R0, R0, R0
ST    R0, R0, 00

4)



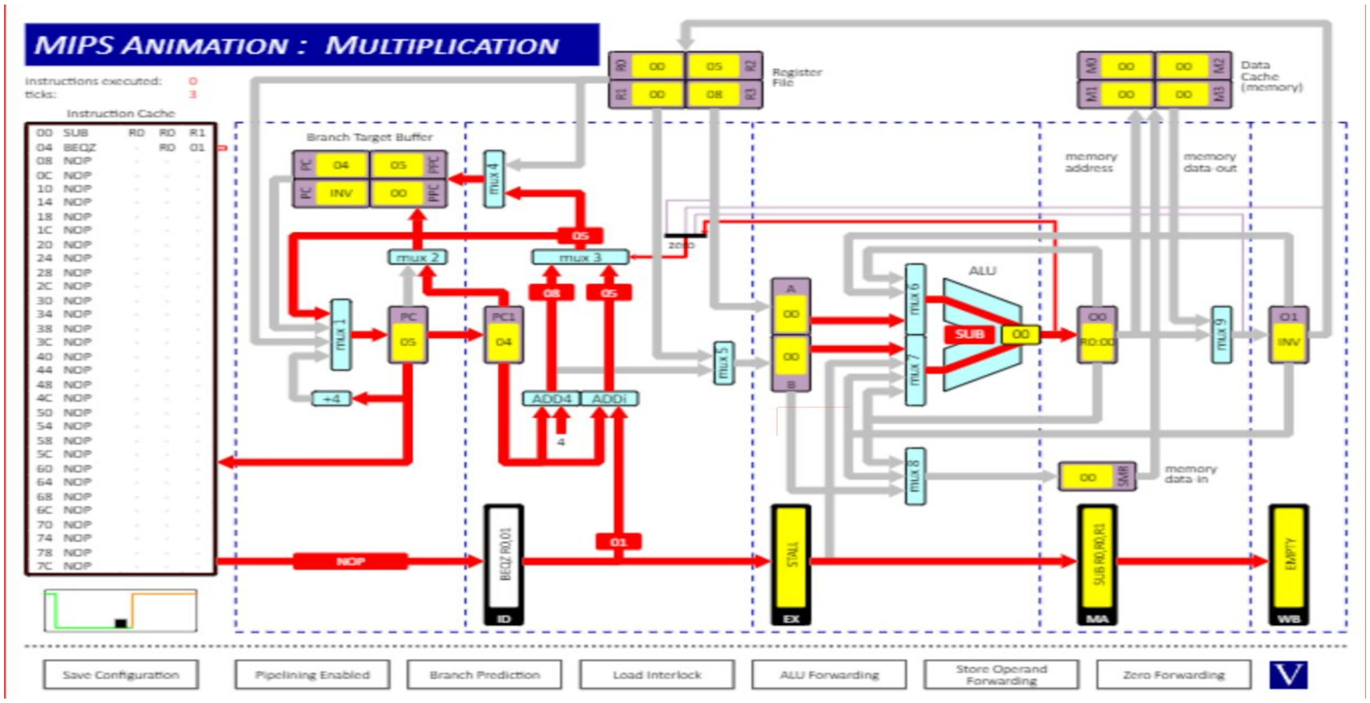
ADDi R0, R0, 01

5)



LD R0,R1,01

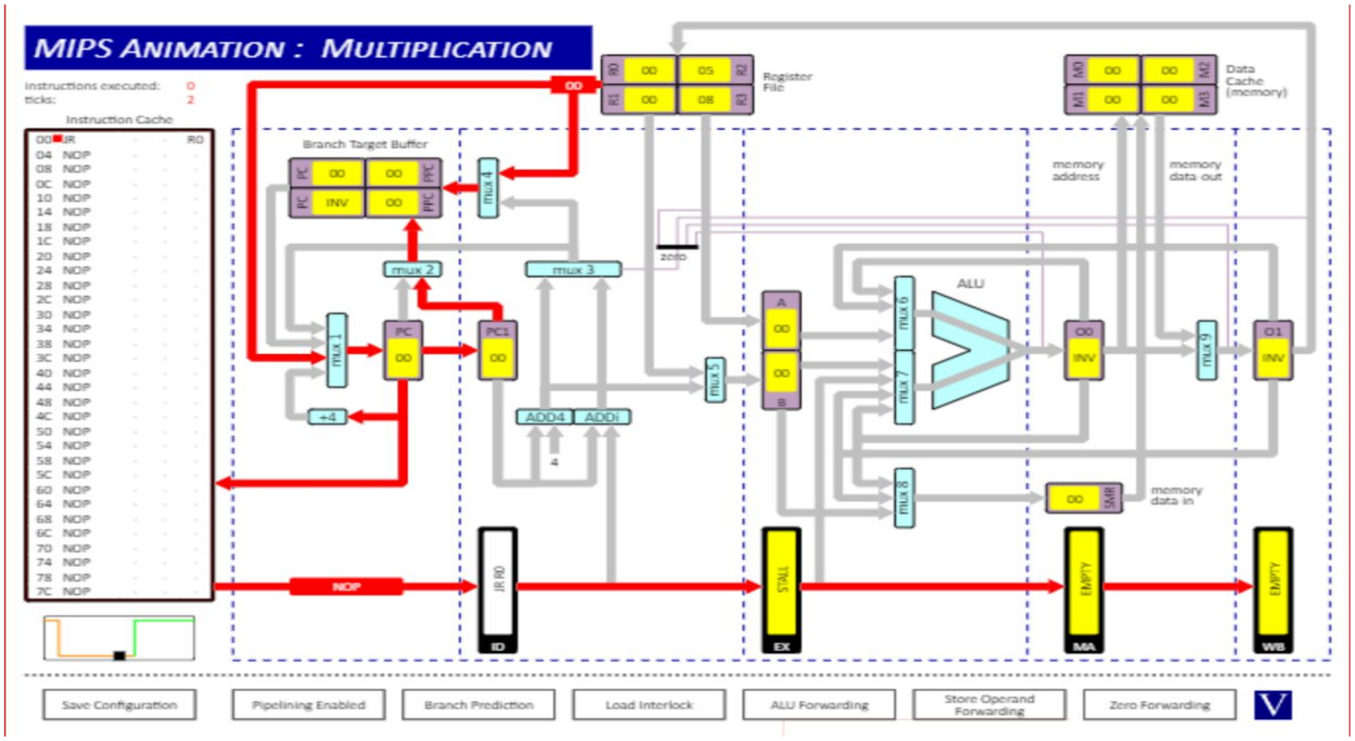
6)



```
00 SUB   R0 R0 R1
04 BEQZ  -  R0 01
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
```

SUB    R0, R0, R1
BEQZ  R0, 01

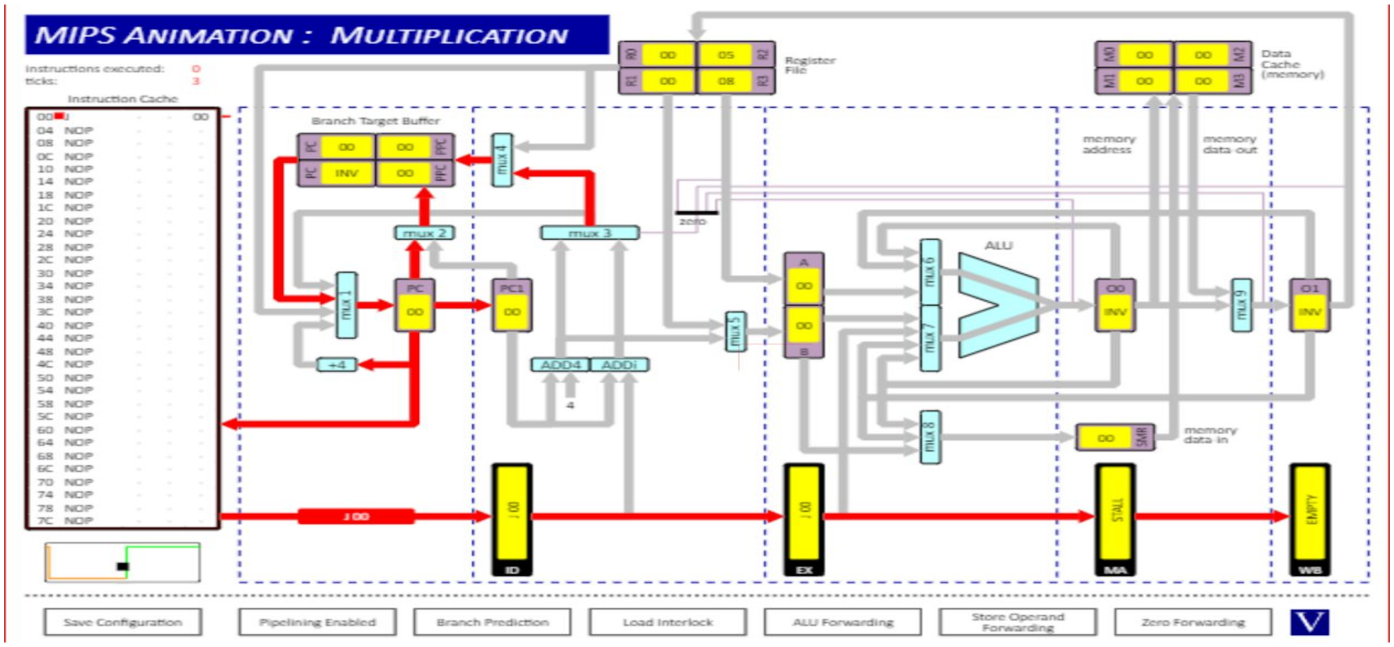
7)



```
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
```

JR  R0

8)


MIPS ANIMATION : MULTIPLICATION

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   -   -   -
```

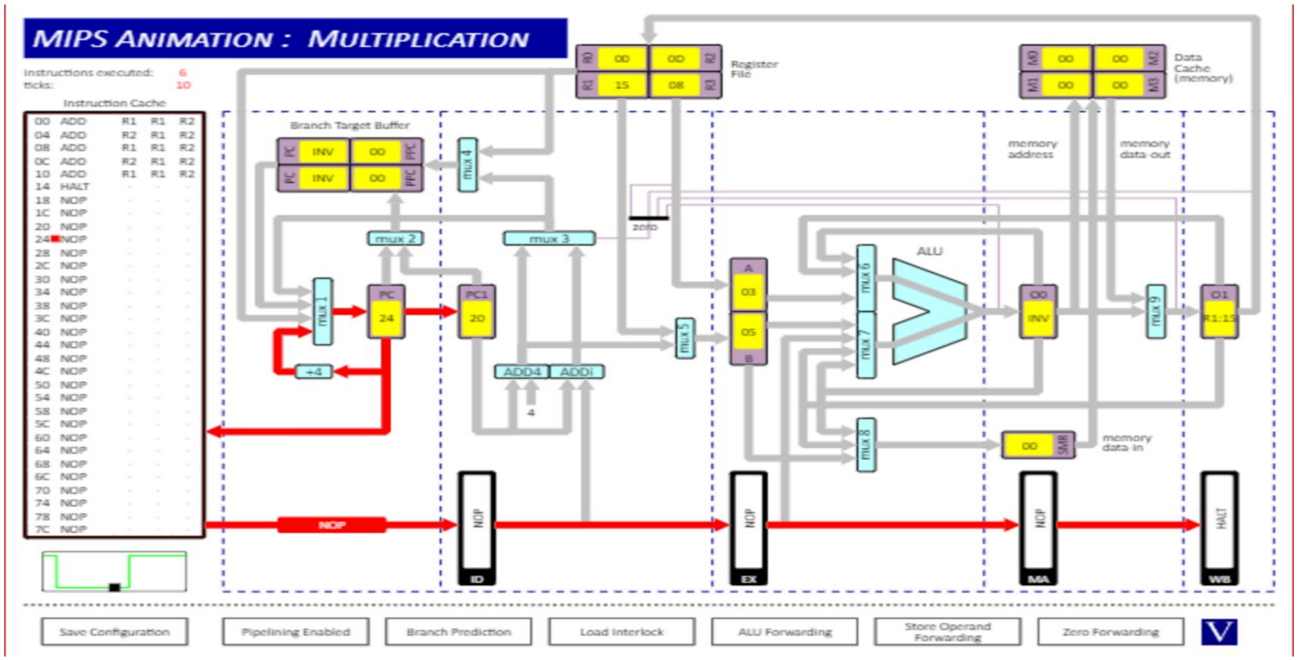
Register File: R0 00, R2 05, R1 00, R3 08

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

J 00

Q2)

i)


MIPS ANIMATION : MULTIPLICATION

instructions executed: 6
ticks: 10

Instruction Cache

```
00  ADD    R1  R1  R2
04  ADD    R2  R1  R2
08  ADD    R1  R1  R2
0C  ADD    R2  R1  R2
10  ADD    R1  R1  R2
14  HALT   -   -   -
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    -   -   -
```

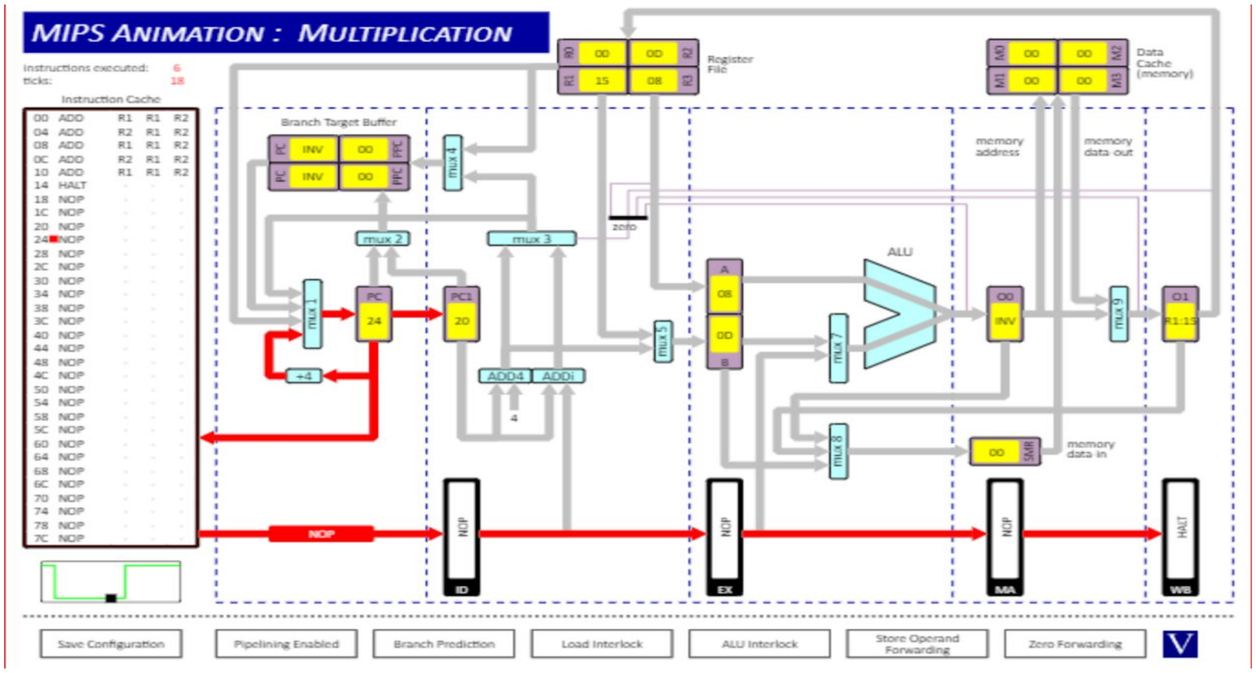
Register File: R0 00, R2 00, R1 15, R3 08

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

R1 = 15
Clock cycles = 10

With ALU forwarding there are no stalls due to O0 and O1 values being used directly without having to use registers, thus clock cycles is 10 and the answer stored in R1 is correct.

ii)



instructions executed: 6
ticks: 18

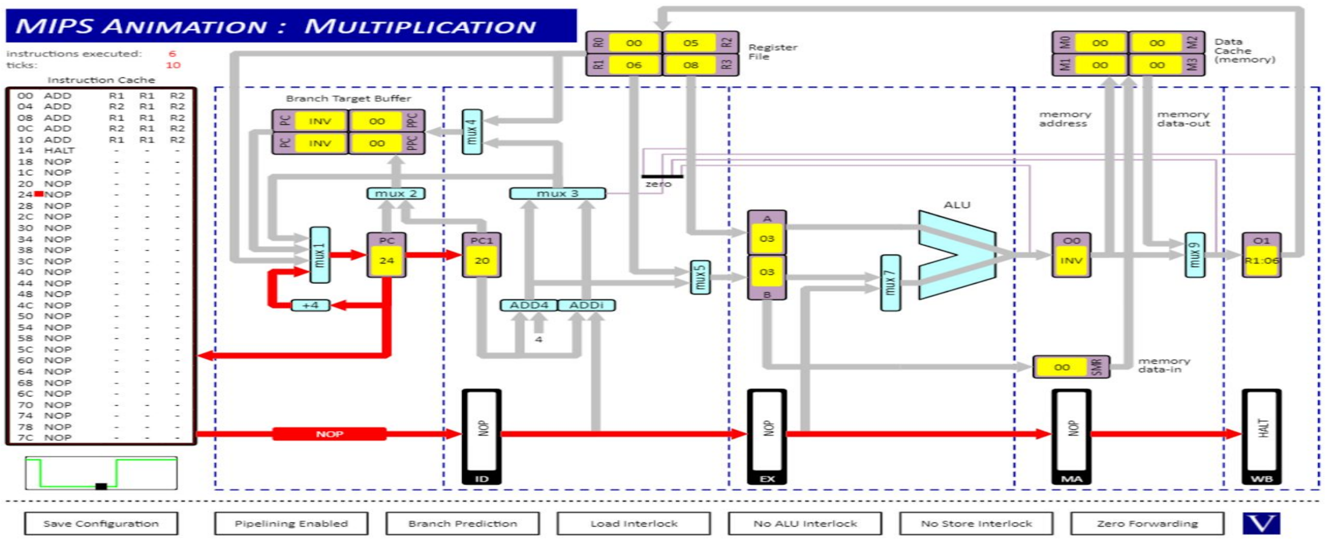
Instruction Cache

```
00 ADD   R1 R1 R2
04 ADD   R2 R1 R2
08 ADD   R1 R1 R2
0C ADD   R2 R1 R2
10 ADD   R1 R1 R2
14 HALT  -  -  -
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   -  -  -
```

R1 = 15
Clock cycles = 18

Without ALU forwarding, stalls occur due to O0 and O1 values being stored to registers before being accessed by the ALU, this causes more clock cycles however the answer in R1 is still correct due to the CPU data dependency interlocks ensuring that the correct register values are used.

iii)



MIPS ANIMATION : MULTIPLICATION

```
instructions executed:    6
ticks:                   10
```

Instruction Cache

```
00  ADD   R1  R1  R2
04  ADD   R2  R1  R2
08  ADD   R1  R1  R2
0C  ADD   R2  R1  R2
10  ADD   R1  R1  R2
14  HALT  -   -   -
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   -   -   -
```

Register File: R0 00, R1 06, R2 05, R3 08

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

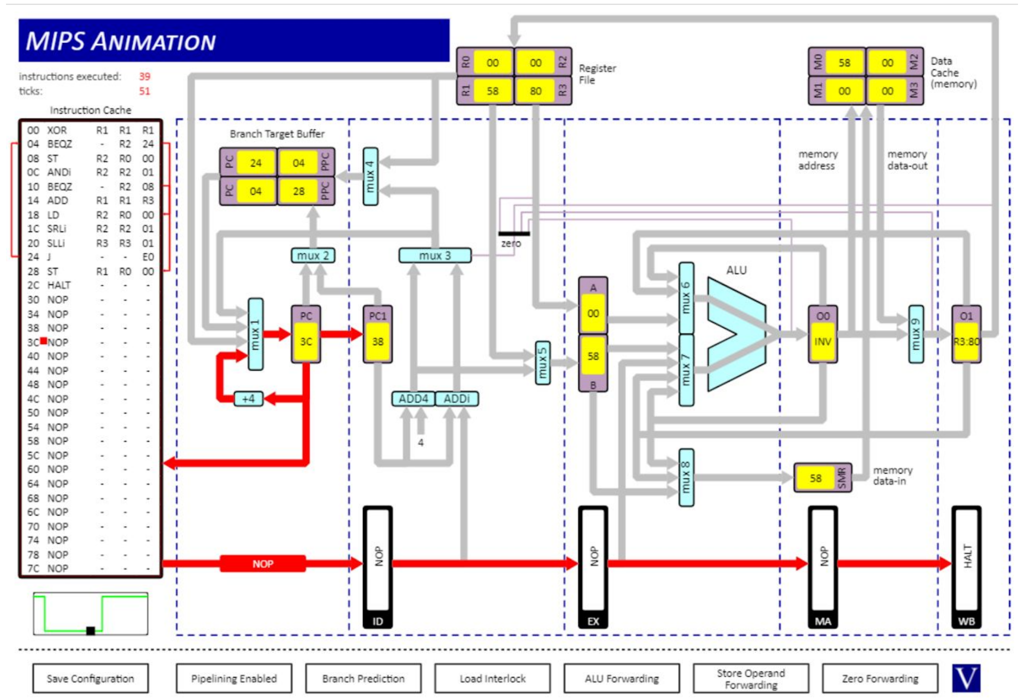
Save Configuration | Pipelining Enabled | Branch Prediction | Load Interlock | No ALU Interlock | No Store Interlock | Zero Forwarding

R1 = 6
Clock cycles = 10

Without ALU forwarding or CPU data dependency stalls will not occur due to the program not waiting for O0 and O1 to be stored into registers, thus giving the same number of clock cycles as part 1. However without ALU forwarding, the program still needs to take values from registers and since it doesn't wait for the correct register values, the answer stored in r1 is incorrect.

## Q3)

i)



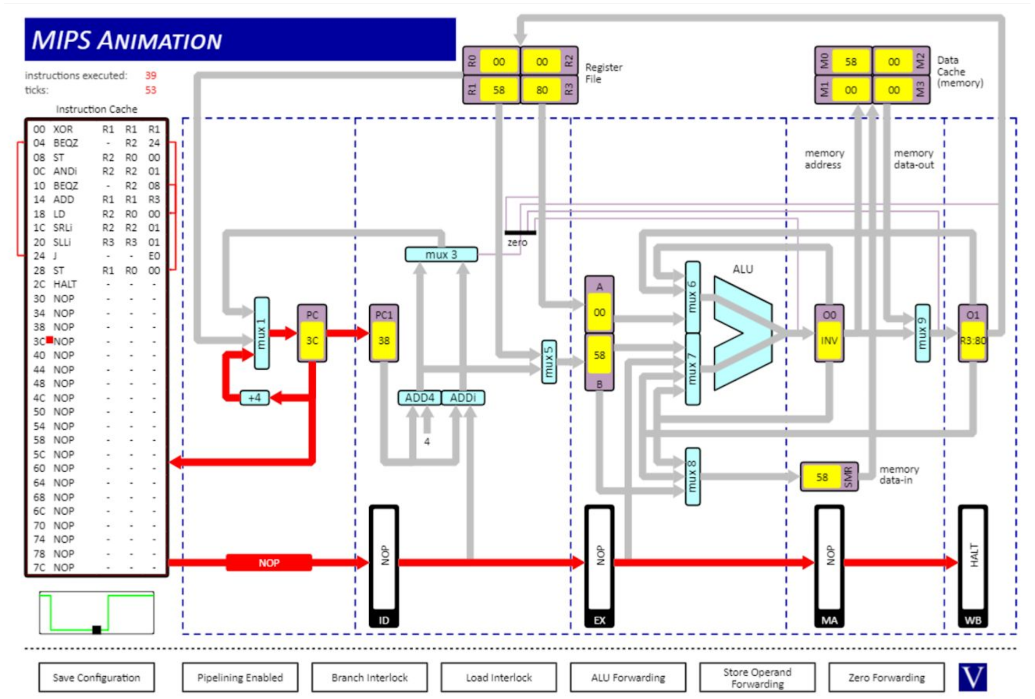
```
instructions executed:   39
ticks:                   51

Instruction Cache
00  XOR    R1  R1  R1
04  BEQZ   -   R2  24
08  ST     R2  R0  00
0C  ANDi   R2  R2  01
10  BEQZ   -   R2  08
14  ADD    R1  R1  R3
18  LD     R2  R0  00
1C  SRLi   R2  R2  01
20  SLLi   R3  R3  01
24  J      -   -   E0
28  ST     R1  R0  00
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 = 58, R2 = 00, R3 = 80
Data Cache: M0 = 58, M1 = 00, M2 = 00, M3 = 00
Branch Target Buffer: PC 24, PPC 04; PC 04, PPC 28

Instructions executed = 39
Clock cycles = 51

The clock cycles and instructions are not equal due to stall cycles which occur when branching initially or shifting, since no instructions are executed during these cycles, the number of cycles increases independently to the number of instructions.

ii)

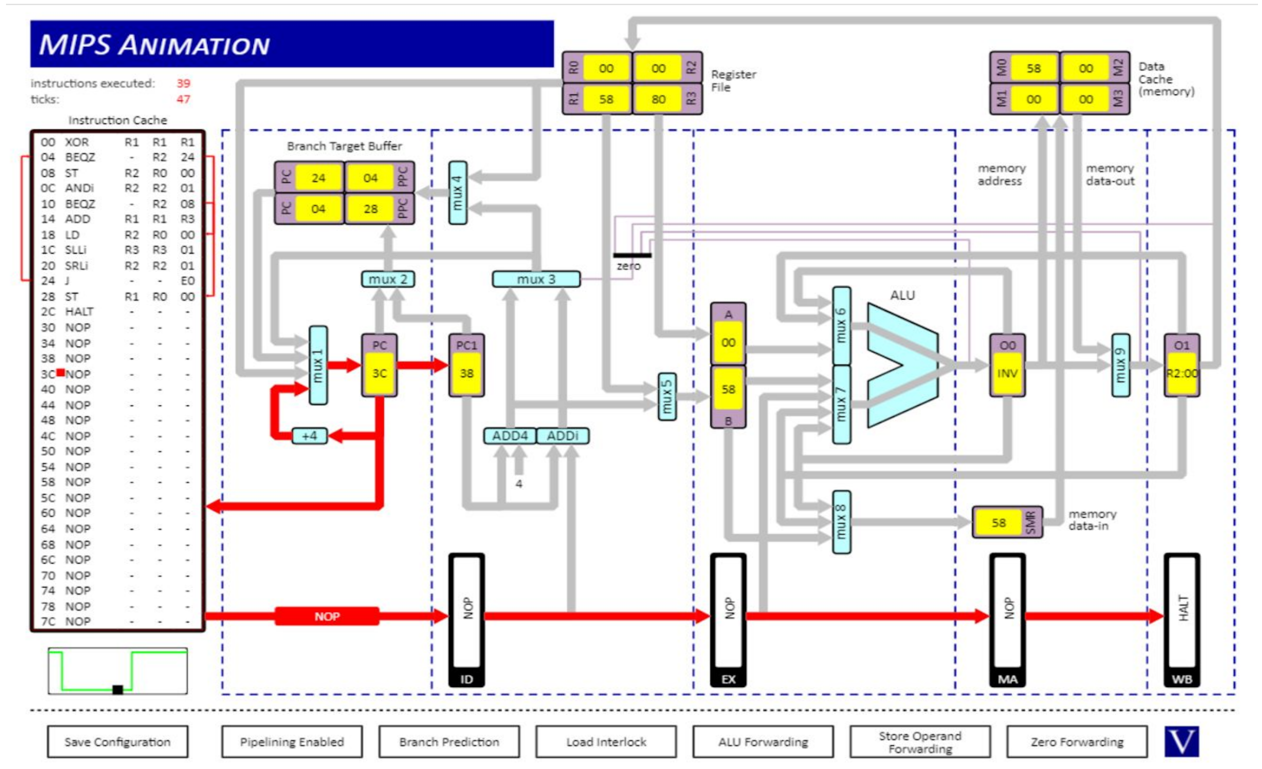


```
00 XOR    R1 R1 R1
04 BEQZ   -  R2 24
08 ST     R2 R0 00
0C ANDi   R2 R2 01
10 BEQZ   -  R2 08
14 ADD    R1 R1 R3
18 LD     R2 R0 00
1C SRLi   R2 R2 01
20 SLLi   R3 R3 01
24 J      -  -  E0
28 ST     R1 R0 00
2C HALT   -  -  -
30 NOP    -  -  -
34 NOP    -  -  -
38 NOP    -  -  -
3C NOP    -  -  -
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    -  -  -
```

Instructions executed = 39
Clock cycles = 53

Similarly to part (i) above, the difference between instructions and clock cycles is again due to stalls when branching and shifting however there are an extra 2 stall cycles due to branch prediction being disabled in this case causing the program to stall for every branch even after the initial branch for every loop.

iii)



```
instructions executed:   39
ticks:                   47

Instruction Cache
00  XOR    R1  R1  R1
04  BEQZ   -   R2  24
08  ST     R2  R0  00
0C  ANDi   R2  R2  01
10  BEQZ   -   R2  08
14  ADD    R1  R1  R3
18  LD     R2  R0  00
1C  SLLi   R3  R3  01
20  SRLi   R2  R2  01
24  J      -   -   E0
28  ST     R1  R0  00
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    -   -   -
```

Instructions executed = 39
Clock cycles = 47

The effect on execution time if the two shift instructions are swapped is that the program will no longer stall before the first shift thus causing the clock cycles to decrease by 4, giving 47 clock cycle instead of 51.

Thomas Fowley
15315353