MICROPROCESSOR AND COMPUTER ARCHITECTURE LABORATORY

UE19CS256

4th Semester, Academic Year 2020-21

Name: Atul Anurag	**SRN:** PES2UG19CS075	**Section:** B

Date: 21-04-2021

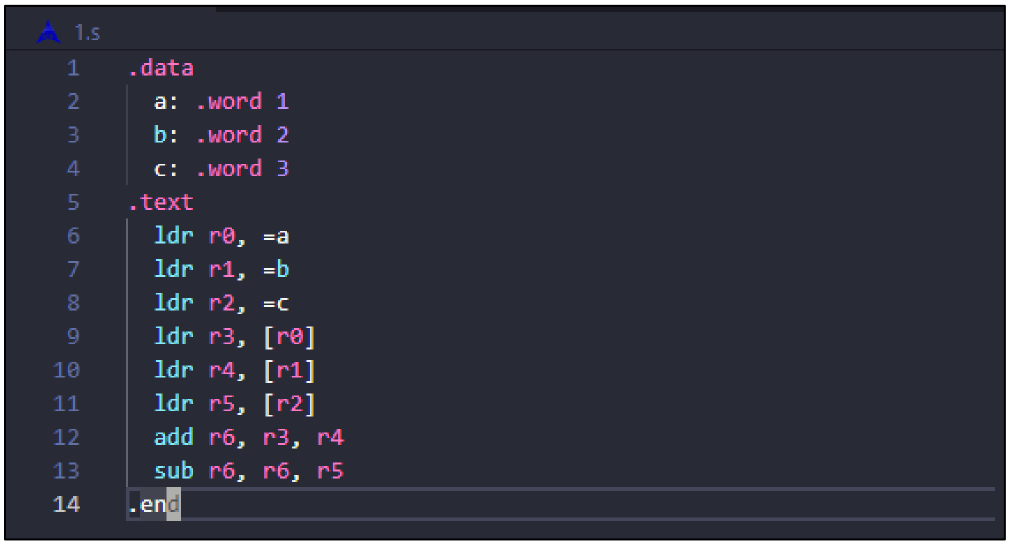
Week#10

Program Number: ____1__

Given a C- Code convert it in its equivalent ARM Code.
These programs need to be executed on ARMSIM Simulator

1) x = (a + b) - c

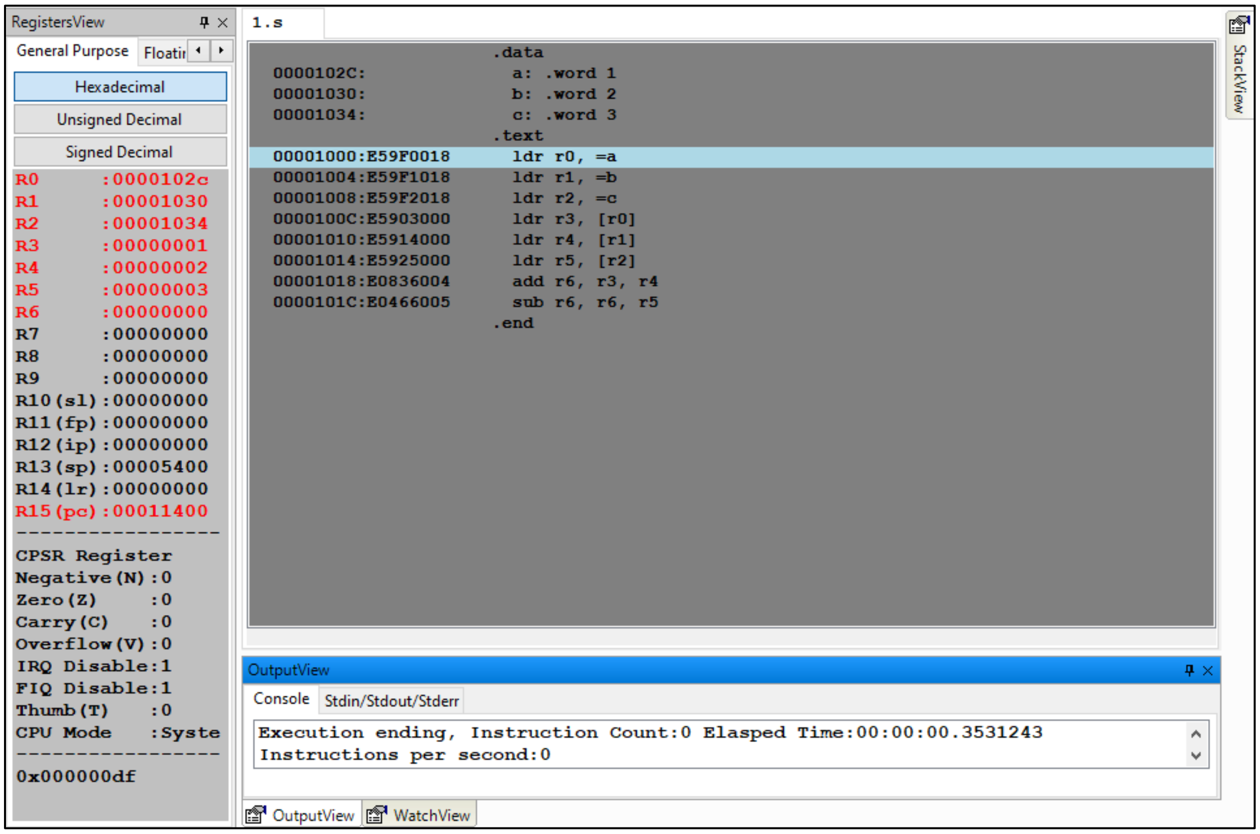
ARM Assembly Language Code



1.s

```
.data
  a: .word 1
  b: .word 2
  c: .word 3
.text
  ldr r0, =a
  ldr r1, =b
  ldr r2, =c
  ldr r3, [r0]
  ldr r4, [r1]
  ldr r5, [r2]
  add r6, r3, r4
  sub r6, r6, r5
.end
```

Screenshot showing the value of x, a, b, c in the register window.



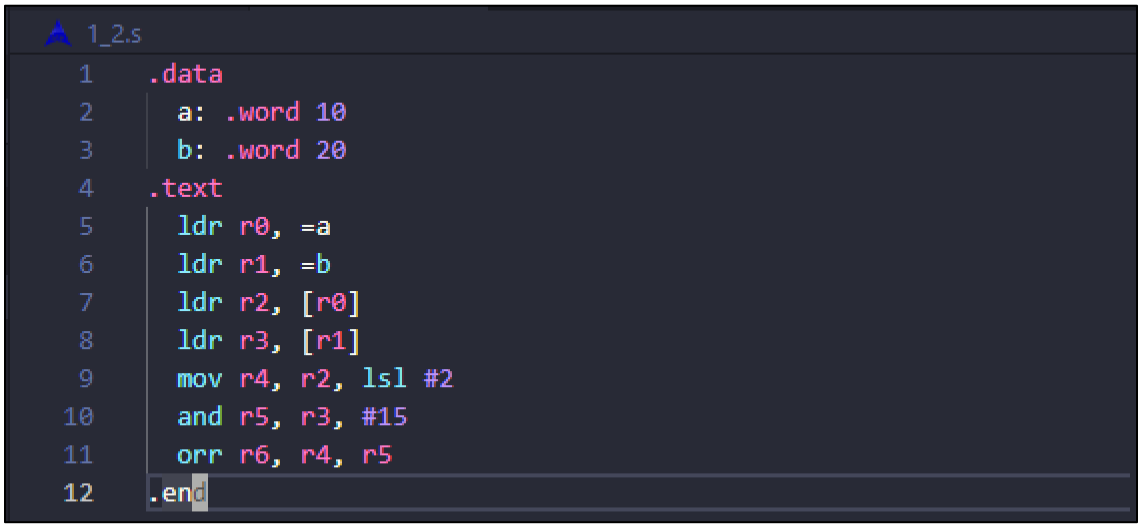
```
R0       :0000102c
R1       :00001030
R2       :00001034
R3       :00000001
R4       :00000002
R5       :00000003
R6       :00000000
R7       :00000000
R8       :00000000
R9       :00000000
R10(sl)  :00000000
R11(fp)  :00000000
R12(ip)  :00000000
R13(sp)  :00005400
R14(lr)  :00000000
R15(pc)  :00011400
-----------------
CPSR Register
Negative(N):0
Zero(Z)    :0
Carry(C)   :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :Syste
-----------------
0x000000df
```

```
                          .data
0000102C:                   a: .word 1
00001030:                   b: .word 2
00001034:                   c: .word 3
                          .text
00001000:E59F0018           ldr r0, =a
00001004:E59F1018           ldr r1, =b
00001008:E59F2018           ldr r2, =c
0000100C:E5903000           ldr r3, [r0]
00001010:E5914000           ldr r4, [r1]
00001014:E5925000           ldr r5, [r2]
00001018:E0836004           add r6, r3, r4
0000101C:E0466005           sub r6, r6, r5
                          .end
```

```
Execution ending, Instruction Count:0 Elasped Time:00:00:00.3531243
Instructions per second:0
```

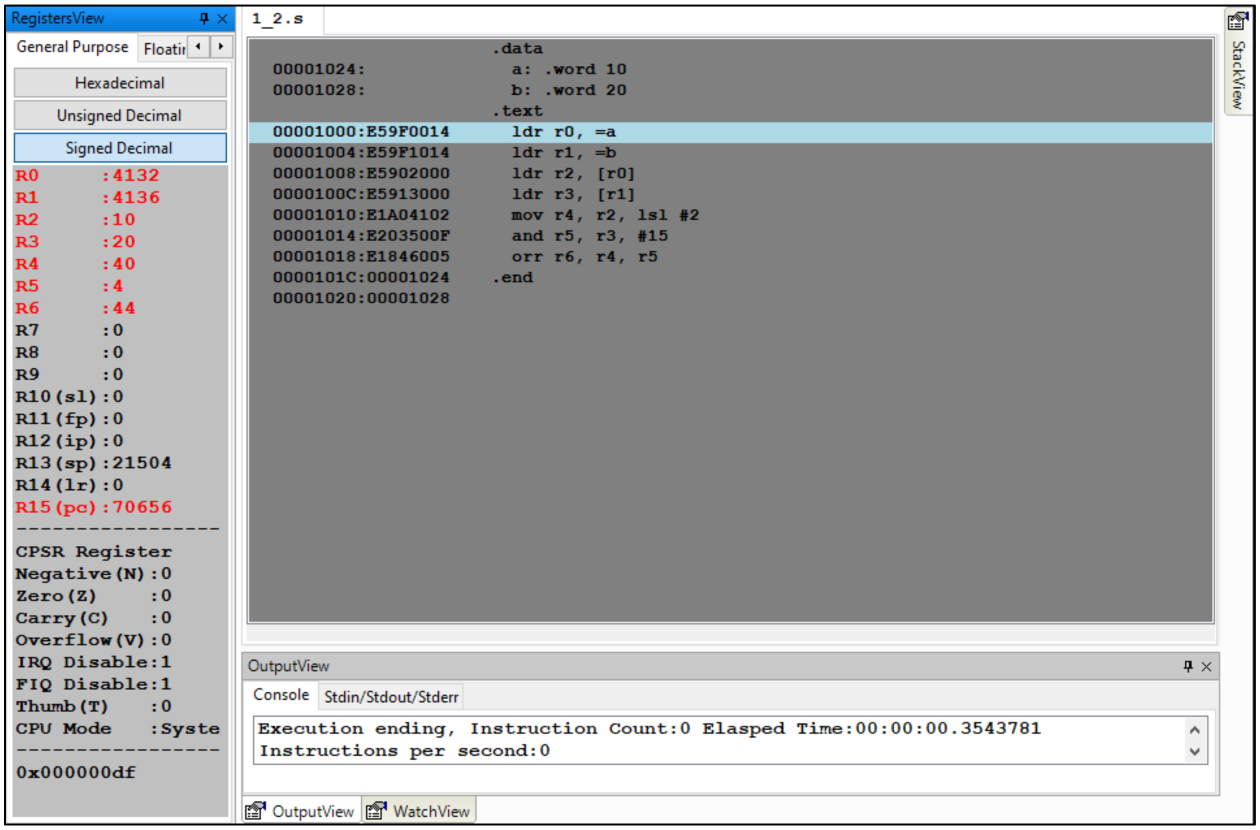
2) z = (a << 2) | (b & 15)

ARM Assembly Language Code



```
1_2.s
.data
  a: .word 10
  b: .word 20
.text
  ldr r0, =a
  ldr r1, =b
  ldr r2, [r0]
  ldr r3, [r1]
  mov r4, r2, lsl #2
  and r5, r3, #15
  orr r6, r4, r5
.end
```

Screenshot showing the value of a, b, z in the register window.



```
R0        :4132
R1        :4136
R2        :10
R3        :20
R4        :40
R5        :4
R6        :44
R7        :0
R8        :0
R9        :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
-----------------
CPSR Register
Negative(N):0
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :Syste
-----------------
0x000000df
```

```
                          .data
00001024:                   a: .word 10
00001028:                   b: .word 20
                          .text
00001000:E59F0014           ldr r0, =a
00001004:E59F1014           ldr r1, =b
00001008:E5902000           ldr r2, [r0]
0000100C:E5913000           ldr r3, [r1]
00001010:E1A04102           mov r4, r2, lsl #2
00001014:E203500F           and r5, r3, #15
00001018:E1846005           orr r6, r4, r5
0000101C:00001024         .end
00001020:00001028
```

```
Execution ending, Instruction Count:0 Elasped Time:00:00:00.3543781
Instructions per second:0
```

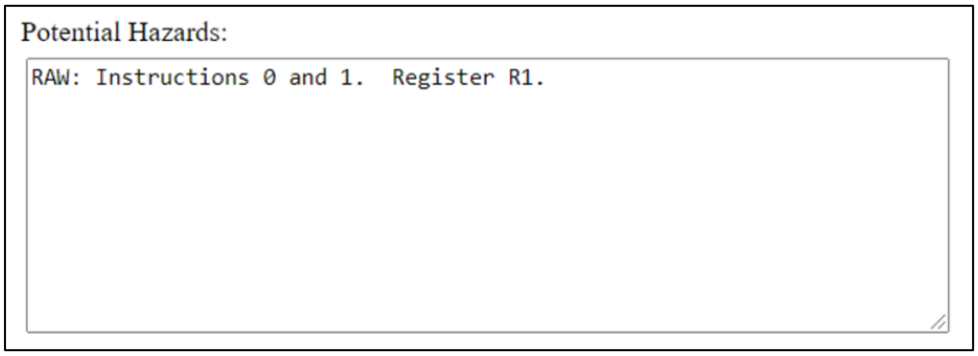
Program Number: ____2__

1) Consider the following instructions. Execute these instructions using simulator of 5 stage pipeline of MIPS architecture.

ADD R0, R1, R2
SUB   R3, R0, R4.

Observe the following and note down the results.

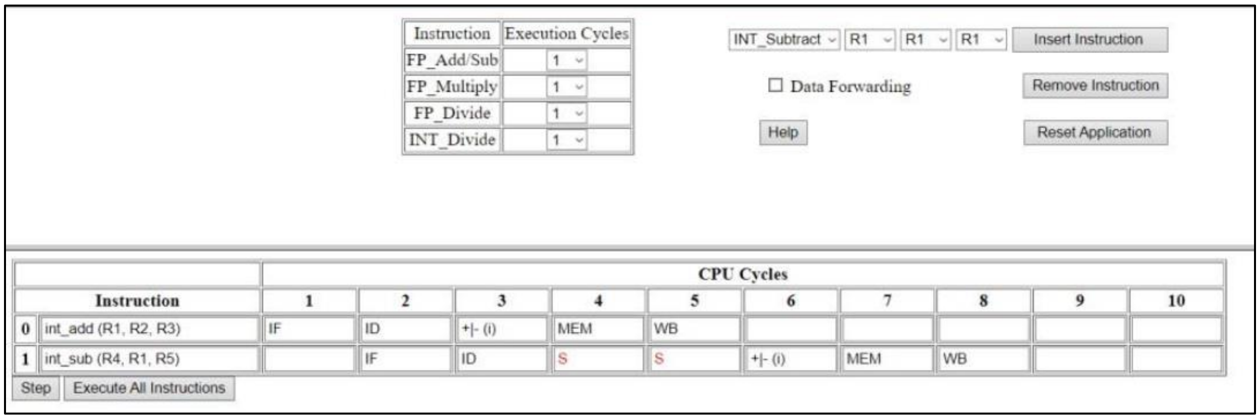
a) Check whether there is data dependency for the second instruction?



Potential Hazards:

```
RAW: Instructions 0 and 1.  Register R1.
```

b) If yes, then, how many stall states have been introduced?



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

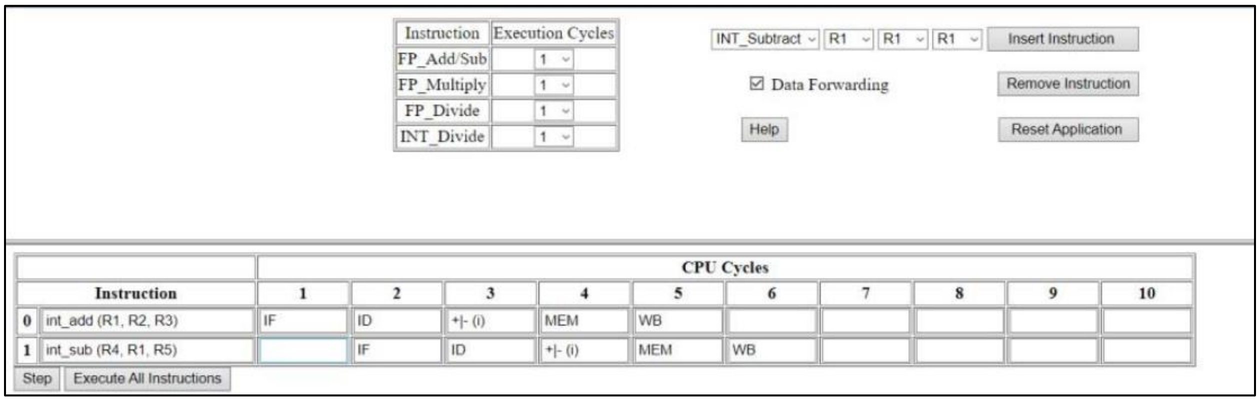
INT_Subtract R1 R1 R1

☐ Data Forwarding

		CPU Cycles									
	Instruction	1	2	3	4	5	6	7	8	9	10
0	int_add (R1, R2, R3)	IF	ID	+|- (i)	MEM	WB					
1	int_sub (R4, R1, R5)		IF	ID	S	S	+|- (i)	MEM	WB		

Without Data Forwarding, are 2 memory stalls.

c) If data forwarding is applied how many stall states have been reduced?



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

INT_Subtract R1 R1 R1

☑ Data Forwarding

		CPU Cycles									
	Instruction	1	2	3	4	5	6	7	8	9	10
0	int_add (R1, R2, R3)	IF	ID	+|- (i)	MEM	WB					
1	int_sub (R4, R1, R5)		IF	ID	+|- (i)	MEM	WB				

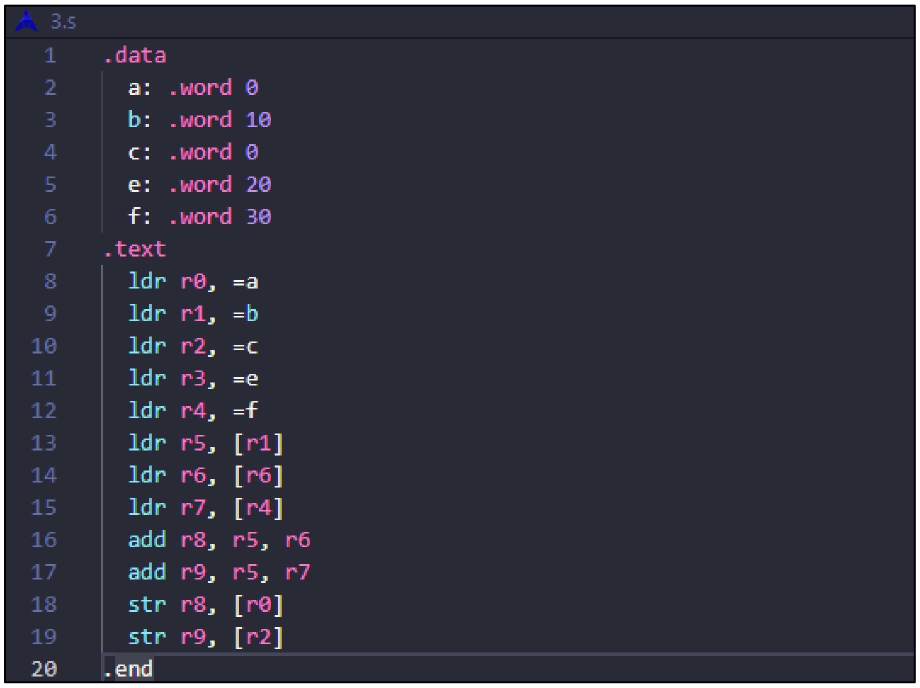
2 stalls have been reduced.

Program Number: ____3__

Consider the following code segment in C.

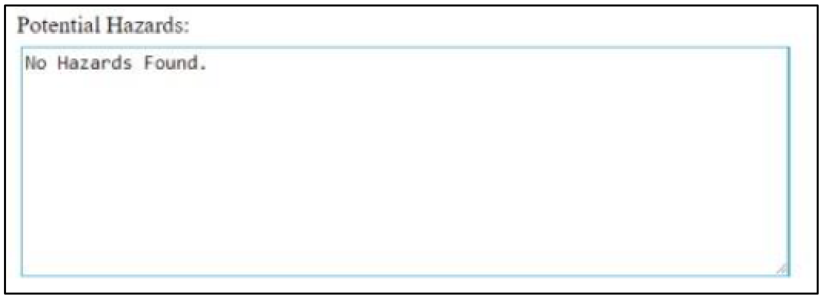
A = B + E;
C = B + F;

a) Write the code using MIPS 5 STAGE pipeline architecture.



```
.data
  a: .word 0
  b: .word 10
  c: .word 0
  e: .word 20
  f: .word 30
.text
  ldr r0, =a
  ldr r1, =b
  ldr r2, =c
  ldr r3, =e
  ldr r4, =f
  ldr r5, [r1]
  ldr r6, [r6]
  ldr r7, [r4]
  add r8, r5, r6
  add r9, r5, r7
  str r8, [r0]
  str r9, [r2]
.end
```

b) Find the hazards.

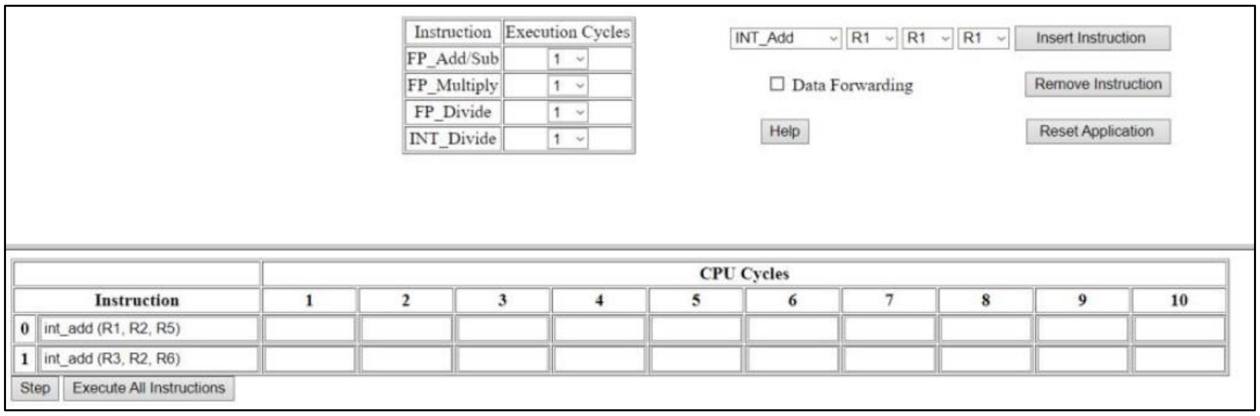


Potential Hazards:

```
No Hazards Found.
```

No hazards found.

c) Reorder the instructions to avoid pipeline stalls.



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

INT_Add R1 R1 R1 Insert Instruction

☐ Data Forwarding

		CPU Cycles									
Instruction		1	2	3	4	5	6	7	8	9	10
0	int_add (R1, R2, R5)										
1	int_add (R3, R2, R6)										

The instructions are already in order.

Program Number: ___4__

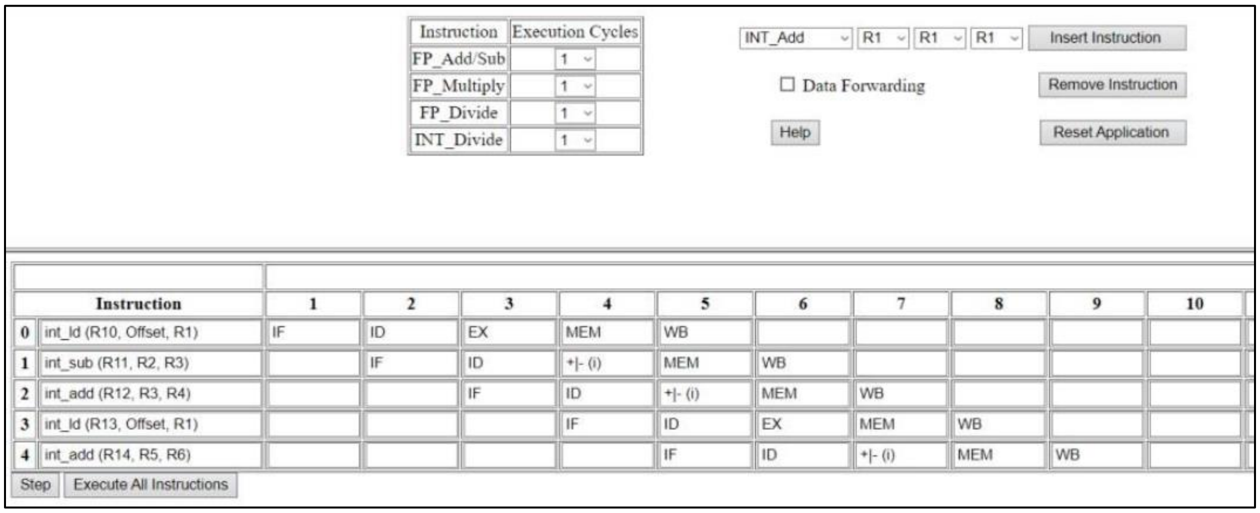
Using MIPS 5 stage pipeline architecture, execute the following instructions and avoid stall states if any.

```
LW     $10, 20($1)
SUB    $11, $2, $3
ADD    $12, $3, $4
LW     $13, 24($1)
ADD    $14, $5, $6
```

a) Related Screenshot with stalls

No stalls observed.

b) Related Screenshot without stalls



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

INT_Add R1 R1 R1 Insert Instruction

☐ Data Forwarding

Remove Instruction

Help

Reset Application

	Instruction	1	2	3	4	5	6	7	8	9	10
0	int_ld (R10, Offset, R1)	IF	ID	EX	MEM	WB					
1	int_sub (R11, R2, R3)		IF	ID	+|- (i)	MEM	WB				
2	int_add (R12, R3, R4)			IF	ID	+|- (i)	MEM	WB			
3	int_ld (R13, Offset, R1)				IF	ID	EX	MEM	WB		
4	int_add (R14, R5, R6)					IF	ID	+|- (i)	MEM	WB	

Program Number: ___5__

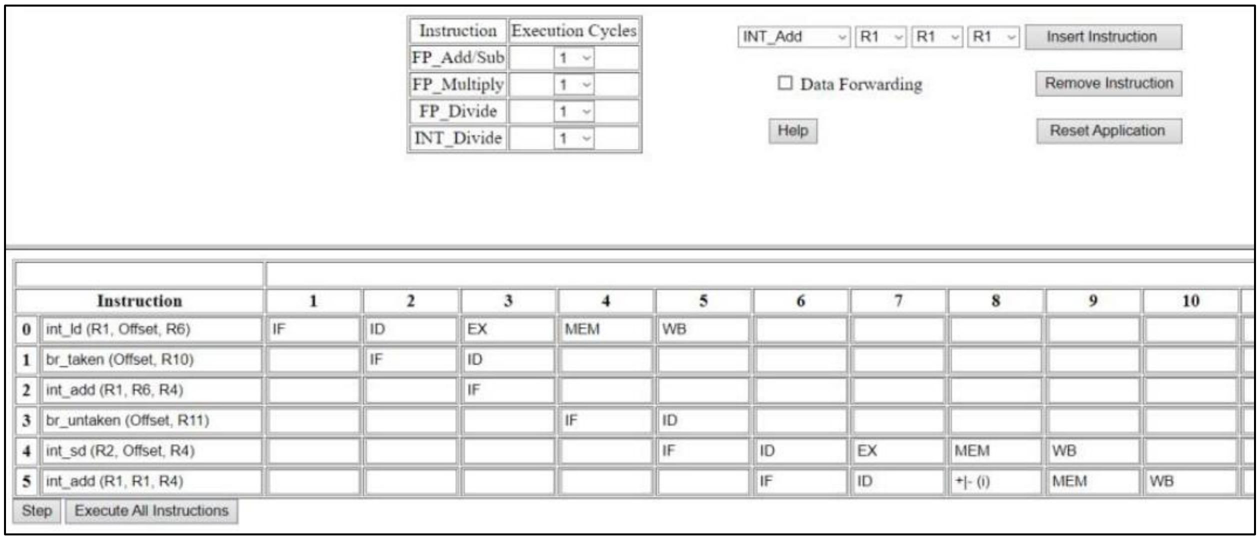
This exercise is to understand the relationship between delay slots, control hazards and branch execution in a 5 stage MIPS pipelined processor.

```
Label 1:   LW    $1, 40($6)
     BEQ   $2, $3, Label2   : branch taken
     ADD   $1, $6, $4
Label2:    BEQ   $1, $2, Label1   : branch not taken
           SW    $2, 20($4)
           ADD   $1, $1, $4
```

Assume full data forwarding and predict- taken branch prediction.

Note the observations.

Related Screenshot



Instruction	Execution Cycles
FP_Add/Sub	1
FP_Multiply	1
FP_Divide	1
INT_Divide	1

INT_Add R1 R1 R1

☐ Data Forwarding

	Instruction	1	2	3	4	5	6	7	8	9	10
0	int_ld (R1, Offset, R6)	IF	ID	EX	MEM	WB					
1	br_taken (Offset, R10)		IF	ID							
2	int_add (R1, R6, R4)			IF							
3	br_untaken (Offset, R11)				IF	ID					
4	int_sd (R2, Offset, R4)					IF	ID	EX	MEM	WB	
5	int_add (R1, R1, R4)						IF	ID	+|- (i)	MEM	WB

Disclaimer:

- The programs and output submitted is duly written, verified and executed by me.
- I have not copied from any of my peers nor from the external resource such as internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature: *Atul Anurag*
Name: Atul Anurag
SRN: PES2UG19CS075
Section: B
Date: 21-04-2021