

# **Microprocessor and Computer Architecture Laboratory**

**UE19CS256**

**4th Semester, Academic Year 2020-21**

Date:10/04/2021

Name: PRIYA MOHATA	SRN:PES2UG19CS301	Section:E
--------------------	-------------------	-----------

Week#10

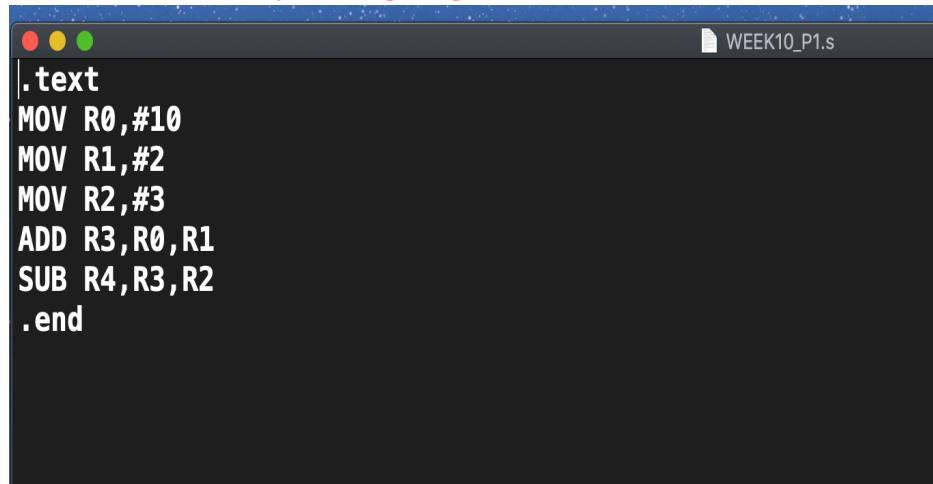
Program Number:1

**Given a C- Code convert it in its equivalent ARM Code.**

**These programs need to be executed on ARMSIM Simulator**

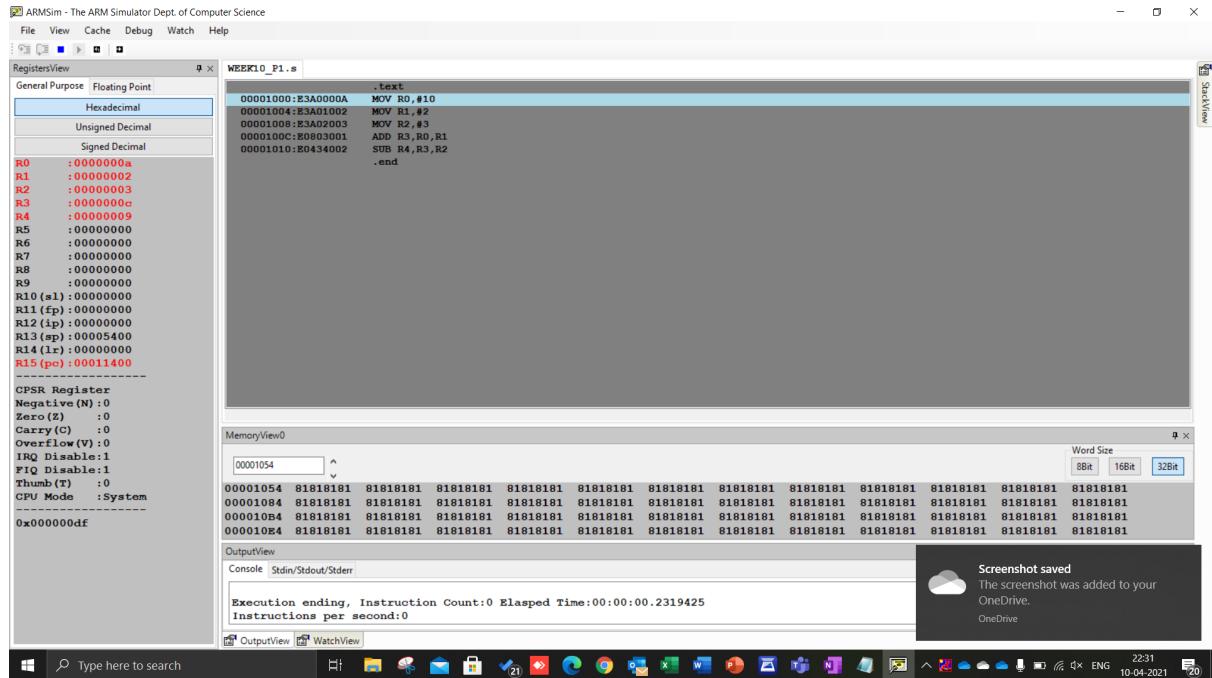
- 1)  $x = (a + b) - c;$

**ARM Assembly Language Code**



```
.text
MOV R0,#10
MOV R1,#2
MOV R2,#3
ADD R3,R0,R1
SUB R4,R3,R2
.end
```

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



$$2) z = (a << 2) | (b \& 15);$$

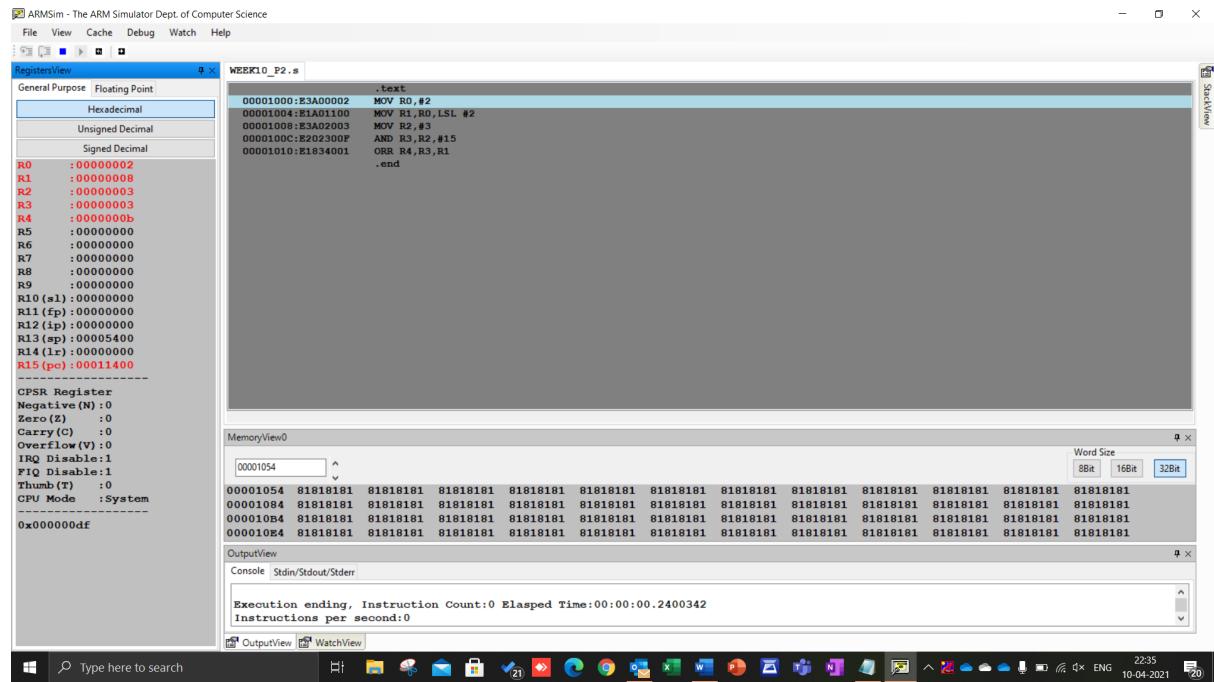
## ARM Assembly Language Code

```

.text
MOV R0,#2
MOV R1,R0,LSL #2
MOV R2,#3
AND R3,R2,#15
ORR R4,R3,R1
.end

```

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



Week# 10

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?

Yes there is RAW data dependency

### Related Screenshot

The screenshot shows a browser window with the title "MIPS Five Stage Pipeline". The URL is "ecs.umass.edu/ece/koren/architecture/windlx/main.html". The application interface includes a table for selecting instructions and their execution cycles, and a main area for viewing the execution timeline.

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

Buttons include "Insert Instruction", "Data Forwarding", "Remove Instruction", "Help", and "Reset Application".

**CPU Cycles**

Instruction	1	2	3	4	5	6	7	8	9	10
0   int_add (R1, R2, R3)										
1   int_sub (R4, R1, R5)										

**Potential Hazards:**  
RAW: Instructions 0 and 1. Register R1.

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

There are two stall states

### Related Screenshot

The screenshot shows a MIPS five-stage pipeline simulation. At the top, there's a toolbar with various tabs and icons. Below the toolbar, a table lists instruction types and their execution cycles. To the right of the table are buttons for inserting, removing, and resetting instructions, along with help and data forwarding checkboxes.

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

Below the table, a CPU cycle timeline shows two instructions being processed across 10 cycles. Instruction 0 (int.add) has stall states at cycles 4 and 5. Instruction 1 (int.sub) also has stall states at cycles 4 and 5. A legend indicates stall states with 'S'.

Potential Hazards:  
RAW: Instructions 0 and 1. Register R1.

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

There are no stall cycles when data forwarding was applied.

### Related Screenshot

Screenshot of a web-based MIPS Five Stage Pipeline simulation interface.

The browser tabs include: (1) Calendar | Microsoft Teams, Profile | Dashboard, PES University - Education, Calendar of Events | PES U..., Watch 'UE19CS256\_050421', MIPS Five Stage Pipeline, and several others like Apps, Gmail, YouTube, Maps, Equations Practice, Laxmi Puja, How to Create Us..., and Reading List.

The main interface shows a table for selecting instructions and their execution cycles:

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

Buttons for pipeline control include: INT\_Subtract, R1, R1, R1, Insert Instruction, Data Forwarding (checked), Remove Instruction, Help, and Reset Application.

---

Instruction	CPU Cycles									
	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				

Buttons at the bottom left: Step, Execute All Instructions.

Potential Hazards:  
No Hazards Found.

Week# 10

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.

### ARM Assembly Language Code

```
WEEK10_P3.s
.text
MOV R0,#2
MOV R1,#3
MOV R2,#4
ADD R3,R0,R1
ADD R4,R0,R2
.end
```

- b) Find the hazards;

### Related Screenshot

No hazards

The screenshot shows the MIPS Five Stage Pipeline application interface. At the top, there is a code editor window containing the following MIPS assembly code:

```
.text
MOV R0,#2
MOV R1,#3
MOV R2,#4
ADD R3,R0,R1
ADD R4,R0,R2
.end
```

Below the code editor is a control panel with the following settings:

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

Control buttons include: INT\_Add dropdown, R1 dropdown, Insert Instruction, Data Forwarding checkbox, Remove Instruction, Help button, and Reset Application button.

Below the control panel is a table showing CPU Cycles for two instructions (0 and 1) over 10 stages. Both instructions are shown as "int\_add" operations.

	CPU Cycles									
Instruction	1	2	3	4	5	6	7	8	9	10
0 int_add (R1, R2, R3)										
1 int_add (R4, R2, R5)										

Buttons at the bottom left are Step and Execute All Instructions.

Potential Hazards section:

No Hazards Found.



c) Reorder the instructions to avoid pipeline stalls.

## Related Screenshot

No reordering required

The screenshot shows a MIPS Five Stage Pipeline application running in a web browser. The interface includes a control panel with dropdown menus for selecting instructions (INT\_Add, FP\_Multiply, FP\_Divide, INT\_Divide) and stages (R1, R2, R3, R4, R5), and buttons for Insert Instruction, Data Forwarding, Remove Instruction, Help, and Reset Application. Below the control panel is a table showing the execution of two integer addition instructions (int\_add) across 10 CPU cycles. Both instructions progress through the pipeline stages sequentially without any visible stalls or hazards. A message at the bottom indicates "No Hazards Found".

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

Potential Hazards:  
No Hazards Found.

Week# 10

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

### c) Related Screenshot without stalls

The screenshot shows a MIPS Five Stage Pipeline simulation interface. At the top, there's a menu bar with Chrome, File, Edit, View, History, Bookmarks, People, Tab, Window, Help. Below the menu is a toolbar with various icons. The main area has a table for selecting instructions and their execution cycles, and a control panel with buttons for Data Forwarding, Insert Instruction, Remove Instruction, Help, and Reset Application. Below this is a large table showing the execution of five instructions over 18 CPU cycles. The instructions are:

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

The execution table shows the following pattern of stages across 18 cycles:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0 int_Id (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_Id (R9, Offset, R1)				IF	ID	EX	MEM	WB										
4 int_add (R8, R5, R6)					IF	ID	+/- (I)	MEM	WB									

At the bottom left, there are buttons for Step and Execute All Instructions. A section labeled "Potential Hazards:" with the message "No Hazards Found." is also present.

Week# 10

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

The screenshot shows a MIPS Five Stage Pipeline simulation running in a Chrome browser. The interface includes a header bar with tabs, a toolbar with various icons, and a main content area divided into two sections: a configuration panel at the top and a pipeline timeline below it.

**Configuration Panel:**

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

Buttons and checkboxes in this panel include: INT\_Add, R1 dropdowns, Insert Instruction, Data Forwarding checkbox, Remove Instruction, Help button, and Reset Application button.

**Pipeline Timeline:**

	CPU Cycles																	
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0 int_Id (R1, Offset, R6)	IF	ID	EX	MEM	WB													
1 br_taken (Offset, R2)		IF	ID															
2 int_add (R1, R6, R4)			IF															
3 br_untaken (Offset, R1)				IF	ID													
4 int_sd (R2, Offset, R4)					IF	ID	EX	MEM	WB									
5 int_add (R1, R1, R4)						IF	ID	+/- (i)	MEM	WB								

Buttons at the bottom of the timeline include: Step and Execute All Instructions.

**Potential Hazards:**

RAW: Instructions 2 and 3. Register R1.

The browser's address bar shows the URL: <https://ecs.umass.edu/ece/koren/architecture/windlx/main.html>.

### **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: PRIYA MOHATA

Name: PRIYA MOHATA

SRN: PES2UG19CS301

Section: E

Date: 10/04/2021