

# **Microprocessor and Computer Architecture Laboratory**

## **UE19CS256**

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

Date:02/01/2021

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

**Week#2.**

**ProgramNumber:1**

**Title of the Program :** Based on the value of the number in R0, Write an ALP to store 1 in R1 if R0 is zero, Store 2 in R1 if R0 is positive, Store 3 in R1 if R0 is negative.

I. ARM Assembly Code for each program

```

.text
MOV R0,#0      TEST CASE 1
CMP R0,#0
BEQ label
BMI label2
MOV R1,#2
B exit
label:MOV R1,#1
B exit
label2:MOV R1,#3
exit:SWI 0x011
.end

```

```
.text  
MOV R0,#10      TEST CASE 2  
CMP R0,#0  
BEQ label  
BMI label2  
MOV R1,#2  
B exit  
label:MOV R1,#1  
B exit  
label2:MOV R1,#3  
exit:SWI 0x011  
.end
```

```
.text      TEST CASE 3  
MOV R0,#-4  
CMP R0,#0  
BEQ label  
BMI label2  
MOV R1,#2  
B exit  
label:MOV R1,#1  
B exit  
label2:MOV R1,#3  
exit:SWI 0x011  
.end
```

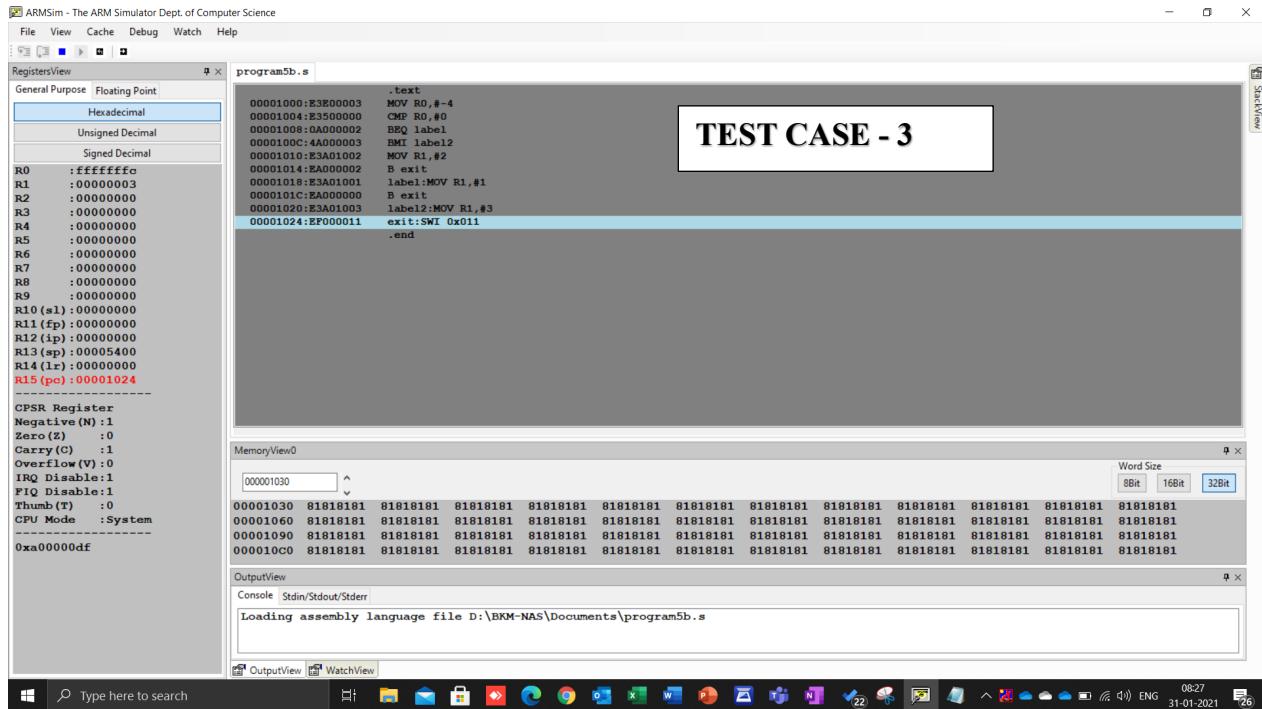
## II. Output Screen Shot

The screenshot shows the ARMSim interface with the following details:

- Registers View:** Shows the state of general purpose registers (R0-R15) and CPSR. R15 (pc) is highlighted in red as **00001024**.
- Memory View 0:** Displays memory starting at address 000001030, showing a repeating pattern of bytes 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81, 81.
- Output View:** Shows the console output for loading the assembly file.
- Text Editor:** The assembly code for **program5.s** is displayed, including labels like **label1**, **label2**, and **exit**, and instructions like **MOV**, **CMP**, **BREQ**, **BMI**, **MOV R1, #2**, **B exit**, **label:MOV R1, #1**, **label2:MOV R1, #3**, and **exit:SWI 0x011**.
- TEST CASE - 1:** A large text box labeled **TEST CASE - 1** is overlaid on the memory dump area.

The screenshot shows the ARM Simulator interface with the following details:

- Registers View:** Shows the ARM register state. R0-R15 and CPSR are initialized to zero, except for R15 (pc) which is set to 00001024.
- Memory View 0:** Displays the memory dump of the program. The memory starts with 000001000 (hex) followed by a series of 01818181 bytes.
- Output View:** Shows the console output, including the loading of the assembly language file.
- Assembly View:** The assembly code for program5a.s is displayed in the text editor. It includes instructions like MOV, CMP, BEQ, BNE, and SWI, along with labels and exits.



### III. Output table for each program

CASES	REGISTER	OPERATION	VALUE
CASE 1	R0		0x00000000
	R1	After compare	1
CASE 2	R0		0x0000000A
	R1	After compare	2
CASE 3	R0		0xFFFFFFF0
	R1	After compare	3

**Week#2.****Program Number:2**

**Title of the Program :** Write an ALP to compare the value of R0 and R1, add if R0 = R1, else subtract.

I. ARM Assembly Code for each program

```
.text          TEST CASE -  
MOV R0,#5      1  
MOV R1,#6  
CMP R0,R1  
BEQ label  
SUB R2,R1,R0  
B exit  
label:ADD R2,R0,R1  
exit:SWI 0x011  
.end
```

```
.text          TEST CASE -  
MOV R0,#5      2  
MOV R1,#5  
CMP R0,R1  
BEQ label  
SUB R2,R1,R0  
B exit  
label:ADD R2,R0,R1  
exit:SWI 0x011  
.end
```

## II.Output Screen Shot

ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00000005 R1 :00000006 R2 :00000001 R3 :00000000 R4 :00000000 R5 :00000000 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :0000101c

CPSR Register Negative(N) :1 Zero(Z) :0 Cx (C) :0 Overflow(V) :0 IRQ Disable:1 FIQ Disable:1 Thumb(T) :0 CPU Mode :System

0x800000df

program6.s

text

00001000: E3A00005 MOV R0, #5  
00001004: E3A00006 MOV R1, #6  
00001008: E1500001 CMP R0, R1  
0000100C: E0000000 BEQ label  
00001010: E9412000 SBC R2, R1, R0  
00001014: EA000000 B exit  
00001018: E90802001 label: ADD R2, R0, R1  
0000101C: EF000001 exit: SWI 0x011  
.end

TEST CASE - 1

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program6.s

OutputView WatchView

Type here to search

ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00000005 R1 :00000005 R2 :0000000a R3 :00000000 R4 :00000000 R5 :00000000 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :0000101c

CPSR Register Negative(N) :0 Zero(Z) :1 Cx (C) :1 Overflow(V) :0 IRQ Disable:1 FIQ Disable:1 Thumb(T) :0 CPU Mode :System

0x600000df

program6.s

text

00001000: E3A00005 MOV R0, #5  
00001004: E3A00005 MOV R1, #5  
00001008: E1500001 CMP R0, R1  
0000100C: E0000000 BEQ label  
00001010: E9412000 SBC R2, R1, R0  
00001014: EA000000 B exit  
00001018: E90802001 label: ADD R2, R0, R1  
0000101C: EF000001 exit: SWI 0x011  
.end

TEST CASE - 2

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program6.s

OutputView WatchView

### III.OUTPUT TABLE

CASES	
CASE 1	$R1=0x06, R0=0x05$ $R2=R1-R0=0x01$
CASE 2	$R1=0x05, R0=0x05$ $R2=R1+R0=0x0A$

**Week#2.****Program Number:3**

**Title of the Program :** Write an ALP to find the factorial of a number stored in R0. Store the value in R1 (without using LDR and STR instructions). Use only registers

I. ARM Assembly Code for each program

```
.text
MOV R0,#5      TEST CASE
MOV R1,#5      -1
LOOP:SUB R0,R0,#1
MUL R2,R1,R0
MOV R1,R2
CMP R0,#1
BEQ EXIT
B LOOP
EXIT:SWI 0x11
.end
```

```
.text
MOV R0,#6      TEST CASE
MOV R1,#6      -2
LOOP:SUB R0,R0,#1
MUL R2,R1,R0
MOV R1,R2
CMP R0,#1
BEQ EXIT
B LOOP
EXIT:SWI 0x11
.end
```

**II. OUTPUT SCREENSHOTS**

ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView **program7.s**

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00000000 R1 :00000078 R2 :00000078 R3 :00000000 R4 :00000000 R5 :00000000 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 **R15 (pc) :00001020**

CPSR Register

Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1 Thumb (T) :0 CPU Mode :System

0x600000df

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program7.s

OutputView WatchView

**TEST CASE – 1**

```
.text
00001000:E3A00005 MOV R0, #5
00001004:E3A01005 MOV R1, #5
00001008:E2400001 LOOP: SUB R0, R0, #1
0000100C:E0020091 MUL R2, R1, R0
00001010:E3500001 MOV R1, R2
00001014:E3500001 CMP R0, #0
00001018:0A000000 BNE EXIT
0000101C:E0FFFF9 B LOOP
00001020:EF000011 EXIT: SWI 0x11
.end
```

ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView **program7.s**

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :1 R1 :120 R2 :120 R3 :0 R4 :0 R5 :0 R6 :0 R7 :0 R8 :0 R9 :0 R10 (s1) :0 R11 (fp) :0 R12 (ip) :0 R13 (sp) :21504 R14 (lr) :0 **R15 (pc) :4128**

CPSR Register

Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1 Thumb (T) :0 CPU Mode :System

0x600000df

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program7.s

OutputView WatchView

**TEST CASE – 1 (Decimal)**

```
.text
00001000:E3A00005 MOV R0, #5
00001004:E3A01005 MOV R1, #5
00001008:E2400001 LOOP: SUB R0, R0, #1
0000100C:E0020091 MUL R2, R1, R0
00001010:E3500001 MOV R1, R2
00001014:E3500001 CMP R0, #0
00001018:0A000000 BNE EXIT
0000101C:E0FFFF9 B LOOP
00001020:EF000011 EXIT: SWI 0x11
.end
```

ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView Floating Point

General Purpose Hexadecimal Unsigned Decimal Signed Decimal

R0 :00000000 R1 :000000240 R2 :000000240 R3 :000000000 R4 :00000000 R5 :00000000 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :00001020

CPSR Register Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1 Thumb (T) :0 CPU Mode :System

0x600000df

program7.a.s

```

.text
00001000:E3A00006 MOV R0, #6
00001004:E3A01006 MOV R1, #6
00001008:E2400001 LOOP: SUB R0, R0, #1
0000100C:E0200091 MUL R2, R1, R0
00001010:E3500001 MOV R1, R2
00001014:E3500001 CMP R0, #0
00001018:0A000000 BNE EXIT
0000101C:E0FFF9 B LOOP
00001020:EF000011 EXIT: SWI 0x11
.end

```

TEST CASE – 2

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program7.a.s

OutputView WatchView

ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView Floating Point

General Purpose Hexadecimal Unsigned Decimal Signed Decimal

R0 :1 R1 :720 R2 :720 R3 :0 R4 :0 R5 :0 R6 :0 R7 :0 R8 :0 R9 :0 R10 (s1) :0 R11 (fp) :0 R12 (ip) :0 R13 (sp) :21504 R14 (lr) :0 R15 (pc) :4128

CPSR Register Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1 Thumb (T) :0 CPU Mode :System

0x600000df

program7.a.s

```

.text
00001000:E3A00006 MOV R0, #6
00001004:E3A01006 MOV R1, #6
00001008:E2400001 LOOP: SUB R0, R0, #1
0000100C:E0200091 MUL R2, R1, R0
00001010:E3500001 MOV R1, R2
00001014:E3500001 CMP R0, #0
00001018:0A000000 BNE EXIT
0000101C:E0FFF9 B LOOP
00001020:EF000011 EXIT: SWI 0x11
.end

```

TEST CASE – 2 (Decimal)

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program7.a.s

OutputView WatchView

### III.OUTPUT TABLE

ITERATIONS	
1 <sup>st</sup> Iteration	R1=0x05 , R0=0x04 R2=0X14=Decimal 20
2 <sup>nd</sup> Iteration	R1=0x14 , R0=0X03 R2=0x3C=Decimal 60
3rd Iteration	R1=0x3C ,R0=0X02 R2=0x78 =Decimal 120
4 <sup>th</sup> Iteration	R1=0x78 , R0=0X01 R2=0x78 =Decimal 120

ITERATIONS	
1 <sup>st</sup> Iteration	R1=0x06 , R0=0x05 R2=0X1e=Decimal 30
2 <sup>nd</sup> Iteration	R1=0x1e , R0=0X04 R2=0x78=Decimal 120
3rd Iteration	R1=0x78 ,R0=0X03 R2=0x168 =Decimal 360
4 <sup>th</sup> Iteration	R1=0x168 , R0=0X02 R2=0x2d0 =Decimal 720
5 <sup>th</sup> Iteration	R1=0x2d0 , R0=0X01 R2=0x2d0 =Decimal 720

**Week#2.****Program Number:4a**

**Title of the Program :** Write an ALP to add two 32 bit numbers loaded from memory and store the result in memory.

I. ARM Assembly Code for each program

```
.data
A: .WORD 10
B: .WORD 20
C:.WORD
.text
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDR R3,[R1]
LDR R4,[R0]
ADD R5,R3,R4
STR R5,[R2]
.end
```

TEST CASE  
- 1

```
.data
A: .WORD 10
B: .WORD 25
C:.WORD
.text
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDR R3,[R1]
LDR R4,[R0]
ADD R5,R3,R4
STR R5,[R2]
.end
```

TEST CASE  
- 2

## II. Output Screenshots

ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00001028 R1 :0000102c R2 :00001030 R3 :00000014 R4 :000000a R5 :0000001e R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :0000101c

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 :System

0x0000000df

program8a.s

```

.data
00001028:    A: .WORD 10
0000102c:    B: .WORD 20
00001030:    C: .WORD
00001031:    .
00001000:E59F0014 LDR R0,-A
00001004:E59F1014 LDR R1,-B
00001008:E59F2014 LDR R2,-C
0000100C:E5913000 LDR R3,[R1]
00001010:E5904000 LDR R4,[R0]
00001014:E0835004 ADD R5,R3,R4
00001018:E5825000 STR R5,[R2]
.end

```

TEST CASE – 1

MemoryView0

Word Size: 8Bit 16Bit 32Bit

00001030 0000001E 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

00001060 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

00001090 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

000010C0 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program8a.s

OutputView WatchView

ARMsim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00001028 R1 :0000102c R2 :00001030 R3 :00000019 R4 :000000a R5 :00000023 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :0000101c

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 :System

0x0000000df

program8a.s

```

.data
00001028:    A: .WORD 10
0000102c:    B: .WORD 25
00001030:    C: .WORD
00001031:    .
00001000:E59F0014 LDR R0,-A
00001004:E59F1014 LDR R1,-B
00001008:E59F2014 LDR R2,-C
0000100C:E5913000 LDR R3,[R1]
00001010:E5904000 LDR R4,[R0]
00001014:E0835004 ADD R5,R3,R4
00001018:E5825000 STR R5,[R2]
.end

```

TEST CASE – 2

MemoryView0

Word Size: 8Bit 16Bit 32Bit

00001030 00000023 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 01818181

00001060 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

00001090 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

000010C0 81818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181 01818181

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program8a.s

OutputView WatchView

### III.Output Table

<b>A=0x0A, B=0x14</b>	
<b>TEST -1</b>	
R0	Address of A
R1	Address of B
R2	Address of C
R3	0x14=Decimal 20 =Content of Location B
R4	0x0A=Decimal 10= Content of Location A
R5	0x1E=Decimal 30
Location C	0x1E=Decimal 30

**Week#2.****Program Number:4b**

**Title of the Program :** Write an ALP to add two 16 bit numbers loaded from memory and store the result in memory.

I. ARM Assembly Code for each program

```
.data
A: .HWORD 10
B: .HWORD 20
C:.HWORD
.TEXT
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDRH R3,[R1]
LDRH R4,[R0]
ADD R5,R3,R4
STRH R5,[R2]
.END
```

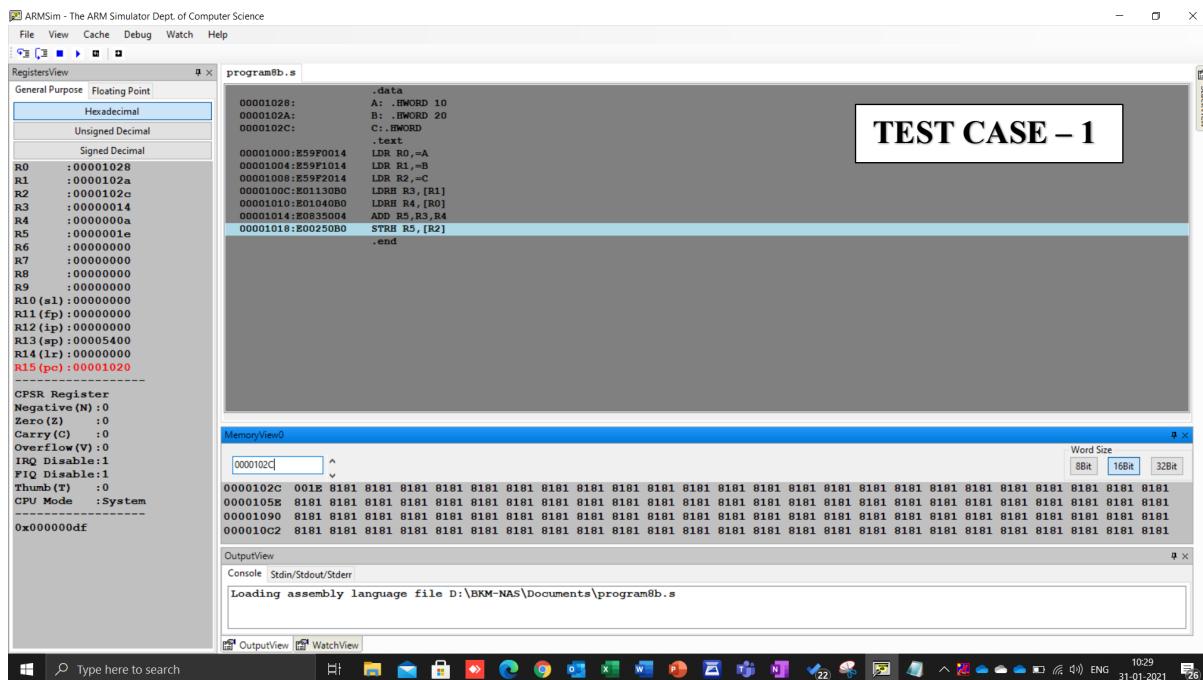
TEST CASE – 1

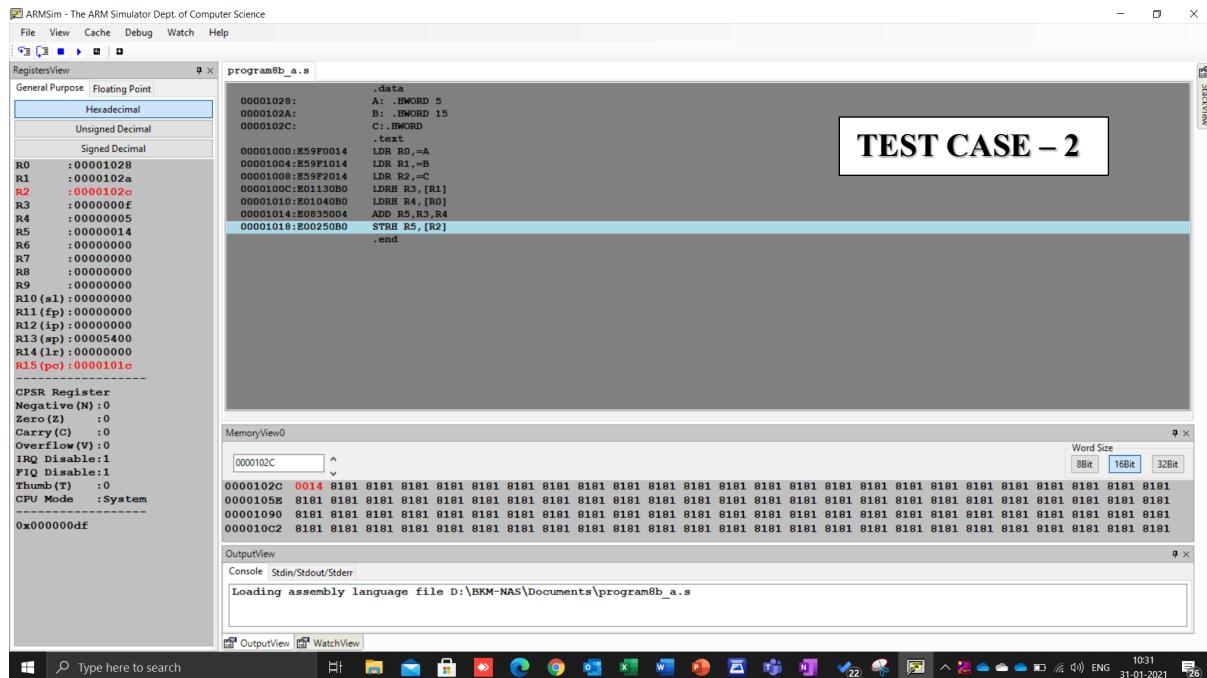
```

.data
A: .HWORD 5
B: .HWORD 15
C:.HWORD
.text
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDRH R3,[R1]
LDRH R4,[R0]
ADD R5,R3,R4
STRH R5,[R2]
.end

```

## II. OUTPUT SCREENSHOTS





### III. Output Table

TEST CASE 1	A=0x0A, B=0x14
R0	Address of A (0x1028)
R1	Address of B (0x102A)
R2	Address of C (0x102C)
R3	0x14=Decimal 20=Content of Location B
R4	0x0A=Decimal 10= Content of Location A
R5	0x1E=Decimal 30
Location C(0x102C)	0x1E=Decimal 30
TEST CASE 2	A=0x05, B=0x0f

<b>R0</b>	<b>Address of A (0x1028)</b>
<b>R1</b>	<b>Address of B (0x102A)</b>
<b>R2</b>	<b>Address of C (0x102C)</b>
<b>R3</b>	<b>0x0f=Decimal 15=Content of Location B</b>
<b>R4</b>	<b>0x05=Decimal 05= Content of Location A</b>
<b>R5</b>	<b>0x14=Decimal 20</b>
<b>Location C(0x102C)</b>	<b>0x14=Decimal 20</b>

**Week#2.****Program Number:5a**

**Title of the Program :** Write an ALP to find GCD of two numbers (without using LDR and STR instructions). Both numbers are in registers. Use only registers

I. ARM Assembly Code for each program

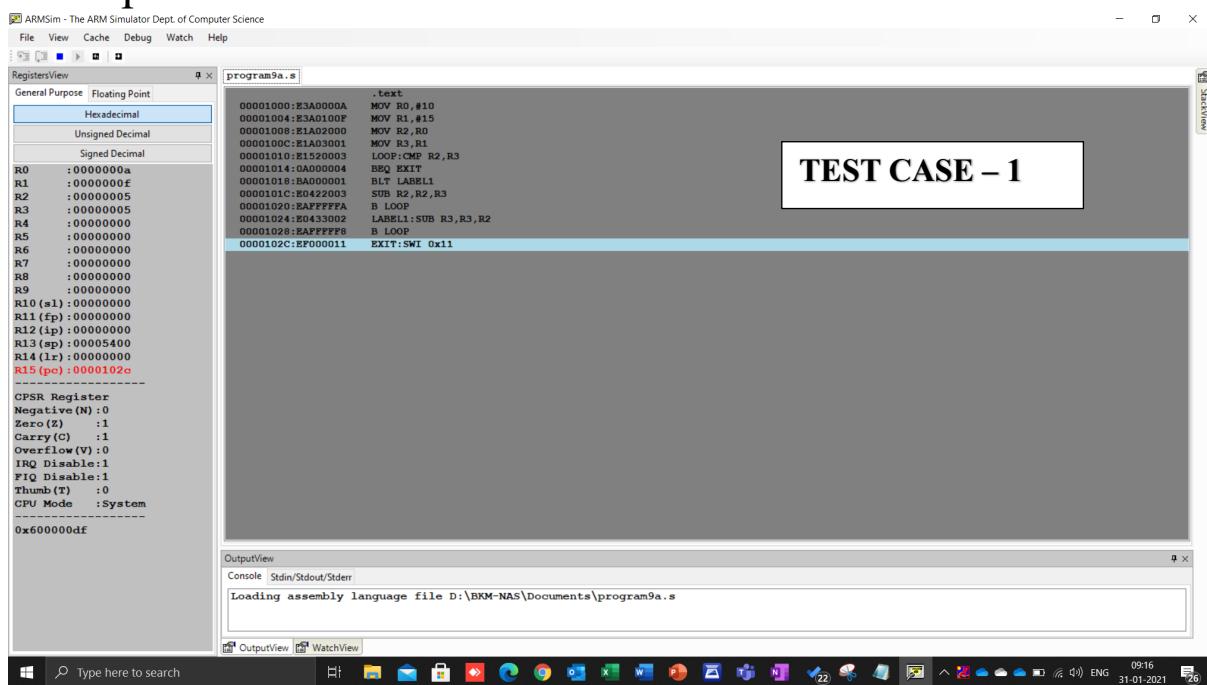
```
.text
MOV R0,#10      TEST CASE
MOV R1,#15
MOV R2,R0
MOV R3,R1
LOOP:CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1:SUB R3,R3,R2
B LOOP
EXIT:SWI 0x11
```

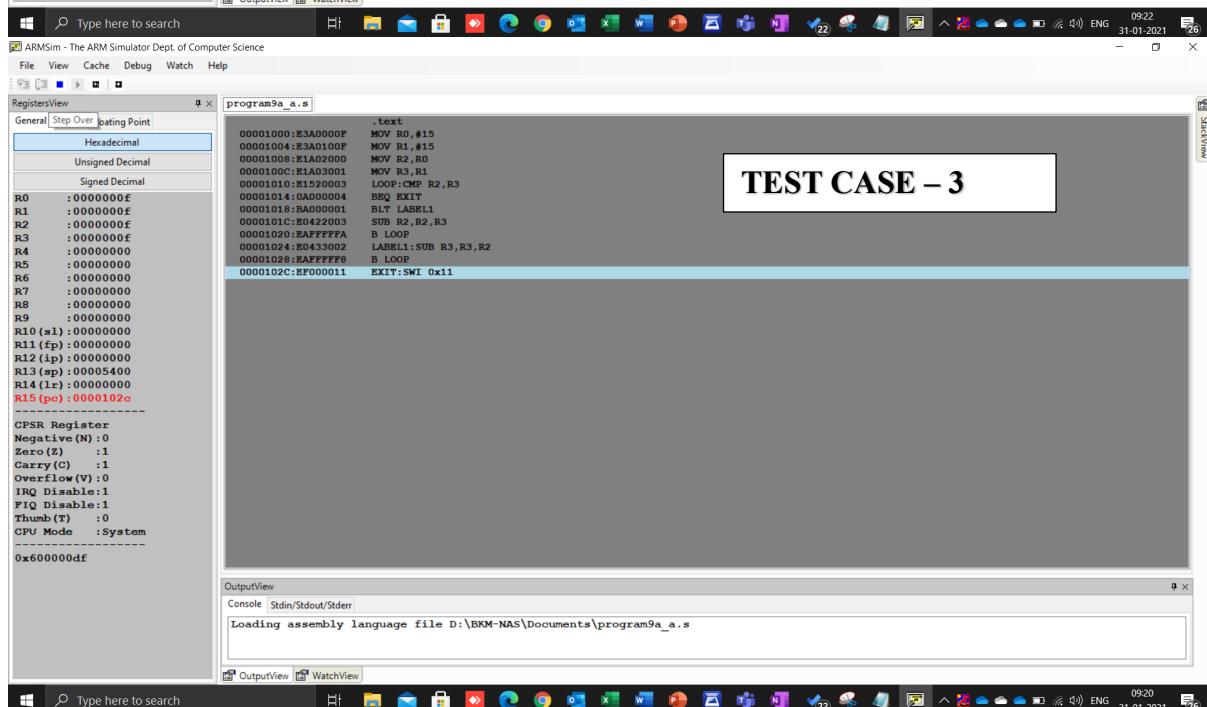
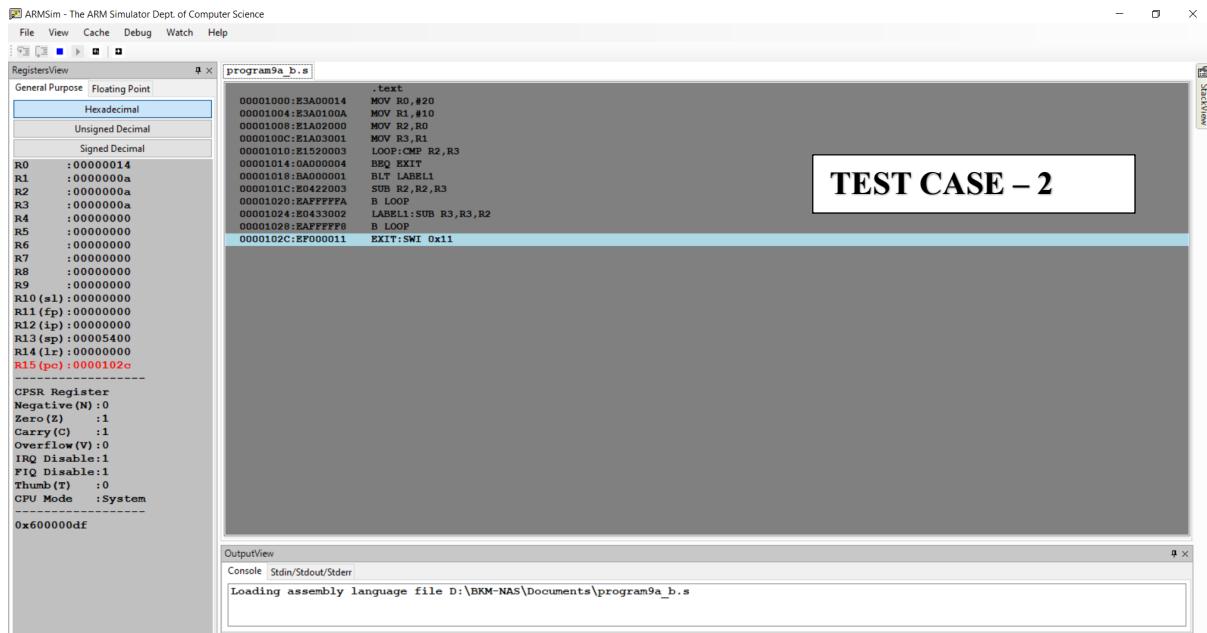
```
.text
MOV R0,#20      TEST CASE
MOV R1,#10
MOV R2,R0
MOV R3,R1
LOOP:CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1:SUB R3,R3,R2
B LOOP
EXIT:SWI 0x11
```

```
.text
MOV R0,#15
MOV R1,#15
MOV R2,R0
MOV R3,R1
LOOP: CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1: SUB R3,R3,R2
B LOOP
EXIT: SWI 0x11
```

---

## II. Output Screenshots





### III. Output Table

CASE 1	R0	0x0A
	R1	0x0F
1 <sup>st</sup> iteration	R2=0x0A , R3=0x0F R3=0x0F-0x0A=0x05	
2 <sup>nd</sup> iteration	R2=0x0A , R3=0x05 R2=0x0A-0x05=0x05	
3 <sup>rd</sup> iteration	R2=0x05 ,R3=0x05 GCD=5	
CASE 2	R0	0x14
	R1	0x0A
1 <sup>st</sup> iteration	R2=0x14 , R3=0x0A R2=0x14-0x0A=0x0A	
2 <sup>nd</sup> iteration	R2=0x0A , R3=0x0A GCD=0x0A (ie 10 in decimal)	
CASE 3	R0	0x0F
	R1	0x0F
1 <sup>st</sup> iteration	R2=0x0F , R3=0x0F GCD = 15 ( 0x0F)	

**Week#2.****Program Number:5b**

**Title of the Program :** Write an ALP to find the GCD of given numbers (both numbers in memory).Store result in memory

I.ARM Assembly Code for each program

```

-.data
A: .WORD 20
B: .WORD 4
C: .WORD
.text
LDR R0,=A
LDR R1,=B
LDR R5,=C
LDR R2,[R0]
LDR R3,[R1]
LOOP:CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1:SUB R3,R3,R2
B LOOP
EXIT:STR R3,[R5]
SWI 0x11
.end

```

TEST CASE  
- 1

```
.data
A: .WORD 10
B: .WORD 15
C: .WORD
.text
LDR R0,=A
LDR R1,=B
LDR R5,=C
LDR R2,[R0]
LDR R3,[R1]
LOOP: CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1: SUB R3,R3,R2
B LOOP
EXIT: STR R3,[R5]
SWI 0x11
.end
```

TEST CASE  
- 2

```

.data
A:.WORD 4
B:.WORD 4
C:.WORD
.text
LDR R0,=A
LDR R1,=B
LDR R5,=C
LDR R2,[R0]
LDR R3,[R1]
LOOP:CMP R2,R3
BEQ EXIT
BLT LABEL1
SUB R2,R2,R3
B LOOP
LABEL1:SUB R3,R3,R2
B LOOP
EXIT:STR R3,[R5]
SWI 0x11
.end

```

TEST CASE  
- 3

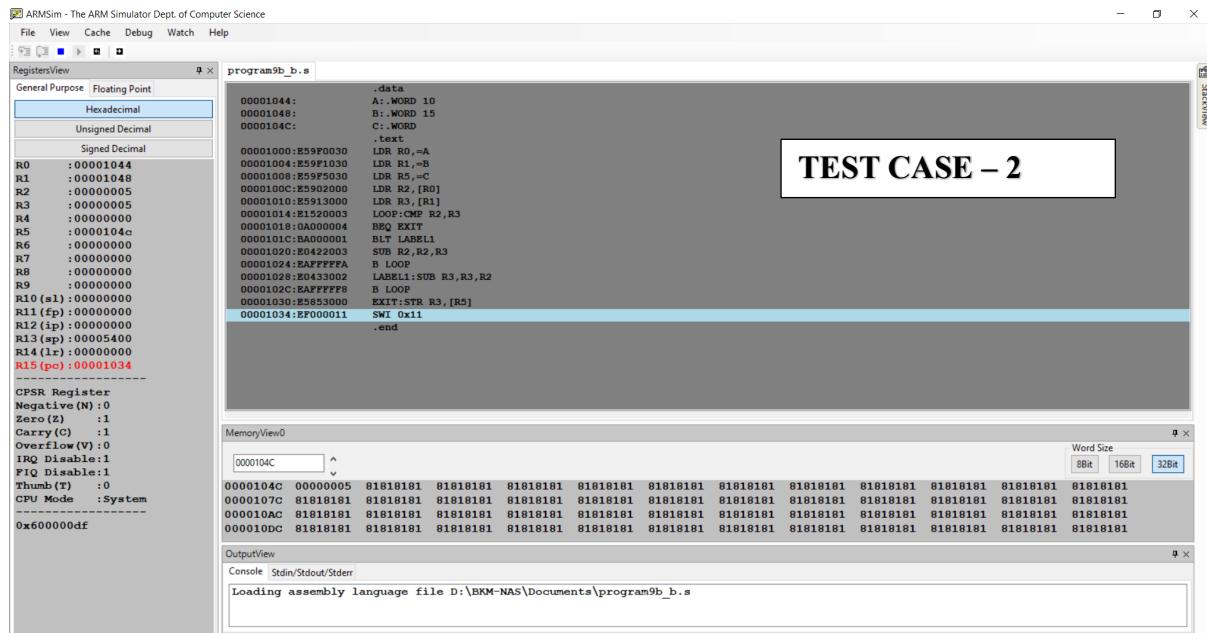
## II. Output Screenshots

The screenshot shows the ARMSim interface with the following windows:

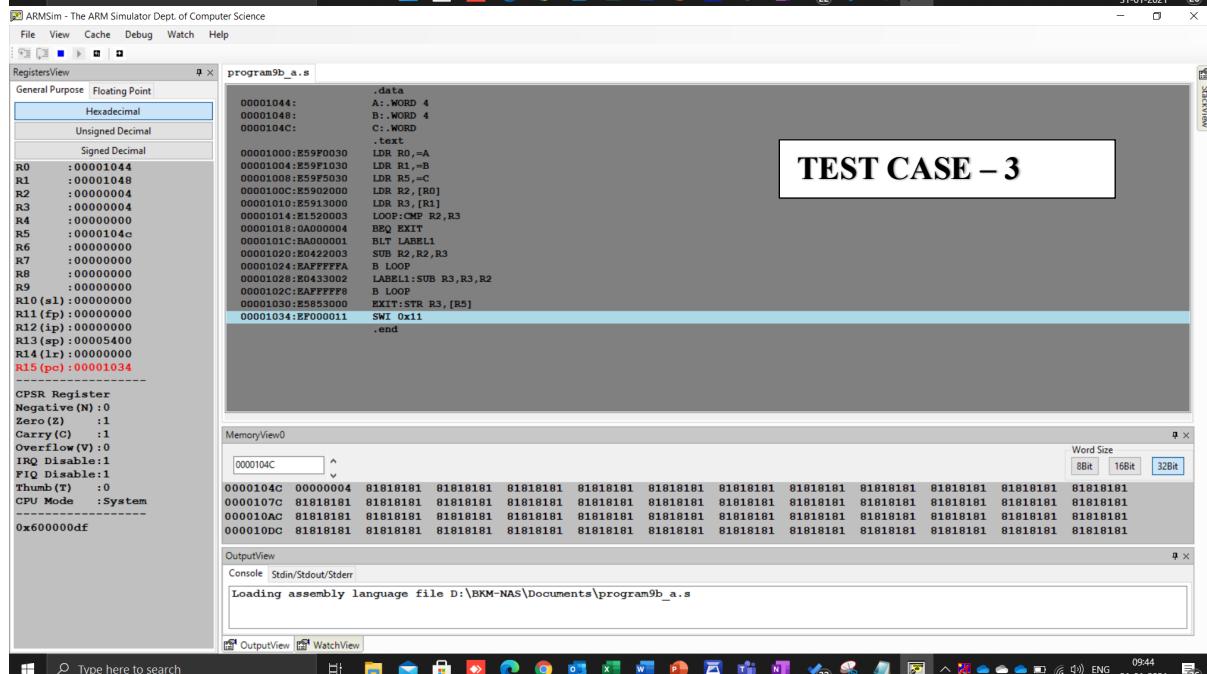
- Registers View:** Shows the ARM register state. The PC register (R15) is highlighted in red with the value 00001034.
- Memory View 0:** Displays memory starting at address 0000104C. The memory contains a repeating sequence of the value 01010101 (hex 00000004).
- Output View:** Shows the command "Loading assembly language file D:\BKM-NAS\Documents\program9b.s".

TEST CASE  
- 1

**TEST CASE – 2**



**TEST CASE – 3**



### III. Output Table

CASE 1	A=0x14,B=0x04
1 <sup>st</sup> iteration	R2=0x14=Decimal 20 R3=0x04 R2=0x14-0x04=0x10

2 <sup>nd</sup> iteration	R2=0x10=Decimal 16 R3=0x04 R2=0x10-0x04=0x0C
3 <sup>rd</sup> iteration	R2=0x0C=Decimal 12 R3=0x04 R2=0x0C-0x04=0x08
4th iteration	R2=0x08 R3=0x04 R2=0x08-0x04=0x04
5th iteration	R2=0x04 R3=0x04 GCD=C=0x04
<b>CASE 2</b>	<b>A=0x0A,B=0x0F</b>
1 <sup>st</sup> iteration	R2=0x0A=Decimal 10 R3=0x0F=Decimal 15 R3=0x0F-0x0A=0x05
2 <sup>nd</sup> iteration	R2=0x0A=Decimal 10 R3=0x05 R2=0x0A-0x05=0x05
3 <sup>rd</sup> iteration	R2=0x05=Decimal 5 R3=0x05 GCD=0x05 ( C=0x05)
<b>CASE 3</b>	<b>A=0x04,B=0x04</b>
1 <sup>st</sup> iteration	R2=0x04=Decimal 04 R3=0x04=Decimal 04 <b>GCD C=0x04</b>

**Week#2****Program Number:6a**

**Title of the Program :** Write an ALP to add an array of ten 32 bit numbers from memory

I.ARM Assembly Code for each program

```
.data
A: .WORD 10,20,30,40,50,60,70,80,90,11
.text
LDR R0,=A
MOV R1,#10
TEST CASE - 1
MOV R3,#0
LOOP: CMP R1,#0
BEQ EXIT
LDR R2, [R0],#4
ADD R3,R3,R2
SUB R1,R1,#1
B LOOP
EXIT: SWI 0x11
.end
```

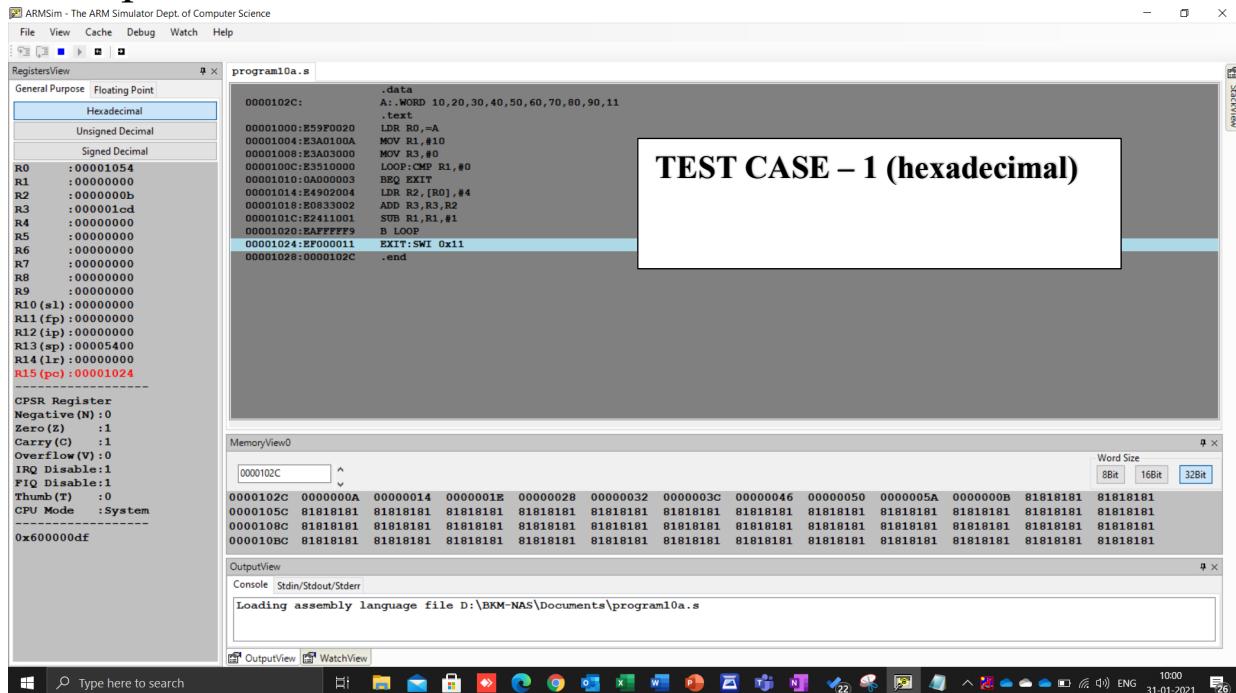
```

.data
A: .WORD 1,2,3,4,5,6,7,8,9,10
.text
LDR R0,=A
MOV R1,#10
MOV R3,#0
LOOP: CMP R1,#0
BEQ EXIT
LDR R2, [R0],#4
ADD R3,R3,R2
SUB R1,R1,#1
B LOOP
EXIT: SWI 0x11
.end

```

TEST CASE – 2

## II. Output Screenshots



ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :4180 R1 :0 R2 :11 R3 :461 R4 :0 R5 :0 R6 :0 R7 :0 R8 :0 R9 :0 R10 (s1) :0 R11 (fp) :0 R12 (ip) :0 R13 (sp) :21504 R14 (lr) :0 R15 (pc) :4132

CPSR Register Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1

Thumb (T) :0 CPU Mode :System

0x600000df

program10a.s

```

.data
0000102C: A: WORD 10,20,30,40,50,60,70,80,90,11
.text
00001000:E59F0020 LDR R0,=A
00001004:E3A0100A MOV R1,#10
00001008:E2510000
0000100C:E3510000 LOOP: CMP R1,#0
00001010:0A000003 BNE EXIT
00001014:E4902004 LDR R2,[R0],#4
00001018:E0833002 ADD R3,R3,R2
0000101C:E2411001 SUB R1,R1,#1
00001020:E4FFFF9 B LOOP
00001024:EF000011 EXIT: SWI 0x1
00001028:0000102C .end

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

0000102C 0000000A 00000014 0000001B 00000029 00000032 0000003C 00000046 00000050 0000005A 0000000B 01010101 01010101

0000105C 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

0000108C 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

000010BC 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

OutputView

Console Stdin/Stdout/Stderr

Loading assembly language file D:\BKM-NAS\Documents\program10a.s

OutputView WatchView

## TEST CASE – 1 (Decimal)

ARMSSim - The ARM Simulator Dept. of Computer Science

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal Unsigned Decimal Signed Decimal

R0 :00001054 R1 :00000000 R2 :00000000 R3 :00000037 R4 :00000000 R5 :00000000 R6 :00000000 R7 :00000000 R8 :00000000 R9 :00000000 R10 (s1) :00000000 R11 (fp) :00000000 R12 (ip) :00000000 R13 (sp) :00005400 R14 (lr) :00000000 R15 (pc) :00001024

CPSR Register Negative (N) :0 Zero (Z) :1 Carry (C) :1 Overflow (V) :0 IRQ Disable:1 FIQ Disable:1

Thumb (T) :0 CPU Mode :System

0x600000df

program10a.s

```

.data
0000102C: A: WORD 1,2,3,4,5,6,7,8,9,10
.text
00001000:E59F0020 LDR R0,=A
00001004:E3A0100A MOV R1,#10
00001008:E2510000
0000100C:E3510000 LOOP: CMP R1,#0
00001010:0A000003 BNE EXIT
00001014:E4902004 LDR R2,[R0],#4
00001018:E0833002 ADD R3,R3,R2
0000101C:E2411001 SUB R1,R1,#1
00001020:E4FFFF9 B LOOP
00001024:EF000011 EXIT: SWI 0x1
00001028:0000102C .end

```

MemoryView0

Word Size 8Bit 16Bit 32Bit

0000102C 00000001 00000002 00000003 00000004 00000005 00000006 00000007 00000008 00000009 0000000A 01010101 01010101

0000105C 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

0000108C 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

000010BC 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101

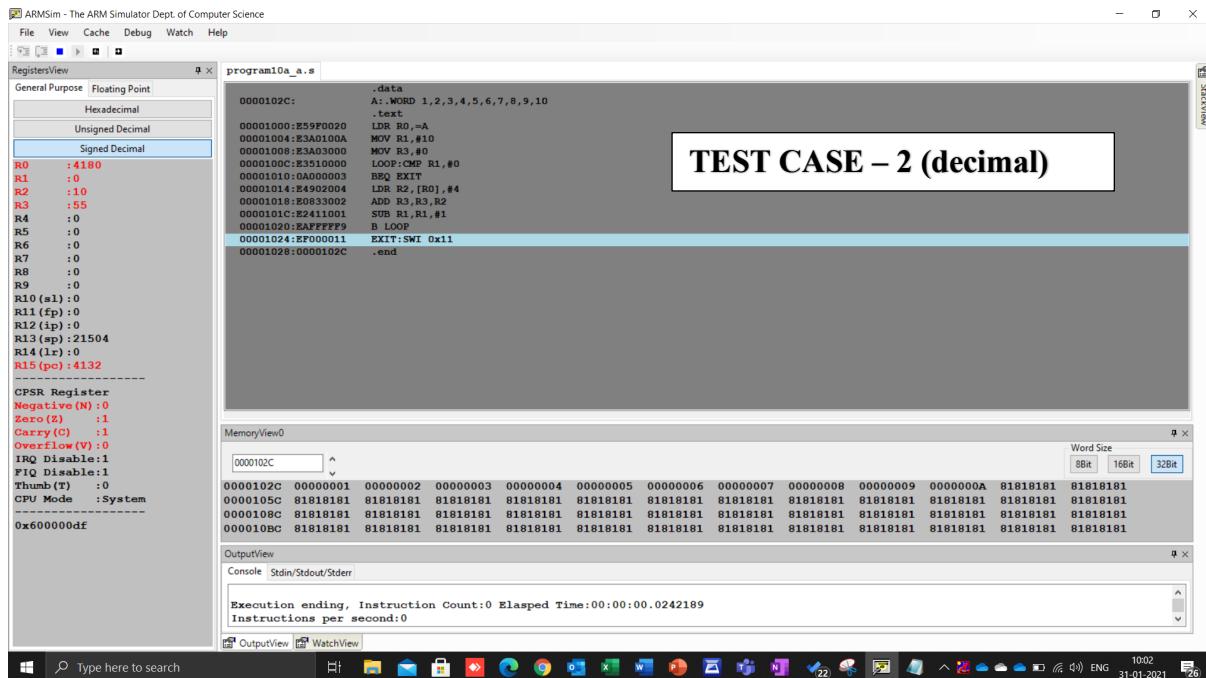
OutputView

Console Stdin/Stdout/Stderr

Execution ending, Instruction Count:0 Elapsed Time:00:00:00.0242189 Instructions per second:0

OutputView WatchView

## TEST CASE – 2 (hexadecimal)



### III. Output Table

#### A:.word 10,20,30,40,50,60,70,80,90,11 [TEST CASE 1]

<b>R1</b>	<b>10</b>	<b>9</b>	<b>8</b>	<b>7</b>	<b>6</b>	<b>5</b>	<b>4</b>	<b>3</b>	<b>2</b>	<b>1</b>
<b>R0</b>	<b>A</b>	<b>A+</b>	<b>A+</b>	<b>A+</b>	<b>A</b>	<b>A</b>	<b>A+</b>	<b>A+</b>	<b>A+</b>	<b>A+3</b>
		<b>4</b>	<b>8</b>	<b>12</b>	<b>+1</b>	<b>+2</b>	<b>24</b>	<b>28</b>	<b>32</b>	<b>6</b>
<b>R2</b>	<b>10</b>	<b>20</b>	<b>30</b>	<b>40</b>	<b>50</b>	<b>60</b>	<b>70</b>	<b>80</b>	<b>90</b>	<b>11</b>
<b>R3</b>	<b>0</b>	<b>10</b>	<b>30</b>	<b>60</b>	<b>10</b>	<b>15</b>	<b>210</b>	<b>280</b>	<b>360</b>	<b>450</b>
<b>R3(a fter exec</b>	<b>10</b>	<b>30</b>	<b>60</b>	<b>100</b>	<b>15</b>	<b>21</b>	<b>280</b>	<b>360</b>	<b>450</b>	<b>461</b>

ution )											
Values (In hexa deci mal)	0x 0A	0x 1E	0x4 C	0x6 4	0x 96	0x D2	0x1 18	0x1 68	0x1 C2	0x1 CD	
<b>A:.word 1,2,3,4,5,6,7,8,9,10 [TEST CASE 2]</b>											
R1	10	9	8	7	6	5	4	3	2	1	
R0	A 4	A+ 8	A+ 12	A +1 6	A +2 0	A+ 24	A+ 28	A+ 32	A+ 36	A+3 6	
R2	1	2	3	4	5	6	7	8	9	10	
R3	0	1	3	6	10	15	21	28	36	45	
R3(after exec ution )	1	3	6	10	15	21	28	36	45	55	
Values (In hexa deci mal)	0x 01	0x 03	0x0 6	0x0 A	0x 0F	0x 15	0x1 c	0x2 4	0x2 d	0x3 7	

**Week#2.****Program Number:6b**

**Title of the Program :** Write an ALP to add an array of five 8 bit numbers from memory(use .byte to store the data instead of .word)

I.ARM Assembly Code for each program

```
.data
A: .BYTE 1,2,3,4,5
.text
LDR R0,=A
MOV R1,#5
MOV R3,#0
LOOP: CMP R1,#0
BEQ EXIT
LDRB R2,[R0],#1
ADD R3,R3,R2
SUB R1,R1,#1
B LOOP
EXIT: SWI 0x11
.end
```

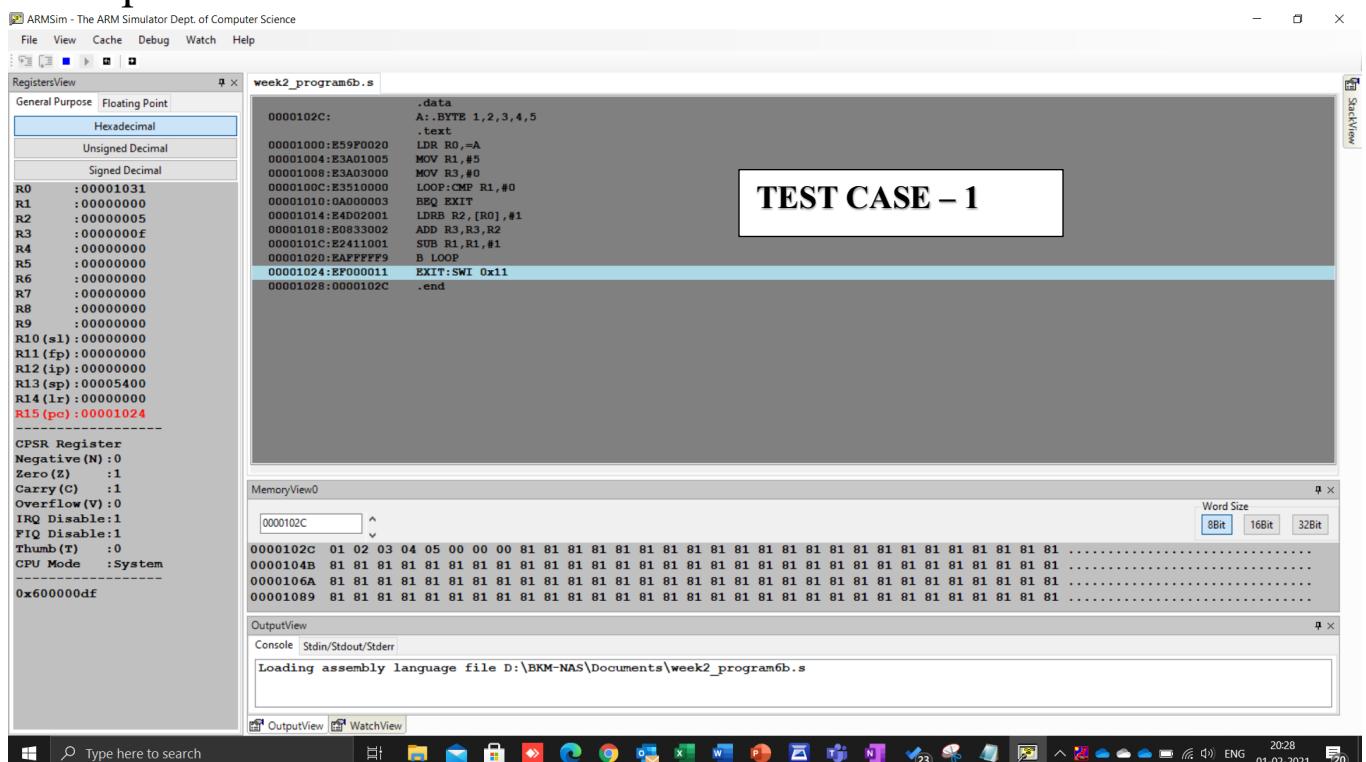
TEST CASE 1

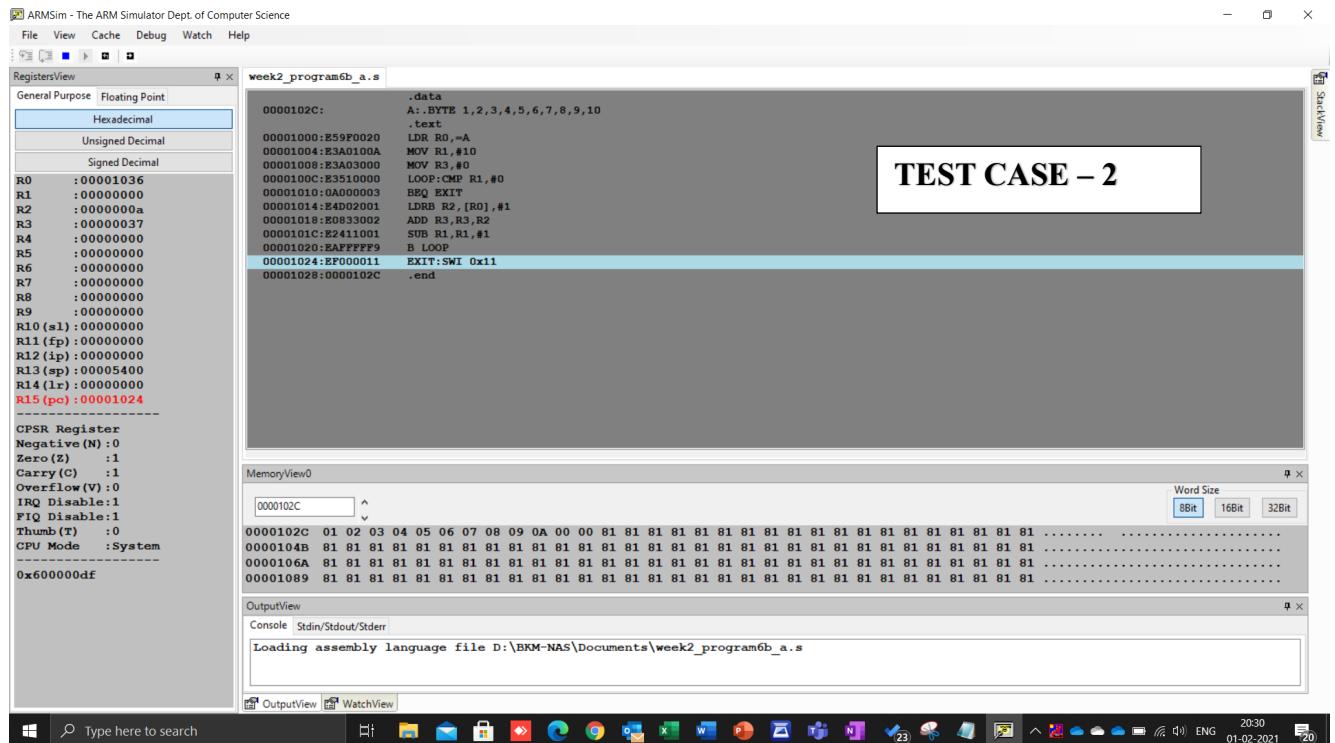
```

.data
A: .BYTE 1,2,3,4,5,6,7,8,9,10
.text
LDR R0,=A
MOV R1,#10
MOV R3,#0
LOOP: CMP R1,#0
BEQ EXIT
LDRB R2,[R0],#1
ADD R3,R3,R2
SUB R1,R1,#1
B LOOP
EXIT: SWI 0x11
.end

```

## II. Output Screenshots





### III. Output Table

#### A::byte 1,2,3,4,5

R1	5	4	3	2	1
R0	A	A+1	A+2	A+3	A+4
R3	0	1	3	6	10
R4	1	2	3	4	5
R3 (After Execution)	1	3	6	10	15
Values in hex	0x01	0x03	0x06	0xA	0xF

A:.byte 1,2,3,4,5,6,7,8,9,10											
R1	10	9	8	7	6	5	4	3	2	1	0
R0	A	A+1	A+2	A+3	A+4	A+5	A+6	A+7	A+8	A+9	A+10
R3	0	1	3	6	10	15	21	28	36	45	55
R4	1	2	3	4	5	6	7	8	9	10	10
R3 (After Execution)	1	3	6	10	15	21	28	36	45	55	55
Values in hex	0x 01	0x 03	0x 06	0x 0A	0x 0F	0x 15	0x 1c	0x 24	0x 2d	0x 37	0x 37

**Week#2**

**ProgramNumber:7**

**Title of the Program : Write an ALP to multiply 35\*R0.Use LSL instruction for multiplication**

**I.ARM ASSEMBLY CODE**

```
.text
MOV R0,#50
MOV R1,R0,LSL #2
ADD R0,R0,R1
MOV R2,R0,LSL #3
SUB R0,R2,R0
SWI 0x11
.end
```

TEST CASE – 1

```

.text
MOV R0,#5
MOV R1,R0,LSL #2
ADD R0,R0,R1
MOV R2,R0,LSL #3
SUB R0,R2,R0
SWI 0x11
.end

```

TEST CASE – 2

## II. Output Screenshots

The screenshot shows the ARMSim interface with the following windows:

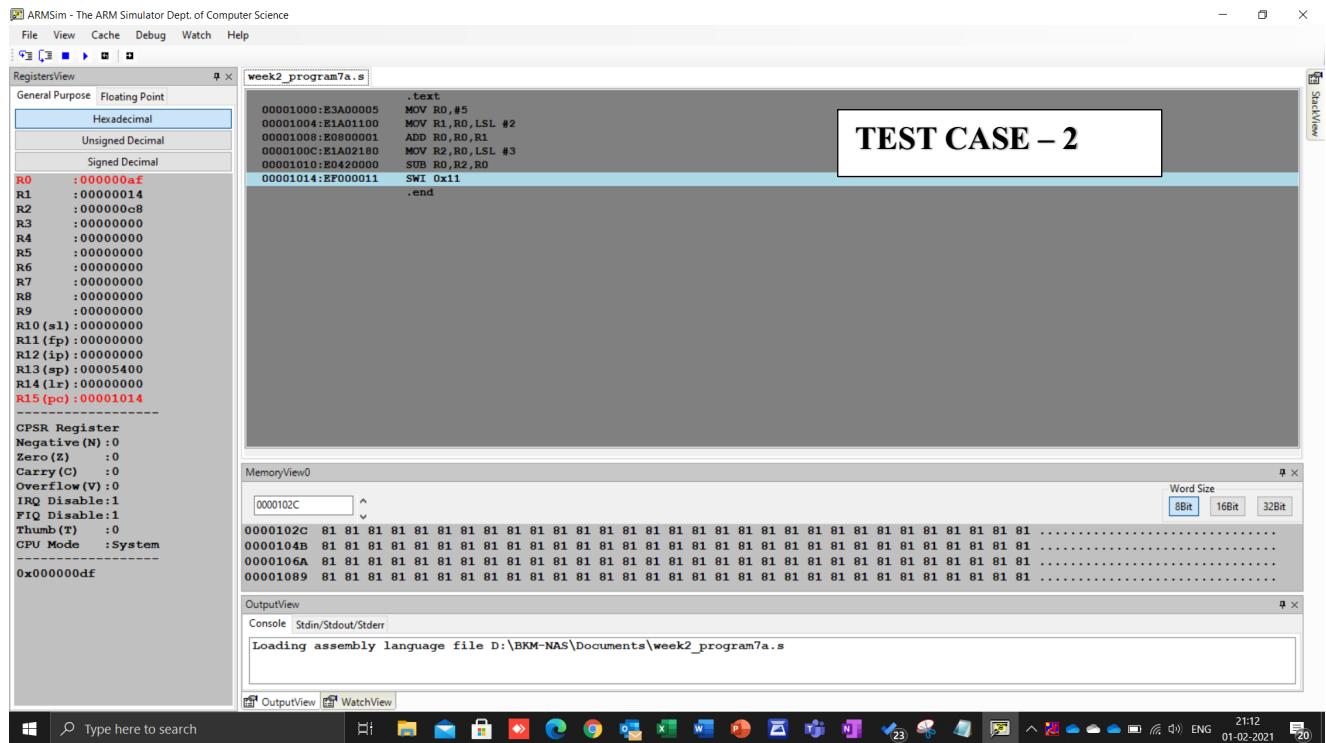
- RegistersView**: Shows general purpose registers (R0-R15) in hexadecimal, decimal, and signed decimal formats. R15 (pc) is highlighted in red as 00001014.
- MemoryView0**: Shows memory starting at address 0000102C, filled with the byte value 81 (hex).
- OutputView**: Shows the console output: "Loading assembly language file D:\BKM-NAS\Documents\week2\_program7.s".

At the top of the interface, a box labeled "TEST CASE – 1" is overlaid on the assembly code area.

```

.week
00001000:E3A00032  .text
00001004:E1A01100  MOV R0,#50
00001008:E08000001  MOV R1,R0,LSL #2
0000100C:E1A02180  ADD R0,R0,R1
0000100D:E0420000  MOV R2,R0,LSL #3
00001010:EF000011  SUB R0,R2,R0
00001014:EF000011  SWI 0x11
.end

```



### III. Output Table

TEST - 1	Decimal	Hexadecimal
R0(1 <sup>st</sup> line of the Program)	50	32
R0(2 <sup>nd</sup> line of the Program)	4R0=200	C8
	4R0+R0=5R0=250	FA
R0(3 <sup>rd</sup> line of	8R0=250*8=2000	0x7D0

## the Program

TEST - 2	Decimal	Hexadecimal
R0(1 <sup>st</sup> line of the Program)	5	5
R0(2 <sup>nd</sup> line of the Program)	4R0=20	14
	4R0+R0= 5R0=20+5=25	19
R0(3 <sup>rd</sup> line of the Program)	8R0=25*8=200	0xc8
	8R0-R0=200- 25=175	af

Week#2

Program Number:8

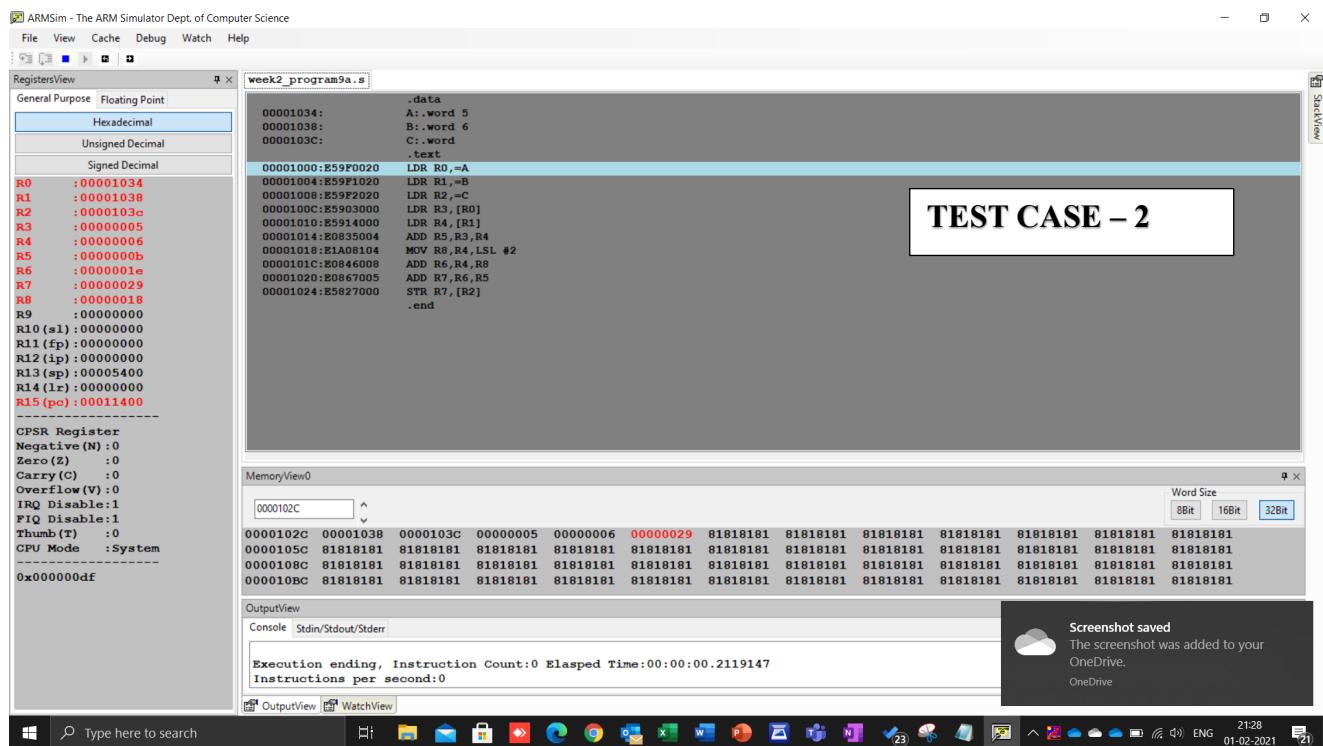
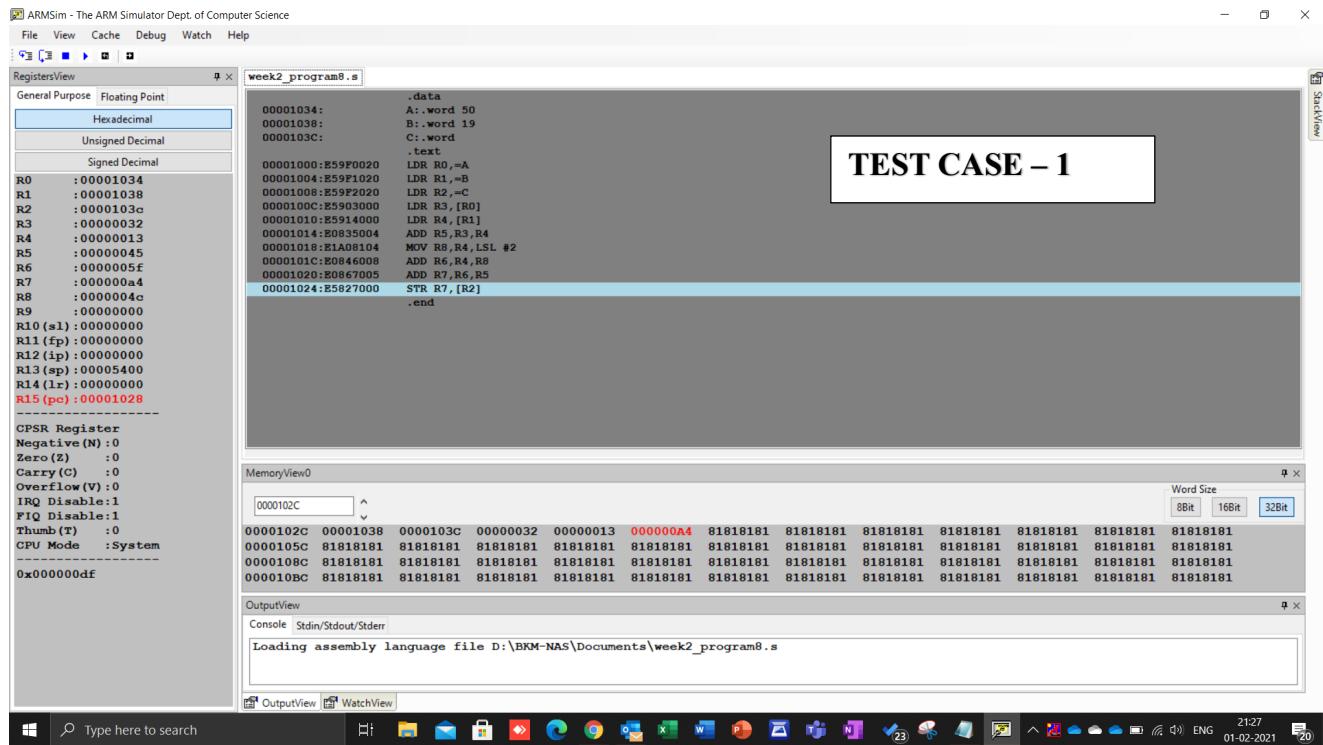
**Title of the Program : Write an ALP to evaluate the expression  $(A+B) + (5*B)$ , where A and B are available in memory location. 9.Store the final result in memory Location C.Use LSL instruction for multiplication**

### I.ARM ASSEMBLY CODE

```
.data    TEST CASE - 1
A: .word 50
B: .word 19
C: .word
.text
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDR R3,[R0]
LDR R4,[R1]
ADD R5,R3,R4
MOV R8,R4,LSL #2
ADD R6,R4,R8
ADD R7,R6,R5
STR R7,[R2]
.end
```

```
.data          TEST CASE - 2
A: .word 5
B: .word 6
C: .word
.text
LDR R0,=A
LDR R1,=B
LDR R2,=C
LDR R3, [R0]
LDR R4, [R1]
ADD R5, R3, R4
MOV R8, R4, LSL #2
ADD R6, R4, R8
ADD R7, R6, R5
STR R7, [R2]
.end
```

## II. Output Screenshots



### III. Output Table

**A=Decimal 50,B=Decimal 19 TEST CASE 1**

	Decimal	Hexadecimal
<b>R3</b>	<b>50</b>	<b>32</b>
<b>R4</b>	<b>19</b>	<b>13</b>
<b>R5</b>  <b>=R3+R4</b>  <b>=A+B</b>	<b>69</b>	<b>45</b>
<b>Calculate 4*B</b>	<b>19*4=76</b>	<b>4C</b>
<b>Calculate 5*B</b>	<b>76+19=95</b>	<b>5F</b>
<b>Calculate (A+B)+5*B</b>	<b>69+95=164</b>	<b>A4</b>

**A=Decimal 5,B=Decimal 6**

	Decimal	Hexadecimal
<b>R3</b>	<b>5</b>	<b>5</b>
<b>R4</b>	<b>6</b>	<b>6</b>

<b>R5</b>  $=R3+R4$  $=A+B$	<b>11</b>	<b>B</b>
<b>Calculate</b> $4*6=24$		<b>18</b>
<b>Calculate</b> $5*B$	<b>24+6=30</b>	<b>1e</b>
<b>Calculate</b> $(A+B)+5*B$	<b>11+30=41</b>	<b>29</b>

**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: 02/01/2021