

Microprocessor and Computer Architecture Laboratory

UE19CS256

4th Semester, Academic Year 2020-21

Date: 10-02-2021

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

Week#3

Program Number: 1

Write an ALP to add two 64-bit numbers loaded from memory and store the result in memory.

I.ARM Assembly Code for the program..

```
[10] 1.s
1  .data
2    a: .word 10000000, 20000000
3    b: .word 30000000, 40000000
4    c: .word 0
5  .text
6    ldr r0, =a
7    ldr r1, =b
8    ldr r2, =c
9    ldr r4, [r0]
10   ldr r5, [r1]
11   add r4, r4, r5
12   str r4, [r2]
13   ldr r4, [r0, #4]
14   ldr r5, [r1, #4]
15   add r6, r4, r5
16   str r6, [r2, #4]
17   .end
```

II. Output Screen Shot (One Example of your choice)

The screenshot shows a debugger interface with four main panes:

- RegistersView:** Displays the state of various registers. R0 through R15 are shown with their current values in hexadecimal. For example, R0 is 00001038, R1 is 00001040, and R15 (PC) is 00011400.
- CodeView:** Shows the assembly code being executed. It includes data sections (a: .word 10000000, 20000000; b: .word 30000000, 40000000; c: .word 0) and text sections with instructions like ldr, add, and str.
- MemoryView:** Displays a memory dump starting at address 0000100C. It shows the raw memory data in hexadecimal and its corresponding ASCII representation.
- OutputView:** Shows the console output of the program. It includes messages like "Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Week3 Student Copy\1.s", "Execution starting ...", and "Execution ending. Instruction Count:16640 Elapsed Time:00:00:00.0900870".

III. Output Table for the program

a: .word 10000000, 20000000 b: .word 30000000, 40000000		
	Upper 32 bits	Lower 32 bits
a: .word	20000000 (00989680)	10000000 (01312D00)
b: .word	40000000 (01C9C380)	30000000 (02625A00)
c: .word	60000000 (02625A00)	40000000 (03938700)

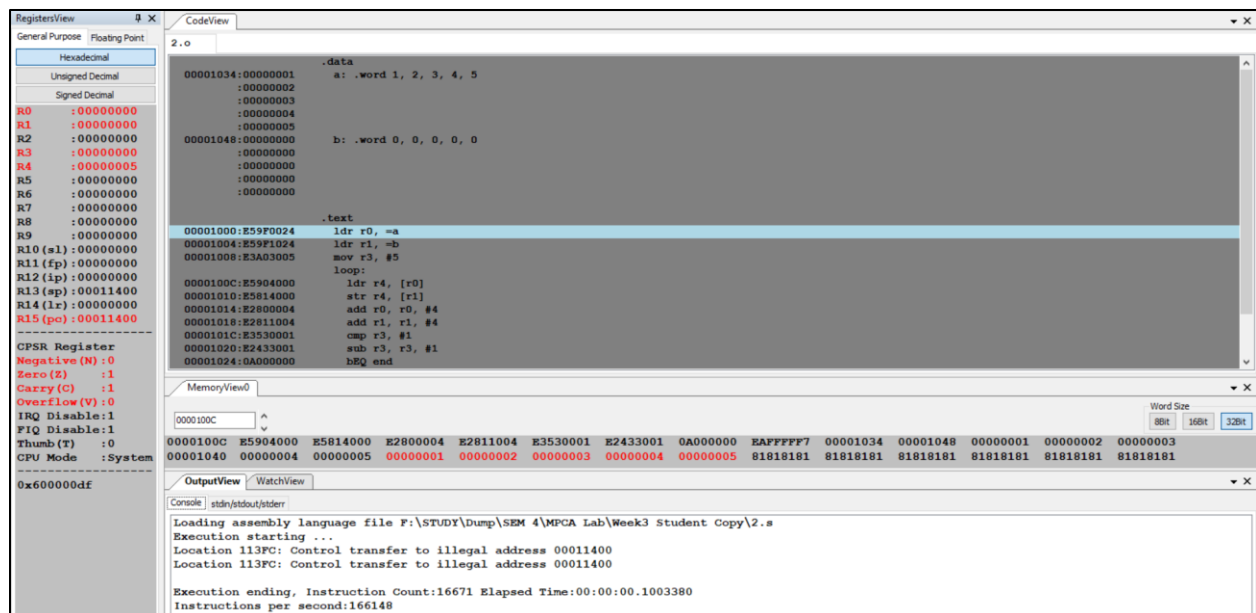
Program Number: 2

Write an ALP to copy n numbers from Memory Location A to Memory Location B

I.ARM Assembly Code for the program.

```
[01] 2.s
1  .data
2  a: .word 1, 2, 3, 4, 5
3  b: .word 0, 0, 0, 0, 0
4
5  .text
6  ldr r0, =a
7  ldr r1, =b
8  mov r3, #5
9  loop:
10 ldr r4, [r0]
11 str r4, [r1]
12 add r0, r0, #4
13 add r1, r1, #4
14 cmp r3, #1
15 sub r3, r3, #1
16 bEQ end
17 b loop
18 end:
19 .end
20
```

II.Output Screen Shot (One Example of your choice)



III. Output Table for the program

.data a: .word 1, 2, 3, 4, 5 b: .word 0, 0, 0, 0, 0	
1 st Iteration	a: .word 1, 2, 3, 4, 5 b: .word 1, 0, 0, 0, 0
2 nd Iteration	a: .word 1, 2, 3, 4, 5 b: .word 1, 2, 0, 0, 0
3 rd Iteration	a: .word 1, 2, 3, 4, 5 b: .word 1, 2, 3, 0, 0
4 th Iteration	a: .word 1, 2, 3, 4, 5 b: .word 1, 2, 3, 4, 0
5 th Iteration	a: .word 1, 2, 3, 4, 5 b: .word 1, 2, 3, 4, 5

Program Number: 3

**Write an ALP to find smallest number in an array
of n 32-bit numbers**

I.ARM Assembly Code for the program.

```
1 .data
2   a: .word 5,8,4,2,6
3
4 .text
5   ldr r0, =a
6   mov r1, #5
7   ldr r2, [r0]
8   add r0, r0, #4
9   loop:
10    cmp r2, #0
11    bEQ end
12    sub r1, r1, #1
13    ldr r3, [r0]
14    add r0, r0, #4
15    cmp r3, r2
16    movLO r2, r3
17    cmp r1, #1
18    bEQ end
19    b loop
20
21 end:
22 .end
```

II.Output Screen Shot (One Example of your choice)

The screenshot displays the execution environment for the ARM assembly program. It includes several panels:

- RegistersView:** Shows the state of 16 registers (R0-R15) and CPSR. R0 is 00000000, R1 is 00000000, R2 is 00000002, R3 is 00000006, R4 is 00000000, R5 is 00000000, R6 is 00000000, R7 is 00000000, R8 is 00000000, R9 is 00000000, R10 (sl) is 00000000, R11 (fp) is 00000000, R12 (ip) is 00000000, R13 (sp) is 00011400, R14 (lr) is 00000000, and R15 (pc) is 00011400. CPSR flags are: Negative(N):0, Zero(Z):1, Carry(C):1, Overflow(V):0, IRQ Disable:1, FIQ Disable:1, Thumb(T):0, CPU Mode: System.
- CodeView:** Shows the assembly code with addresses. The code is as follows:

```
0000103C:00000005 .data
0000103C:00000008 a: .word 5,8,4,2,6
0000103C:00000004
0000103C:00000002
0000103C:00000006
00001000:E59F0030 .text
00001000:E59F0030 ldr r0, =a
00001004:E3A01005 mov r1, #5
00001008:E5902000 ldr r2, [r0]
0000100C:E2800004 add r0, r0, #4
00001010:E3520000 loop:
00001010:E3520000 cmp r2, #0
00001014:0A000007 bEQ end
00001018:E2411001 sub r1, r1, #1
0000101C:E5903000 ldr r3, [r0]
00001020:E2800004 add r0, r0, #4
00001024:E1530002 cmp r3, r2
00001028:31A02003 movLO r2, r3
0000102C:E3510001 cmp r1, #1
00001030:0A000000 bEQ end
00001034:EAF0FF5 b loop
```
- MemoryView0:** Shows memory at address 0000100C, containing the array [5, 8, 4, 2, 6].
- OutputView:** Shows the execution log:

```
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Week3 Student Copy\3.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending. Instruction Count:1669 Elapsed Time:00:00:00.0503329
Instructions per second:331175
```

III. Output Table for the program

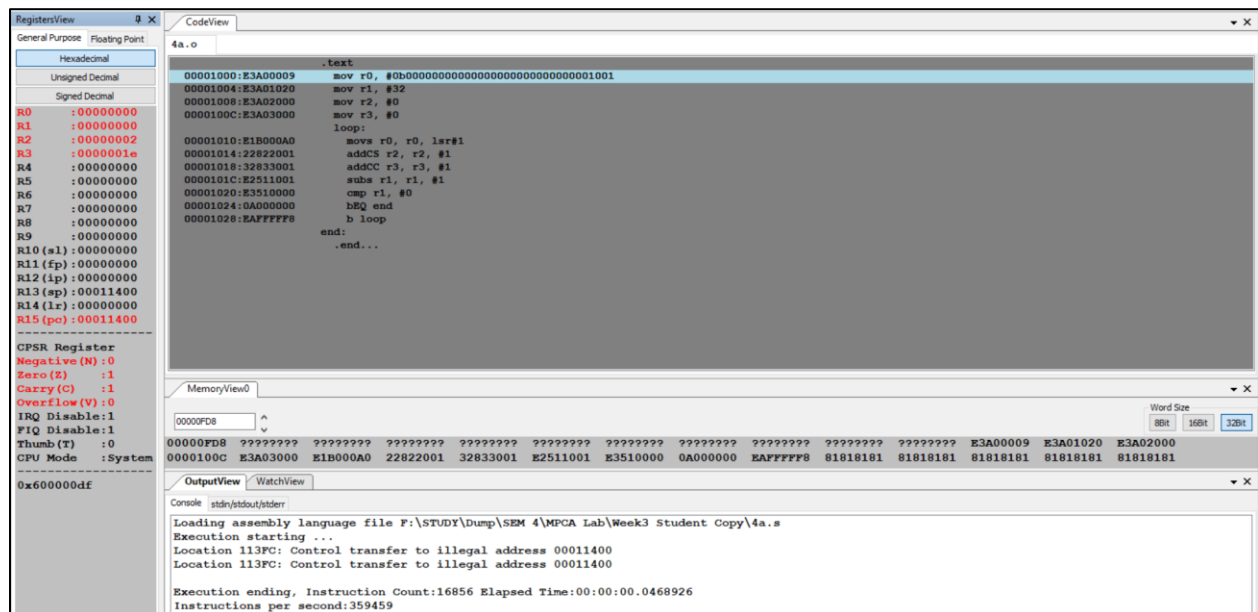
a: .word 5, 8, 4, 2, 6	
1 st Iteration	R2=5, R3=8 (R3>R2)
2 nd Iteration	R2=5, R3=4 (R3<R2)
3 rd Iteration	R2=4, R3=2 (R3<R2)
4 th Iteration	R2=2, R3=6 (R3>R2)
Smallest number 2 is present in R2	

Write an ALP to count the number of 1's and 0's in a given 32-bit number.

I.ARM Assembly Code for the program.

[illegible]

II. Output Screen Shot (One Example of your choice)



III. Output Table for the program

r0= 0b000000000000000000000000000000001001		
r1	32	
r2	After execution	2 (=2 in hex)
r3	After execution	30 (=1e in hex)

Program Number: 4b

Write an ALP to find the number of zeroes, positive and negative numbers in a given array

I.ARM Assembly Code for the program.

```
[a0] 4b.s
1  .data
2  a: .word 1, 2, 0, -3, -4, 0, 0, 5, 6, 0
3  .text
4  ldr r0, =a
5  mov r1, #10
6  mov r5, #0
7  mov r4, #0
8  mov r3, #0
9  loop:
10 ldr r2, [r0]
11 add r0, r0, #4
12 cmp r2, #0
13 addEQ r4, r4, #1
14 addGT r5, r5, #1
15 addLT r3, r3, #1
16 subs r1, r1, #1
17 bNE loop
18 .end
19
```

II.Output Screen Shot (One Example of your choice)

The screenshot displays the ARM assembly debugger interface. The **Register View** on the left shows the state of registers R0 through R15, with R0 at 00000000, R1 at 00000000, R2 at 00000000, R3 at 00000000, R4 at 00000000, R5 at 00000000, R6 at 00000000, R7 at 00000000, R8 at 00000000, R9 at 00000000, R10 (s1) at 00000000, R11 (fp) at 00000000, R12 (ip) at 00000000, R13 (sp) at 00011400, R14 (lr) at 00001047, and R15 (pc) at 00002042. The **Code View** shows the assembly code for 4b.o, with the current instruction at 00001000: E59F002C, ldr r0, =a. The **Memory View** shows the memory contents at 00001000, with the current instruction at 00001000: E59F002C, ldr r0, =a. The **Output View** shows the execution log, including the loading of the assembly language file, execution starting, location 2042: Store to invalid memory location 0x0, execution ending, instruction count: 93, elapsed time: 00:00:00.0199512, and instructions per second: 4661.

```
RegistersView 4b.o
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 00000000
R1 : 00000000
R2 : 00000000
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): 00011400
R14 (lr): 00001047
R15 (pc): 00002042
CPSR Register
Negative (N): 0
Zero (Z): 1
Carry (C): 1
Overflow (V): 0
IRQ Disable: 1
FIQ Disable: 1
Thumb (T): 1
CPU Mode : System
0x600000ff

CodeView
4b.o
.data
00001038:00000001 a: .word 1, 2, 0, -3, -4, 0, 0, 5, 6, 0
:00000002
:00000000
:FFFFFFFD
:FFFFFFFC
.text
00001000:E59F002C ldr r0, =a
00001004:E3A0100A mov r1, #10
00001008:E3A05000 mov r5, #0
0000100C:E3A04000 mov r4, #0
00001010:E3A03000 mov r3, #0
loop:
00001014:E5902000 ldr r2, [r0]
00001018:E2800004 add r0, r0, #4
0000101C:E3520000 cmp r2, #0
00001020:02844001 addEQ r4, r4, #1
00001024:C2855001 addGT r5, r5, #1
00001028:B2833001 addLT r3, r3, #1
0000102C:E2511001 subs r1, r1, #1
00001030:1AFFFFF7 bNE loop
00001034:00000000 .end

MemoryView
0000100C E3A04000 E3A03000 E5902000 E2800004 E3520000 02844001 C2855001 B2833001 E2511001 1AFFFFF7 00001038 00000001 00000002
00001040 00000000 FFFFFFFD FFFFFFFC 00000000 00000000 00000005 00000006 00000000 81818181 81818181 81818181 81818181

OutputView
WatchView
Console
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Week3 Student Copy\4b.s
Execution starting ...
Location 2042: Store to invalid memory location 0x0
Execution ending, Instruction Count:93 Elapsed Time:00:00:00.0199512
Instructions per second:4661
```

III. Output Table for the program

a: word 1, 2, 0, -3, -4, 0, 0, 5, 6, 0	
R3	2
R4	4
R5	4

Program Number: 5

Write an ALP to check whether a given number is present in array using Linear Search (Without SWI 0x02), if found move +1 to R6 and key position to R7 else move -1 to R6 (if number not found)

I.ARM Assembly Code for the program.

```
[a0] 5.s
1  .data
2  a: .word 5, 10, 15, 20, 25, 30, 35, 40, 45, 50
3  .text
4  mov r2, #31
5  mov r3, #10
6  ldr r0, =a
7  loop:
8      ldr r1, [r0]
9      add r0, r0, #4
10     cmp r1, r2
11     bEQ exit
12     subs r3, r3, #1
13     bNE loop
14     mov r6, #-1
15     b end
16 exit:
17     mov r4, #11
18     sub r7, r4, r3
19 end:
20 .end
```

II.Output Screen Shot (One Example of your choice)

The screenshot displays the execution of the ARM assembly program. The top panel shows the assembly code with memory addresses. The middle panel shows the registers, with R0-R15 and CPSR. The bottom panel shows the console output, which includes the loading of the assembly file, execution starting, control transfer to illegal addresses, and execution ending with instruction count and elapsed time.

```
RegistersView: 5.s
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 00000000
R1 : 00000000
R2 : 0000001F
R3 : 00000000
R4 : 00000000
R5 : 00000000
R6 : FFFFFFFF
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10(s1): 00000000
R11(fp): 00000000
R12(ip): 00000000
R13(sp): 00011400
R14(lr): 00000000
R15(pc): 00011400
CPSR Register
Negative(N): 0
Zero(Z): 1
Carry(C): 1
Overflow(O): 0
IRQ Disable: 1
FIQ Disable: 1
Thumb(T): 0
CPU Mode : System
0x600000df

CodeView: 5.s
.data
00001038:00000005 a: .word 5, 10, 15, 20, 25, 30, 35, 40, 45, 50
:0000000A :0000000F
:0000000C :00000014
:0000000E :00000019
.text
00001000:E3A0201F mov r2, #31
00001004:E3A0300A mov r3, #10
00001008:E59F0024 ldr r0, =a
loop:
0000100C:E5901000 ldr r1, [r0]
00001010:E2800004 add r0, r0, #4
00001014:R1510002 cmp r1, r2
00001018:0A000003 bEQ exit
0000101C:E2533001 subs r3, r3, #1
00001020:1AFFFFF9 bNE loop
00001024:E3E06000 mov r6, #-1
00001028:EAD00001 b end
exit:
0000102C:E3A0400B mov r4, #11
00001030:E0470003 sub r7, r4, r3
end:
00001034:00000000 .end...

MemoryView0
Word Size
8Bit 16Bit 32Bit
0000100C
0000100C E5901000 E2800004 E1510002 0A000003 E2533001 1AFFFFF9 E3E06000 EA000001 E3A0400B E0470003 00001038 00000005 0000000A 0000000F
00001040 0000000F 00000014 00000019 0000001E 00000023 0000002D 00000032 81818181 81818181 81818181 81818181 81818181 81818181

OutputView / WatchView
Console
Loading assembly language file F:\STUDY\Dump\SEM 4\MPCA Lab\Week3 Student Copy\5.s
Execution starting ...
Location 113FC: Control transfer to illegal address 00011400
Location 113FC: Control transfer to illegal address 00011400
Execution ending, Instruction Count:16692 Elapsed Time:00:00:00.0591990
Instructions per second:281964
```

III. Output Table for the program

A:.WORD 5, 10, 15, 20, 25, 30, 35, 40, 45, 50		
		HEX value
R2	KEY =31	1F
R3	COUNT =10	
R0	Address of A	00001038
R3	After Execution =0 R6= -1 (key not present)	

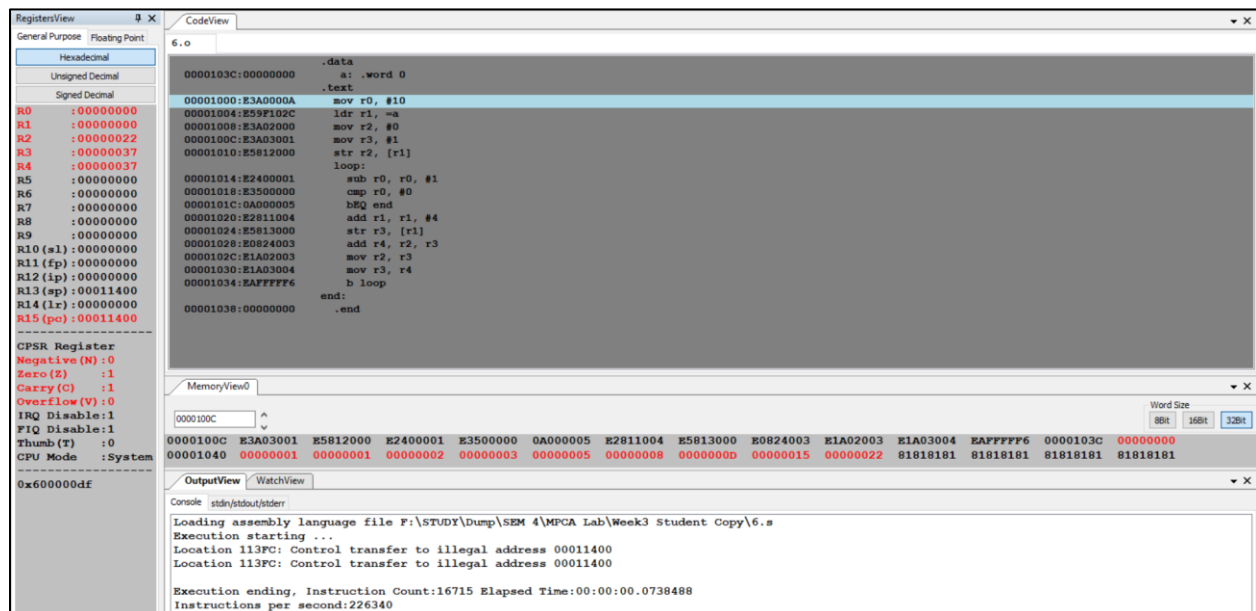
Program Number: 6

Write an ALP to generate Fibonacci Series and store them in an array

I.ARM Assembly Code for the program.

```
[6] 6.s
1  .data
2  a: .word 0
3  .text
4  mov r0, #10
5  ldr r1, =a
6  mov r2, #0
7  mov r3, #1
8  str r2, [r1]
9  loop:
10     sub r0, r0, #1
11     cmp r0, #0
12     beq end
13     add r1, r1, #4
14     str r3, [r1]
15     add r4, r2, r3
16     mov r2, r3
17     mov r3, r4
18     b loop
19 end:
20 .end
21
```

II.Output Screen Shot (One Example of your choice)



III. Output Table for the program

FIBONACCI SEQUENCE		
R0	Fibonacci Count	10
R1	Address of A	0000103C
R2	Initially 0	0
R3	Initially 1	1
R4	1 st Iteration	0 + 1 = 1
R4	2 nd Iteration	1 + 1 = 2
R4	3 rd Iteration	2 + 1 = 3
R4	4 th Iteration	3 + 2 = 5
R4	5 th Iteration	5 + 3 = 8
R4	6 th Iteration	8 + 5 = 13 = 0000000D
R4	7 th Iteration	13 + 8 = 21 = 00000015
R4	8 th Iteration	21 + 13 = 34 = 00000022

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: 10-02-21