

Microprocessor and Computer Architecture Laboratory

UE19CS256

4th Semester, Academic Year 2020-21

Date:

Name: A Narendiran	SRN: PES1UG19CS001	Section
		A

Week# 3 Program Number: 1

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

ARM Assembly Code for the program

```
text
    LDR r0,=a
    LDR r1,=b
    LDR r3,=c
    LDR r4,[r0]
    LDR r5,[r1]
    ADD r4,r4,r5
    STR r4,[r3]
    ADD r0,r0,#4
    ADD r1,r1,#4
    ADD r3,r3,#4
    LDR r4,[r0]
    LDR r5,[r1]
    ADD r6,r4,r5
    STR r6,[r3]
    SWI 0x011
.data
a:.word 12345678, 57689457
b:.word 98564532, 46237824
c:.word 0,0
```

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 : 0000104c
 R1 : 00001054
 R2 : 00000000
 R3 : 0000105c
 R4 : 03704571
 R5 : 02c18880
 R6 : 0631cdf1
 R7 : 00000000
 R8 : 00000000
 R9 : 00000000
 R10 (s1) : 00000000
 R11 (fp) : 00000000
 R12 (ip) : 00000000
 R13 (sp) : 00000000
 R14 (lr) : 0000103c
 R15 (pc) : 00000008

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 : Supervisor

0x000000d3

p1.s

.text

```
00001000:E59F0034 LDR r0,=a
00001004:E59F1034 LDR r1,=b
00001008:E59F3034 LDR r3,=c
0000100C:E5904000 LDR r4,[r0]
00001010:E5915000 LDR r5,[r1]
00001014:E0844005 ADD r4,r4,r5
00001018:E5834000 STR r4,[r3]
0000101C:E2800004 ADD r0,r0,#4
00001020:E2811004 ADD r1,r1,#4
00001024:E2833004 ADD r3,r3,#4
00001028:E5904000 LDR r4,[r0]
0000102C:E5915000 LDR r5,[r1]
00001030:E0846005 ADD r6,r4,r5
00001034:E5836000 STR r6,[r3]
00001038:EF000011 SWI 0x011
```

.data

```
00001048: a:.word 12345678, 57689457
00001050: b:.word 98564532, 46237824
00001058: c:.word 0,0
```

MemoryView0

1058

00001058	069C5B02	0631CDF1	81818181	81818181
0000108C	81818181	81818181	81818181	81818181
000010C0	81818181	81818181	81818181	81818181
000010F4	81818181	81818181	81818181	81818181
00001128	81818181	81818181	81818181	81818181
0000115C	81818181	81818181	81818181	81818181

Week# 3 Program Number: 2

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

ARM Assembly Code for the program

```
.data
a: .word 1,3,5,7,11
b: .word 0,0,0,0,0

.text
LDR r0, =a
LDR r1, =b
MOV r2, #5
loop:
LDR r3, [r0]
STR r3, [r1]
ADD r0, r0, #4
ADD r1, r1, #4
SUB r2, r2, #1
CMP r2, #0
BNE loop
SWI 0x11
.end
```

Output Screen Shot

File View Cache Debug Watch Help

RegistersView ✕

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 00001048
R1 : 0000105c
R2 : 00000000
R3 : 0000000b
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00000000
R14 (lr) : 0000102c
R15 (pc) : 00000008

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 : Supervisor

0x600000d3

p2.s

```
.data
a: .word 1,3,5,7,11
b: .word 0,0,0,0,0

.text
00001000:E59F0024 LDR r0, =a
00001004:E59F1024 LDR r1, =b
00001008:E3A02005 MOV r2, #5
0000100C:          loop:
0000100C:E5903000 LDR r3, [r0]
00001010:E5813000 STR r3, [r1]
00001014:E2800004 ADD r0, r0, #4
00001018:E2811004 ADD r1, r1, #4
0000101C:E2422001 SUB r2, r2, #1
00001020:E3520000 CMP r2, #0
00001024:1AFFFFF8 BNE loop
00001028:EF000011 SWI 0x11
0000102C:00001034 .end
00001030:00001048
```

MemoryView0

1048

00001048	00000001	00000003	00000005	00000007	0000000B
0000107C	81818181	81818181	81818181	81818181	81818181
000010B0	81818181	81818181	81818181	81818181	81818181
000010E4	81818181	81818181	81818181	81818181	81818181
00001118	81818181	81818181	81818181	81818181	81818181

Week# 3 Program Number: 3

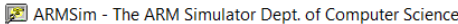
3. Write an ALP to find smallest number in an array of n - 32 bit numbers

ARM Assembly Code for the program

```
.data
a: .word 92,257,22,53,62,73
.text
LDR r0, =a
MOV r1, #6
LDR r2, [r0], #4
SUB r1, r1, #1
```

```
loop:
CMP r2,#0
BEQ exit
LDR r3,[r0],#4
SUBS r4,r3,r2
BMI l1
SUBS r1,r1,#1
BEQ exit
BNE loop
l1:
MOV r2,r3
SUBS r1,r1,#1
BEQ exit
BNE loop
exit:
SWI 0x11
.end
```

Output Screen Shot



The screenshot displays the ARM DevStudio interface with two main windows open: RegistersView and MemoryView0.

RegistersView: This window shows the state of the ARM registers. The "General Purpose" tab is selected, and the "Hexadecimal" view is chosen. The registers R0 through R15 are listed, with their current values in hexadecimal. R0 is 00001060, R1 is 00000000, R2 is 00000016, R3 is 00000049, R4 is 00000033, R5 is 00000000, R6 is 00000000, R7 is 00000000, R8 is 00000000, R9 is 00000000, R10 (s1) is 00000000, R11 (fp) is 00000000, R12 (ip) is 00000000, R13 (sp) is 00000000, R14 (lr) is 00001044, and R15 (pc) is 00000008. Below the registers, the CPSR Register is shown with various flags: Negative (N) is 0, Zero (Z) is 1, Carry (C) is 1, Overflow (V) is 0, IRQ Disable is 1, FIQ Disable is 1, Thumb (T) is 0, and CPU Mode is Supervisor.

MemoryView0: This window shows the memory dump for the current execution. The address 1048 is entered in the address field. The memory dump shows a sequence of instructions in hexadecimal, with the first instruction being 00001048 0000005C 00000101 00000016 00000035 0000003E 00000049.

Week#____3_____

Program Number: __4a__

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

ARM Assembly Code for the program

```
.data
A:.word 0x11011010
.text
LDR R0,=A
LDR R1,[R0]
MOV R2,#0
MOV R3,#0
LOOP: MOVS R1,R1,LSL #1
ADDCS R2,R2,#1
CMP R1,#0
BNE LOOP
RSC R3,R2,#32
SWI 0x1011
.end
```

Output Screen Shot

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0000102c
R1 : 00000000
R2 : 00000005
R3 : 0000001b
R4 : 00000000
R5 : 00000000
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00000000
R14 (lr) : 00001028
R15 (pc) : 00000008

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 : Supervisor

0x600000d3

p4a.s

```
.data
0000102C:      A: .word 0x11011010
.text
00001000:E59F0020  LDR R0,=A
00001004:E5901000  LDR R1,[R0]
00001008:E3A02000  MOV R2,#0
0000100C:E3A03000  MOV R3,#0
00001010:E1B01081  LOOP: MOVS R1,R1,LSL #1
00001014:22822001  ADDCS R2,R2,#1
00001018:E3510000  CMP R1,#0
0000101C:1AFFFFFB  BNE LOOP
00001020:E2E23020  RSC R3,R2,#32
00001024:EF001011  SWI 0x1011
00001028:0000102C  .end
```

MemoryView0

00001000

00001000	E59F0020	E5901000	E3A02000	E3A03000
00001034	81818181	81818181	81818181	81818181
00001068	81818181	81818181	81818181	81818181
0000109C	81818181	81818181	81818181	81818181
000010D0	81818181	81818181	81818181	81818181
00001104	81818181	81818181	81818181	81818181
00001138	81818181	81818181	81818181	81818181
0000116C	81818181	81818181	81818181	81818181
000011A0	81818181	81818181	81818181	81818181
000011D4	81818181	81818181	81818181	81818181
00001208	81818181	81818181	81818181	81818181

Week# 3 Program Number: 4b

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

ARM Assembly Code for the program

```
.data
a: .word 1,2,-2,3,0,-1,0,-5
.text
```

```
LDR r0,=a
MOV r1,#8
MOV r3,#0
MOV r4,#0
MOV r5,#0
loop:
CMP r1,#0
BEQ l4
SUB r1,r1,#1
LDR r2,[r0]
ADD r0,r0,#4
CMP r2,#0
BEQ l2
CMP r2,#0
BGT l1
BLT l3
l1:
ADD r5,r5,#1
B loop
l2:
ADD r4,r4,#1
B loop
l3:
ADD r3,r3,#1
B loop
l4:
SWI 0x11
.end
```

Output Screen Shot

R0 : 0000107c
R1 : 00000000
R2 : ffffffff
R3 : 00000003
R4 : 00000002
R5 : 00000003
R6 : 00000000
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (sl) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00000000
R14 (lr) : 00001058
R15 (pc) : 00000008

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 : Supervisor
0x600000d3

```
.data
a: .word 1,2,-2,3,0,-1,0,-5
.text
0000105C:
00001000:E59F0050 LDR r0,=a
00001004:E3A01008 MOV r1,#8
00001008:E3A03000 MOV r3,#0
0000100C:E3A04000 MOV r4,#0
00001010:E3A05000 MOV r5,#0
00001014: loop:
00001014:E3510000 CMP r1,#0
00001018:0A00000D BEQ 14
0000101C:E2411001 SUB r1,r1,#1
00001020:E5902000 LDR r2,[r0]
00001024:E2800004 ADD r0,r0,#4
00001028:E3520000 CMP r2,#0
0000102C:0A000004 BEQ 12
00001030:E3520000 CMP r2,#0
00001034:CA000000 BGT 11
00001038:BA000003 BLT 13
```

00001000

00001000	E59F0050	E3A01008	E3A03000	E3A04000
00001034	CA000000	BA000003	E2855001	EAF00003
00001068	00000003	00000000	FFFFFFFF	00000000
0000109C	81818181	81818181	81818181	81818181
000010D0	81818181	81818181	81818181	81818181
00001104	81818181	81818181	81818181	81818181
00001138	81818181	81818181	81818181	81818181
0000116C	81818181	81818181	81818181	81818181
000011A0	81818181	81818181	81818181	81818181
000011D4	81818181	81818181	81818181	81818181
00001208	81818181	81818181	81818181	81818181

Week# 3 Program Number: 5

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)

ARM Assembly Code for the program

```
.data
a: .WORD 3,9,10,12,16,38,40,58,65,79
.text
LDR r0, =a
```

```
MOV r1, #10
MOV r4, #0 ;to count the position
Loop:
LDR r2, [r0]
CMP r2, r1
BEQ l1
ADD r4, r4, #1
ADD r0, r0, #4
SUBS r1, r1, #1
BNE Loop
MOV R3, #-1 ;if not found
SWI 0x011
l1:
MOV R3, #1 ;if found
SWI 0x011
.end
```

Output Screen Shot

File View Cache Debug Watch Help

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 0000104c
R1 : 00001054
R2 : 00000000
R3 : 0000105c
R4 : 03704571
R5 : 02c18880
R6 : 0631cdf1
R7 : 00000000
R8 : 00000000
R9 : 00000000
R10 (s1) : 00000000
R11 (fp) : 00000000
R12 (ip) : 00000000
R13 (sp) : 00000000
R14 (lr) : 0000103c
R15 (pc) : 00000008

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 : Supervisor

0x000000d3

p1.s

.text

00001000:E59F0034 LDR r0,=a
00001004:E59F1034 LDR r1,=b
00001008:E59F3034 LDR r3,=c
0000100C:E5904000 LDR r4,[r0]
00001010:E5915000 LDR r5,[r1]
00001014:E0844005 ADD r4,r4,r5
00001018:E5834000 STR r4,[r3]
0000101C:E2800004 ADD r0,r0,#4
00001020:E2811004 ADD r1,r1,#4
00001024:E2833004 ADD r3,r3,#4
00001028:E5904000 LDR r4,[r0]
0000102C:E5915000 LDR r5,[r1]
00001030:E0846005 ADD r6,r4,r5
00001034:E5836000 STR r6,[r3]
00001038:EF000011 SWI 0x011

.data

00001048: a:.word 12345678, 57689457
00001050: b:.word 98564532, 46237824
00001058: c:.word 0,0

Week# 3 Program Number: 6

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

ARM Assembly Code for the program

```
.text
MOV r0, #10
MOV r1, #0
MOV r2, #1
LDR r3, =a
STR r1, [r3], #4
STR r2, [r3], #4
CMP r0, #1
BNE Loop
Loop:
ADD r5, r1, r2
MOV r1, r2
MOV r2, r5
STR r5, [r3], #4
SUBS r0, r0, #1
CMP r0, #1
BEQ exit
BL Loop
exit:
SWI 0x011
.data
a: .WORD
```

Output Screen Shot

