

Microprocessor and Computer Architecture Laboratory

UE19CS256

4th Semester, Academic Year 2020-21

Date: 1/2/2021

Name: TUSHAR Y S	SRN: PES1UG19CS545	Section I
------------------	-----------------------	--------------

Week# 2 Program Number: 1

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.

ARM Assembly Code:

Case 1 (zero):

.text

MOV R0,#0

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

c2:

MOV R1,#3

SWI 0x11

.end

Output:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'RegistersView', shows the state of various registers. The right panel, titled 'a1.s', shows the assembly code being executed.

RegistersView Panel:

- General Purpose: Floating Point
- Hexadecimal (selected)
- Unsigned Decimal
- Signed Decimal
- R0: 00000000
- R1: 00000001
- 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): 00005400
- R14 (lr): 00000000
- R15 (pc): 0000101c
-
- 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

a1.s Panel:

```
.text
00001000:E3A00000    MOV R0,#0
00001004:E3500000    CMP R0,#0
00001008:0A000002    BEQ c1
0000100C:4A000003    BMI c2
00001010:E3A01002    MOV R1,#2
00001014:EF000011    SWI 0x11
00001018:                c1:
00001018:E3A01001    MOV R1,#1
0000101C:EF000011    SWI 0x11
00001020:                c2:
00001020:E3A01003    MOV R1,#3
00001024:EF000011    SWI 0x11
.end
```

Case 2 (positive number):

.text

MOV R0,#18

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

c2:

MOV R1,#3

SWI 0x11

.end

Output:

The screenshot displays a debugger interface with two main panes. The left pane, titled 'RegistersView', shows the state of various registers. The right pane, titled 'a1.s', shows the assembly code being executed.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal (selected)
- Unsigned Decimal
- Signed Decimal
- R0: 00000012
- R1: 00000002
- 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): 00005400
- R14 (lr): 00000000
- R15 (pc): 00001014
-
- CPSR Register
- Negative (N): 0
- Zero (Z): 0
- Carry (C): 1
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System
-
- 0x200000df

a1.s:

```
.text
00001000:E3A00012    MOV R0,#18
00001004:E3500000    CMP R0,#0
00001008:0A000002    BEQ c1
0000100C:4A000003    BMI c2
00001010:E3A01002    MOV R1,#2
00001014:EF000011    SWI 0x11
00001018:          c1:
00001018:E3A01001          MOV R1,#1
0000101C:EF000011          SWI 0x11
00001020:          c2:
00001020:E3A01003          MOV R1,#3
00001024:EF000011          SWI 0x11
.end
```

Case 3 (negative number):

.text

MOV R0,#-18

CMP R0,#0

BEQ c1

BMI c2

MOV R1,#2

SWI 0x11

c1:

MOV R1,#1

SWI 0x11

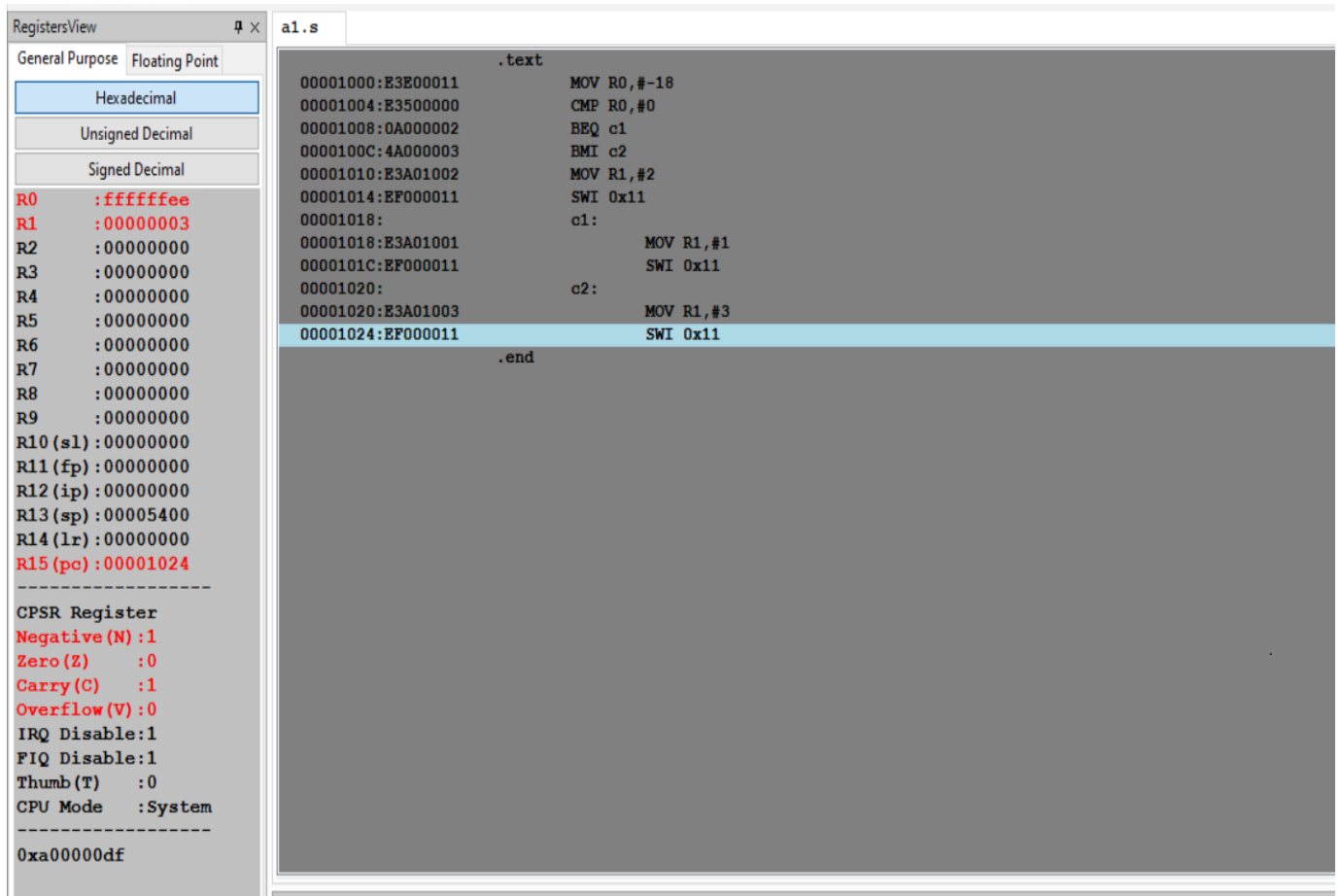
c2:

MOV R1,#3

SWI 0x11

.end

Output:



Week# 2 Program Number: 2

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

ARM Assembly Code:

Case 1:

.text

MOV R0,#13

MOV R1,#13

CMP R0,R1

BEQ equal

SUB R2,R0,R1

SWI 0x11

equal:

ADD R2,R1,R0

SWI 0x11

.end

Output:

The screenshot displays a debugger interface with two main panes. The left pane, titled 'RegistersView', shows the state of various registers. The right pane shows the assembly code for a file named 'a2.s'.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0: 0000000d
- R1: 0000000d
- R2: 0000001a
- 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
- Carry (C): 1
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System

Assembly Code (a2.s):

```
.text
00001000:E3A0000D    MOV R0,#13
00001004:E3A0100D    MOV R1,#13
00001008:E1500001    CMP R0,R1
0000100C:0A000001    BEQ equal
00001010:E0402001    SUB R2,R0,R1
00001014:EF000011    SWI 0x11
00001018:                equal:
00001018:E0812000                ADD R2,R1,R0
0000101C:EF000011                SWI 0x11
.end
```

Case 2:

.text

MOV R0,#10

MOV R1,#4

CMP R0,R1

BEQ equal

SUB R2,R0,R1

SWI 0x11

equal:

ADD R2,R1,R0

SWI 0x11

.end

Output:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'RegistersView', shows the state of various registers. The right panel, titled 'a2.s', shows the assembly code being executed.

RegistersView Panel:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0: 0000000a
- R1: 00000004
- R2: 00000006
- R3: 00000000
- R4: 00000000
- R5: 00000000
- 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): 00001014
-
- CPSR Register
- Negative (N): 0
- Zero (Z): 0
- Carry (C): 1
- Overflow (V): 0
- IRQ Disable: 1
- FIQ Disable: 1
- Thumb (T): 0
- CPU Mode: System
-
- 0x200000df

a2.s Panel:

```
.text
00001000:E3A0000A    MOV R0,#10
00001004:E3A01004    MOV R1,#4
00001008:E1500001    CMP R0,R1
0000100C:0A000001    BEQ equal
00001010:E0402001    SUB R2,R0,R1
00001014:EF000011    SWI 0x11
00001018:                equal:
00001018:E0812000                ADD R2,R1,R0
0000101C:EF000011                SWI 0x11
.end
```

Week#____2_____

Program Number: _ 3_____

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.

ARM Assembly Code:

.text

MOV R0,#3

MOV R1,R0

CMP R1,#1

BEQ j0

j1:

SUB R1,R1,#1

MUL R2,R1,R0

MOV R0,R2

CMP R1,#1

BNE j1

BEQ j0

j0:

SWI 0x11

.end

Output:

RegistersView
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 : 00000006
R1 : 00000001
R2 : 00000006
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) : 00001028
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

a3.s

```

.text
00001000:E3A00003    MOV R0,#3
00001004:E1A01000    MOV R1,R0
00001008:E3510001    CMP R1,#1
0000100C:0A000005    BEQ j0
00001010:           j1:
00001010:E2411001    SUB R1,R1,#1
00001014:E0020091    MUL R2,R1,R0
00001018:E1A00002    MOV R0,R2
0000101C:E3510001    CMP R1,#1
00001020:1AFFFFFA    BNE j1
00001024:0AFFFFFF    BEQ j0
00001028:           j0:
00001028:EF000011    SWI 0x11
.end

```

Week# 2 Program Number: 4a

Write an ALP to add two 32 bit numbers loaded from memory and store the result in memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDR R3,[R0]

LDR R4,[R1]

ADD R5,R4,R3

STR R5,[R2]

SWI 0x11

.data

A:.WORD 50

B:.WORD 100

C:.WORD 0

Output:

The screenshot displays an ARM assembly simulator interface. On the left, the 'RegistersView' window shows the state of various registers. R0 through R14 are at 0x00000000, while R15 (PC) is at 0x000101c. The CPSR register shows flags: Negative (N) is 0, Zero (Z) is 0, Carry (C) is 0, Overflow (V) is 0, IRQ Disable is 1, FIQ Disable is 1, Thumb (T) is 0, and CPU Mode is System. The main window shows the assembly code for 'a4a.s'. The .text section contains instructions to load constants A, B, and C into registers R0, R1, and R2, add R0 to R3, and store the result of R4 + R3 into R5, followed by a software interrupt (SWI 0x11). The .data section defines three words: A (50), B (100), and C (0). The instruction 'STR R5,[R2]' is highlighted in blue.

Register	Value
R0	0x000102c
R1	0x0001030
R2	0x0001034
R3	0x0000032
R4	0x0000064
R5	0x0000096
R6	0x0000000
R7	0x0000000
R8	0x0000000
R9	0x0000000
R10 (s1)	0x0000000
R11 (fp)	0x0000000
R12 (ip)	0x0000000
R13 (sp)	0x0000540
R14 (lr)	0x0000000
R15 (pc)	0x000101c

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
0x000000df

```
.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:E0845003 ADD R5,R4,R3
00001018:E5825000 STR R5,[R2]
0000101C:EF000011 SWI 0x11

.data
0000102C: A: .WORD 50
00001030: B: .WORD 100
00001034: C: .WORD 0
```

Week# ____2____

Program Number: _ 4b_

Write an ALP to add two 16 bit numbers loaded from memory and store the result in memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDRH R3,[R0]

LDRH R4,[R1]

ADD R5,R4,R3

STRH R5,[R2]

SWI 0x11

.data

A:.HWORD 5

B:.HWORD 10

C:.HWORD 0

Output:

RegistersView		a4b.s
General Purpose		
Floating Point		
Hexadecimal		
Unsigned Decimal		
Signed Decimal		
R0	:0000102a	
R1	:0000102e	
R2	:00001030	
R3	:00000005	
R4	:0000000a	
R5	:0000000f	
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)	:0000101a	

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		

0x000000df		

a4b.s	
.text	
00001000:E59F0018	LDR R0,=A
00001004:E59F1018	LDR R1,=B
00001008:E59F2018	LDR R2,=C
0000100C:E01030B0	LDRH R3,[R0]
00001010:E01140B0	LDRH R4,[R1]
00001014:E0845003	ADD R5,R4,R3
00001018:E00250B0	STRH R5,[R2]
0000101C:EF000011	SWI 0x11
.data	
0000102C:	A:.HWORD 5
0000102E:	B:.HWORD 10
00001030:	C:.HWORD 0

Week#____2_____

Program Number: _ 5a_____

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

ARM Assembly Code:

.text

MOV R0,#100

MOV R1,#5

MOV R2,R0

MOV R3,R1

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

.end

Output:

The screenshot shows an ARM assembly editor with two panes. The left pane, titled 'RegistersView', displays the state of 16 registers (R0-R15) and the CPSR register. The right pane, titled 'a5a.s', shows the assembly code for a program.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0 : 00000064
- R1 : 00000005
- R2 : 00000005
- R3 : 00000005
- 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) : 0000103c
- 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

a5a.s:

```
.text
00001000:E3A00064      MOV R0,#100
00001004:E3A01005      MOV R1,#5
00001008:E1A02000      MOV R2,R0
0000100C:E1A03001      MOV R3,R1

00001010:              L1:
00001010:E1520003      CMP R2,R3
00001014:0A000008      BEQ OUT
00001018:BA000003      BLT L2
0000101C:E0422003      SUB R2,R2,R3
00001020:E1530002      CMP R3,R2
00001024:0A000004      BEQ OUT
00001028:1AFFFFF8      BNE L1
0000102C:              L2:
0000102C:E0433002      SUB R3,R3,R2
00001030:E1530002      CMP R3,R2
00001034:0A000000      BEQ OUT
00001038:1AFFFFF4      BNE L1
0000103C:              OUT:
0000103C:EF000011      SWI 0x11

.end
```

Week# 2 Program Number: 5b

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

ARM Assembly Code:

Ex 1 (A=B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

 CMP R2,R3

 BEQ OUT

 BLT L2

 SUB R2,R2,R3

 CMP R3,R2

 BEQ OUT

 BNE L1

L2:

 SUB R3,R3,R2

 CMP R3,R2

 BEQ OUT

 BNE L1

OUT:

 SWI 0x11

.data

A:.WORD 55

B:.WORD 55

C:.WORD 0

Output:

RegistersView

General Purpose

Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 00001048
R1 : 0000104c
R2 : 00000037
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) : 0000103c

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

a5b.s

```

.text
00001000:E59F0038      LDR R0,=A
00001004:E59F1038      LDR R1,=B
00001008:E5902000      LDR R2,[R0]
0000100C:E5913000      LDR R3,[R1]
00001010:              L1:
00001010:E1520003              CMP R2,R3
00001014:0A000008              BEQ OUT
00001018:BA000003              BLT L2
0000101C:E0422003              SUB R2,R2,R3
00001020:E1530002              CMP R3,R2
00001024:0A000004              BEQ OUT
00001028:1AFFFFF8              BNE L1
0000102C:              L2:
0000102C:E0433002              SUB R3,R3,R2
00001030:E1530002              CMP R3,R2
00001034:0A000000              BEQ OUT
00001038:1AFFFFF4              BNE L1
0000103C:              OUT:
0000103C:EF000011      SWI 0x11

.data
00001048:              A: .WORD 55
0000104C:              B: .WORD 55
00001050:              C: .WORD 0

```

Ex 2 (A>B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

.data

A:.WORD 55

B:.WORD 10

C:.WORD 0

Output:

The screenshot displays a debugger interface with two main panels. The left panel, titled 'RegistersView', shows the state of various registers. The right panel, titled 'a5b.s', shows the assembly code being executed.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0: 00001048
- R1: 0000104c
- R2: 00000005
- R3: 00000005
- 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): 0000103c

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

Assembly Code (a5b.s):

```
.text
00001000:E59F0038    LDR R0,=A
00001004:E59F1038    LDR R1,=B
00001008:E5902000    LDR R2,[R0]
0000100C:E5913000    LDR R3,[R1]
00001010:           L1:
00001010:E1520003           CMP R2,R3
00001014:0A000008           BEQ OUT
00001018:BA000003           BLT L2
0000101C:E0422003           SUB R2,R2,R3
00001020:E1530002           CMP R3,R2
00001024:0A000004           BEQ OUT
00001028:1AFFFFF8           BNE L1
0000102C:           L2:
0000102C:E0433002           SUB R3,R3,R2
00001030:E1530002           CMP R3,R2
00001034:0A000000           BEQ OUT
00001038:1AFFFFF4           BNE L1
0000103C:           OUT:
0000103C:EF000011           SWI 0x11

.data
00001048:           A:.WORD 55
0000104C:           B:.WORD 10
00001050:           C:.WORD 0
```


Ex 3 (A<B):

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

L1:

CMP R2,R3

BEQ OUT

BLT L2

SUB R2,R2,R3

CMP R3,R2

BEQ OUT

BNE L1

L2:

SUB R3,R3,R2

CMP R3,R2

BEQ OUT

BNE L1

OUT:

SWI 0x11

.data

A:.WORD 20

B:.WORD 30

C:.WORD 0

Output:

The screenshot shows a debugger window titled 'RegistersView' with a tab 'a5b.s'. The left pane displays the state of ARM registers and the CPSR. The right pane shows the assembly code for the file 'a5b.s'.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal

Registers:

- R0 : 00001048
- R1 : 0000104c
- R2 : 0000000a
- R3 : 0000000a
- 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) : 0000103c

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

Assembly Code (a5b.s):

```
.text
00001000:E59F0038    LDR R0,=A
00001004:E59F1038    LDR R1,=B
00001008:E5902000    LDR R2,[R0]
0000100C:E5913000    LDR R3,[R1]
00001010:           L1:
00001010:E1520003           CMP R2,R3
00001014:0A000008           BEQ OUT
00001018:BA000003           BLT L2
0000101C:E0422003           SUB R2,R2,R3
00001020:E1530002           CMP R3,R2
00001024:0A000004           BEQ OUT
00001028:1AFFFFF8           BNE L1
0000102C:           L2:
0000102C:E0433002           SUB R3,R3,R2
00001030:E1530002           CMP R3,R2
00001034:0A000000           BEQ OUT
00001038:1AFFFFF4           BNE L1
0000103C:           OUT:
0000103C:EF000011           SWI 0x11

.data
00001048:           A:.WORD 20
0000104C:           B:.WORD 30
00001050:           C:.WORD 0
```

Week# 2 Program Number: 6a

Write an ALP to add an array of ten 32 bit numbers from memory.

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R4,[R1]

MOV R3,#0

L:

LDR R2,[R0],#4

ADD R3,R3,R2

SUB R4,R4,#1

CMP R4,#0

BEQ OUT

BNE L

OUT:SWI 0x11

.data

A:.WORD 1,3,5,7,9,11,13,15,17,19

B:.WORD 10

Output:

The screenshot displays a debugger interface with two main panes. The left pane, titled 'RegistersView', shows the state of various registers. The right pane, titled 'a6a.s', shows the assembly code being executed.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0: 0000105c
- R1: 0000105c
- R2: 00000013
- R3: 00000064
- 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): 00001028
-
- 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

a6a.s:

```
.text
00001000:E59F0024 LDR R0,=A
00001004:E59F1024 LDR R1,=B
00001008:E5914000 LDR R4,[R1]
0000100C:E3A03000 MOV R3,#0
00001010: L:
00001010:E4902004 LDR R2,[R0],#4
00001014:E0833002 ADD R3,R3,R2
00001018:E2444001 SUB R4,R4,#1
0000101C:E3540000 CMP R4,#0
00001020:0A000000 BEQ OUT
00001024:1AFFFFFF9 BNE L
00001028:EF000011 OUT:SWI 0x11

.data
00001034: A:.WORD 1,3,5,7,9,11,13,15,17,19
0000105C: B:.WORD 10
```

Week#____2_____

Program Number: _ 6b_____

Write an ALP to add array of ten 8 bit numbers taking data from memory location stored as byte data (use .byte to store the data instead of .word)

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDRB R4,[R1]

MOV R3,#0

L1:

LDRB R2,[R0]

ADD R3,R3,R2

SUB R4,R4,#1

CMP R4,#0

BEQ L2

ADD R0,R0,#1

BNE L1

L2:

SWI 0x11

.data

A:.BYTE 2,4,6,8,10

B:.BYTE 5

Output:

The screenshot shows an ARM assembly debugger interface. On the left, the 'RegistersView' window displays the state of 16 registers (R0-R15) in hexadecimal, with R15 (PC) highlighted. Below the registers, the 'CPSR Register' window shows various status flags: Negative (N): 0, Zero (Z): 1, Carry (C): 1, Overflow (V): 0, IRQ Disable: 1, FIQ Disable: 1, Thumb (T): 0, and CPU Mode: System. The main window displays the assembly code for a file named 'a6b.s'. The code is divided into two sections: '.text' and '.data'. The '.text' section contains instructions for loading data from memory, performing arithmetic operations, and branching. The '.data' section defines two memory locations, A and B, as byte arrays. The assembly code is as follows:

```
.text
00001000:E59F0028    LDR R0,=A
00001004:E59F1028    LDR R1,=B
00001008:E5D14000    LDRB R4,[R1]
0000100C:E3A03000    MOV R3,#0
00001010:                L1:
00001010:E5D02000                LDRB R2,[R0]
00001014:E0833002                ADD R3,R3,R2
00001018:E2444001                SUB R4,R4,#1
0000101C:E3540000                CMP R4,#0
00001020:0A000001                BEQ L2
00001024:E2800001                ADD R0,R0,#1
00001028:1AFFFF8                BNE L1
0000102C:                L2:
0000102C:EF000011                SWI 0x11

.data
00001038:                A: .BYTE 2,4,6,8,10
0000103D:                B: .BYTE 5
```

Week# ____2____

Program Number: _ 7 _

Write an ALP to multiply using barrel shifter.

35*R0

ARM Assembly Code:

.text

MOV R0,#10

MOV R1,R0,LSL #5

ADD R2,R0,R0,LSL #1

ADD R3,R1,R2

SWI 0x11

.end

Output:

The screenshot shows the ARM RegistersView window with the 'General Purpose' tab selected. The 'Hexadecimal' view is chosen. The registers R0 through R15 are listed with their values in hexadecimal. R0 is 0000000a, R1 is 00000140, R2 is 0000001e, R3 is 0000015e, and R15 (pc) is 00001010. The CPSR Register is also shown with various flags set. The assembly code window on the right shows the following code:

```
.text
00001000:E3A0000A    MOV R0,#10
00001004:E1A01280    MOV R1,R0,LSL #5
00001008:E0802080    ADD R2,R0,R0,LSL #1
0000100C:E0813002    ADD R3,R1,R2
00001010:EF000011    SWI 0x11
.end
```

Week# 2 Program Number: 8

Write an ALP to evaluate the expression $(A+B) + (3*B)$, where A and B are memory location.

* Use LSL instruction for multiplication

ARM Assembly Code:

.text

LDR R0,=A

LDR R1,=B

LDR R2,[R0]

LDR R3,[R1]

ADD R5,R2,R3,LSL #2

SWI 0x11

.data

A:.WORD 2

B:.WORD 10

Output:

The screenshot displays a debugger interface with two main panes. The left pane, titled 'RegistersView', shows the state of 16 registers (R0-R15) and the CPSR register. The right pane shows the assembly code for a file named 'a8.s'.

RegistersView:

- General Purpose: Floating Point
- Hexadecimal
- Unsigned Decimal
- Signed Decimal
- R0: 00001020
- R1: 00001024
- R2: 00000002
- R3: 0000000a
- R4: 00000000
- R5: 0000002a
- 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): 00001014
- 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
- 0x000000df

Assembly Code (a8.s):

```
.text
00001000:E59F0010    LDR R0,=A
00001004:E59F1010    LDR R1,=B
00001008:E5902000    LDR R2,[R0]
0000100C:E5913000    LDR R3,[R1]
00001010:E0825103    ADD R5,R2,R3,LSL #2
00001014:EF000011    SWI 0x11

.data
00001020:                A:.WORD 2
00001024:                B:.WORD 10
```

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: Tushar

Name: TUSHAR Y S

SRN: PES1UG19CS545

Section: I

Date: 1/2/2021