

# MPCA LAB – 3

Name : Y Uday Sampreeth Reddy

SRN : PES1UG21CS736

Section : L

Roll No : 5

1. Generate Fibonacci Series and store them in an array.

.DATA

N: .WORD 10

A: .WORD

.TEXT

LDR R5,=N

LDR R4,[R5]

LDR R6,=A

MOV R0,#0

MOV R1,#1

STR R0,[R6],#4

STR R1,[R6],#4

SUB R4,R4,#1

loop:

ADD R3,R0,R1

MOV R1,R0

MOV R0,R3

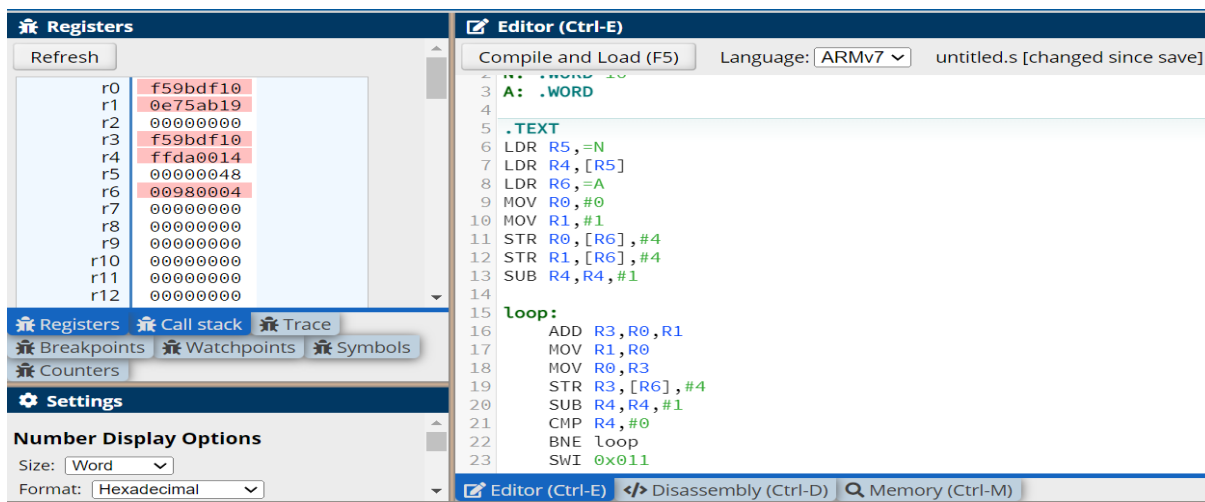
STR R3,[R6],#4

SUB R4,R4,#1

CMP R4,#0

BNE loop

SWI 0x011

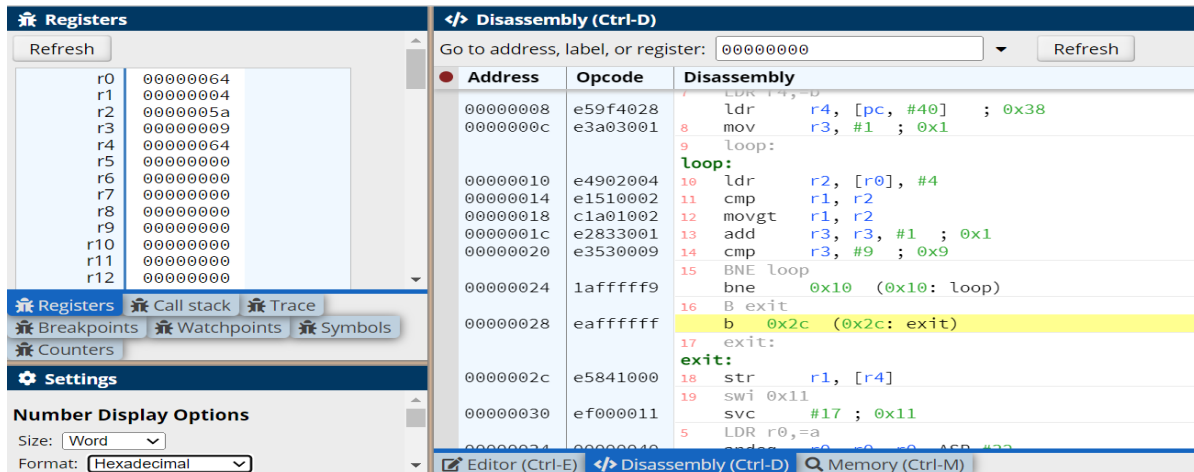


2. Write an ALP to find smallest number in an array of n 32 bit numbers

.DATA

a: .word 16,10,32,52,4,9,20,13,90

```
        b: .word -1
.TEXT
        LDR r0,=a
        LDR r1,[r0],#4
        LDR r4,=b
        MOV r3,#1
loop:
        LDR r2,[r0],#4
        CMP r1,r2
        MOVGT r1,r2
        ADD r3,r3,#1
        CMP r3,#9
        BNE loop
        B exit
exit:
        STR r1,[r4]
        swi 0x11
.end
```



3. To perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)

.DATA

A: .WORD 10,20,30,40,50,60,70,80,90,100

B: .WORD 1,2,3,4,5,6,7,8,9,10

C: .WORD

N: .WORD 10

.TEXT

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDR R8,=N

LDR R9,[R8]

loop:

LDR R4,[R0],#4

LDR R5,[R1],#4

MUL R6,R4,R5

ADD R7,R6,R7

STR R7,[R2],#4

SUB R9,R9,#1

CMP R9,#0

BNE loop

SWI 0x011

The screenshot shows a debugger window with two main panes. The left pane, titled 'Registers', lists registers r0 through r12 with their current values in hexadecimal. The right pane, titled 'Disassembly (Ctrl-D)', shows a list of instructions with their addresses, opcodes, and disassembled code. The instruction at address 00000034 is highlighted in yellow.

Address	Opcode	Disassembly
0000000c	e59f8030	ldr r8, [pc, #48] ; 0x44
00000010	e5989000	ldr r9, [r8]
00000014	e4904004	ldr r4, [r0], #4
00000018	e4915004	ldr r5, [r1], #4
0000001c	e0060594	mul r6, r4, r5
00000020	e0867007	add r7, r6, r7
00000024	e4827004	str r7, [r2], #4
00000028	e2499001	sub r9, r9, #1 ; 0x1
0000002c	e3590000	cmp r9, #0 ; 0x0
00000030	1afffff7	bne 0x14 (0x14: loop)
00000034	ef000011	svc #17 ; 0x11
00000038	00000048	LDR R0,=A
0000003c	00000070	andeq r0, r0, r8, ASR #32
		LDR R1,=B
		andeq r0, r0, r0, ROR r0

4. To perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

.DATA

A: .WORD 10,20,30,40,50,60,70,80,90,100

B: .WORD 1,2,3,4,5,6,7,8,9,10

C: .WORD

N: .WORD 10

.TEXT

LDR R0,=A

LDR R1,=B

LDR R2,=C

LDR R8,=N

LDR R9,[R8]

loop:

LDR R4,[R0],#4

LDR R5,[R1],#4

MLA R6,R4,R5,R7

MOV R7,R6

STR R7,[R2],#4

SUB R9,R9,#1

CMP R9,#0

BNE loop

SWI 0x011

The screenshot shows a debugger window with two main panes. The left pane, titled 'Registers', displays a list of registers from r0 to r12, each with a hexadecimal value. The right pane, titled 'Disassembly (Ctrl-D)', shows a list of instructions with their addresses, opcodes, and disassembled code. The instructions are as follows:

Address	Opcode	Disassembly
0000000c	e59f8030	ldr r8, [pc, #48] ; 0x44
00000010	e5989000	ldr r9, [r8]
00000014	e4904004	ldr r4, [r0], #4
00000018	e4915004	ldr r5, [r1], #4
0000001c	e0267594	mla r6, r4, r5, r7
00000020	e1a07006	mov r7, r6
00000024	e4827004	str r7, [r2], #4
00000028	e2499001	sub r9, r9, #1 ; 0x1
0000002c	e3590000	cmp r9, #0 ; 0x0
00000030	1afffff7	bne 0x14 (0x14: loop)
00000034	ef000011	swi 0x011
00000038	00000048	andeq r0, r0, r8, ASR #32
0000003c	00000070	ldr r1, =B

5. Write an ALP to find mul(add(a,b),c)

.data

a: .word 0

stk: .word 0

.text

LDR r0,=a

MOV r1,#10

MOV r2,#20

MOV r3,#30

BL mulADD /\*mul(add(10,20),30)\*/

STR r6,[r0]

B exit

mulADD:

LDR r4,=stk

STR LR,[r4]

BL add

MUL r6,r5,r3

LDR LR,[r4]

MOV PC,LR

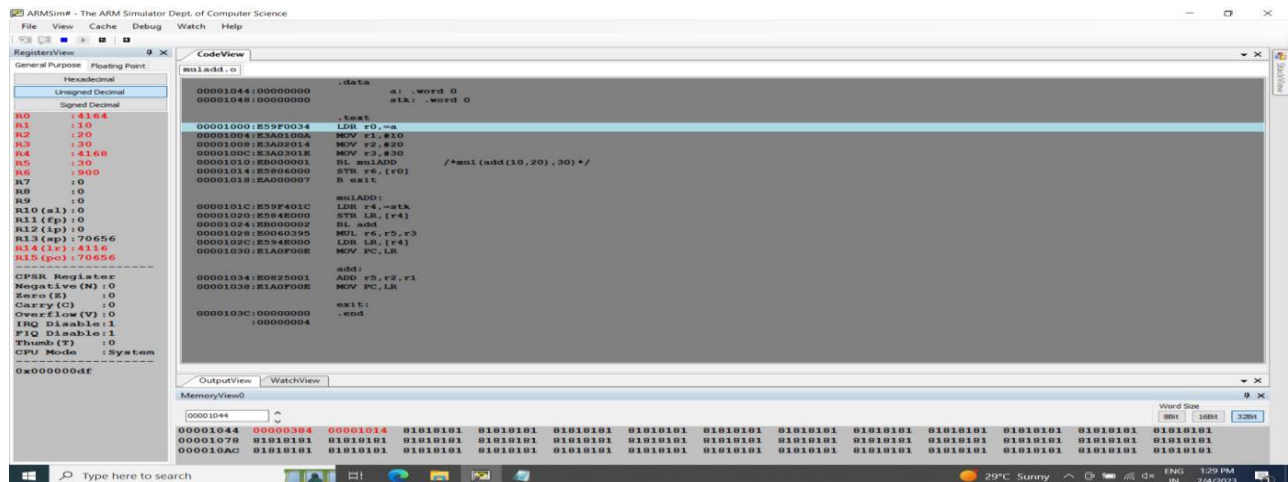
add:

ADD r5,r2,r1

MOV PC,LR

exit:

.end



## 6. Write an ALP to find factorial using subroutine

.data

a: .word 0

.text

LDR r0,=a

MOV r1,#10

BL fact

STR r2,[r0]

B exit

fact:

MOV r2,#1

loop:

MUL r2,r2,r1

SUB r1,r1,#1

CMP r1,#0

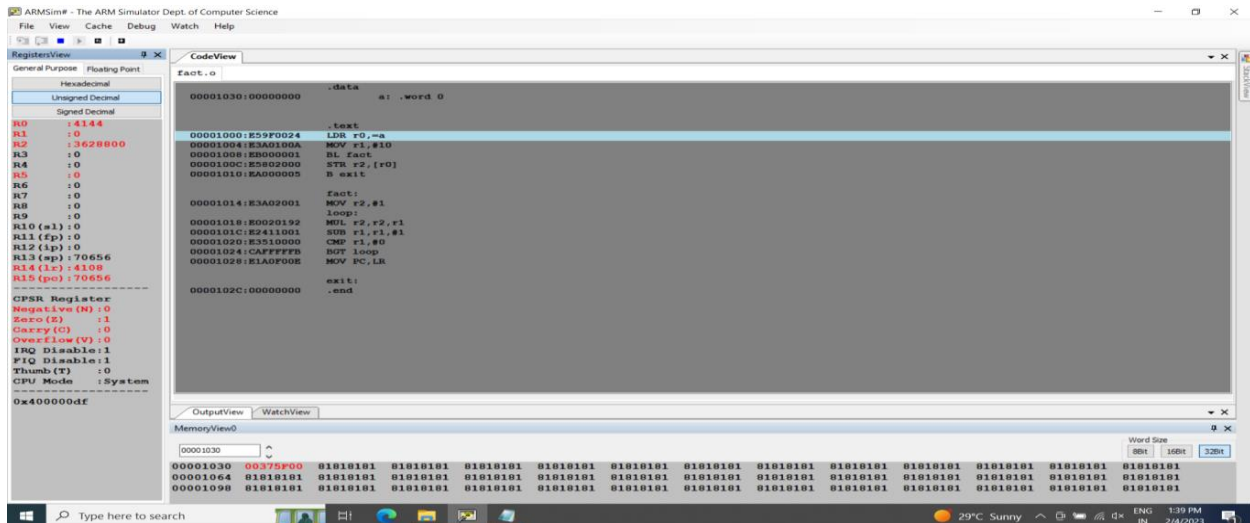
BGT loop



MOV PC,LR

exit:

.end



7. Write an ALP to perform multiplication using shift method (without using MUL)

mov r0,#5

mov r1,#9

mov r2,#0

loop:

cmp r1,#0

ble exit

tst r1,#1

addne r2,r2,r0

mov r0,r0,lsl #1

mov r1,r1,lsr #1

b loop

exit:

swi 0x011

The screenshot shows a development environment for ARMv7 assembly. On the left is a 'Registers' panel with a 'Refresh' button and a table of 13 registers (r0-r12). Register r0 contains the value 00000050, while all other registers are 00000000. Below the registers are tabs for 'Registers', 'Call stack', 'Trace', 'Breakpoints', 'Watchpoints', 'Symbols', and 'Counters'. A 'Settings' section at the bottom left shows 'Number Display Options' with 'Size' set to 'Word' and 'Format' set to 'Hexadecimal'. The main 'Editor (Ctrl-E)' window on the right contains assembly code for a program. The code includes a loop that shifts register r1 right by one bit and increments register r2 until r1 reaches zero, followed by a system call (swi 0x011) to exit. The language is set to 'ARMv7' and the file is named 'untitled.s [changed since save]'. At the bottom, there are tabs for 'Editor (Ctrl-E)', 'Disassembly (Ctrl-D)', and 'Memory (Ctrl-M)'.

Register	Value
r0	00000050
r1	00000000
r2	00000000
r3	00000000
r4	00000000
r5	00000000
r6	00000000
r7	00000000
r8	00000000
r9	00000000
r10	00000000
r11	00000000
r12	00000000

```
1  mov r0,#5
2  mov r1,#9
3  mov r2,#0
4
5  loop:
6      cmp r1,#0
7      ble exit
8      tst r1,#1
9      addne r2,r2,r0
10     mov r0,r0,lsr #1
11     mov r1,r1,lsr #1
12     b loop
13 exit:
14     swi 0x011
15
16
```