



Department of Computer Science & Engineering (AI & ML)
Microprocessor & Computer Architecture Lab

Lab 2 Programs

UE23CS251B

- 1 Write an ALP using ARM7TDMI to perform to multiplication of 16X25 without using MUL instructions.
(Hint: barrel shifter instructions.)
(Note :any number can be considered as multiplier)

Program screen shot:

.text

.global _start

_start:

MOV R0, #16

MOV R1, #25

MOV R2, #0

@25 = 16 + 8 + 1

@ Ans = 16*16 + 16*8 +16*1

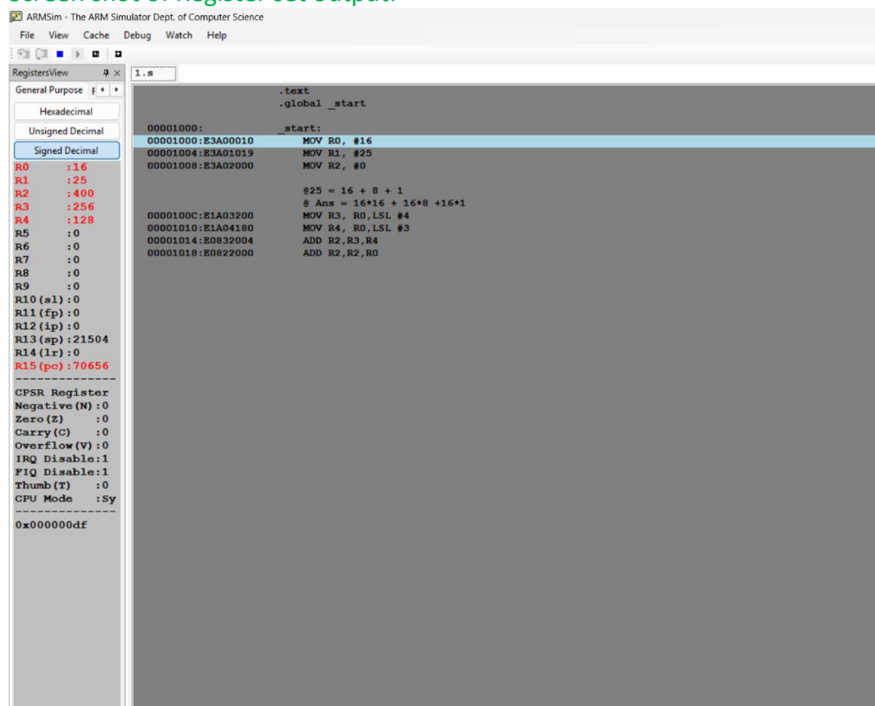
MOV R3, R0,LSL #4

MOV R4, R0,LSL #3

ADD R2,R3,R4

ADD R2,R2,R0

Screen shot of Register set output:



- 2 Write an ALP using ARM7TDMI to add only even numbers stored in memory location for a given set of numbers and store the sum in the memory location.

Array:.WORD 15,10,12,13,9,45,16,8,25,33
evensum:.WORD

Program screen shot:

.data

Array: .word 15,10,12,13,9,45,16,8,25,33
evensum: .word 0

.text

.global _start

_start:

LDR R0,=Array

MOV R1,#10 @Array size

MOV R2,#0 @Sum register

loop:

LDR R3,[R0],#4 @using post increment mode

TST R3,#1

BNE skip

ADD R2,R2,R3

skip:

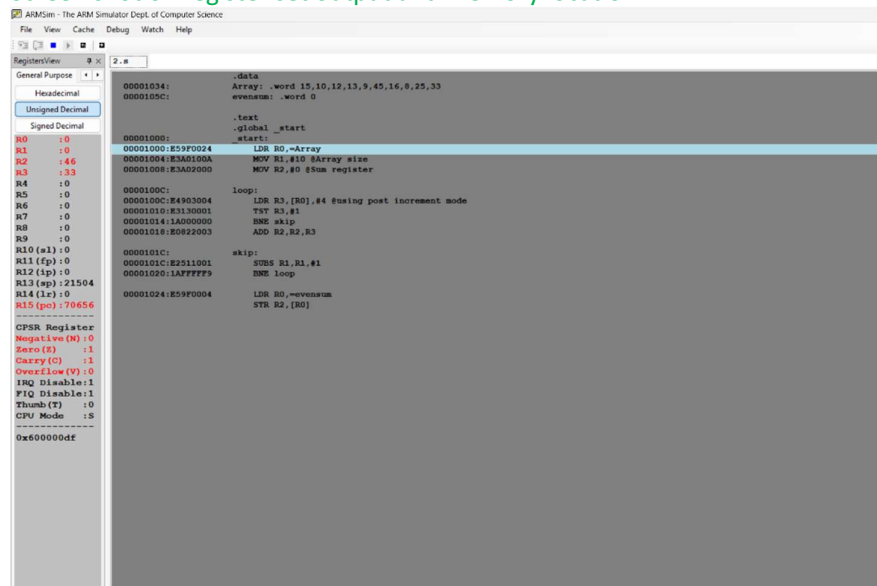
SUBS R1,R1,#1

BNE loop

LDR R0,=evensum

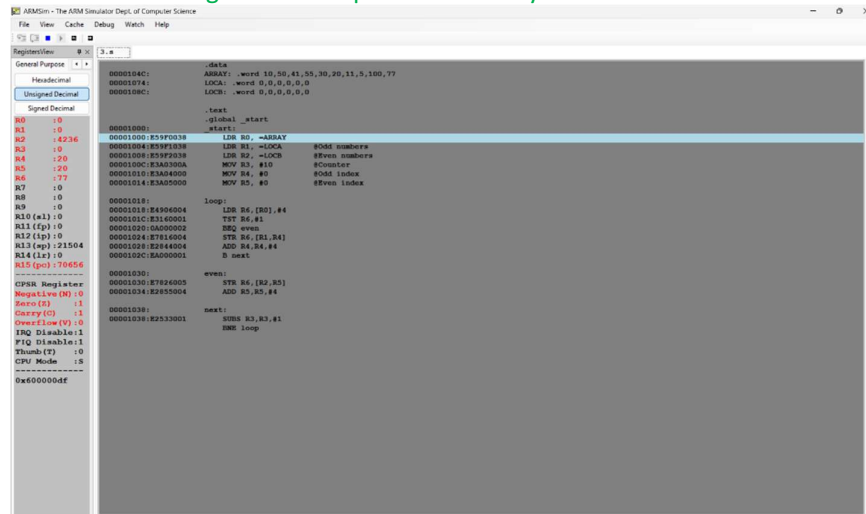
STR R2,[R0]

Screen shot of Register set output and memory location:



3	<p>Write a ALP using ARMTDMI-ISA to store odd and even numbers in separate memory locations starting from LOCA and LOCB respectively</p> <pre> ARRAY: .word 10,50,41,55,30,20,11,5,100,77 LOCA: .word 0,0,0,0,0,0 LOCB: .word 0,0,0,0,0,0 Program screen shot: .data ARRAY: .word 10,50,41,55,30,20,11,5,100,77 LOCA: .word 0,0,0,0,0,0 LOCB: .word 0,0,0,0,0,0 .text .global _start _start: LDR R0, =ARRAY LDR R1, =LOCA @Odd numbers LDR R2, =LOCB @Even numbers MOV R3, #10 @Counter MOV R4, #0 @Odd index MOV R5, #0 @Even index loop: LDR R6,[R0],#4 TST R6,#1 BEQ even STR R6,[R1,R4] ADD R4,R4,#4 B next even: STR R6,[R2,R5] ADD R5,R5,#4 next: SUBS R3,R3,#1 BNE loop </pre>
---	--

Screen shot of Register set output and memory location:



- 4 Write an ALP using ARM7TDMI to find the largest number from a given set of numbers:

A: .word 10,50,41,55,30,20,11,5,100,77

Program screen shot:

.data

A: .WORD 10,50,41,55,30,20,11,5,100,77

.text

.global _start

_start:

LDR R0,=A @load A

MOV R1,#10 @count

MOV R3,#0 @greatest element

L1:

LDR R2,[R0],#4

SUB R1,R1,#1

CMP R3,R2

BLT L2

CMP R1,#0

BNE L1

BEQ L3

L2:

MOV R3,R2

CMP R1,#0

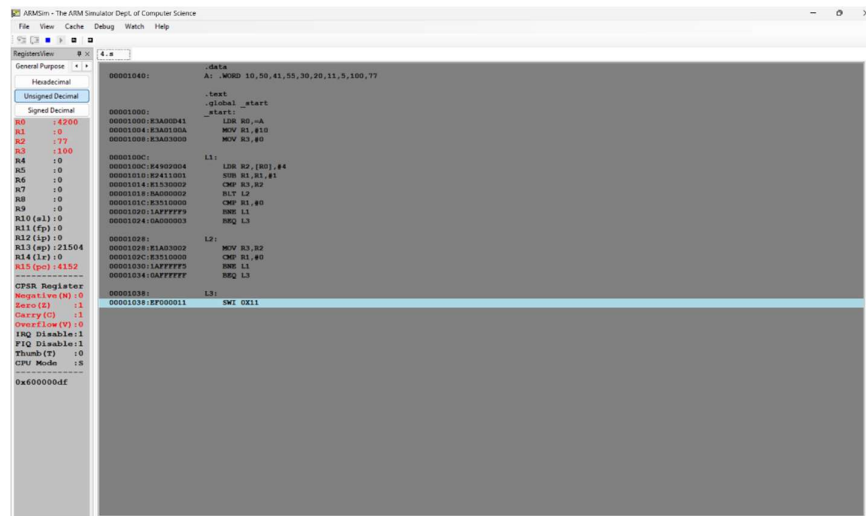
BNE L1

BEQ L3

L3:

SWI 0x11

Screen shot of Register set output and memory location:



Assignments Questions

- Write an ALP using ARM7TDMI to find whether the given number is even parity.

Program screen shot:

```
.text
.global _start
```

_start:

```
MOV R0,#3 @the number to check
MOV R1,#0 @ counter
MOV R2,#32 @ Bit counter
```

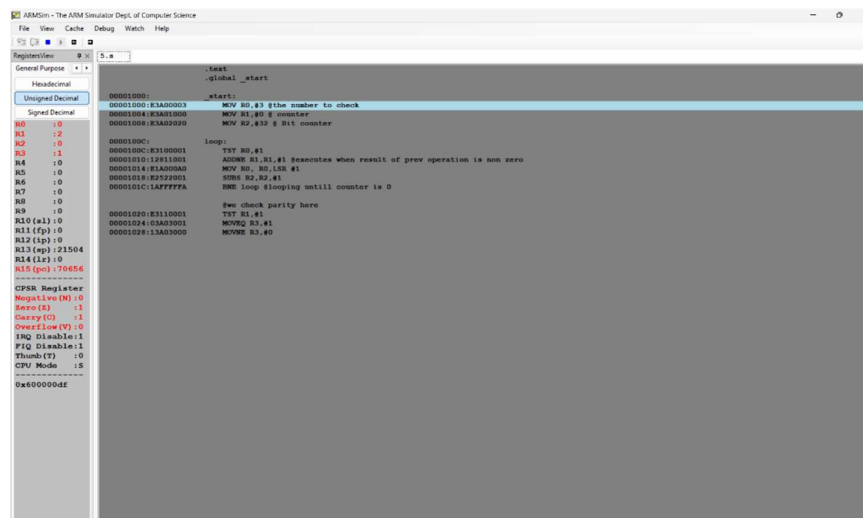
loop:

```
TST R0,#1
ADDNE R1,R1,#1 @executes when result of prev operation is non zero
MOV R0, R0,LSR #1
SUBS R2,R2,#1
BNE loop @looping untill counter is 0
```

@we check parity here

```
TST R1,#1
MOVEQ R3,#1
MOVNE R3,#0
```

Screen shot of Register set output:



- 6 Write an ALP using ARM7TDMI to multiplication of 38X72 without using MUL instructions.
(Hint: barrel shifter instructions.)
(Note :any number can be considered as multiplier)

Program screen shot:

```
.text
.global _start

_start:
    @38X72 multiplication
    @do not use MUL
    @method 1: try using 2#pow method
    MOV R0,#38 @first number
    MOV R1,#72 @second number
    MOV R5,R0 @store first number in R5
    MOV R0,#0 @result
    @72 = 64 + 8 ( 2^6 + 2^3)
    @38*72 = 38*64 + 38*8

    MOV R3,R5,LSL #6 @38*64
    MOV R4,R5,LSL #3 @38*8
    ADD R2,R3,R4 @final result in R2
```

Screen shot of Register set output:

The screenshot displays the ARMv8-M simulator interface. The top menu bar includes File, View, Cache, Debug, and Help. The main window is divided into two panes. The left pane shows the Register File with the following values:

Register	Value
R0	0
R1	72
R2	2736
R3	2432
R4	384
R5	38
R6	0
R7	0
R8	0
R9	0
R10	0
R11	0
R12	0
R13	21504
R14	0
R15	70656

The right pane shows the assembly code with the following instructions:

```

;task
;global _start
_start:
    @32x12 multiplication
    @do not use MUL
    @method 1: try using 28pow method
    MOV R0,R1 @first number
    MOV R1,R2 @second number
    MOV R2,R3 @store first number in R5
    MOV R0,R0 @result
    @R2 = R4 * 3 (2^4 * 3^3)
    @32x12 = 3844 * 384
    MOV R3,R5,LSL #6 @38^64
    MOV R4,R5,LSL #3 @38^3
    ADD R2,R3,R4 @final result in R2
    
```

The bottom status bar shows the CPU Mode as 0x0000000d.