PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 3 submission

Name of the Student	Pranav Hemanth
SRN	PES1UG23CS433
Section	G
Department	CSE
Campus	RR

Jan -May 2025 LAB SUBMISSION_**UE23CS251B**

Department of Computer Science & Engineering
Microprocessor & Computer Architecture Lab

UE23CS251B

1 Write an ALP using ARM7TDMI to find the remainder of a number.(ie 10/3, remainder is 1)

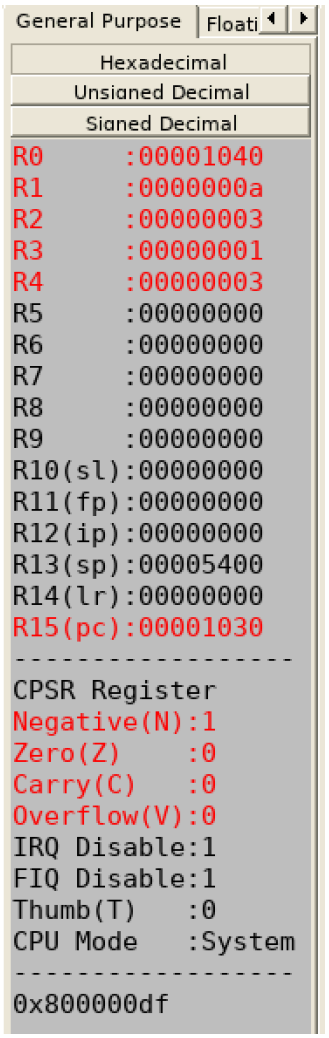
.DATA
A: .word 10
B: .word 3

Program screen shot:

ARMSim_files > lab3_q1.s

```
1   @ Write an ALP using ARM7TDMI to find the remainder of a number.(ie 10/3, remainder is 1)
2
3   .data
4       A: .word 10
5       B: .word 3
6       R: .word 0
7
8   .text
9
10  init:
11      LDR R0, =A
12      LDR R1, [R0]
13      LDR R0, =B
14      LDR R2, [R0]
15
16  start:
17      MOV R3, R1
18      MOV R4, #0
19
20  Division_Loop:
21      CMP R3, R2
22      BLT Store_Result
23      SUB R3, R3, R2
24      ADD R4, R4, #1
25
26      B Division_Loop
27
28  Store_Result:
29      STR R3, [R0]
30
31  end:
32      SWI 0x011
33
```

Screen shot of Register set output:



2 Write an ALP using ARM7TDMI to search for an element in an array of 16 bit each using Linear search technique

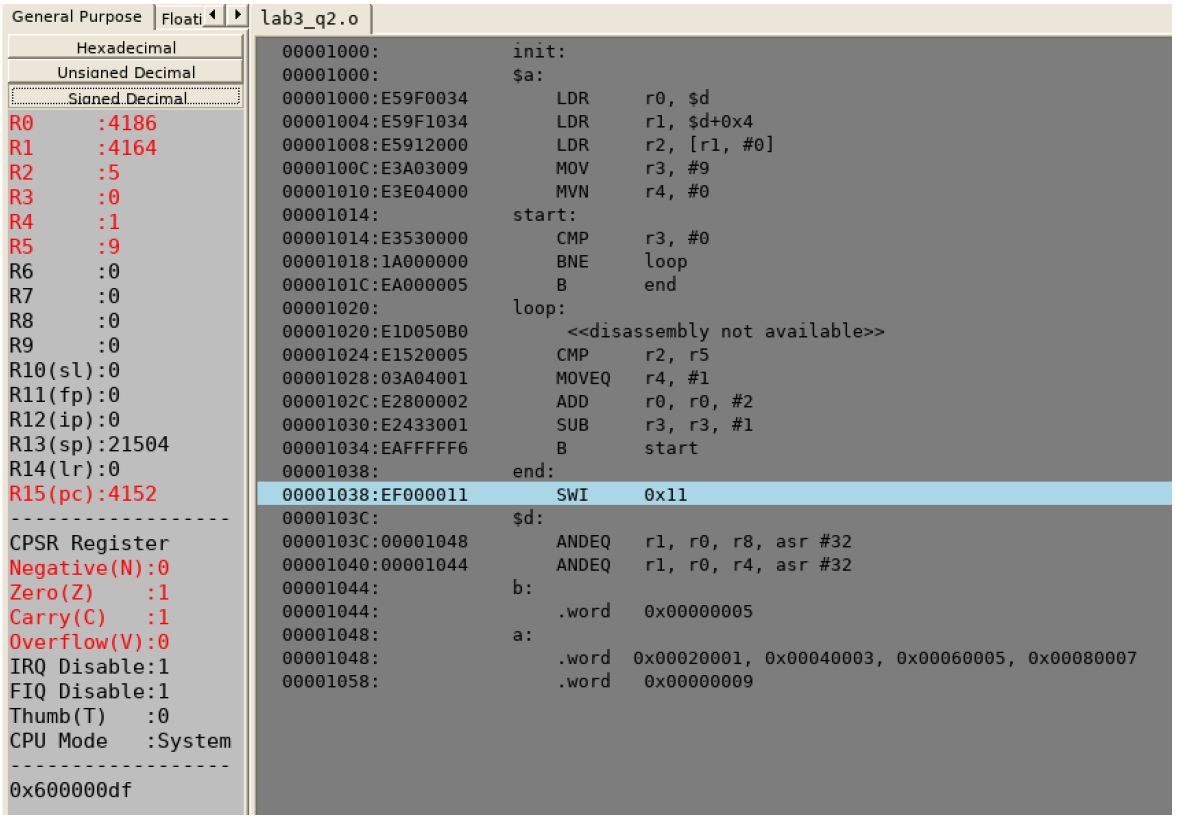
.DATA
A:.hword 1,2,3,4,5,6,7,8,9

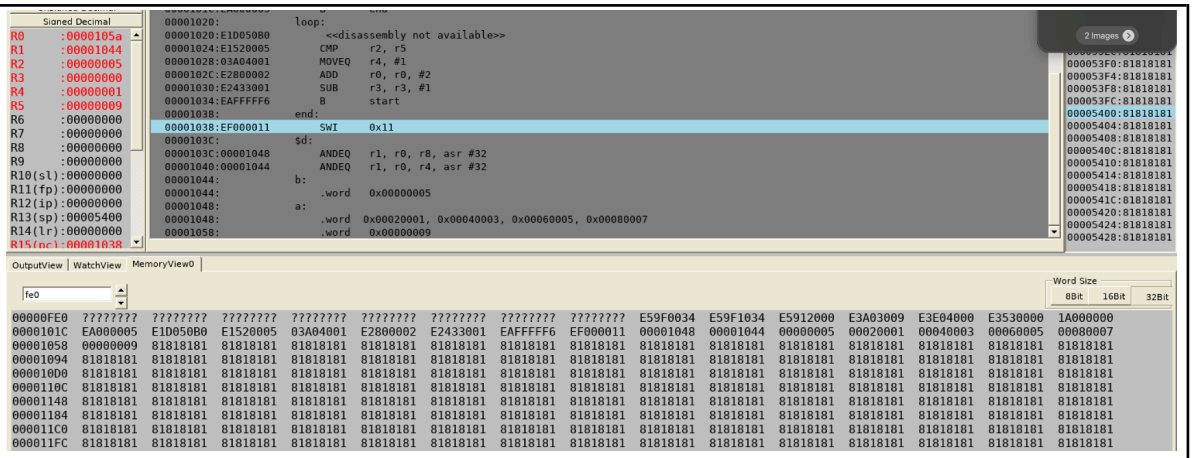
Program screen shot:

ARMSim_files > lab3_q2.s

```
1   @ Write an ALP using ARM7TDMI to search for an element in an array of 16 bit each using Linear search technique
2   @ Store -1 in R4 if not found, else 1
3
4   .data
5   B:.word 5
6   A:.hword 1,2,3,4,5,6,7,8,9
7
8   .text
9
10  init:
11      LDR R0, =A
12      LDR R1, =B
13      LDR R2, [R1]
14      MOV R3, #9
15      MOV R4, #-1
16
17  start:
18      CMP R3, #0
19      BNE loop
20      B end
21
22  loop:
23      LDRH R5, [R0]
24      CMP R2, R5
25      MOVEQ R4, #1
26      ADD R0, R0, #2
27      SUB R3, R3, #1
28      B start
29
30  end:
31      SWI 0x011
32
```

Screen shot of Register set output and memory location:





3 Write an ALP using ARM7TDMI to to copy a block 128 bytes of data from location A to location B if the rate of data transfer rate is 16 bytes, LDM and STM instructions.

and

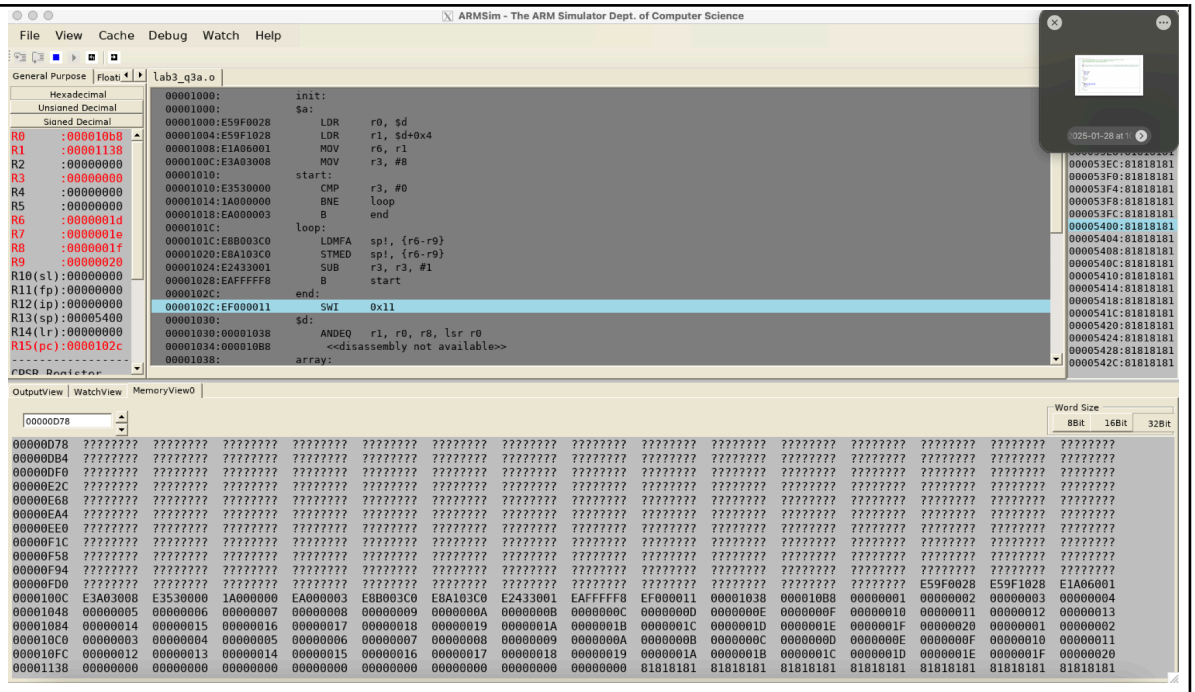
For the same transfer the block with auto-indexing.

Program screen shot:

ARMSim_files > ASM lab3_q3a.s

```
1  @ Write an ALP using ARM7TDMI to to copy a block 128 bytes of data from location A to location B
2  @ if the rate of data transfer rate is 16 bytes, LDM and STM instructions.
3  @ And For the same transfer the block with auto-indexing.
4
5  .data
6  array: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
7  copy: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
8
9  .text
10
11 init:
12     LDR R0, =array
13     LDR R1, =copy
14     MOV R6, R1
15     MOV R3, #8
16
17 start:
18     CMP R3, #0
19     BNE loop
20     B end
21
22 loop:
23     LDMIA R0!, {R6, R7, R8, R9}
24     STMIA R1!, {R6, R7, R8, R9}
25     SUB R3, R3, #1
26     B start
27
28 end:
29     SWI 0x011
30
```

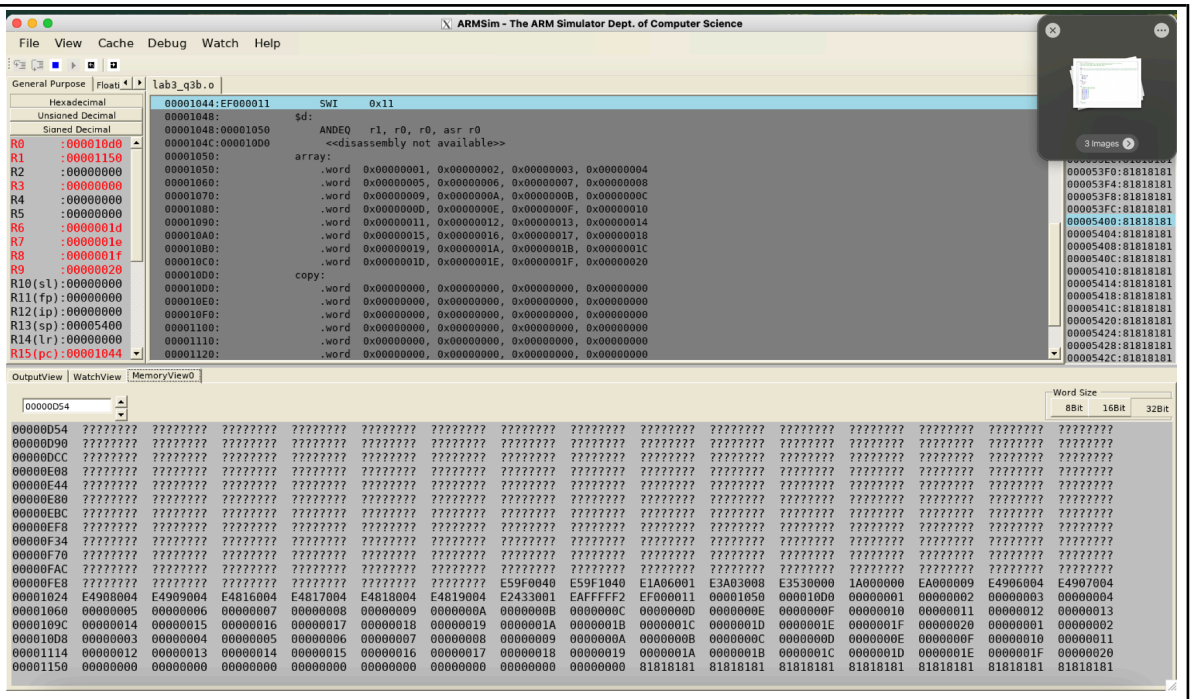
Screen shot of Register set output and memory location:



2nd approach:

ARMSim_files > ASM lab3_q3b.s

```
1  @ Write an ALP using ARM7TDMI to to copy a block 128 bytes of data from location A to location B
2  @ if the rate of data transfer rate is 16 bytes, LDM and STM instructions.
3  @ And For the same transfer the block with auto-indexing.
4
5  .data
6  array: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
7  copy: .word 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
8
9  .text
10
11 init:
12     LDR R0, =array
13     LDR R1, =copy
14     MOV R6, R1
15     MOV R3, #8
16
17 start:
18     CMP R3, #0
19     BNE loop
20     B end
21
22 loop:
23     @ LDMIA R0!, {R6, R7, R8, R9}
24     @ STMIA R1!, {R6, R7, R8, R9}
25     LDR R6, [R0], #4
26     LDR R7, [R0], #4
27     LDR R8, [R0], #4
28     LDR R9, [R0], #4
29     STR R6, [R1], #4
30     STR R7, [R1], #4
31     STR R8, [R1], #4
32     STR R9, [R1], #4
33     SUB R3, R3, #1
34     B start
35
36 end:
37     SWI 0x011
38
```



4 Write an ALP using ARM7TDMI, for the given matric arranged in row major order, find the index of an element if coordinates of a matrix is given and also find the address of the indexed element. (Using MLA instruction)

1	00000000
2	00000001
3	
4	
5	
6	
7	
8	00000020
9	

A = 1 2 3 / 4 5 6 / 7 8 9 => A(0,0) A(0,1) A(0,2) / A(1,0) A(1,1) A(1,2) / A(2,0) A(2,1) A(2,2)

Row Major Order for a 3 x 3 matrix :

A [2, 2] = Row Num. x No. of elements per row + Column Num.

2 x 3 + 2 = 8th location.

- Since each location is of 4 bytes, the address of the 3rd element in the 2nd
- row is
  8 x 4 = 32.
- This is achieved in ARM processor, using MLA instruction as follows.

MLA R1, R2, R3, R4

Mnemonic Dst Row # # of elements per row col #

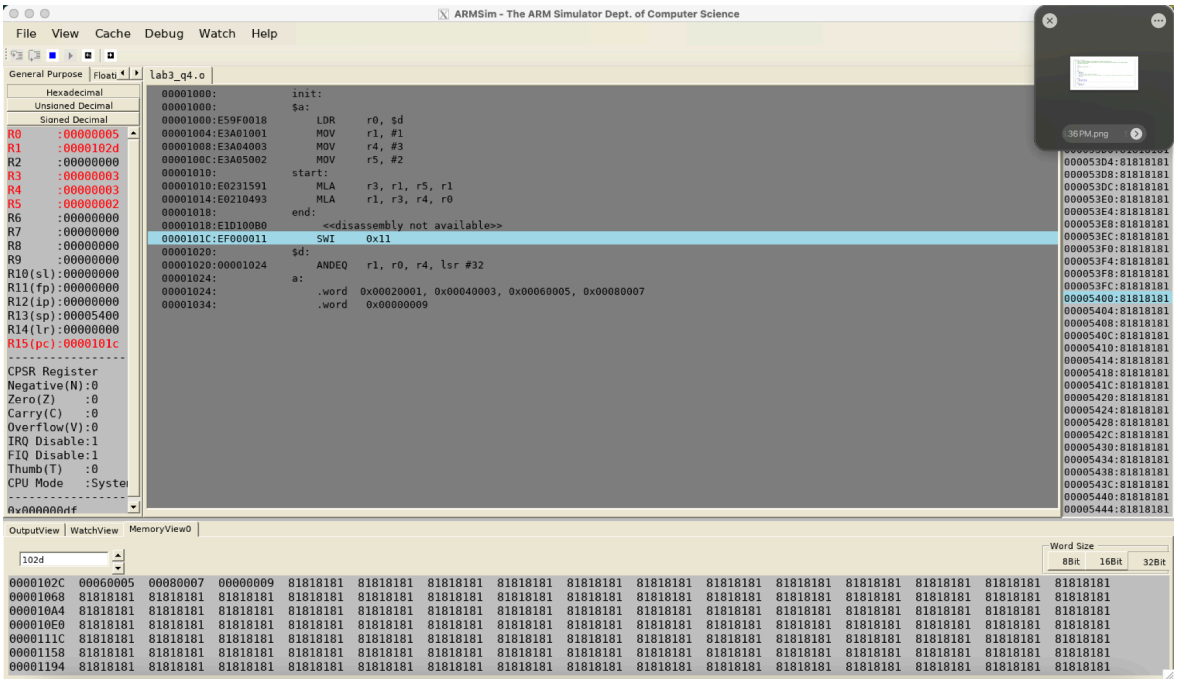
- To get consecutive data elements in a @D Array.

Program screen shot:

ARMSim_files > lab3_q4.s

```
1   @ Write an ALP using ARM7TDMI, for the given matrix arranged in row major order,
2   @ find the index of an element if coordinates of a matrix is given and also find the address of the indexed element.
3   @ (Using MLA instruction)
4
5   .data
6   A:.hword 1,2,3,4,5,6,7,8,9
7
8   .text
9
10  init:
11      LDR R0, =A
12      MOV R1, #1
13      @ Lets assume [i,j] = [2,2] as an example
14      MOV R4, #3  @ This is set to 3 and not 2 eventhough i = 2. This is because indexing starts from 0, we hv to correct for that
15      MOV R5, #2
16
17  start:
18      MLA R3, R1, R5, R1
19      MLA R1, R3, R4, R0
20
21  end:
22      LDRH R0, [R1]
23      SWI 0x011
24
```

Screen shot of Register set output and memory location:



Assignments Questions

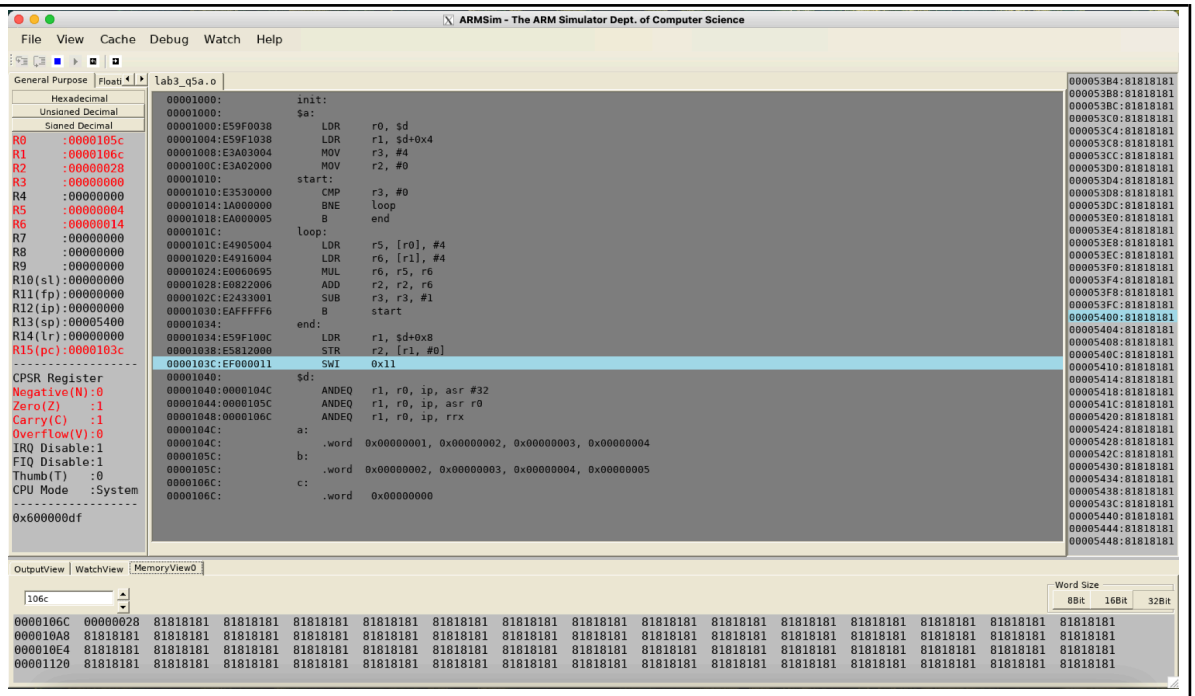
5 a)Write an ALP using ARM7TDMI to perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)
b Write an ALP using ARM7TDMI to perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

Program screen shot:

ARMSim_files > lab3_q5a.s

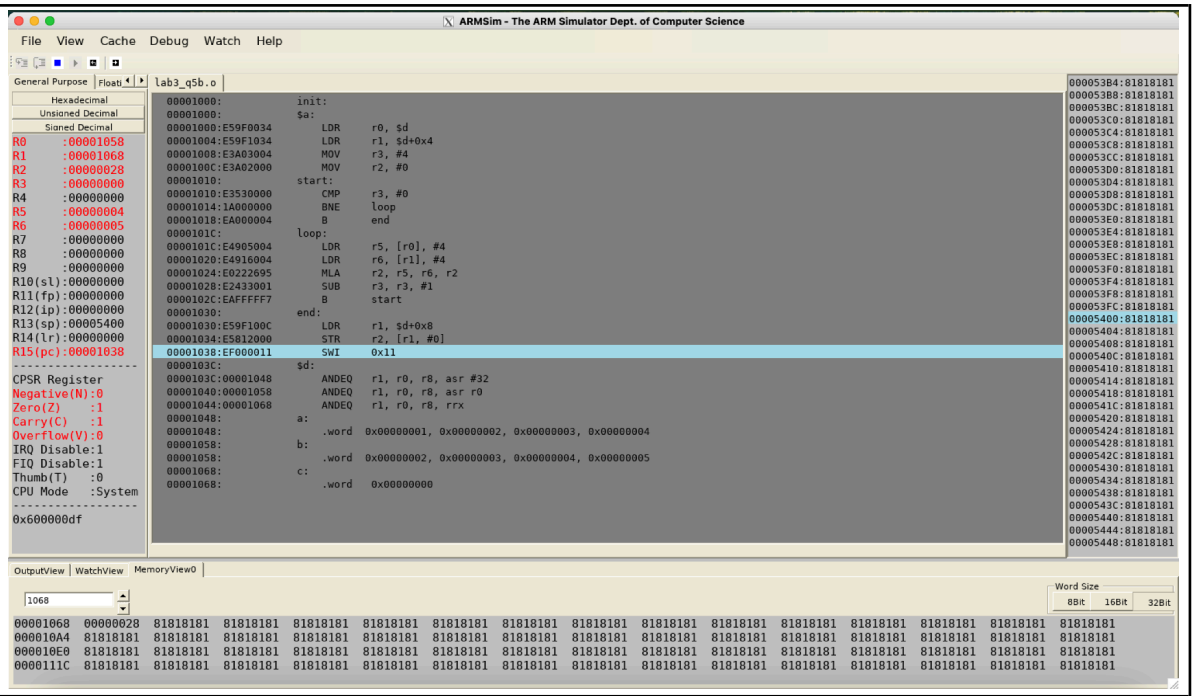
```
1   @ a )Write an ALP using ARM7TDMI  to perform Convolution using MUL  instruction
2   @ (Addition of multiplication of respective numbers of loc A and loc B)
3
4   .data
5   A:      .word 1,2,3,4
6   B:      .word 2,3,4,5
7   C:  .WORD 0
8
9   .text
10
11  init:
12      LDR R0, =A
13      LDR R1, =B
14      MOV R3, #4
15      MOV R2, #0
16
17  start:
18      CMP R3, #0
19      BNE loop
20      B end
21
22  loop:
23      LDR R5,[R0],#4
24      LDR R6,[R1],#4
25      MUL R6,R5,R6
26      ADD R2,R2,R6
27      SUB R3,R3,#1
28      B start
29
30  end:
31      LDR R1,=C
32      STR R2,[R1]
33      SWI 0x011
34
```

Screen shot of Register set output:


ARMSim - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
General Purpose
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0000105c
R1 :0000106c
R2 :00000028
R3 :00000000
R4 :00000000
R5 :00000004
R6 :00000014
R7 :00000000
R8 :00000000
R9 :00000000
R10(sl):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
lab3_q5a.o
00001000: init:
00001000: $a:
00001000:E59F0038 LDR r0, $d
00001004:E59F1038 LDR r1, $d+0x4
00001008:E3A03004 MOV r3, #4
0000100C:E3A02000 MOV r2, #0
00001010: start:
00001010:E3530000 CMP r3, #0
00001014:1A000000 BNE loop
00001018:EA000005 B end
0000101C: loop:
0000101C:E4905004 LDR r5, [r0], #4
00001020:E4916004 LDR r6, [r1], #4
00001024:E0060695 MUL r6, r5, r6
00001028:E0822006 ADD r2, r2, r6
0000102C:E2433001 SUB r3, r3, #1
00001030:EAFFFFF6 B start
00001034: end:
00001034:E59F100C LDR r1, $d+0x8
00001038:E5812000 STR r2, [r1, #0]
0000103C:EF000011 SWI 0x11
00001040: $d:
00001040:0000104C ANDEQ r1, r0, ip, asr #32
00001044:0000105C ANDEQ r1, r0, ip, asr r0
00001048:0000106C ANDEQ r1, r0, ip, rrx
0000104C: a:
0000104C: .word 0x00000001, 0x00000002, 0x00000003, 0x00000004
0000105C: b:
0000105C: .word 0x00000002, 0x00000003, 0x00000004, 0x00000005
0000106C: c:
0000106C: .word 0x00000000
000053B4:81818181
000053B8:81818181
000053BC:81818181
000053C0:81818181
000053C4:81818181
000053C8:81818181
000053CC:81818181
000053D0:81818181
000053D4:81818181
000053D8:81818181
000053DC:81818181
000053E0:81818181
000053E4:81818181
000053E8:81818181
000053EC:81818181
000053F0:81818181
000053F4:81818181
000053F8:81818181
000053FC:81818181
00005400:81818181
00005404:81818181
00005408:81818181
0000540C:81818181
00005410:81818181
00005414:81818181
00005418:81818181
0000541C:81818181
00005420:81818181
00005424:81818181
00005428:81818181
0000542C:81818181
00005430:81818181
00005434:81818181
00005438:81818181
0000543C:81818181
00005440:81818181
00005444:81818181
00005448:81818181
OutputView WatchView MemoryView0
106c
Word Size
8Bit 16Bit 32Bit
0000106C 00000028 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010A8 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010E4 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00001120 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

ARMSim_files > lab3_q5b.s

```
3
4   .data
5   A:      .word 1,2,3,4
6   B:      .word 2,3,4,5
7   C:  .WORD 0
8
9   .text
10
11  init:
12      LDR R0, =A
13      LDR R1, =B
14      MOV R3, #4
15      MOV R2, #0
16
17  start:
18      CMP R3, #0
19      BNE loop
20      B end
21
22  loop:
23      LDR R5,[R0],#4
24      LDR R6,[R1],#4
25      MLA R2,R5,R6,R2
26      SUB R3,R3,#1
27      B start
28
29  end:
30      LDR R1,=C
31      STR R2,[R1]
32      SWI 0x011
33
```

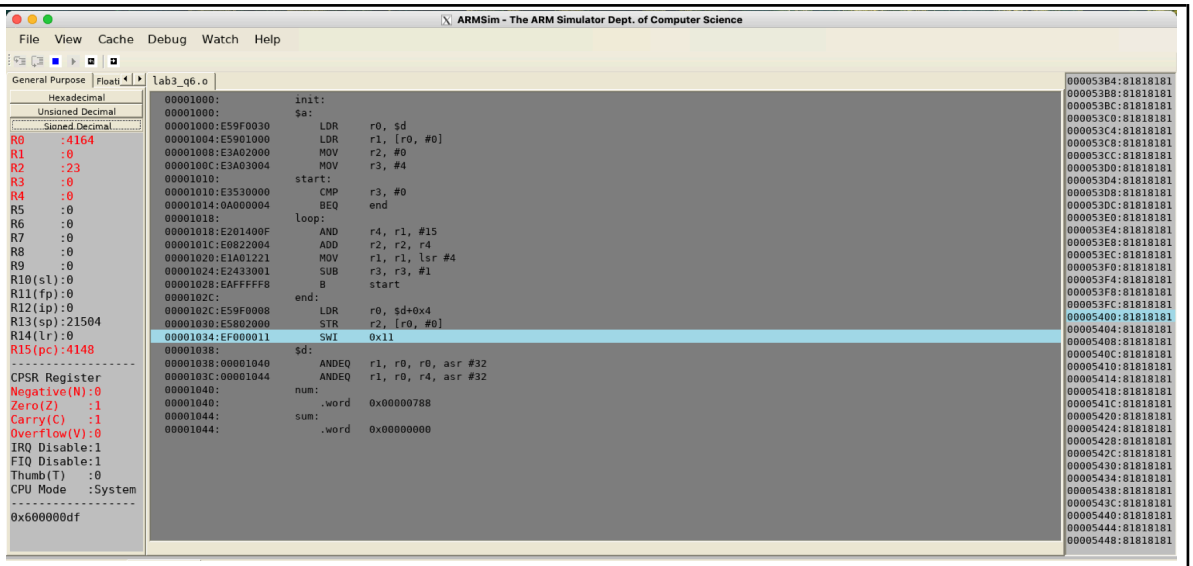


6 Write an ALP using ARM7TDMI to find the sum of all the BCD digits of a given 32 bit number. (hint:788 =7+8+8)

Program screen shot:

ARMSim_files > ASM lab3_q6.s

```
1   @ Write an ALP using ARM7TDMI to find the sum of all the BCD digits of a given 32 bit number.
2   @ (hint:788 =7+8+8)
3
4   .data
5   num: .word 0x00000788
6   sum: .word 0
7
8   .text
9
10  init:
11      LDR R0, =num
12      LDR R1, [R0]
13      MOV R2, #0
14      MOV R3, #4
15
16  start:
17      CMP R3, #0
18      BEQ end
19
20  loop:
21      AND R4, R1, #0xF
22      ADD R2, R2, R4
23      MOV R1, R1, LSR #4
24      SUB R3, R3, #1
25      B start
26
27  end:
28      LDR R0, =sum
29      STR R2, [R0]
30      SWI 0x011
31
```



Screen shot of Register set output: