# PES UNIVERSITY

**Department of Computer Science & Engineering**

## Microprocessor & Computer Architecture Lab

## UE23CS251B

## WEEK 3 submission

| | |
|---|---|
| **Name of the Student** | *CHIRAG K M* |
| **SRN** | *PES1UG23CS167* |
| **Section** | *C* |
| **Department** | *CSE* |
| **Campus** | *RR/EC* |

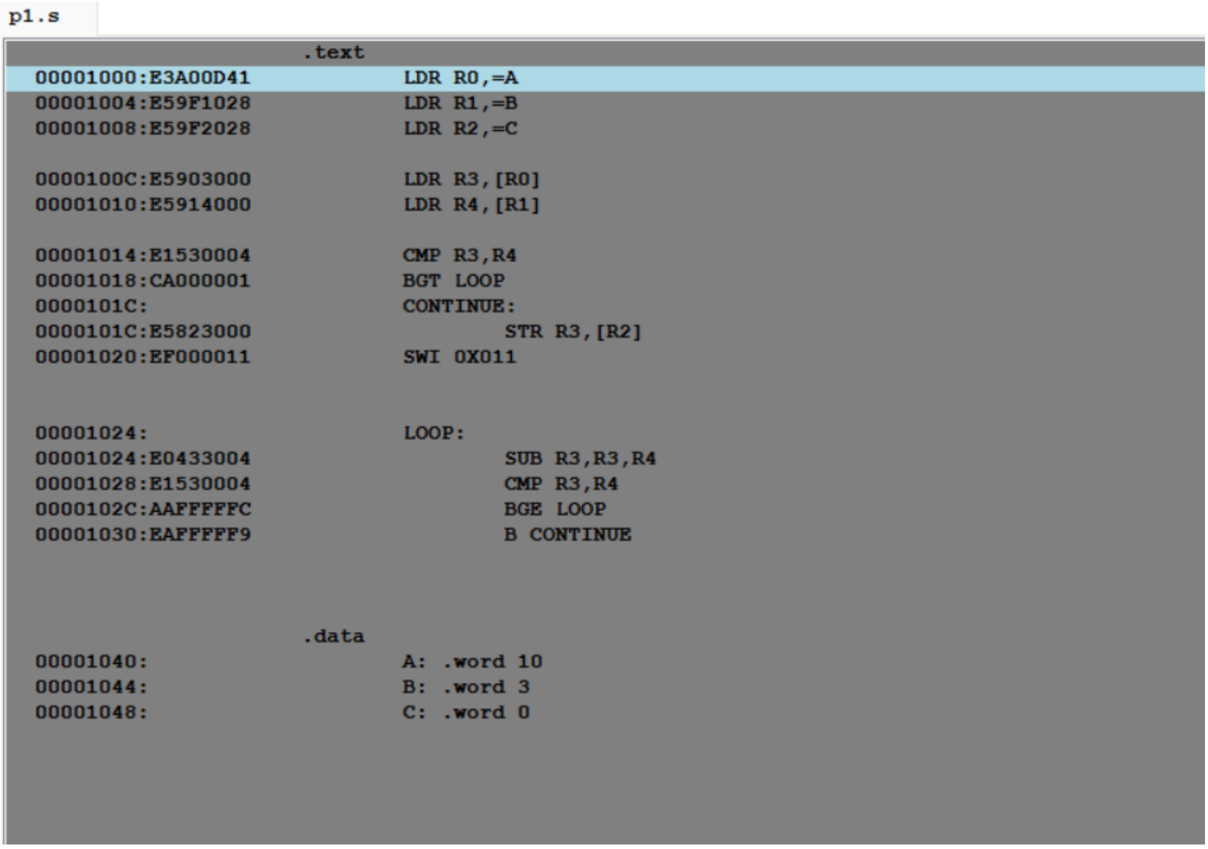
***Q1.***Write an ALP using ARM7TDMI to find the remainder of a number.(ie 10/3, remainder is 1)

.DATA

A: .word 10

B: .word 3

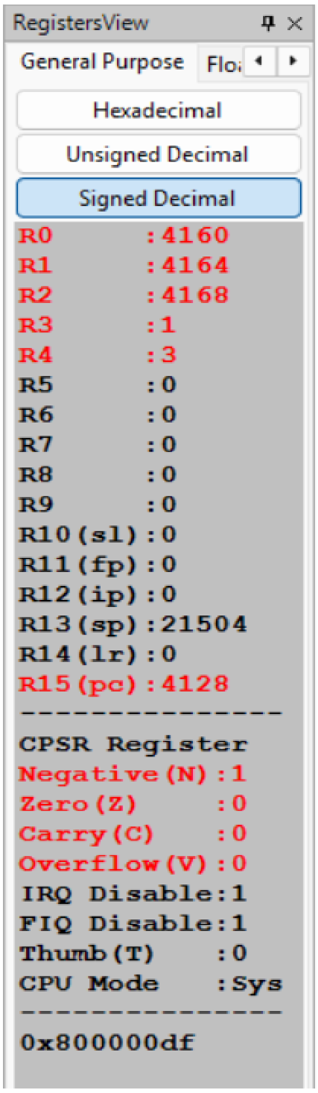
Program screen shot:



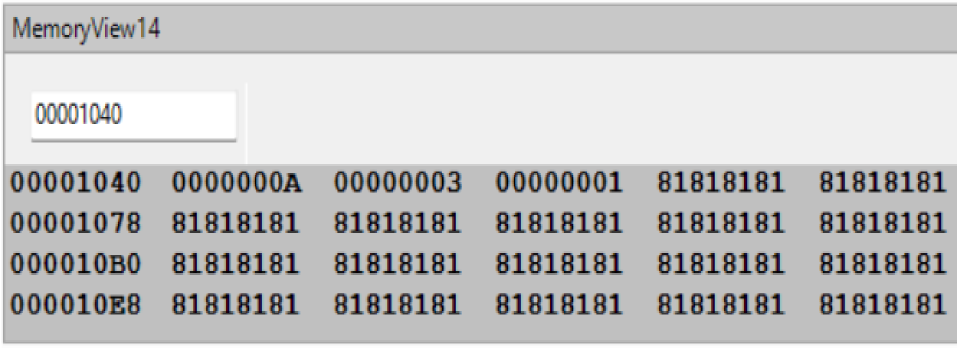
p1.s

```
                        .text
00001000:E3A00D41               LDR R0,=A
00001004:E59F1028               LDR R1,=B
00001008:E59F2028               LDR R2,=C

0000100C:E5903000               LDR R3,[R0]
00001010:E5914000               LDR R4,[R1]

00001014:E1530004               CMP R3,R4
00001018:CA000001               BGT LOOP
0000101C:                       CONTINUE:
0000101C:E5823000                       STR R3,[R2]
00001020:EF000011               SWI 0X011


00001024:                       LOOP:
00001024:E0433004                       SUB R3,R3,R4
00001028:E1530004                       CMP R3,R4
0000102C:AAFFFFFC                       BGE LOOP
00001030:EAFFFFF9                       B CONTINUE



                        .data
00001040:                       A: .word 10
00001044:                       B: .word 3
00001048:                       C: .word 0
```

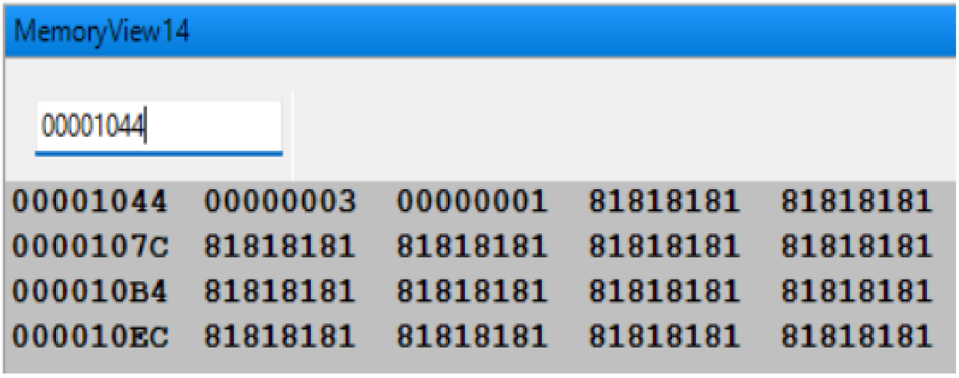
Screen shot of Register set output



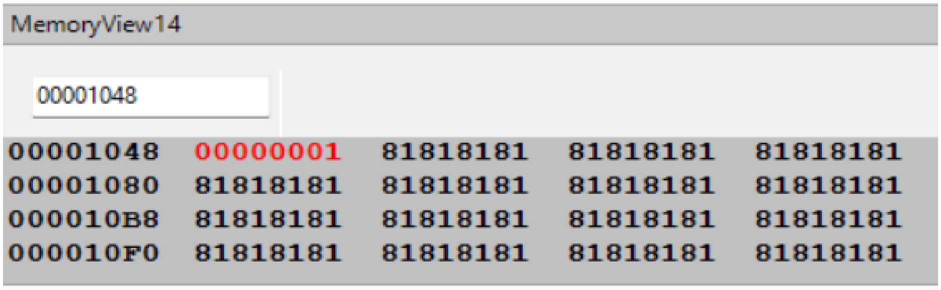
```
R0        :4160
R1        :4164
R2        :4168
R3        :1
R4        :3
R5        :0
R6        :0
R7        :0
R8        :0
R9        :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4128
---------------
CPSR Register
Negative(N):1
Zero(Z)     :0
Carry(C)    :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :Sys
---------------
0x800000df
```



```
00001040
00001040  0000000A  00000003  00000001  81818181  81818181
00001078  81818181  81818181  81818181  81818181  81818181
000010B0  81818181  81818181  81818181  81818181  81818181
000010E8  81818181  81818181  81818181  81818181  81818181
```



```
00001044
00001044  00000003  00000001  81818181  81818181
0000107C  81818181  81818181  81818181  81818181
000010B4  81818181  81818181  81818181  81818181
000010EC  81818181  81818181  81818181  81818181
```



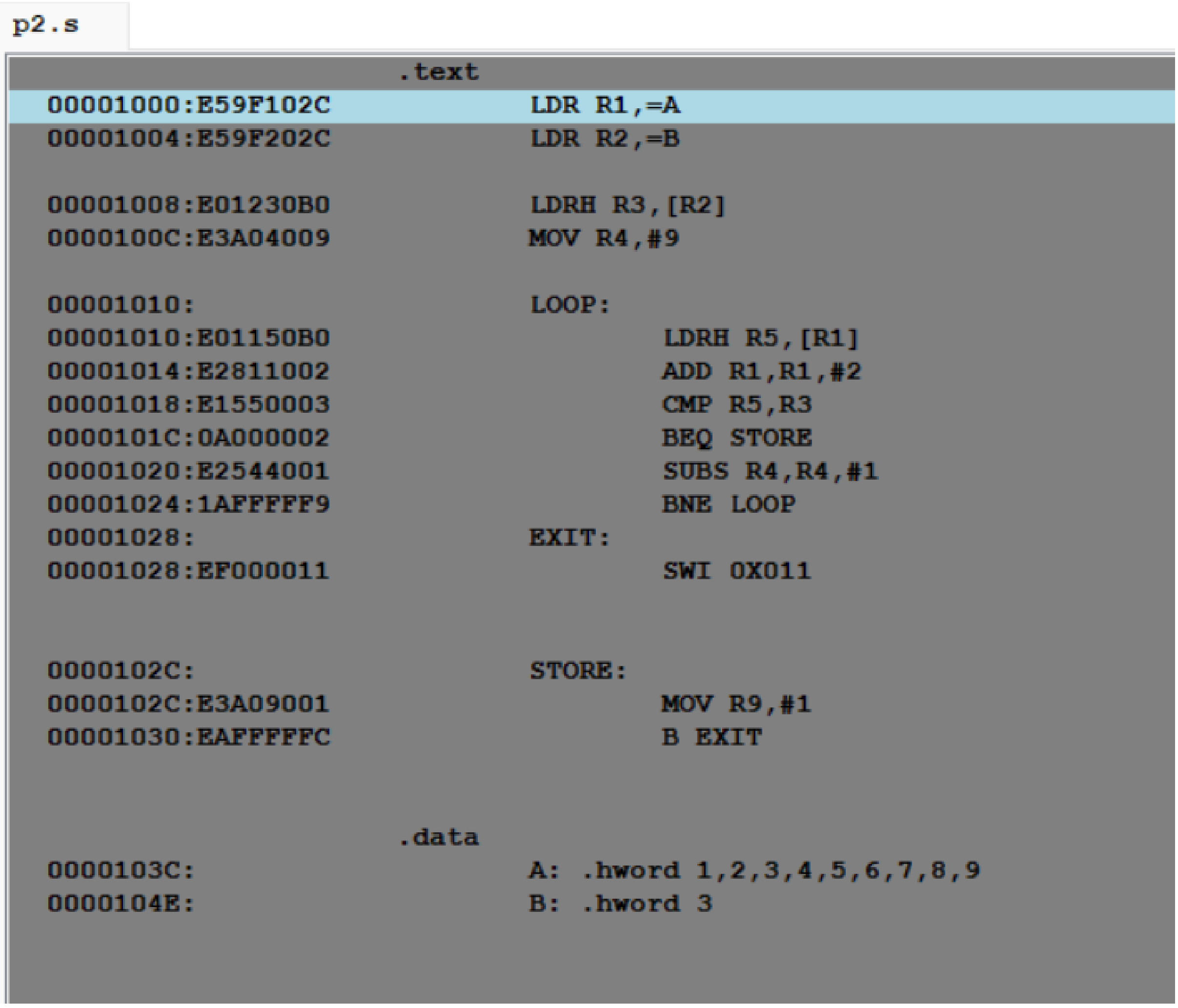
```
00001048
00001048  00000001  81818181  81818181  81818181
00001080  81818181  81818181  81818181  81818181
000010B8  81818181  81818181  81818181  81818181
000010F0  81818181  81818181  81818181  81818181
```

Jan -May 2025 LAB SUBMISSION_**UE23CS251B**

**Q2.** Write an ALP using ARM7TDMI to search for an element in an array of 16 bit each using Linear search technique
.DATA

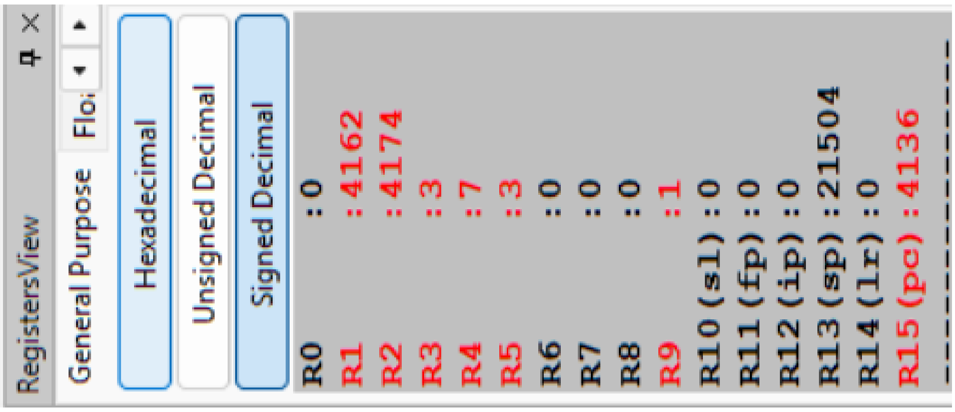
A:.hword 1,2,3,4,5,6,7,8,9

Program screen shot:



```
                         .text
00001000:E59F102C              LDR R1,=A
00001004:E59F202C              LDR R2,=B

00001008:E01230B0              LDRH R3,[R2]
0000100C:E3A04009              MOV R4,#9

00001010:                      LOOP:
00001010:E01150B0                     LDRH R5,[R1]
00001014:E2811002                     ADD R1,R1,#2
00001018:E1550003                     CMP R5,R3
0000101C:0A000002                     BEQ STORE
00001020:E2544001                     SUBS R4,R4,#1
00001024:1AFFFFF9                     BNE LOOP
00001028:                      EXIT:
00001028:EF000011                     SWI 0X011


0000102C:                      STORE:
0000102C:E3A09001                     MOV R9,#1
00001030:EAFFFFFC                     B EXIT


                         .data
0000103C:                      A: .hword 1,2,3,4,5,6,7,8,9
0000104E:                      B: .hword 3
```

Screen shot of Register set output



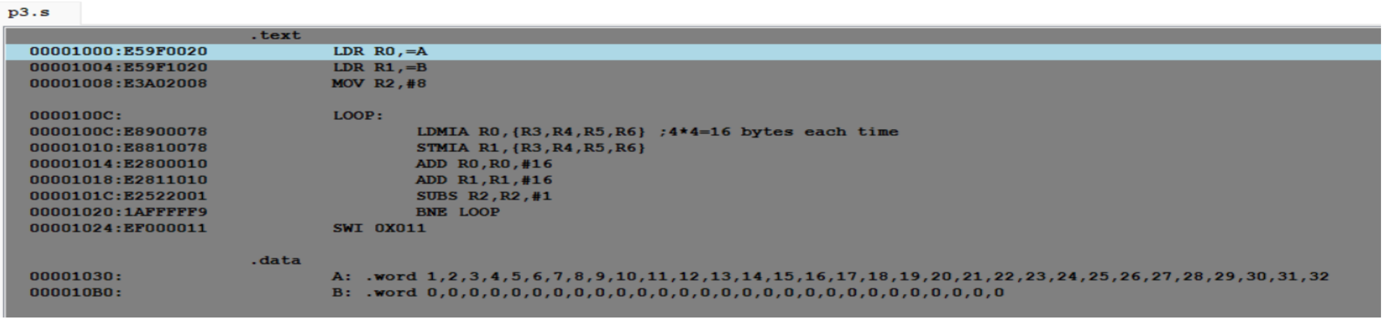
```
R0          :0
R1          :4162
R2          :4174
R3          :3
R4          :7
R5          :3
R6          :0
R7          :0
R8          :0
R9          :1
R10(sl)     :0
R11(fp)     :0
R12(ip)     :0
R13(sp)     :21504
R14(lr)     :0
R15(pc)     :4136
--------------------
```

# Jan -May 2025 LAB SUBMISSION_**UE23CS251B**

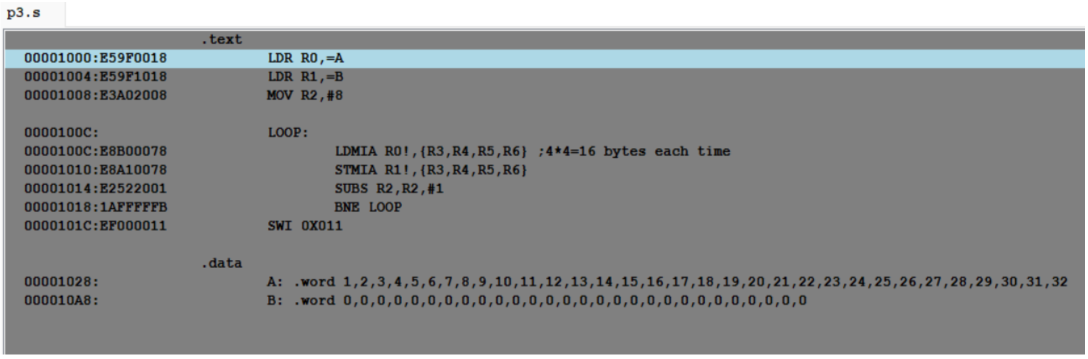
**Q3.** Write an ALP using ARM7TDMI to to copy a block 128 bytes of data from location A to location B if the rate of data transfer rate is 16 bytes, LDM and STM instructions and

For the same transfer the block **with** auto-indexing.

Program screen shot: without auto incr



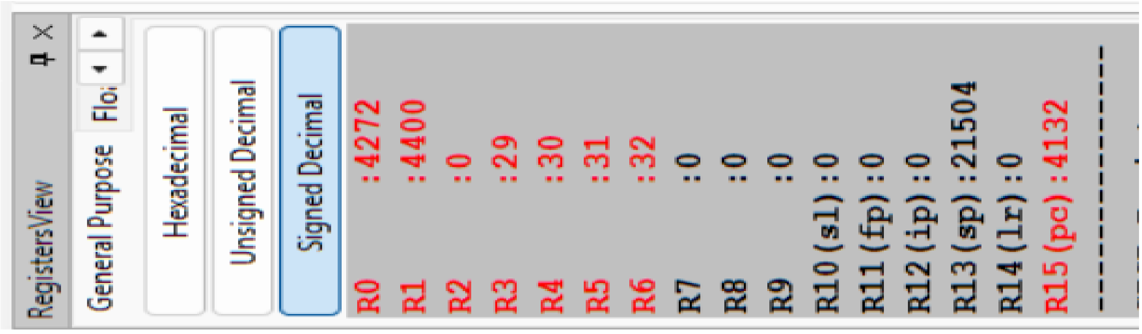
```
p3.s
                          .text
 00001000:E59F0020              LDR R0,=A
 00001004:E59F1020              LDR R1,=B
 00001008:E3A02008              MOV R2,#8

 0000100C:                      LOOP:
 0000100C:E8900078                    LDMIA R0,{R3,R4,R5,R6} ;4*4=16 bytes each time
 00001010:E8810078                    STMIA R1,{R3,R4,R5,R6}
 00001014:E2800010                    ADD R0,R0,#16
 00001018:E2811010                    ADD R1,R1,#16
 0000101C:E2522001                    SUBS R2,R2,#1
 00001020:1AFFFFF9                    BNE LOOP
 00001024:EF000011              SWI 0X011

                          .data
 00001030:                      A: .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
 000010B0:                      B: .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
```



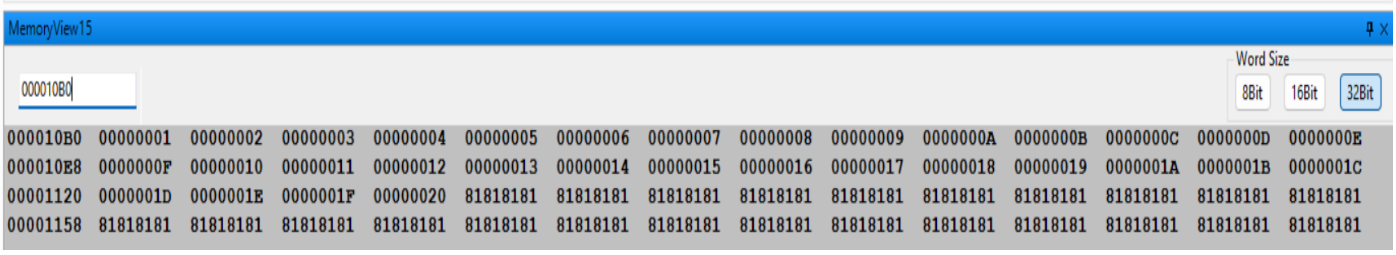
```
p3.s
                          .text
 00001000:E59F0018              LDR R0,=A
 00001004:E59F1018              LDR R1,=B
 00001008:E3A02008              MOV R2,#8

 0000100C:                      LOOP:
 0000100C:E8B00078                    LDMIA R0!,{R3,R4,R5,R6} ;4*4=16 bytes each time
 00001010:E8A10078                    STMIA R1!,{R3,R4,R5,R6}
 00001014:E2522001                    SUBS R2,R2,#1
 00001018:1AFFFFFB                    BNE LOOP
 0000101C:EF000011              SWI 0X011

                          .data
 00001028:                      A: .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
 000010A8:                      B: .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
```

With auto increment

**Screen shot of Register set output**

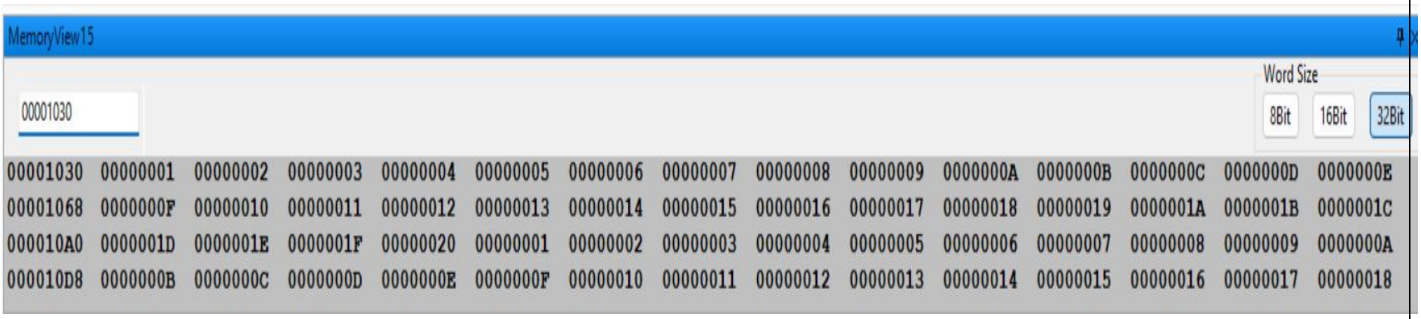


```
R0      :4272
R1      :4400
R2      :0
R3      :29
R4      :30
R5      :31
R6      :32
R7      :0
R8      :0
R9      :0
R10(sl) :0
R11(fp) :0
R12(ip) :0
R13(sp) :21504
R14(lr) :0
R15(pc) :4132
```



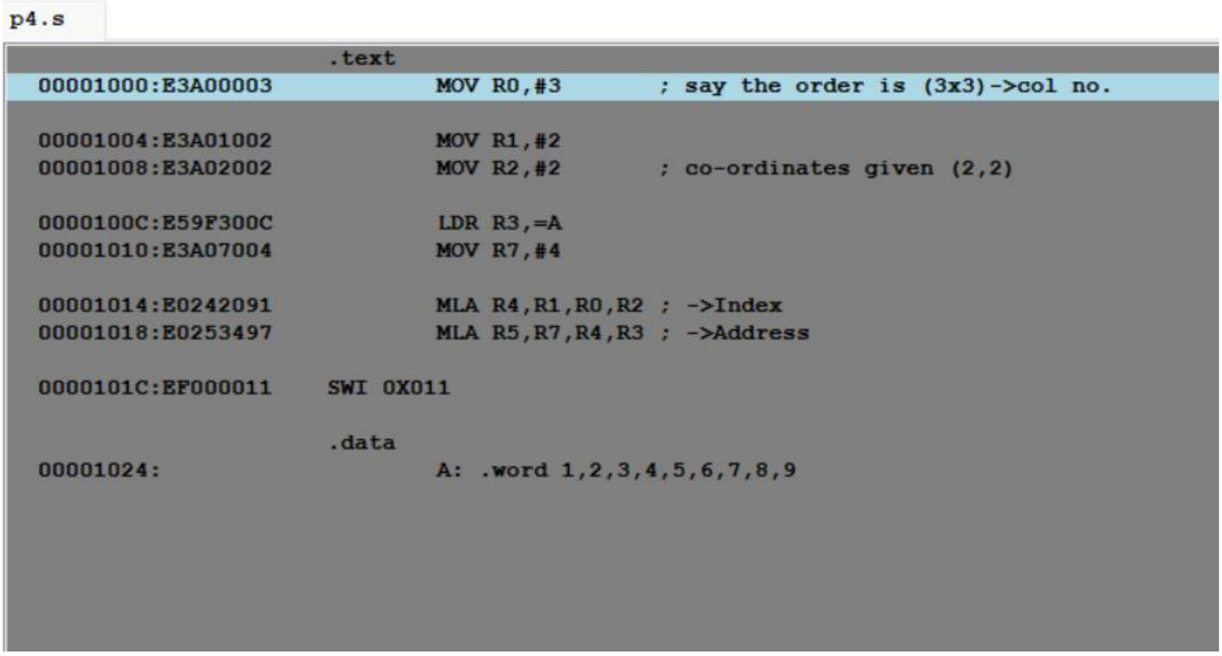
000010B0

```
000010B0  00000001  00000002  00000003  00000004  00000005  00000006  00000007  00000008  00000009  0000000A  0000000B  0000000C  0000000D  0000000E
000010E8  0000000F  00000010  00000011  00000012  00000013  00000014  00000015  00000016  00000017  00000018  00000019  0000001A  0000001B  0000001C
00001120  0000001D  0000001E  0000001F  00000020  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001158  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181
```



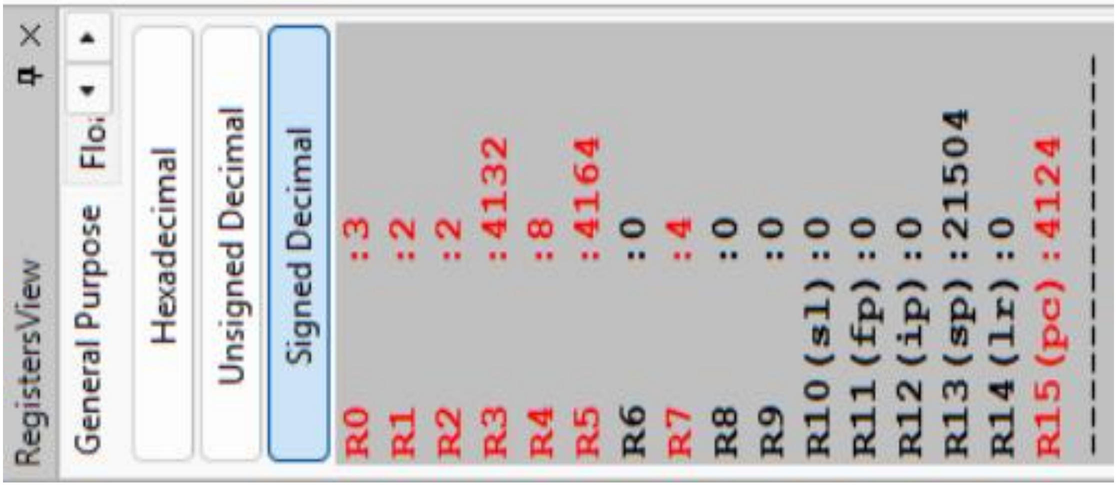
```
00001030

00001030  00000001  00000002  00000003  00000004  00000005  00000006  00000007  00000008  00000009  0000000A  0000000B  0000000C  0000000D  0000000E
00001068  0000000F  00000010  00000011  00000012  00000013  00000014  00000015  00000016  00000017  00000018  00000019  0000001A  0000001B  0000001C
000010A0  0000001D  0000001E  0000001F  00000020  00000001  00000002  00000003  00000004  00000005  00000006  00000007  00000008  00000009  0000000A
000010D8  0000000B  0000000C  0000000D  0000000E  0000000F  00000010  00000011  00000012  00000013  00000014  00000015  00000016  00000017  00000018
```

**Q4.** 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)

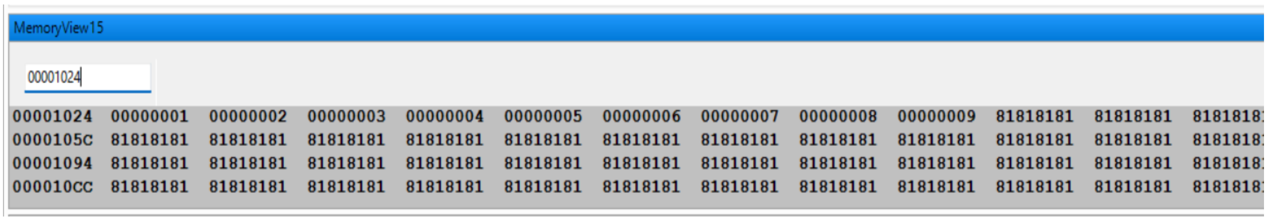


```
                        .text
00001000:E3A00003               MOV R0,#3        ; say the order is (3x3)->col no.

00001004:E3A01002               MOV R1,#2
00001008:E3A02002               MOV R2,#2        ; co-ordinates given (2,2)

0000100C:E59F300C               LDR R3,=A
00001010:E3A07004               MOV R7,#4

00001014:E0242091               MLA R4,R1,R0,R2 ; ->Index
00001018:E0253497               MLA R5,R7,R4,R3 ; ->Address

0000101C:EF000011       SWI 0X011

                        .data
00001024:                       A:  .word 1,2,3,4,5,6,7,8,9
```

Screen shot of Register set output and memory location:

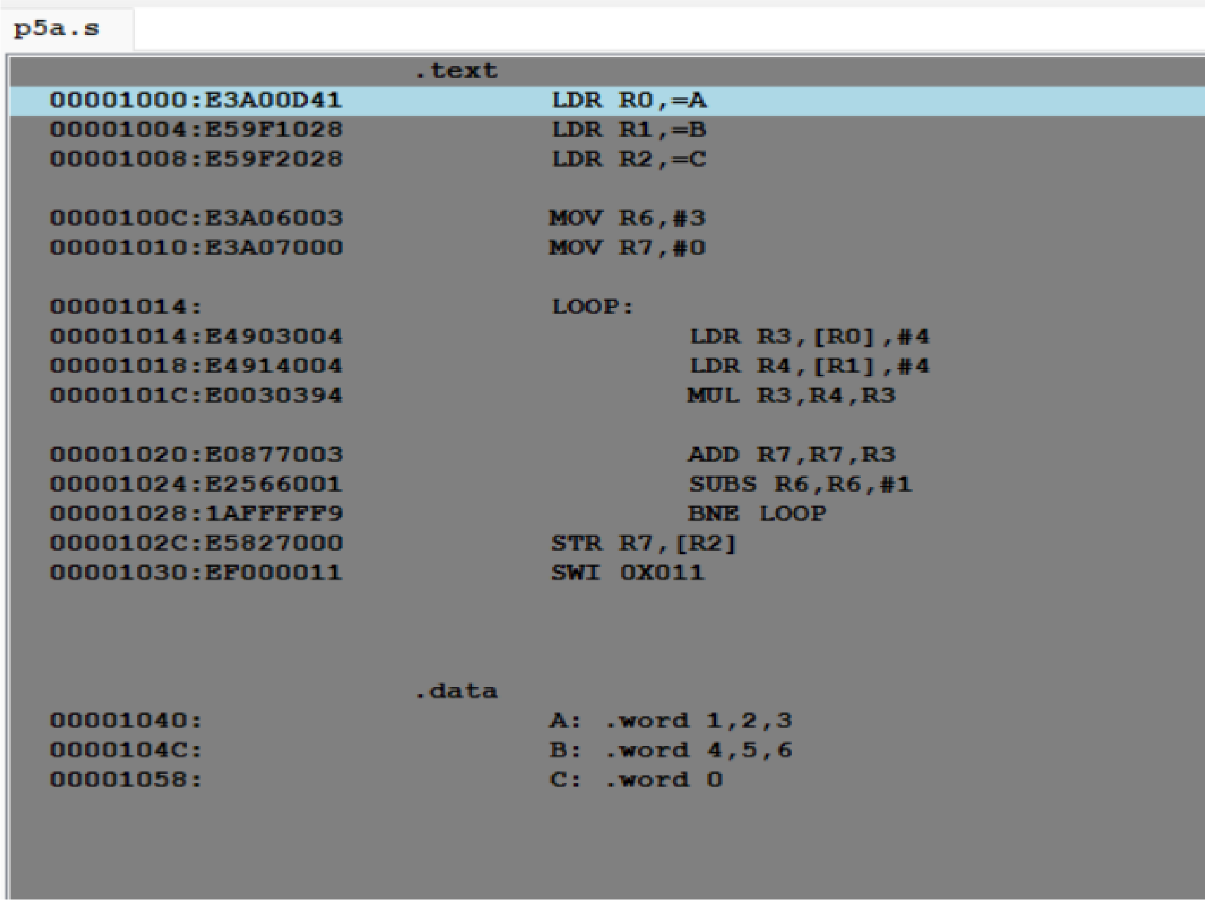


```
R0      :3
R1      :2
R2      :2
R3      :4132
R4      :8
R5      :4164
R6      :0
R7      :4
R8      :0
R9      :0
R10(sl) :0
R11(fp) :0
R12(ip) :0
R13(sp) :21504
R14(lr) :0
R15(pc) :4124
------------------
```



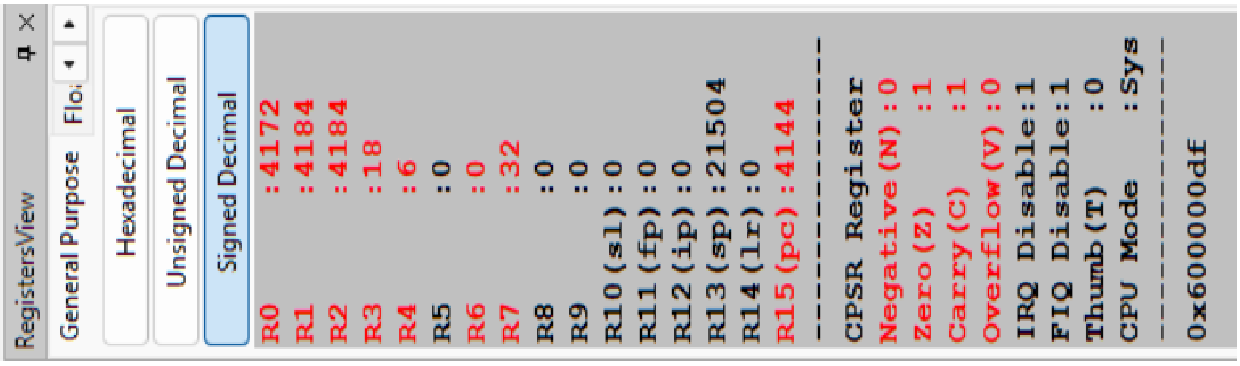
```
00001024  00000001  00000002  00000003  00000004  00000005  00000006  00000007  00000008  00000009  81818181  81818181  8181818
0000105c  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  8181818
00001094  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  8181818
000010cc  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  81818181  8181818
```

**Q5**. a )Write an ALP using ARM7TDMI to perform Convolution using MUL instruction (Addition of multiplication of respective numbers of loc A and loc B)

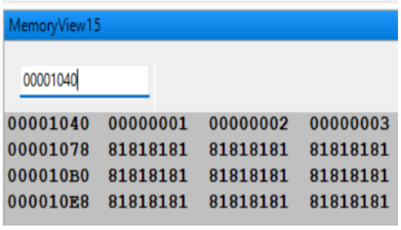


```
                        .text
  00001000:E3A00D41               LDR R0,=A
  00001004:E59F1028               LDR R1,=B
  00001008:E59F2028               LDR R2,=C

  0000100C:E3A06003               MOV R6,#3
  00001010:E3A07000               MOV R7,#0

  00001014:                       LOOP:
  00001014:E4903004                       LDR R3,[R0],#4
  00001018:E4914004                       LDR R4,[R1],#4
  0000101C:E0030394                       MUL R3,R4,R3

  00001020:E0877003                       ADD R7,R7,R3
  00001024:E2566001                       SUBS R6,R6,#1
  00001028:1AFFFFF9                       BNE LOOP
  0000102C:E5827000               STR R7,[R2]
  00001030:EF000011               SWI 0X011



                        .data
  00001040:                       A:  .word 1,2,3
  0000104C:                       B:  .word 4,5,6
  00001058:                       C:  .word 0
```

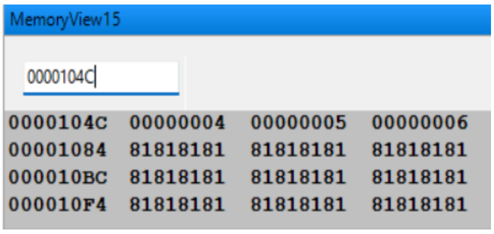
**Screen shot of Register set output:**



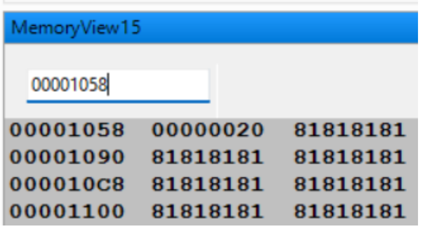
```
R0       :4172
R1       :4184
R2       :4184
R3       :18
R4       :6
R5       :0
R6       :0
R7       :32
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4144
------------
CPSR Register
Negative(N):0
Zero(Z)    :1
Carry(C)   :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)   :0
CPU Mode   :Sys
------------
0x600000df
```



```
00001040 00000001 00000002 00000003
00001078 81818181 81818181 81818181
000010B0 81818181 81818181 81818181
000010E8 81818181 81818181 81818181
```



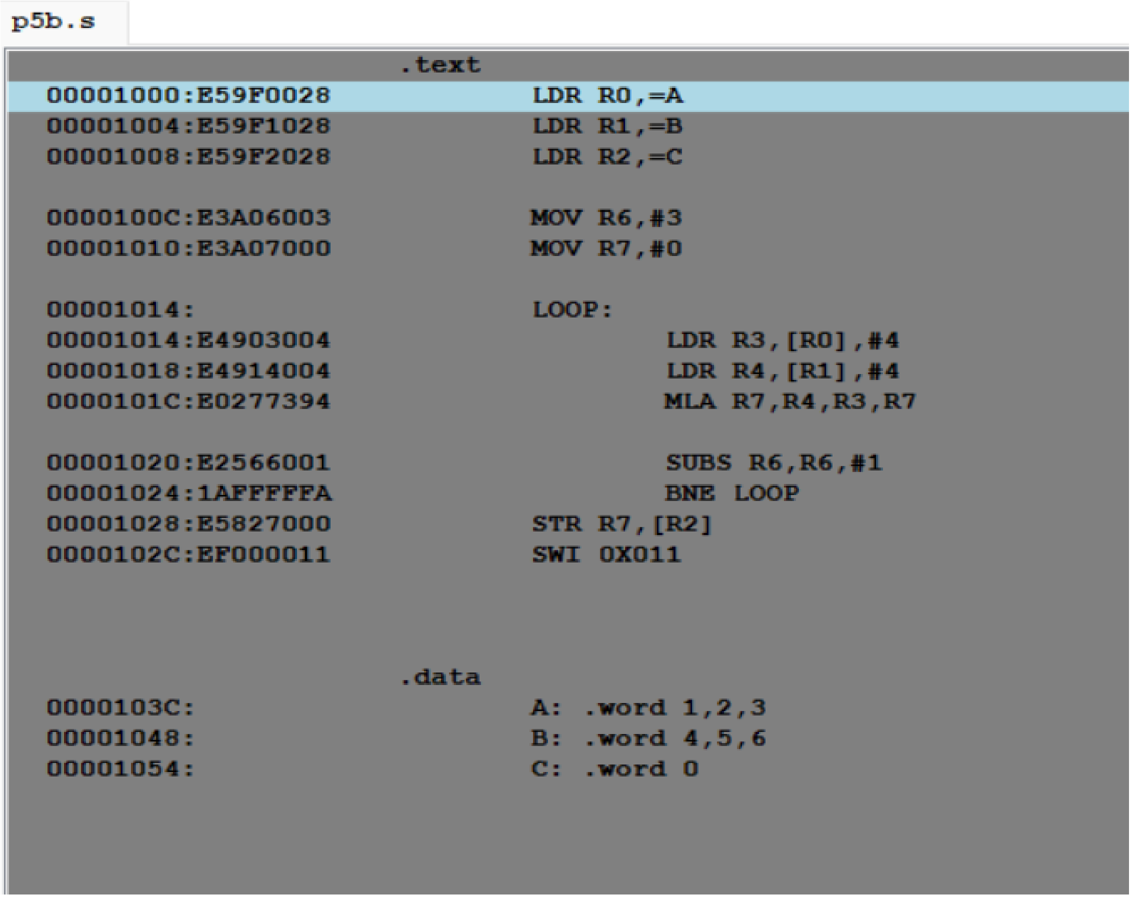
```
0000104C 00000004 00000005 00000006
00001084 81818181 81818181 81818181
000010BC 81818181 81818181 81818181
000010F4 81818181 81818181 81818181
```



```
00001058 00000020 81818181
00001090 81818181 81818181
000010C8 81818181 81818181
00001100 81818181 81818181
```

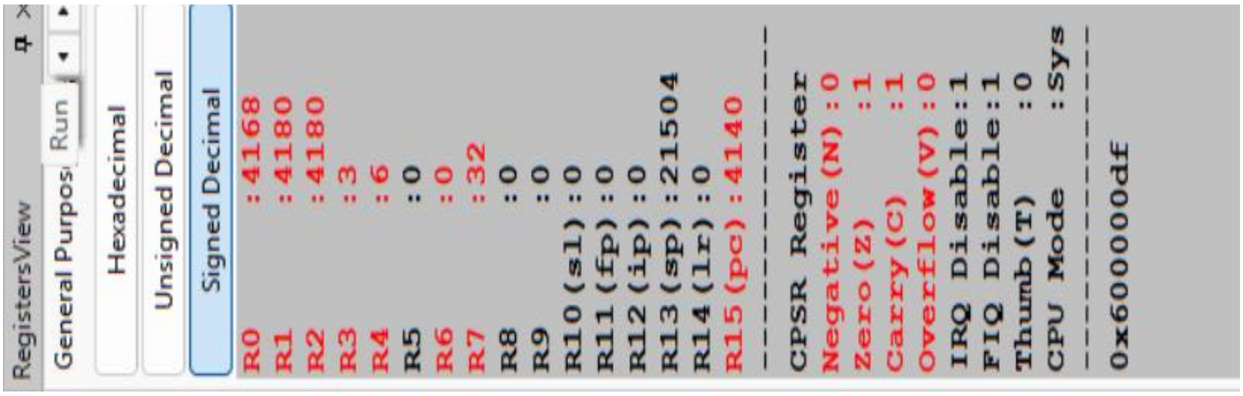
**Q5.** b Write an ALP using ARM7TDMI to perform Convolution using MLA instruction (Addition of multiplication of respective numbers of loc A and loc B).

Program screen shot:



```
                              .text
  00001000:E59F0028                   LDR R0,=A
  00001004:E59F1028                   LDR R1,=B
  00001008:E59F2028                   LDR R2,=C

  0000100C:E3A06003                   MOV R6,#3
  00001010:E3A07000                   MOV R7,#0

  00001014:                           LOOP:
  00001014:E4903004                           LDR R3,[R0],#4
  00001018:E4914004                           LDR R4,[R1],#4
  0000101C:E0277394                           MLA R7,R4,R3,R7

  00001020:E2566001                           SUBS R6,R6,#1
  00001024:1AFFFFFA                           BNE LOOP
  00001028:E5827000                   STR R7,[R2]
  0000102C:EF000011                   SWI 0X011



                              .data
  0000103C:                           A:  .word 1,2,3
  00001048:                           B:  .word 4,5,6
  00001054:                           C:  .word 0
```

**Screen shot of Register set output:**

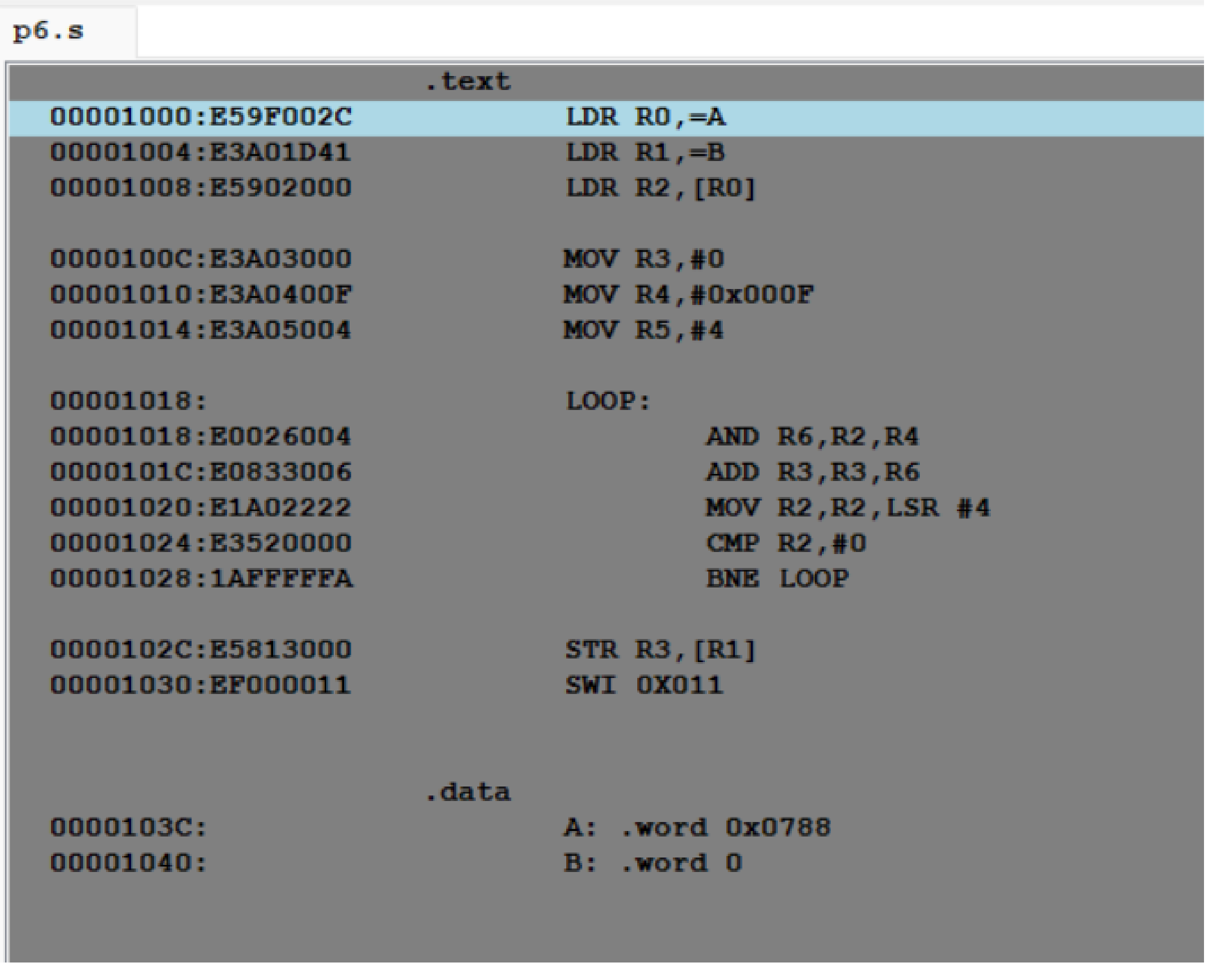


```
R0         :4168
R1         :4180
R2         :4180
R3         :3
R4         :6
R5         :0
R6         :0
R7         :32
R8         :0
R9         :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4140
----------------
CPSR Register
Negative(N):0
Zero(Z)     :1
Carry(C)    :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T)    :0
CPU Mode    :Sys
----------------
0x600000df
```

**Q6**. Write an ALP using ARM7TDMI to find the sum of all the BCD digits of a given 32 bit number.

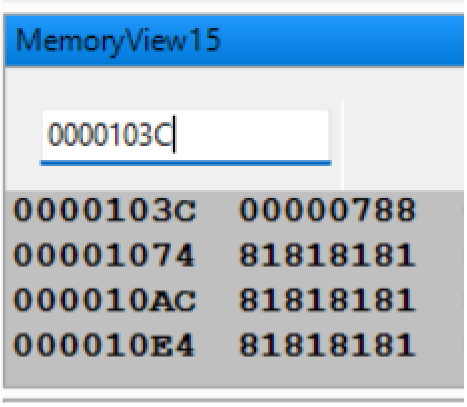
(hint:788 =7+8+8)

Program screen shot:

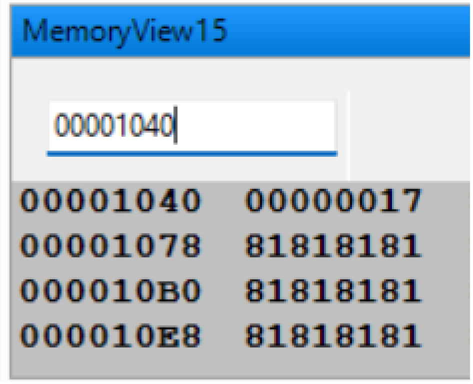


```
                     .text
00001000:E59F002C            LDR R0,=A
00001004:E3A01D41            LDR R1,=B
00001008:E5902000            LDR R2,[R0]

0000100C:E3A03000            MOV R3,#0
00001010:E3A0400F            MOV R4,#0x000F
00001014:E3A05004            MOV R5,#4

00001018:                    LOOP:
00001018:E0026004                   AND R6,R2,R4
0000101C:E0833006                   ADD R3,R3,R6
00001020:E1A02222                   MOV R2,R2,LSR #4
00001024:E3520000                   CMP R2,#0
00001028:1AFFFFFA                   BNE LOOP

0000102C:E5813000            STR R3,[R1]
00001030:EF000011            SWI 0X011


                     .data
0000103C:                    A: .word 0x0788
00001040:                    B: .word 0
```

**Screen shot of Register set output:**



```
0000103C

0000103C  00000788
00001074  81818181
000010AC  81818181
000010E4  81818181
```

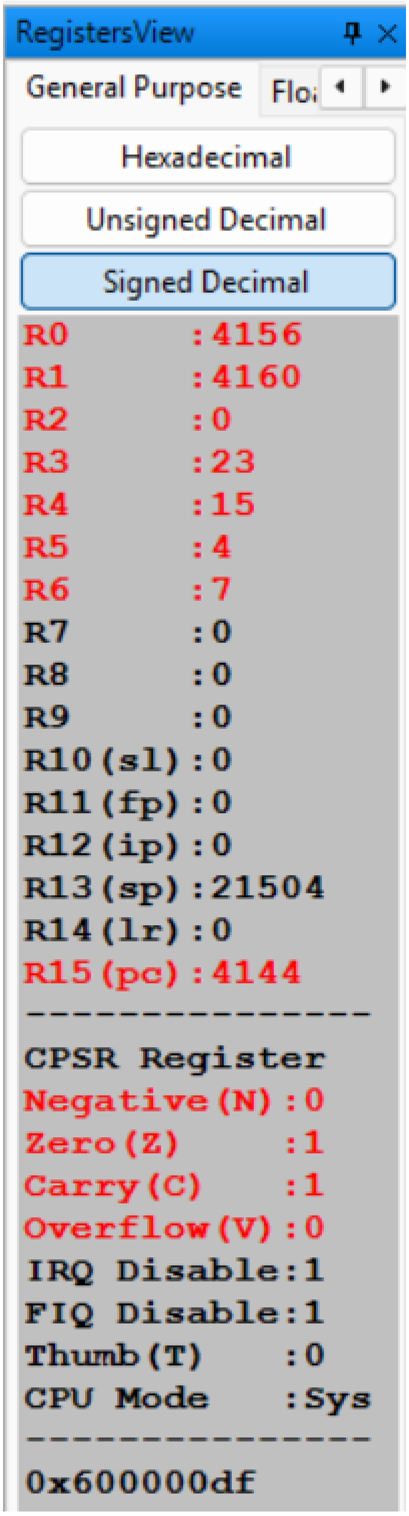


```
00001040

00001040  00000017
00001078  81818181
000010B0  81818181
000010E8  81818181
```

**Res = 17**

**(in hexa)**

**i.e 16+7=23**



```
R0      :4156
R1      :4160
R2      :0
R3      :23
R4      :15
R5      :4
R6      :7
R7      :0
R8      :0
R9      :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4144
---------------
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   :Sys
---------------
0x600000df
```

**R3 IS OUTPUT REG**

# THANK YOU