PES UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture Lab

UE23CS251B

WEEK 4 submission

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

Q1. Write an ALP using ARM7TDMI to generate a matrix of order 3 to store natural numbers

USE column MAJOR ORDER

Before:
data
MATA: .WORD 0,0,0,0,0,0,0,0,0

After:
data
MATA: .WORD 1,4,7,2,5,8,3,6,9

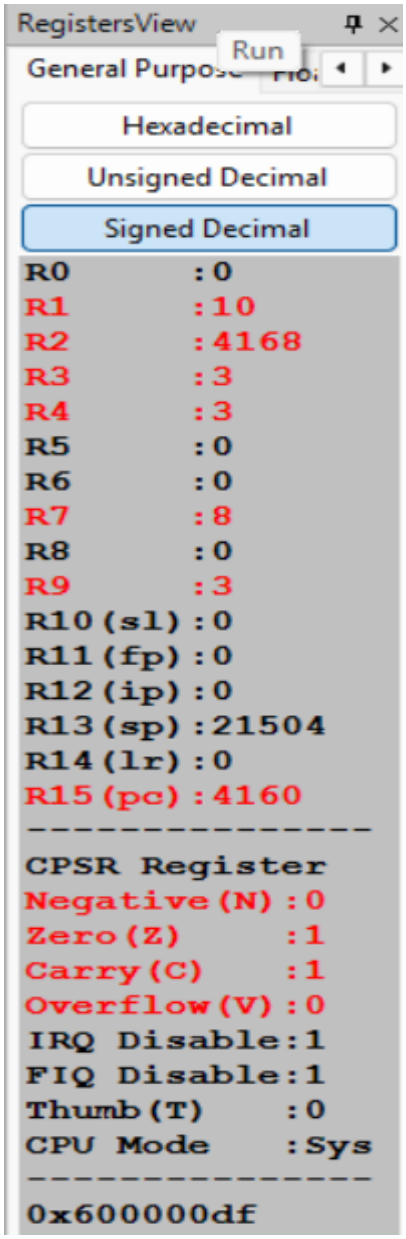
q1.s

```
                    .text
00001000:E3A03000           MOV R3,#0  ; indicates i (col)
00001004:E3A04000           MOV R4,#0  ; indicates j (row)
00001008:E3A09003           MOV R9,#3  ; Order -> col

0000100C:E3A01001           MOV R1,#1
00001010:E3A07000           MOV R7,#0
00001014:E59F2028           LDR R2,=A

00001018:                   LOOP1:                          ; j loop
00001018:E3A04000                   MOV R4,#0

0000101C:                           LOOP2:                  ; i loop
0000101C:E0273994                           MLA R7,R4,R9,R3
00001020:E7821107                           STR R1,[R2,R7,LSL #2]
00001024:E2811001                           ADD R1,R1,#1

00001028:E2844001                           ADD R4,R4,#1
0000102C:E3540003                           CMP R4,#3
00001030:1AFFFFF9                           BNE LOOP2

00001034:E2833001           ADD R3,R3,#1
00001038:E3530003           CMP R3,#3
0000103C:1AFFFFF5           BNE LOOP1

00001040:EF000011           SWI 0X011


                    .data
00001048:                   A: .word 0,0,0,0,0,0,0,0,0
```

MemoryView15

00001048

00001048	00000001	00000004	00000007	00000002	00000005	00000008	00000003	00000006	00000009
00001080	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
000010B8	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181
000010F0	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181	81818181



Q2. Write an ALP using ARM7TDMI to find the sum of 2 BCD numbers in a function using stack parameter passing technique

REF: DDCO

The addition of two *n*-digit unsigned BCD numbers follows the same procedure. Consider the addition of 184 + 576 = 760 in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

q2.s

```
                    .text
00001000:E3A01099           MOV R1,#0x99                    ; inp (larger one)
00001004:E3A02045           MOV R2,#0x45
00001008:E3A05000           MOV R5,#0                       ; result

0000100C:E3A0300F           MOV R3,#0xF
00001010:E3A08006           MOV R8,#6
00001014:E1A0900D           MOV R9,R13

00001018:                   PUSH:                           ; add corresponding BYTE and push each
00001018:E0014003                   AND R4,R1,R3
0000101C:E0025003                   AND R5,R2,R3
00001020:E0A44005                   ADC R4,R4,R5
00001024:E3540009                   CMP R4,#9
00001028:CA00000E                   BGT CNG
0000102C:                   CONT:
0000102C:E8AD0010                   STMEA R13!,{R4}
00001030:E1A01221                   MOV R1,R1,LSR #4
00001034:E1A02222                   MOV R2,R2,LSR #4
00001038:E3510000                   CMP R1,#0
0000103C:1AFFFFF5                   BNE PUSH

00001040:                   CARY:                           ; if carry from MSB
00001040:E3A04000                   MOV R4,#0
00001044:E3A05000                   MOV R5,#0
00001048:E2A44000                   ADC R4,R4,#0
0000104C:E8AD0010                   STMEA R13!,{R4}


00001050:                   POP:                            ; APPEND Poped res
00001050:E93D0010                   LDMEA R13!,{R4}
00001054:E1A05205                   MOV R5,R5,LSL #4
00001058:E0855004                   ADD R5,R5,R4
0000105C:E159000D                   CMP R9,R13
00001060:1AFFFFFA                   BNE POP
00001064:EF000011           SWI 0x011

00001068:                   CNG:
00001068:E2844006                   ADD R4,R4,#6
0000106C:E204400F                   AND R4,R4,#0xF
00001070:EAFFFFED                   B CONT
```

StackView

```
000053F0:81818181
000053F4:81818181
000053F8:81818181
000053FC:81818181
00005400:00000004
00005404:00000004
00005408:00000001
0000540C:81818181
```

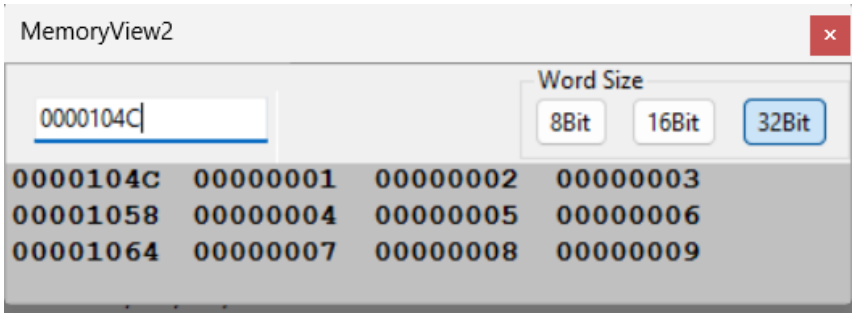
Hexadecimal | Unsigned Decimal | Signed Decimal

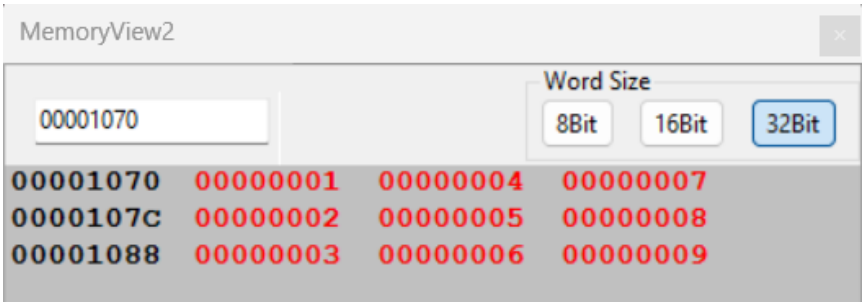
```
R0        :00000000
R1        :00000000
R2        :00000000
R3        :0000000f
R4        :00000004
R5        :00000144
R6        :00000000
R7        :00000000
R8        :00000006
R9        :00005400
R10(sl)   :00000000
R11(fp)   :00000000
R12(ip)   :00000000
R13(sp)   :00005400
R14(lr)   :00000000
R15(pc)   :00001064
```

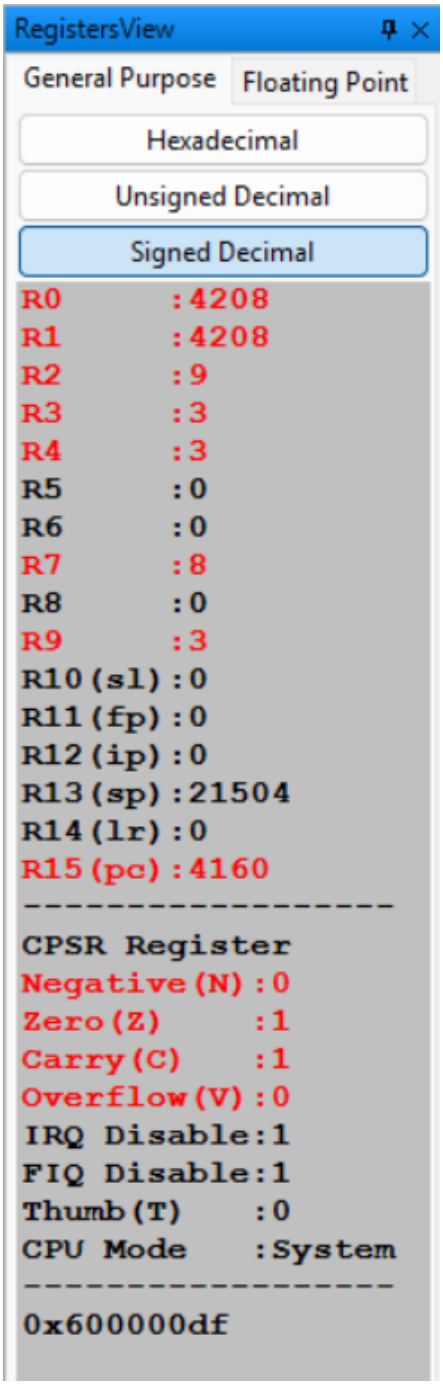
Q3. Write an ALP using ARM7TDMI to find the transpose of a matrix.

q3.s

```
                .text
00001000:E59F003C       LDR R0,=matrix
00001004:E59F103C       LDR R1,=transpose

00001008:E3A03000       MOV R3,#0  ; indicates i (col)
0000100C:E3A04000       MOV R4,#0  ; indicates j (row)
00001010:E3A09003       MOV R9,#3  ; Order -> col

00001014:E3A07000       MOV R7,#0

00001018:               LOOP1:                          ; j loop
00001018:E3A04000               MOV R4,#0

0000101C:                       LOOP2:                  ; i loop
0000101C:E0273994                       MLA R7,R4,R9,R3
00001020:E4902004                       LDR R2,[R0],#4
00001024:E7812107                       STR R2,[R1,R7,LSL #2]

00001028:E2844001                       ADD R4,R4,#1
0000102C:E3540003                       CMP R4,#3
00001030:1AFFFFF9                       BNE LOOP2

00001034:E2833001       ADD R3,R3,#1
00001038:E3530003       CMP R3,#3
0000103C:1AFFFFF5       BNE LOOP1

00001040:EF000011       SWI 0X011



                .data
0000104C:               matrix:     .word   1, 2, 3     ;Original 3x3 matrix
                                    .word   4, 5, 6
                                    .word   7, 8, 9

00001070:               transpose:  .word   0, 0, 0     ;Space for transposed matrix
                                    .word   0, 0, 0
                                    .word   0, 0, 0
```

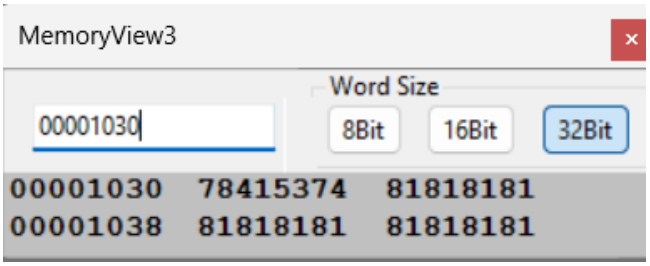






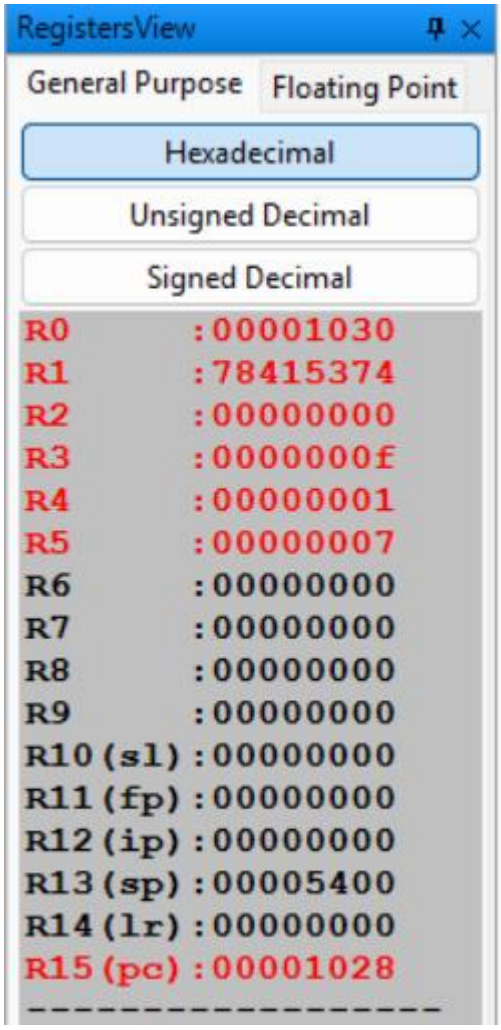
Q4. Write an ALP using ARM7TDMI to find the smallest of all the BCD digits of a given 32bit number.

(hint: If R1=78415374 the smallest digit is 1)



q4.s

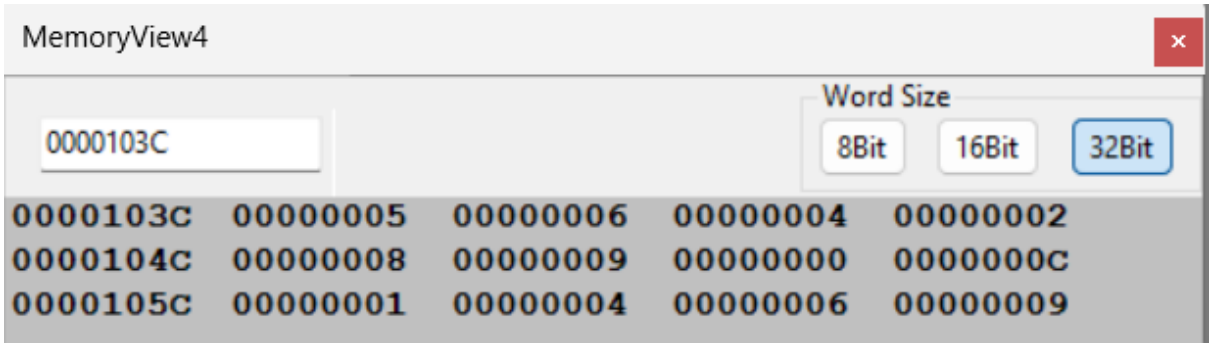
```
                        .text
00001000:E59F0024               LDR R0,=A
00001004:E5901000               LDR R1,[R0]

00001008:E1A02001               MOV R2,R1
0000100C:E3A0300F               MOV R3,#0x0000000F
00001010:E3A04009               MOV R4,#9       ; result  (initialized with max value)

00001014:                       LOOP:
00001014:E0035002                       AND R5,R3,R2
00001018:E1550004                       CMP R5,R4
0000101C:B1A04005                       MOVLT R4,R5

00001020:E1B02222                       MOVS R2,R2,LSR #4
00001024:1AFFFFFA                       BNE LOOP
00001028:EF000011               SWI 0x011




                        .data
00001030:                       A: .word 0x78415374
```

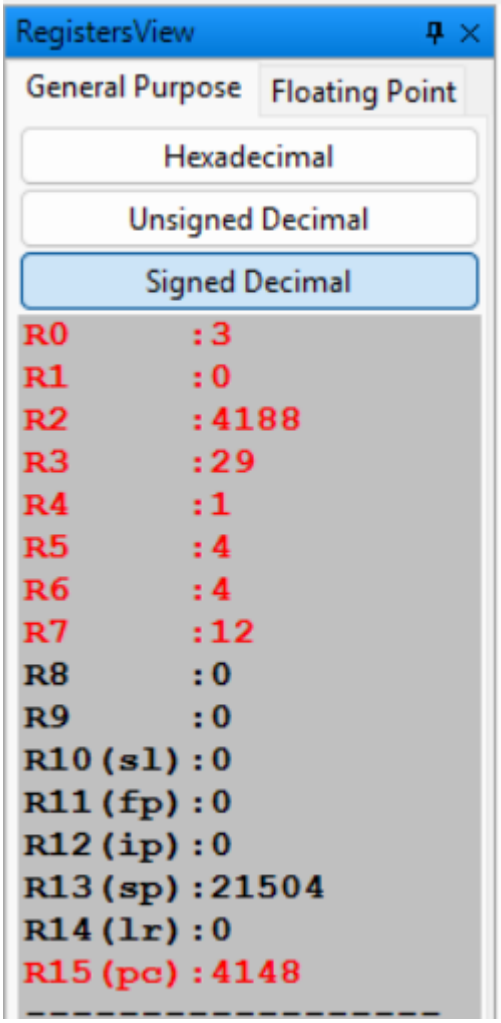


Q5. Write an ALP using ARM7TDMI to find the sum of all elements for a given row number if size of matrix is mxn.

q5.s

```
                        .text
00001000:E3A00003               MOV R0,#3               ; No. of rows
00001004:E3A01004               MOV R1,#4               ; No. of cols

00001008:E59F2028               LDR R2,=A
0000100C:E3A03000               MOV R3,#0               ; SUM

00001010:E3A04001               MOV R4,#1               ; inp: 2nd row
00001014:E1A05001               MOV R5,R1
00001018:E0060194               MUL R6,R4,R1
0000101C:E0822106               ADD R2,R2,R6, LSL #2

00001020:                       Loop:
00001020:E5927000                       LDR R7,[R2]
00001024:E0833007                       ADD R3,R3,R7
00001028:E2822004                       ADD R2,R2,#4

0000102C:E2511001                       SUBS R1,R1,#1
00001030:1AFFFFFA                       BNE Loop
00001034:EF000011               SWI 0x011



                        .data
0000103C:                       A: .word 5,6,4,2
                                   .word 8,9,0,12
                                   .word 1,4,6,9
```



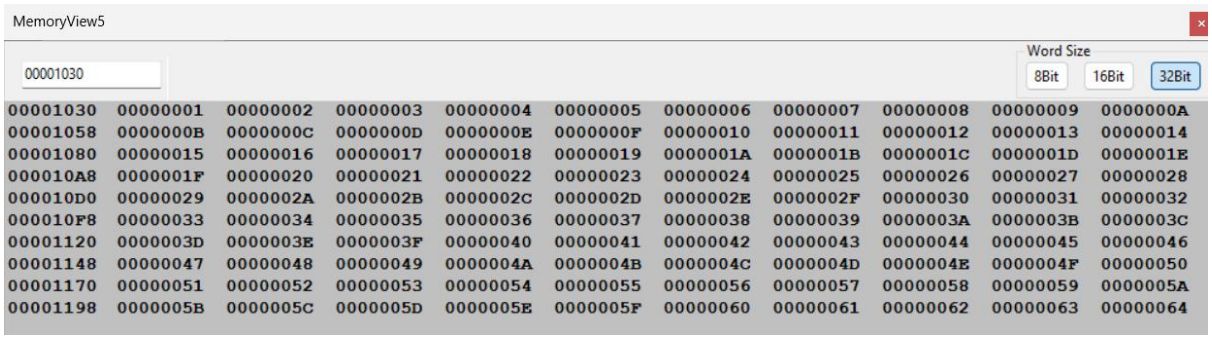


EXTRA:

Write an ALP using ARM7TDMI to copy a block 400 bytes of data from location A to location B if the rate of data transfer rate is 40 bytes, LDM and STM instructions.

and

For the same transfer the block with auto-indexing



MemoryView5

000011C0

Word Size: 8Bit | 16Bit | 32Bit

000011C0	00000001	00000002	00000003	00000004	00000005	00000006	00000007	00000008	00000009	0000000A
000011E8	0000000B	0000000C	0000000D	0000000E	0000000F	00000010	00000011	00000012	00000013	00000014
00001210	00000015	00000016	00000017	00000018	00000019	0000001A	0000001B	0000001C	0000001D	0000001E
00001238	0000001F	00000020	00000021	00000022	00000023	00000024	00000025	00000026	00000027	00000028
00001260	00000029	0000002A	0000002B	0000002C	0000002D	0000002E	0000002F	00000030	00000031	00000032
00001288	00000033	00000034	00000035	00000036	00000037	00000038	00000039	0000003A	0000003B	0000003C
000012B0	0000003D	0000003E	0000003F	00000040	00000041	00000042	00000043	00000044	00000045	00000046
000012D8	00000047	00000048	00000049	0000004A	0000004B	0000004C	0000004D	0000004E	0000004F	00000050
00001300	00000051	00000052	00000053	00000054	00000055	00000056	00000057	00000058	00000059	0000005A
00001328	0000005B	0000005C	0000005D	0000005E	0000005F	00000060	00000061	00000062	00000063	00000064

qe1.s

```
                    .text
00001000:E59F0020           LDR R0,=A
00001004:E3A01D47           LDR R1,=B
00001008:E3A0200A           MOV R2,#10

0000100C:                   LOOP:
0000100C:E8901FF8                   LDMIA R0,{R3-R12} ;4*10=40 bytes each time
00001010:E8811FF8                   STMIA R1,{R3-R12}
00001014:E2800028                   ADD R0,R0,#40
00001018:E2811028                   ADD R1,R1,#40
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
                               .word 26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50
                               .word 51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75
                               .word 76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100

000011C0:                   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
                               .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
                               .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
                               .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
```

RegistersView

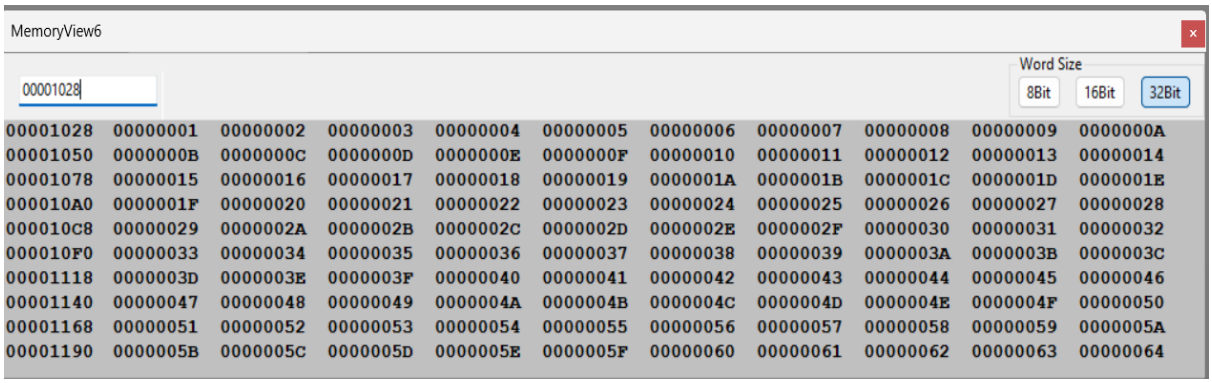
General Purpose | Floating Point | Hexadecimal | Unsigned Decimal | Signed Decimal

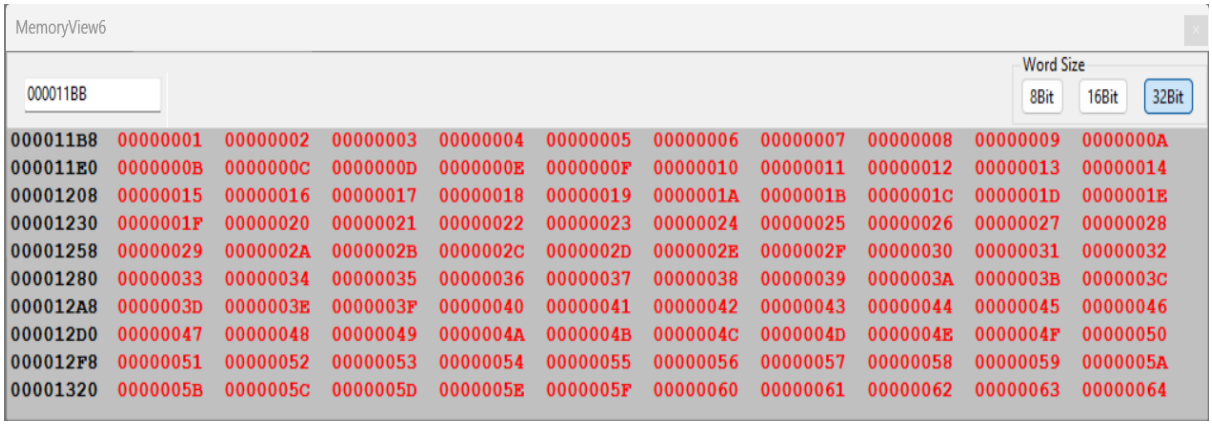
R0	:4544
R1	:4944
R2	:0
R3	:91
R4	:92
R5	:93
R6	:94
R7	:95
R8	:96
R9	:97
R10 (sl)	:98
R11 (fp)	:99
R12 (ip)	:100
R13 (sp)	:21504
R14 (lr)	:0
R15 (pc)	:4132
----------	----------

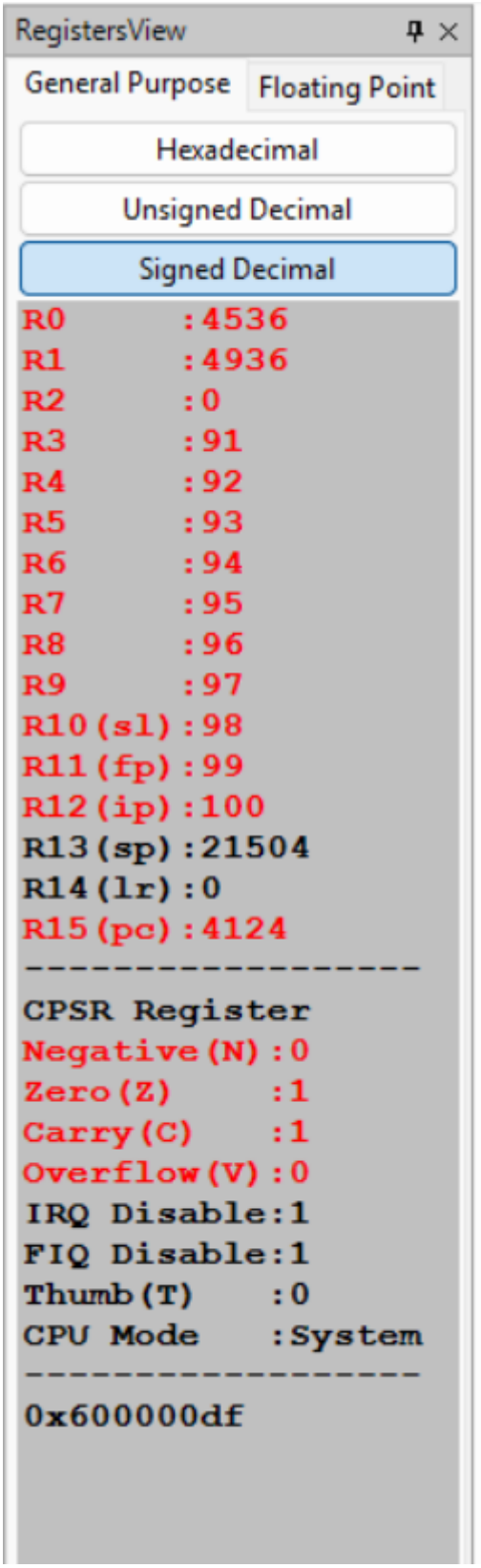
WITH AUTO-INDEXING :

qe2.s

```
                    .text
00001000:E59F0018           LDR R0,=A
00001004:E59F1018           LDR R1,=B
00001008:E3A0200A           MOV R2,#10

0000100C:                   LOOP:
0000100C:E8B01FF8                   LDMIA R0!,{R3-R12} ;4*10=40 bytes each time
00001010:E8A11FF8                   STMIA R1!,{R3-R12}
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
                               .word 26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50
                               .word 51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75
                               .word 76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100

000011B8:                   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
                               .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
                               .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
                               .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
```





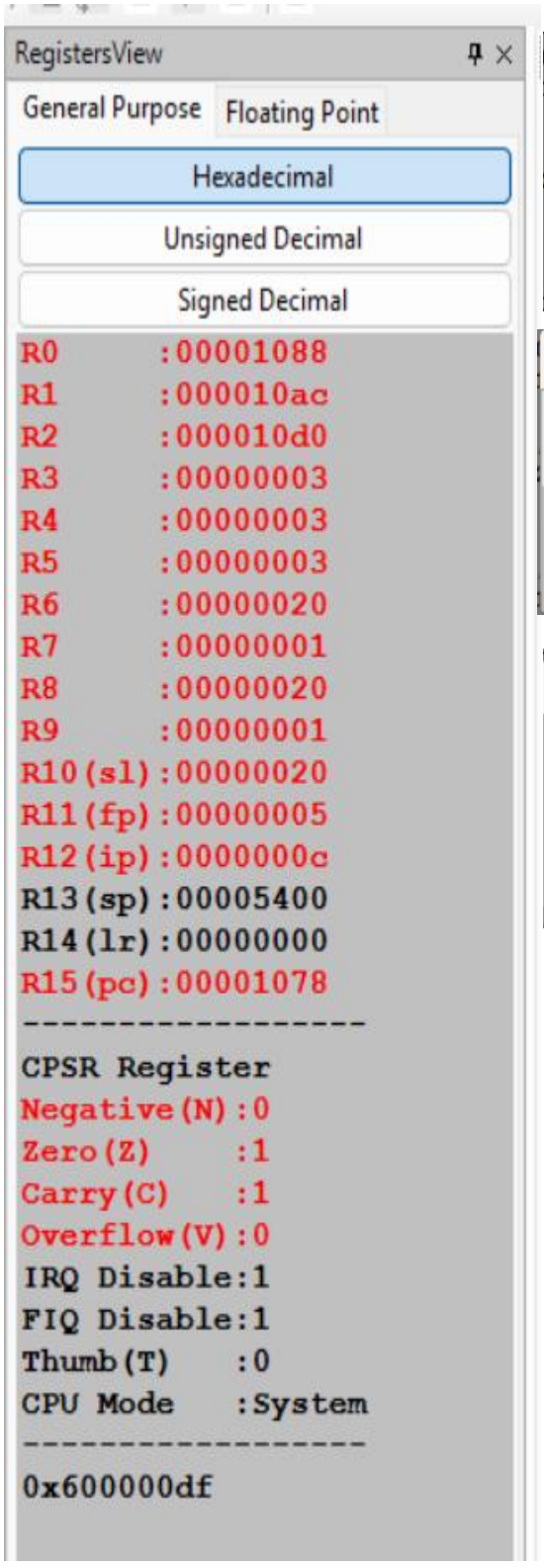

RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4536
R1 :4936
R2 :0
R3 :91
R4 :92
R5 :93
R6 :94
R7 :95
R8 :96
R9 :97
R10(sl):98
R11(fp):99
R12(ip):100
R13(sp):21504
R14(lr):0
R15(pc):4124
CPSR Register
Negative(N):0
Zero(Z) :1
Carry(C) :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x600000df

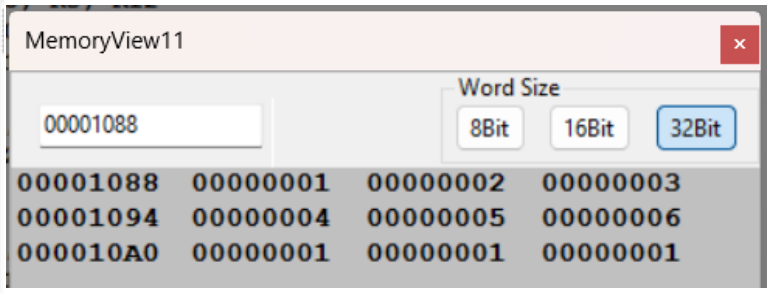
MATRIX MUL:

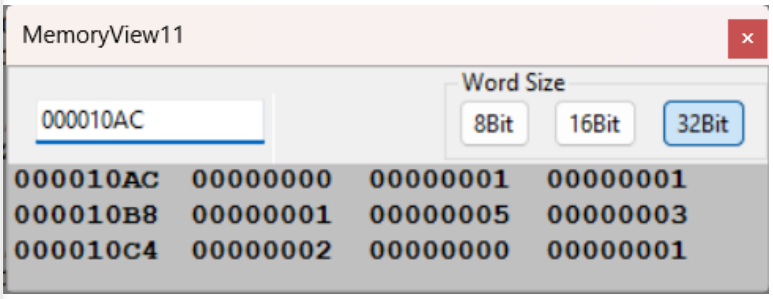
mat.s

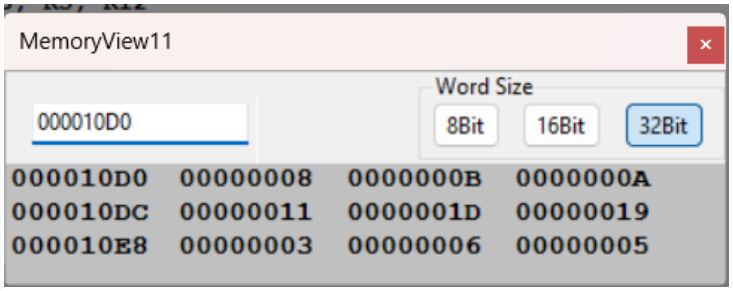
```
                .text
00001000:E59F0074       LDR R0, =A
00001004:E59F1074       LDR R1, =B
00001008:E59F2074       LDR R2, =C

0000100C:E3A03000       MOV R3, #0
00001010:E3A0C00C           MOV R12,#12

00001014:                   LOOP1:
00001014:E3530003                   CMP R3, #3
00001018:AA000016                   BGE END
0000101C:E3A04000                   MOV R4, #0

00001020:                           LOOP2:
00001020:E3540003                               CMP R4, #3
00001024:AA000011                               BGE INCREMENT_I
00001028:E3A0B000                               MOV R11, #0
0000102C:E3A05000                               MOV R5, #0

00001030:                                   LOOP3:
00001030:E3550003                                       CMP R5, #3
00001034:AA000008                                       BGE INCREMENT_J

00001038:E0060C93                                       MUL R6, R3, R12
0000103C:E0866105                                       ADD R6, R6, R5, LSL #2
00001040:E7907006                                       LDR R7, [R0, R6]

00001044:E0080C95                                       MUL R8, R5, R12
00001048:E0888104                                       ADD R8, R8, R4, LSL #2
0000104C:E7919008                                       LDR R9, [R1, R8]

00001050:E02BB997                                       MLA R11, R7, R9, R11

00001054:E2855001                                       ADD R5, R5, #1
00001058:EAFFFFF4                               B LOOP3

0000105C:                   INCREMENT_J:
0000105C:E00A0C93                   MUL R10, R3, R12
00001060:E08AA104                   ADD R10, R10, R4, LSL #2
00001064:E782B00A                   STR R11, [R2, R10]

00001068:E2844001                   ADD R4, R4, #1
0000106C:EAFFFFEB                   B LOOP2
```

```
00001070:                       INCREMENT_I:
00001070:E2833001                       ADD R3, R3, #1
00001074:EAFFFFE6                       B LOOP1

00001078:               END:
00001078:EF000011           SWI 0X011

                        .data
00001088:               A:  .word 1,2,3
                            .word 4,5,6
                            .word 1,1,1

000010AC:               B:  .word 0,1,1
                            .word 1,5,3
                            .word 2,0,1

000010D0:               C:  .word 0,0,0
                            .word 0,0,0
                            .word 0,0,0
```


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :00001088
R1 :000010ac
R2 :000010d0
R3 :00000003
R4 :00000003
R5 :00000003
R6 :00000020
R7 :00000001
R8 :00000020
R9 :00000001
R10(sl):00000020
R11(fp):00000005
R12(ip):0000000c
R13(sp):00005400
R14(lr):00000000
R15(pc):00001078
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


MemoryView11
00001088
Word Size
8Bit
16Bit
32Bit
00001088 00000001 00000002 00000003
00001094 00000004 00000005 00000006
000010A0 00000001 00000001 00000001


MemoryView11
000010AC
Word Size
8Bit
16Bit
32Bit
000010AC 00000000 00000001 00000001
000010B8 00000001 00000005 00000003
000010C4 00000002 00000000 00000001


MemoryView11
000010D0
Word Size
8Bit
16Bit
32Bit
000010D0 00000008 0000000B 0000000A
000010DC 00000011 0000001D 00000019
000010E8 00000003 00000006 00000005


THANK YOU