PES
UNIVERSITY

Department of Computer Science & Engineering

Microprocessor & Computer Architecture

MPCA-Laboratory/Assignment/Hands-on/Project

UE20CS252

NAME: VISHWAS M **SEC : F**

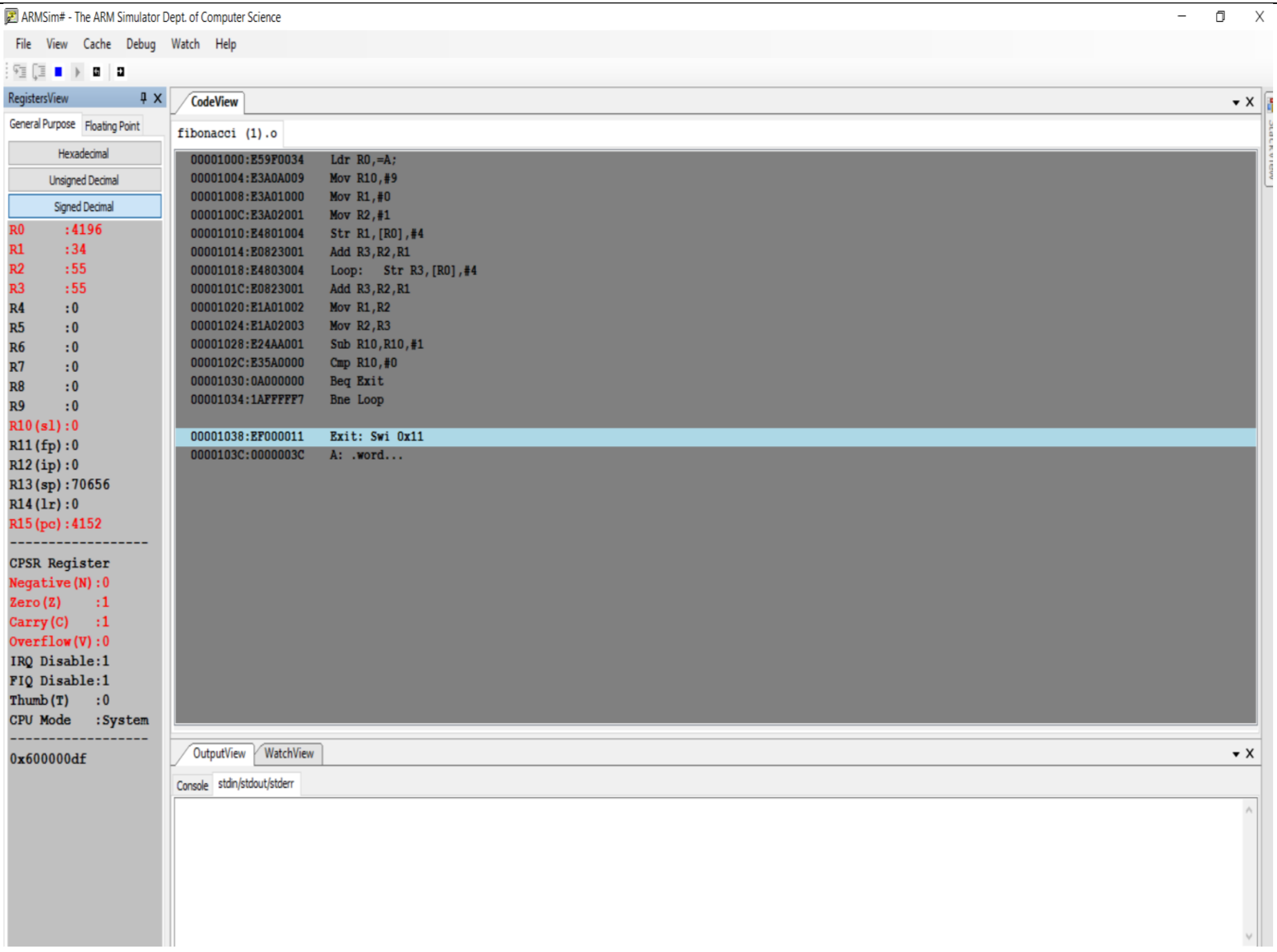
SRN : PES2UG20CS390 **DATE: 22/02/2022**

Sl. No	Programs
Week No.5	1. Write a program in ARM7TDMI-ISA to generate Fibonacci Series and store them in an array.

Program:

```
Ldr R0,=A;
Mov R10,#9
Mov R1,#0
Mov R2,#1
Str R1,[R0],#4
Add R3,R2,R1
Loop:   Str R3,[R0],#4
        Add R3,R2,R1
        Mov R1,R2
        Mov R2,R3
        Sub R10,R10,#1
        Cmp R10,#0
        Beq Exit
        Bne Loop

Exit: Swi 0x11
A: .word
```

Screenshot:



```
R0      :4196
R1      :34
R2      :55
R3      :55
R4      :0
R5      :0
R6      :0
R7      :0
R8      :0
R9      :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4152
------------------
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
```

fibonacci (1).o

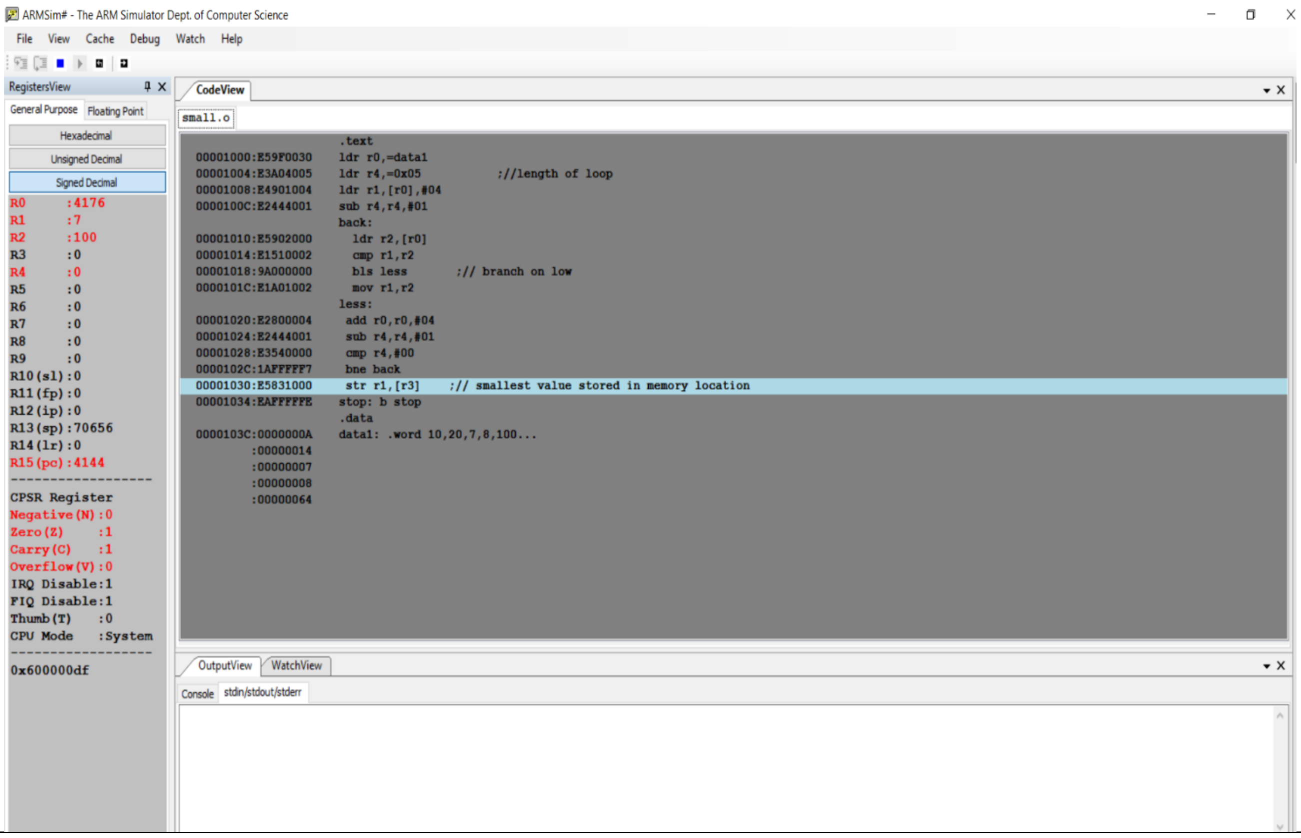
```
00001000:E59F0034    Ldr R0,=A;
00001004:E3A0A009    Mov R10,#9
00001008:E3A01000    Mov R1,#0
0000100C:E3A02001    Mov R2,#1
00001010:E4801004    Str R1,[R0],#4
00001014:E0823001    Add R3,R2,R1
00001018:E4803004    Loop:   Str R3,[R0],#4
0000101C:E0823001    Add R3,R2,R1
00001020:E1A01002    Mov R1,R2
00001024:E1A02003    Mov R2,R3
00001028:E24AA001    Sub R10,R10,#1
0000102C:E35A0000    Cmp R10,#0
00001030:0A000000    Beq Exit
00001034:1AFFFFF7    Bne Loop

00001038:EF000011    Exit: Swi 0x11
0000103C:0000003C    A: .word...
```

2. Write a program in ARM7TDMI-ISA to find smallest number in an array of n 32 bit numbers

Program:

```
.text
ldr r0,=data1
ldr r4,=0x05          ;//length of loop
ldr r1,[r0],#04
sub r4,r4,#01
back:
  ldr r2,[r0]
  cmp r1,r2
  bls less      ;// branch on low
  mov r1,r2
less:
 add r0,r0,#04
 sub r4,r4,#01
 cmp r4,#00
 bne back
 str r1,[r3]    ;// smallest value stored in memory location
stop: b stop
.data
data1: .word 10,20,7,8,100
```

Screenshot:



```
R0       :4176
R1       :7
R2       :100
R3       :0
R4       :0
R5       :0
R6       :0
R7       :0
R8       :0
R9       :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
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    :System
-----------------
0x600000df
```

```
                    .text
00001000:E59F0030   ldr r0,=data1
00001004:E3A04005   ldr r4,=0x05          ;//length of loop
00001008:E4901004   ldr r1,[r0],#04
0000100C:E2444001   sub r4,r4,#01
                    back:
00001010:E5902000     ldr r2,[r0]
00001014:E1510002     cmp r1,r2
00001018:9A000000     bls less      ;// branch on low
0000101C:E1A01002     mov r1,r2
                    less:
00001020:E2800004    add r0,r0,#04
00001024:E2444001    sub r4,r4,#01
00001028:E3540000    cmp r4,#00
0000102C:1AFFFFF7    bne back
00001030:E5831000    str r1,[r3]    ;// smallest value stored in memory location
00001034:EAFFFFFE   stop: b stop
                    .data
0000103C:0000000A   data1: .word 10,20,7,8,100...
        :00000014
        :00000007
        :00000008
        :00000064
```

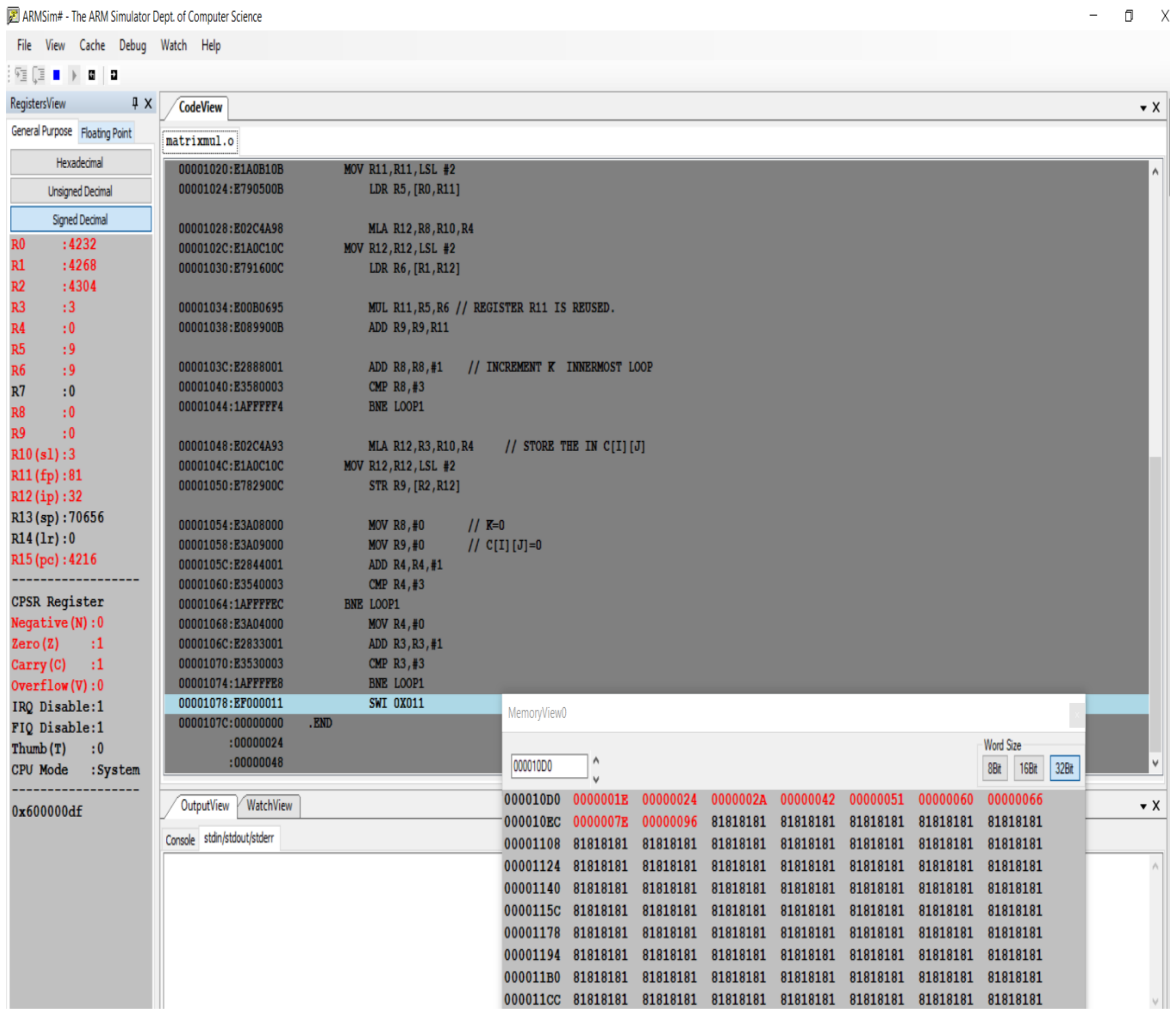
3. Write a program in ARM7TDMI-ISA to multiply 2 matrices of order3.
   i.e., implement c[i][j]=c[i][j] + a[i][j] x b[i][j].
   a. Use MLA instruction
   b. Use MUL instruction

Program:

```
; MULTIPLICATION OF 2 MATRICES.
.DATA
A: .WORD 1,2,3,4,5,6,7,8,9
B: .WORD 1,2,3,4,5,6,7,8,9
C: .WORD 0,0,0,0,0,0,0,0,0

.TEXT
    LDR R0,=A
    LDR R1,=B
    LDR R2,=C

    MOV R3,#0  ;INNER LOOP COUNT I INDEX
    MOV R4,#0  ;OUTER LOOP COUNT J INDEX
    MOV R10,#3 ; NUMBER OF ELEMENTS IN A ROW
    MOV R8,#0 ;VALUE OF K

LOOP1:MLA R11,R3,R10,R8
    MOV R11,R11,LSL #2
     LDR R5,[R0,R11]

     MLA R12,R8,R10,R4
    MOV R12,R12,LSL #2
     LDR R6,[R1,R12]

     MUL R11,R5,R6 ; REGISTER R11 IS REUSED.
     ADD R9,R9,R11

     ADD R8,R8,#1    ; INCREMENT K  INNERMOST LOOP
     CMP R8,#3
     BNE LOOP1

     MLA R12,R3,R10,R4     ; STORE THE IN C[I][J]
    MOV R12,R12,LSL #2
     STR R9,[R2,R12]
```

```
        MOV R8,#0       ; K=0
        MOV R9,#0       ; C[I][J]=0
        ADD R4,R4,#1
        CMP R4,#3
       BNE LOOP1
        MOV R4,#0
        ADD R3,R3,#1
        CMP R3,#3
        BNE LOOP1
        SWI 0X011
.END
```

Screenshot:



```
R0      :4232
R1      :4268
R2      :4304
R3      :3
R4      :0
R5      :9
R6      :9
R7      :0
R8      :0
R9      :0
R10(sl):3
R11(fp):81
R12(ip):32
R13(sp):70656
R14(lr):0
R15(pc):4216
-----------------
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
```

```
00001020:E1A0B10B          MOV R11,R11,LSL #2
00001024:E790500B              LDR R5,[R0,R11]

00001028:E02C4A98              MLA R12,R8,R10,R4
0000102C:E1A0C10C          MOV R12,R12,LSL #2
00001030:E791600C              LDR R6,[R1,R12]

00001034:E00B0695              MUL R11,R5,R6 // REGISTER R11 IS REUSED.
00001038:E089900B              ADD R9,R9,R11

0000103C:E2888001              ADD R8,R8,#1    // INCREMENT K  INNERMOST LOOP
00001040:E3580003              CMP R8,#3
00001044:1AFFFFF4              BNE LOOP1

00001048:E02C4A93              MLA R12,R3,R10,R4     // STORE THE IN C[I][J]
0000104C:E1A0C10C          MOV R12,R12,LSL #2
00001050:E782900C              STR R9,[R2,R12]

00001054:E3A08000              MOV R8,#0       // K=0
00001058:E3A09000              MOV R9,#0       // C[I][J]=0
0000105C:E2844001              ADD R4,R4,#1
00001060:E3540003              CMP R4,#3
00001064:1AFFFFEC          BNE LOOP1
00001068:E3A04000              MOV R4,#0
0000106C:E2833001              ADD R3,R3,#1
00001070:E3530003              CMP R3,#3
00001074:1AFFFFE8              BNE LOOP1
00001078:EF000011              SWI 0X011
0000107C:00000000      .END
         :00000024
         :00000048
```

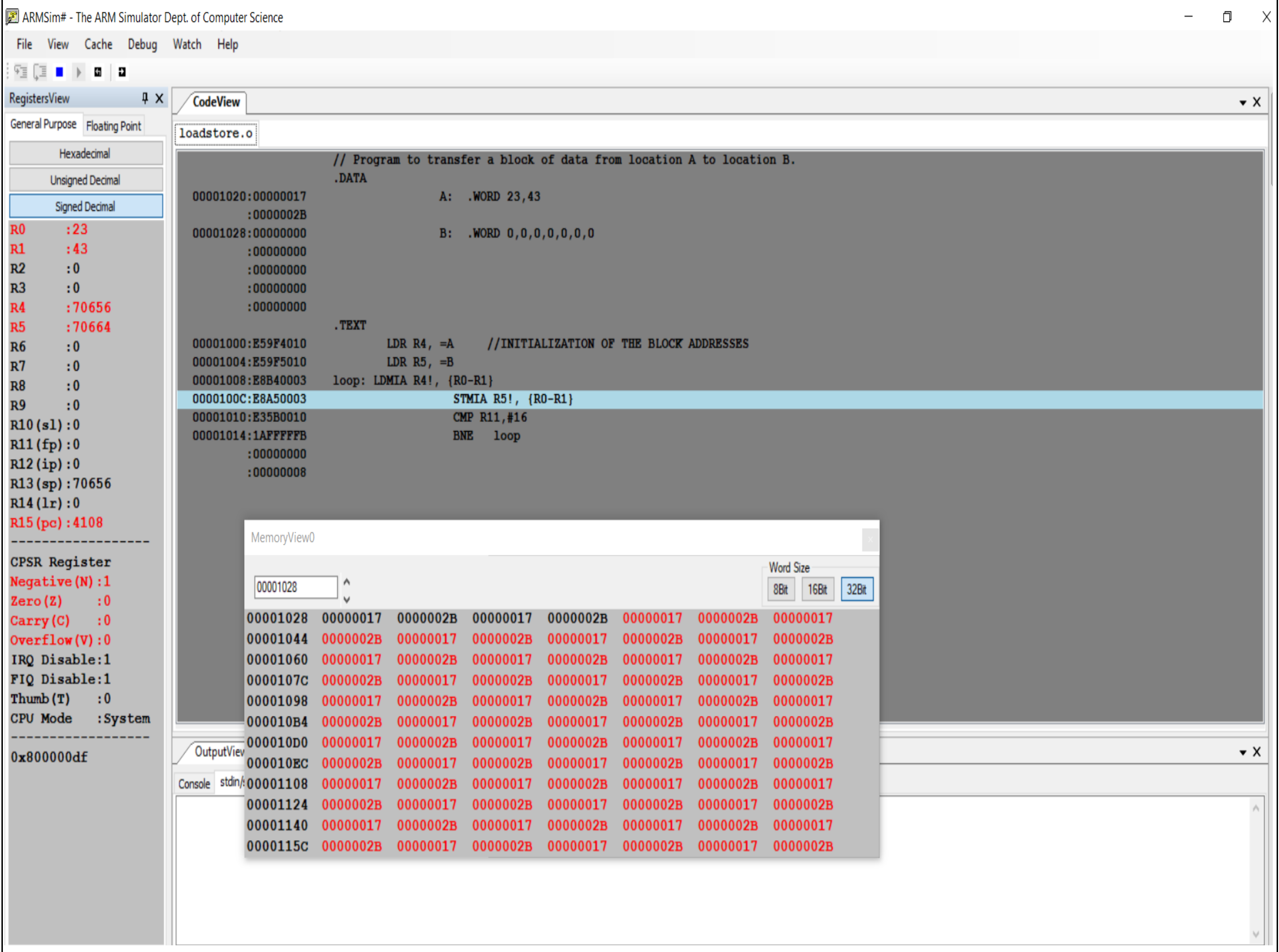
```
000010D0
000010D0  0000001E  00000024  0000002A  00000042  00000051  00000060  00000066
000010EC  0000007E  00000096  81818181  81818181  81818181  81818181  81818181
00001108  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001124  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001140  81818181  81818181  81818181  81818181  81818181  81818181  81818181
0000115C  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001178  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001194  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000011B0  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000011CC  81818181  81818181  81818181  81818181  81818181  81818181  81818181
```

4. Write a program in ARM7TDMI-ISA to transfer a block of 256 words stored at memory location X to memory location Y using Load Multiple and Store Multiple instructions. The rate of transfer is 32 bytes.

Programs:

```
    // Program to transfer a block of data from location X to location Y.
.DATA
                A:  .WORD 23,43
                B:  .WORD 0,0,0,0,0,0,0
.TEXT
        LDR R4, =A     //INITIALIZATION OF THE BLOCK ADDRESSES
        LDR R5, =B
loop: LDMIA R4!, {R0-R1}
                STMIA R5!, {R0-R1}
                CMP R11,#16
                BNE   loop
```

Screenshot:



ARMSim# - The ARM Simulator Dept. of Computer Science
File View Cache Debug Watch Help
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
CodeView
loadstore.o
MemoryView0
OutputView
Console stdin/

```
R0      :23
R1      :43
R2      :0
R3      :0
R4      :70656
R5      :70664
R6      :0
R7      :0
R8      :0
R9      :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4108
------------------
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    :System
------------------
0x800000df
```

```
                     // Program to transfer a block of data from location A to location B.
                     .DATA
00001020:00000017                A:  .WORD 23,43
        :0000002B
00001028:00000000                B:  .WORD 0,0,0,0,0,0,0
        :00000000
        :00000000
        :00000000
        :00000000
                     .TEXT
00001000:E59F4010          LDR R4, =A     //INITIALIZATION OF THE BLOCK ADDRESSES
00001004:E59F5010          LDR R5, =B
00001008:E8B40003    loop: LDMIA R4!, {R0-R1}
0000100C:E8A50003                STMIA R5!, {R0-R1}
00001010:E35B0010                CMP R11,#16
00001014:1AFFFFFB                BNE   loop
        :00000000
        :00000008
```

```
00001028                                     Word Size  8Bit  16Bit  32Bit
00001028 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
00001044 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
00001060 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
0000107C 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
00001098 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
000010B4 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
000010D0 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
000010EC 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
00001108 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
00001124 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
00001140 00000017 0000002B 00000017 0000002B 00000017 0000002B 00000017
0000115C 0000002B 00000017 0000002B 00000017 0000002B 00000017 0000002B
```

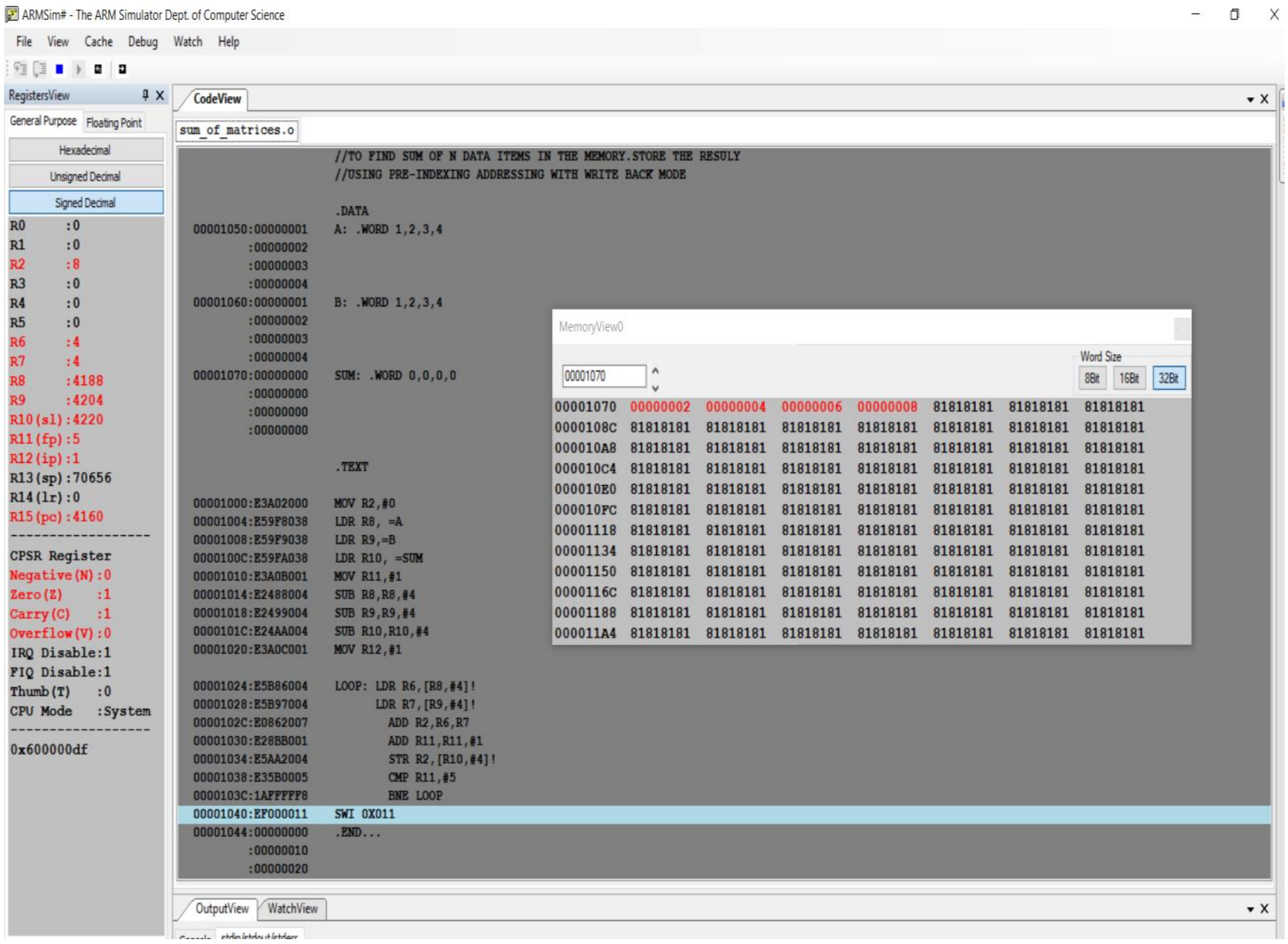
Student exercises:

1. Write a program in ARM7TDMI-ISA to add 2 matrices of order3.
   i.e., Implement c[i][j]= a[i][j] + b[i][j].

Program:

```
  //TO FIND SUM OF N DATA ITEMS IN THE MEMORY.STORE THE RESULY
//USING PRE-INDEXING ADDRESSING WITH WRITE BACK MODE

.DATA
A: .WORD 1,2,3,4
B: .WORD 1,2,3,4
SUM: .WORD 0,0,0,0

.TEXT

MOV R2,#0
LDR R8, =A
LDR R9,=B
LDR R10, =SUM
MOV R11,#1
SUB R8,R8,#4
SUB R9,R9,#4
SUB R10,R10,#4
MOV R12,#1

LOOP: LDR R6,[R8,#4]!
    LDR R7,[R9,#4]!
        ADD R2,R6,R7
        ADD R11,R11,#1
      STR R2,[R10,#4]!
        CMP R11,#5
        BNE LOOP
SWI 0X011
.END
```

Screenshot:



```
R0       :0
R1       :0
R2       :8
R3       :0
R4       :0
R5       :0
R6       :4
R7       :4
R8       :4188
R9       :4204
R10(sl):4220
R11(fp):5
R12(ip):1
R13(sp):70656
R14(lr):0
R15(pc):4160
------------------
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
```

```
                      //TO FIND SUM OF N DATA ITEMS IN THE MEMORY.STORE THE RESULY
                      //USING PRE-INDEXING ADDRESSING WITH WRITE BACK MODE

                      .DATA
00001050:00000001     A: .WORD 1,2,3,4
        :00000002
        :00000003
        :00000004
00001060:00000001     B: .WORD 1,2,3,4
        :00000002
        :00000003
        :00000004
00001070:00000000     SUM: .WORD 0,0,0,0
        :00000000
        :00000000
        :00000000

                      .TEXT

00001000:E3A02000     MOV R2,#0
00001004:E59F8038     LDR R8, =A
00001008:E59F9038     LDR R9,=B
0000100C:E59FA038     LDR R10, =SUM
00001010:E3A0B001     MOV R11,#1
00001014:E2488004     SUB R8,R8,#4
00001018:E2499004     SUB R9,R9,#4
0000101C:E24AA004     SUB R10,R10,#4
00001020:E3A0C001     MOV R12,#1

00001024:E5B86004     LOOP: LDR R6,[R8,#4]!
00001028:E5B97004           LDR R7,[R9,#4]!
0000102C:E0862007             ADD R2,R6,R7
00001030:E28BB001             ADD R11,R11,#1
00001034:E5AA2004             STR R2,[R10,#4]!
00001038:E35B0005             CMP R11,#5
0000103C:1AFFFFF8             BNE LOOP
00001040:EF000011     SWI 0X011
00001044:00000000     .END...
        :00000010
        :00000020
```

```
00001070
00001070  00000002  00000004  00000006  00000008  81818181  81818181  81818181
0000108C  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010A8  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010C4  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010E0  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010FC  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001118  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001134  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001150  81818181  81818181  81818181  81818181  81818181  81818181  81818181
0000116C  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001188  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000011A4  81818181  81818181  81818181  81818181  81818181  81818181  81818181
```

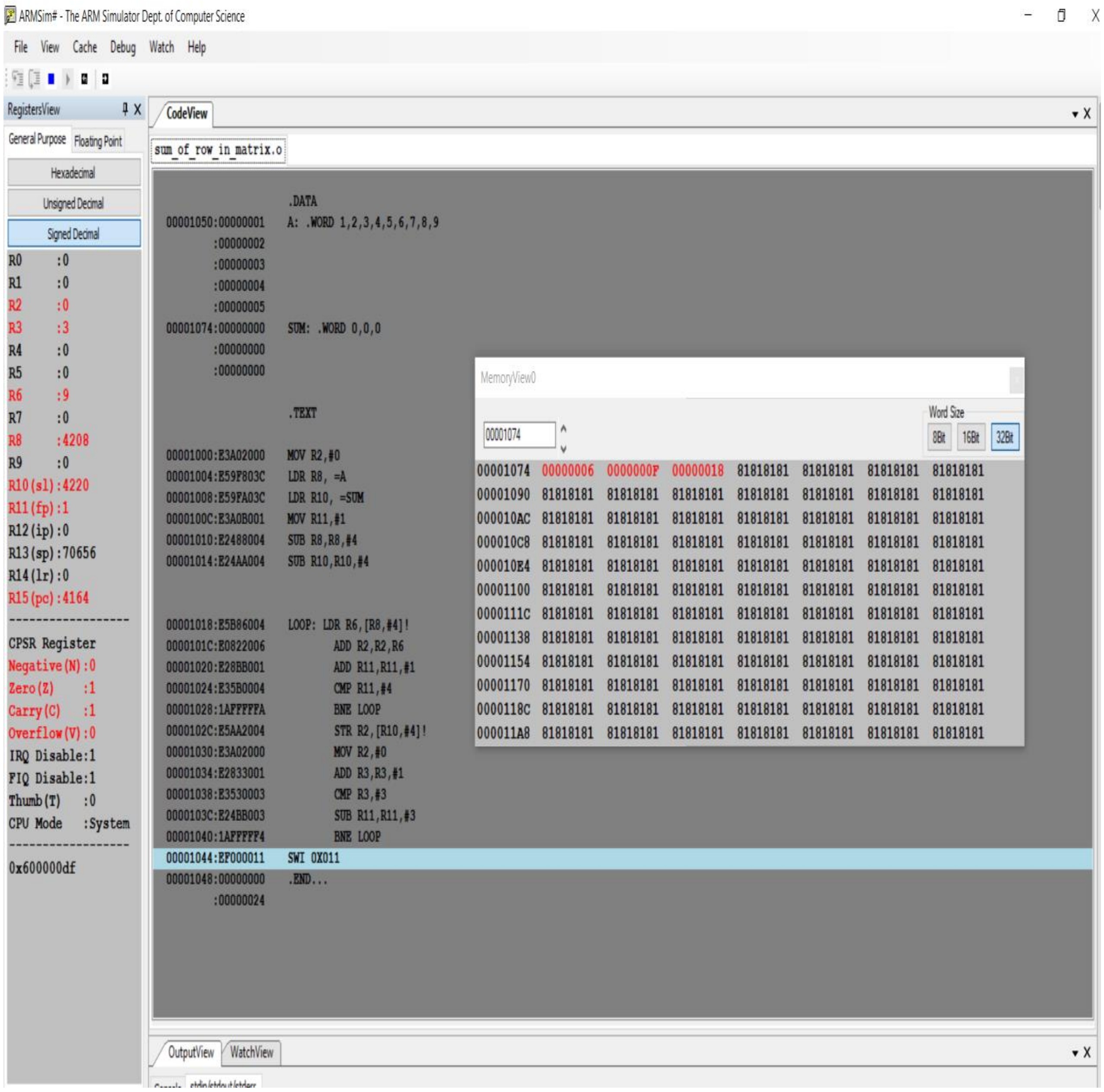
2. Write a program in ARM7TDMI-ISA to find the ROWSUM of a matrix.

Program:

```
.DATA
A: .WORD 1,2,3,4,5,6,7,8,9
SUM: .WORD 0,0,0

.TEXT

MOV R2,#0
LDR R8, =A
LDR R10, =SUM
MOV R11,#1
SUB R8,R8,#4
SUB R10,R10,#4
```

```
LOOP: LDR R6,[R8,#4]!
     ADD R2,R2,R6
     ADD R11,R11,#1
     CMP R11,#4
     BNE LOOP
       STR R2,[R10,#4]!
       MOV R2,#0
       ADD R3,R3,#1
       CMP R3,#3
       SUB R11,R11,#3
       BNE LOOP
SWI 0X011
.END
```

Screenshot:



```
R0       :0
R1       :0
R2       :0
R3       :3
R4       :0
R5       :0
R6       :9
R7       :0
R8       :4208
R9       :0
R10(sl):4220
R11(fp):1
R12(ip):0
R13(sp):70656
R14(lr):0
R15(pc):4164
------------------
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
```

```
                        .DATA
00001050:00000001       A: .WORD 1,2,3,4,5,6,7,8,9
         :00000002
         :00000003
         :00000004
         :00000005
00001074:00000000       SUM: .WORD 0,0,0
         :00000000
         :00000000

                        .TEXT

00001000:E3A02000       MOV R2,#0
00001004:E59F803C       LDR R8, =A
00001008:E59FA03C       LDR R10, =SUM
0000100C:E3A0B001       MOV R11,#1
00001010:E2488004       SUB R8,R8,#4
00001014:E24AA004       SUB R10,R10,#4


00001018:E5B86004       LOOP: LDR R6,[R8,#4]!
0000101C:E0822006             ADD R2,R2,R6
00001020:E28BB001             ADD R11,R11,#1
00001024:E35B0004             CMP R11,#4
00001028:1AFFFFFA             BNE LOOP
0000102C:E5AA2004             STR R2,[R10,#4]!
00001030:E3A02000             MOV R2,#0
00001034:E2833001             ADD R3,R3,#1
00001038:E3530003             CMP R3,#3
0000103C:E24BB003             SUB R11,R11,#3
00001040:1AFFFFF4             BNE LOOP
00001044:EF000011       SWI 0X011
00001048:00000000       .END...
         :00000024
```

```
00001074
00001074  00000006  0000000F  00000018  81818181  81818181  81818181  81818181
00001090  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010AC  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010C8  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000010E4  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001100  81818181  81818181  81818181  81818181  81818181  81818181  81818181
0000111C  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001138  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001154  81818181  81818181  81818181  81818181  81818181  81818181  81818181
00001170  81818181  81818181  81818181  81818181  81818181  81818181  81818181
0000118C  81818181  81818181  81818181  81818181  81818181  81818181  81818181
000011A8  81818181  81818181  81818181  81818181  81818181  81818181  81818181
```