MPCA LAB WEEK 3

NAME: DHARANI S

SRN: PES2UG24CS157

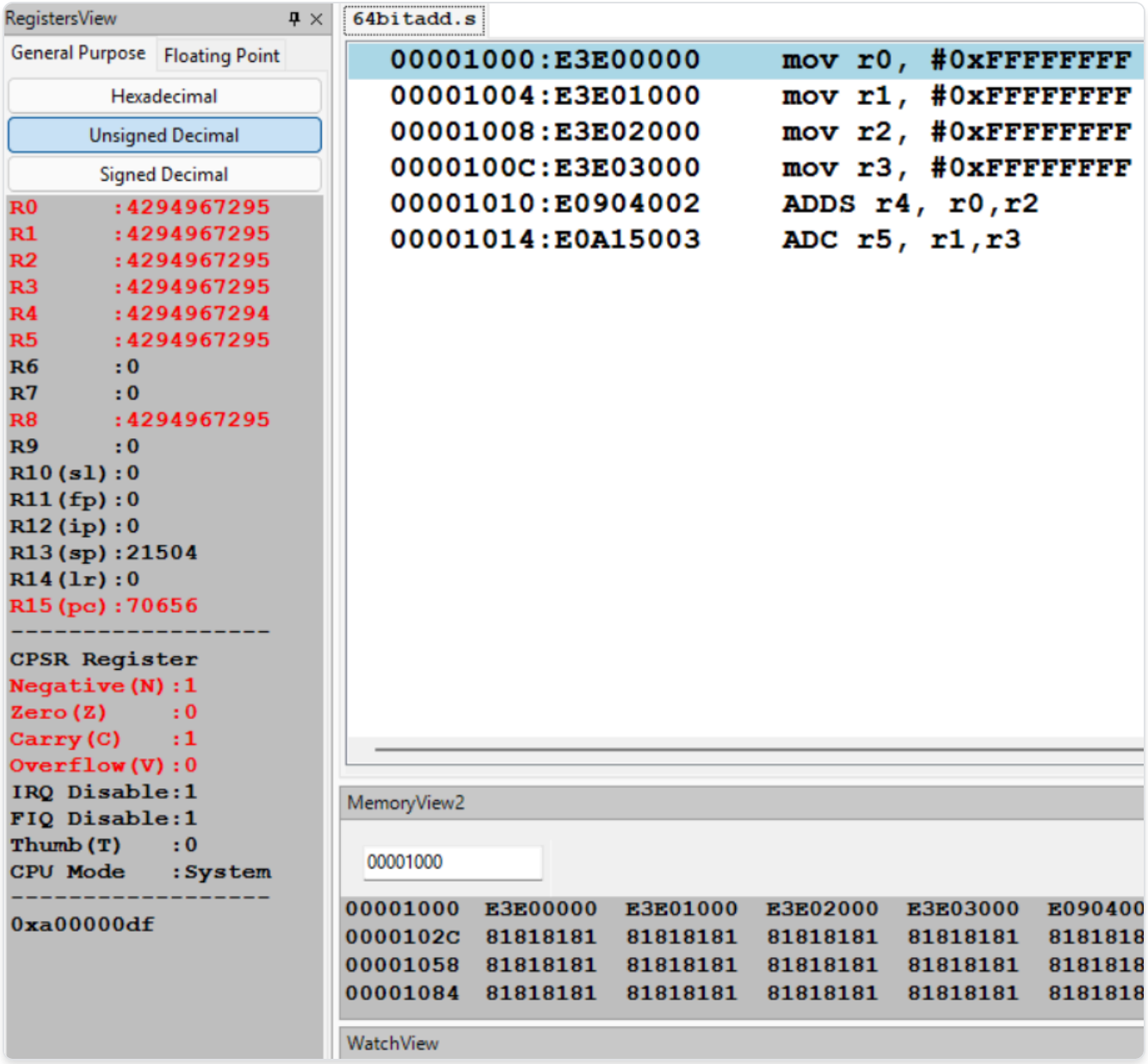
SECTION: C

DATE: 29/01/2026

> program 1:Write an ALP to perform Addition for of two numbers of sizes 64 bit and 128 bit and save the result in register (reuse the register to store the result).

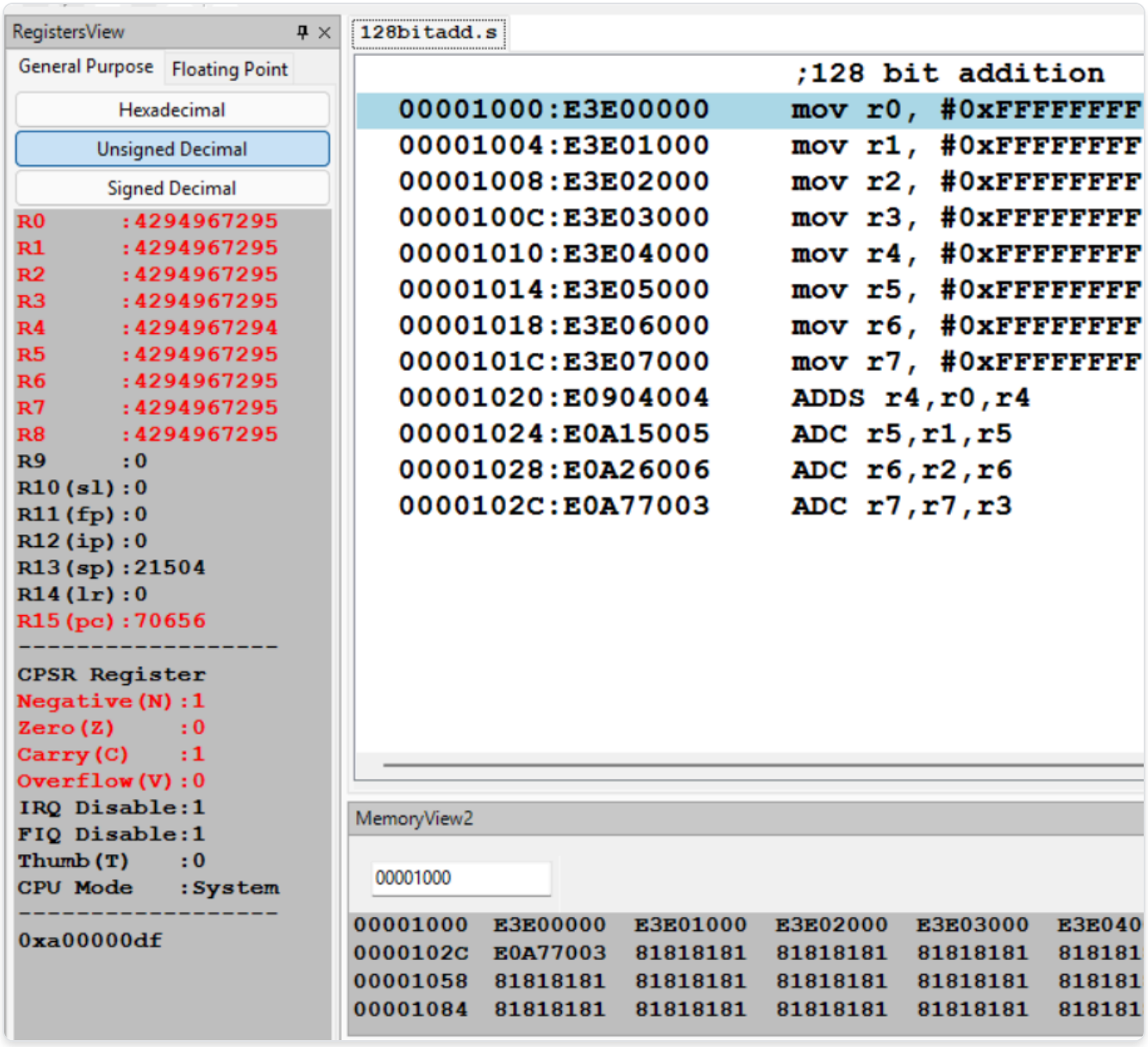
› 64 bit addition:

```
mov r0, #0xFFFFFFFF
mov r1, #0xFFFFFFFF
mov r2, #0xFFFFFFFF
mov r3, #0xFFFFFFFF
ADDS r4, r0,r2
ADC r5, r1,r3
```


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4294967295
R1 :4294967295
R2 :4294967295
R3 :4294967295
R4 :4294967294
R5 :4294967295
R6 :0
R7 :0
R8 :4294967295
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
CPSR Register
Negative(N):1
Zero(Z) :0
Carry(C) :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0xa00000df
64bitadd.s
00001000:E3E00000 mov r0, #0xFFFFFFFF
00001004:E3E01000 mov r1, #0xFFFFFFFF
00001008:E3E02000 mov r2, #0xFFFFFFFF
0000100C:E3E03000 mov r3, #0xFFFFFFFF
00001010:E0904002 ADDS r4, r0,r2
00001014:E0A15003 ADC r5, r1,r3
MemoryView2
00001000
00001000 E3E00000 E3E01000 E3E02000 E3E03000 E090400
0000102C 81818181 81818181 81818181 81818181 8181818
00001058 81818181 81818181 81818181 81818181 8181818
00001084 81818181 81818181 81818181 81818181 8181818
WatchView

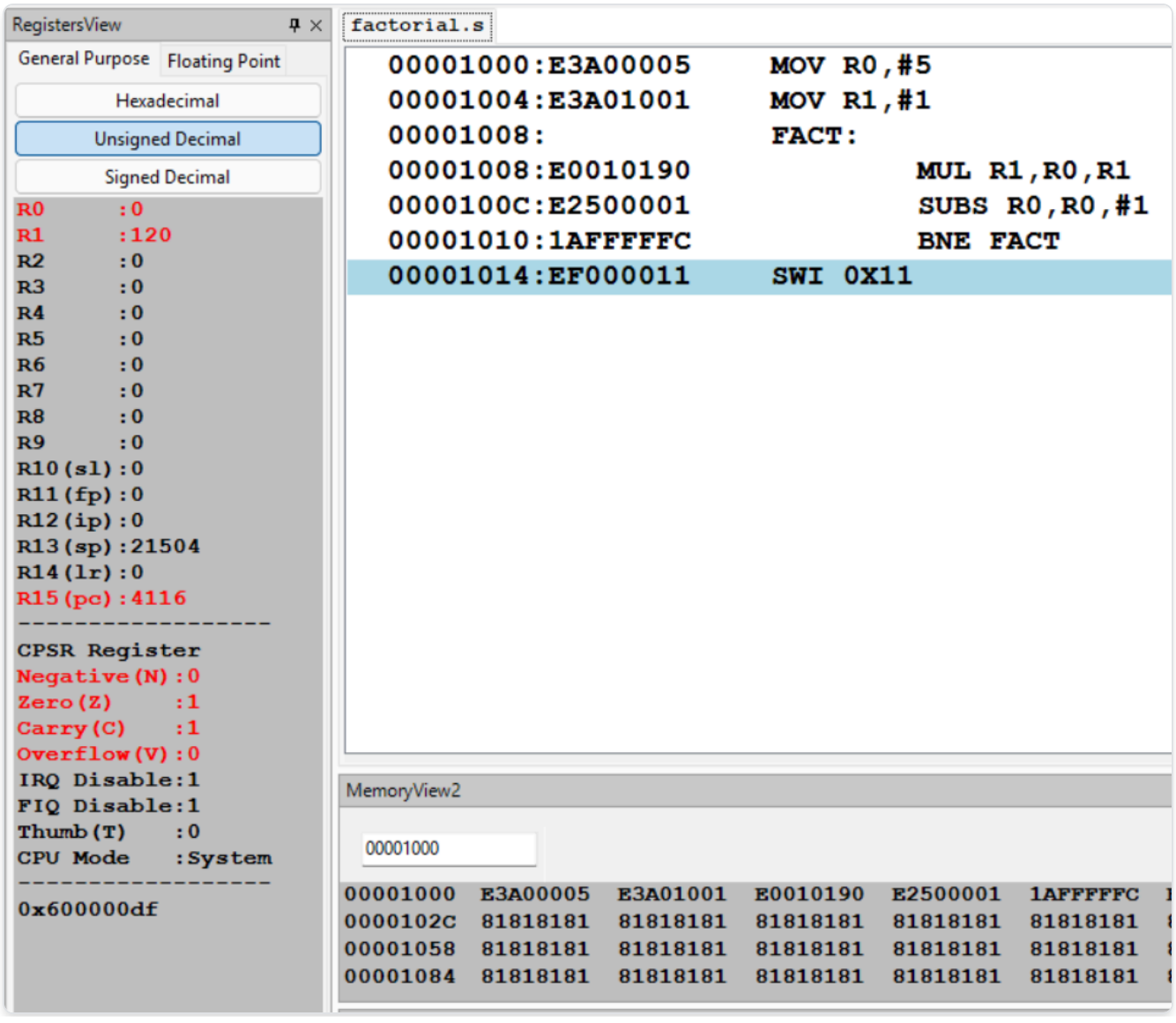
› 128 bit addition:

```
;128 bit addition
mov r0, #0xFFFFFFFF
mov r1, #0xFFFFFFFF
mov r2, #0xFFFFFFFF
mov r3, #0xFFFFFFFF
mov r4, #0xFFFFFFFF
mov r5, #0xFFFFFFFF
mov r6, #0xFFFFFFFF
mov r7, #0xFFFFFFFF
ADDS r4,r0,r4
ADC r5,r1,r5
ADC r6,r2,r6
ADC r7,r7,r3
```


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4294967295
R1 :4294967295
R2 :4294967295
R3 :4294967295
R4 :4294967294
R5 :4294967295
R6 :4294967295
R7 :4294967295
R8 :4294967295
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):70656
CPSR Register
Negative(N):1
Zero(Z) :0
Carry(C) :1
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0xa00000df
128bitadd.s
;128 bit addition
00001000:E3E00000 mov r0, #0xFFFFFFFF
00001004:E3E01000 mov r1, #0xFFFFFFFF
00001008:E3E02000 mov r2, #0xFFFFFFFF
0000100C:E3E03000 mov r3, #0xFFFFFFFF
00001010:E3E04000 mov r4, #0xFFFFFFFF
00001014:E3E05000 mov r5, #0xFFFFFFFF
00001018:E3E06000 mov r6, #0xFFFFFFFF
0000101C:E3E07000 mov r7, #0xFFFFFFFF
00001020:E0904004 ADDS r4,r0,r4
00001024:E0A15005 ADC r5,r1,r5
00001028:E0A26006 ADC r6,r2,r6
0000102C:E0A77003 ADC r7,r7,r3
MemoryView2
00001000
00001000 E3E00000 E3E01000 E3E02000 E3E03000 E3E040
0000102C E0A77003 81818181 81818181 81818181 818181
00001058 81818181 81818181 81818181 81818181 818181
00001084 81818181 81818181 81818181 81818181 818181

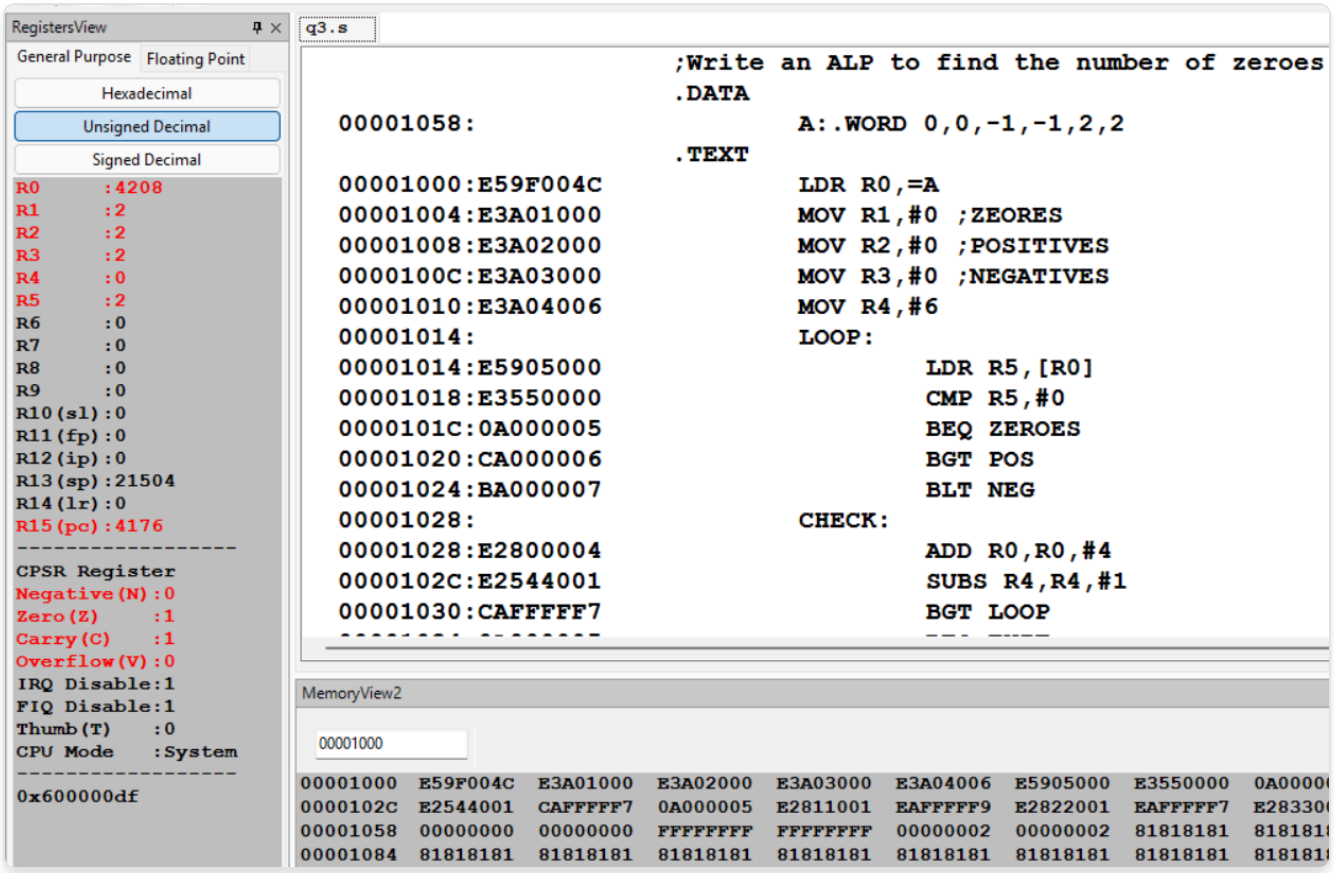
program 2: Write a program to find the factorial of a given number.

```
MOV R0,#5
MOV R1,#1
FACT:
        MUL R1,R0,R1
        SUBS R0,R0,#1
        BNE FACT
SWI 0X11
```



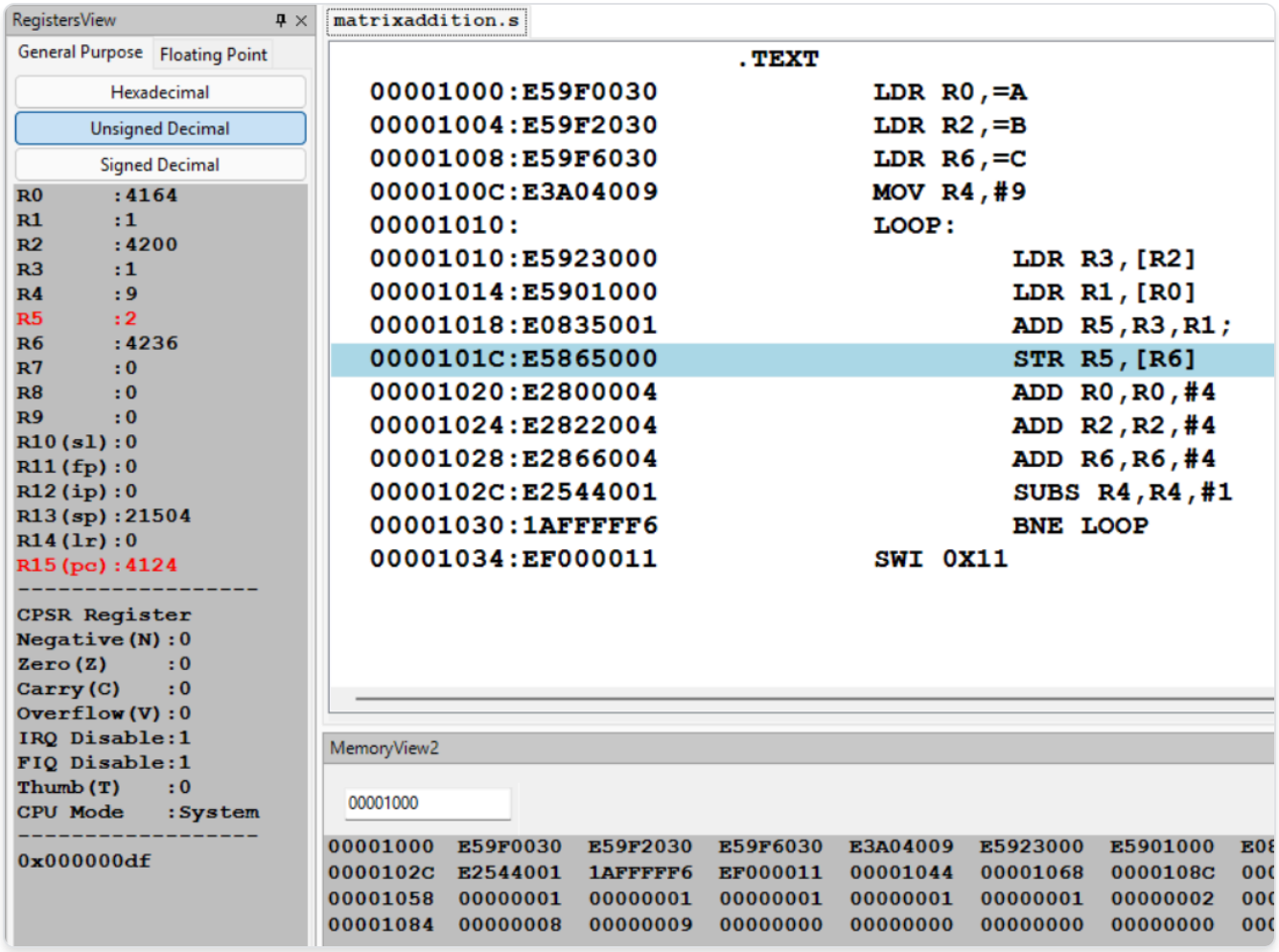
program 3: Write an ALP to find the number of zeroes, positive and negative numbers in a given array:

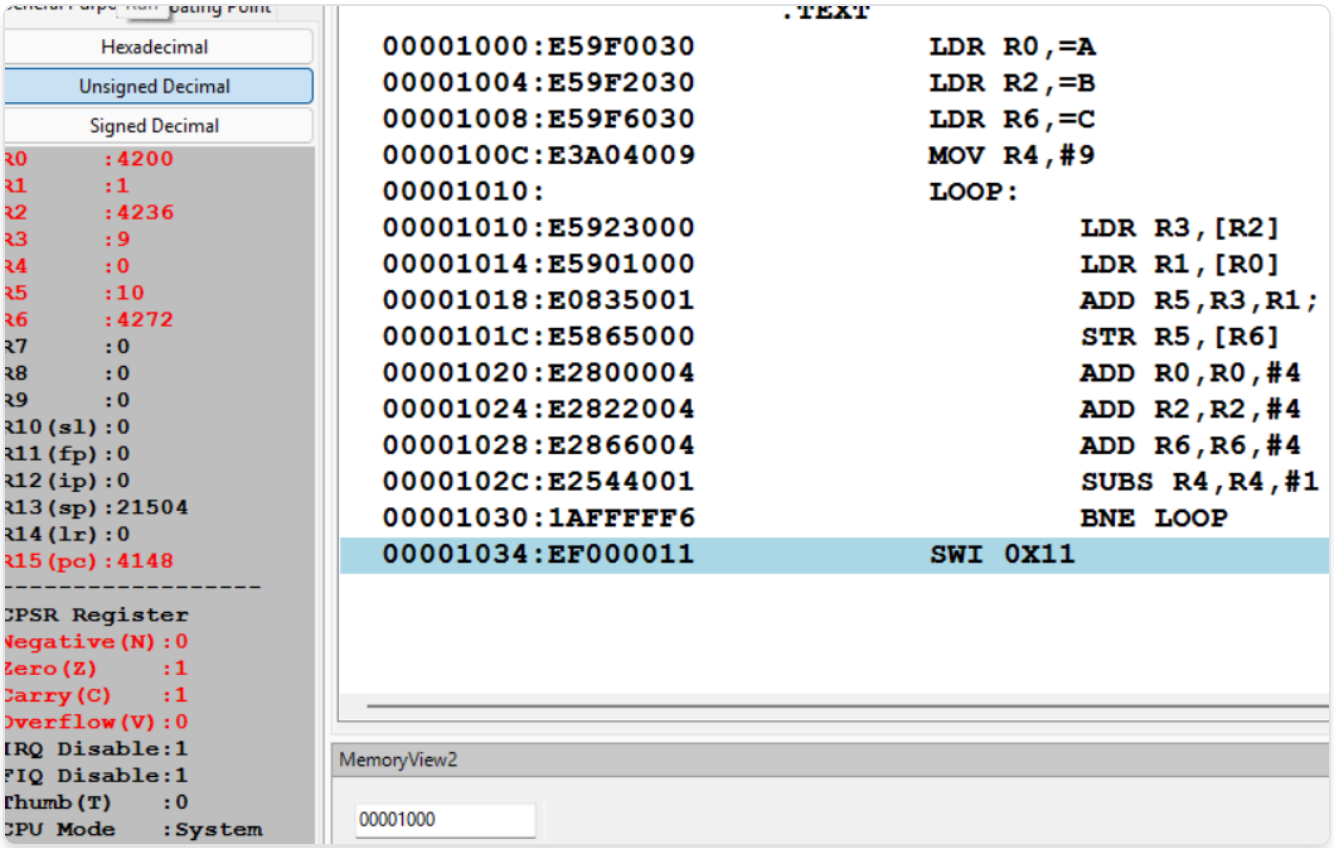
```
.DATA
        A:.WORD 0,0,-1,-1,2,2
.TEXT
        LDR R0,=A
        MOV R1,#0 ;ZEORES
        MOV R2,#0 ;POSITIVES
        MOV R3,#0 ;NEGATIVES
        MOV R4,#6
        LOOP:
                LDR R5,[R0]
                CMP R5,#0
                BEQ ZEROES
                BGT POS
                BLT NEG
        CHECK:
                ADD R0,R0,#4
                SUBS R4,R4,#1
                BGT LOOP
                BEQ EXIT
        ZEROES:
                ADD R1,R1,#1
                B CHECK

        POS:
                ADD R2,R2,#1
                B CHECK
        NEG:
                ADD R3,R3,#1
                B CHECK

        EXIT:
                SWI 0X11
```


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4208
R1 :2
R2 :2
R3 :2
R4 :0
R5 :2
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):4176
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
q3.s
;Write an ALP to find the number of zeroes
.DATA
00001058: A:.WORD 0,0,-1,-1,2,2
.TEXT
00001000:E59F004C LDR R0,=A
00001004:E3A01000 MOV R1,#0 ;ZEORES
00001008:E3A02000 MOV R2,#0 ;POSITIVES
0000100C:E3A03000 MOV R3,#0 ;NEGATIVES
00001010:E3A04006 MOV R4,#6
00001014: LOOP:
00001014:E5905000 LDR R5,[R0]
00001018:E3550000 CMP R5,#0
0000101C:0A000005 BEQ ZEROES
00001020:CA000006 BGT POS
00001024:BA000007 BLT NEG
00001028: CHECK:
00001028:E2800004 ADD R0,R0,#4
0000102C:E2544001 SUBS R4,R4,#1
00001030:CAFFFFF7 BGT LOOP
MemoryView2
00001000
00001000 E59F004C E3A01000 E3A02000 E3A03000 E3A04006 E5905000 E3550000 0A0000
0000102C E2544001 CAFFFFF7 0A000005 E2811001 EAFFFFF9 E2822001 EAFFFFF7 E2833
00001058 00000000 00000000 FFFFFFFF FFFFFFFF 00000002 00000002 81818181 8181
00001084 81818181 81818181 81818181 81818181 81818181 81818181 81818181 8181

program 4: Write a program to perform 3X3 matrix addition.

```
.DATA
        A:.WORD 1,1,1,1,1,1,1,1,1
        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 R2,=B
        LDR R6,=C
        MOV R4,#9
        LOOP:
                LDR R3,[R2]
                LDR R1,[R0]
                ADD R5,R3,R1;
                STR R5,[R6]
                ADD R0,R0,#4
                ADD R2,R2,#4
                ADD R6,R6,#4
                SUBS R4,R4,#1
                BNE LOOP
        SWI 0X11
```

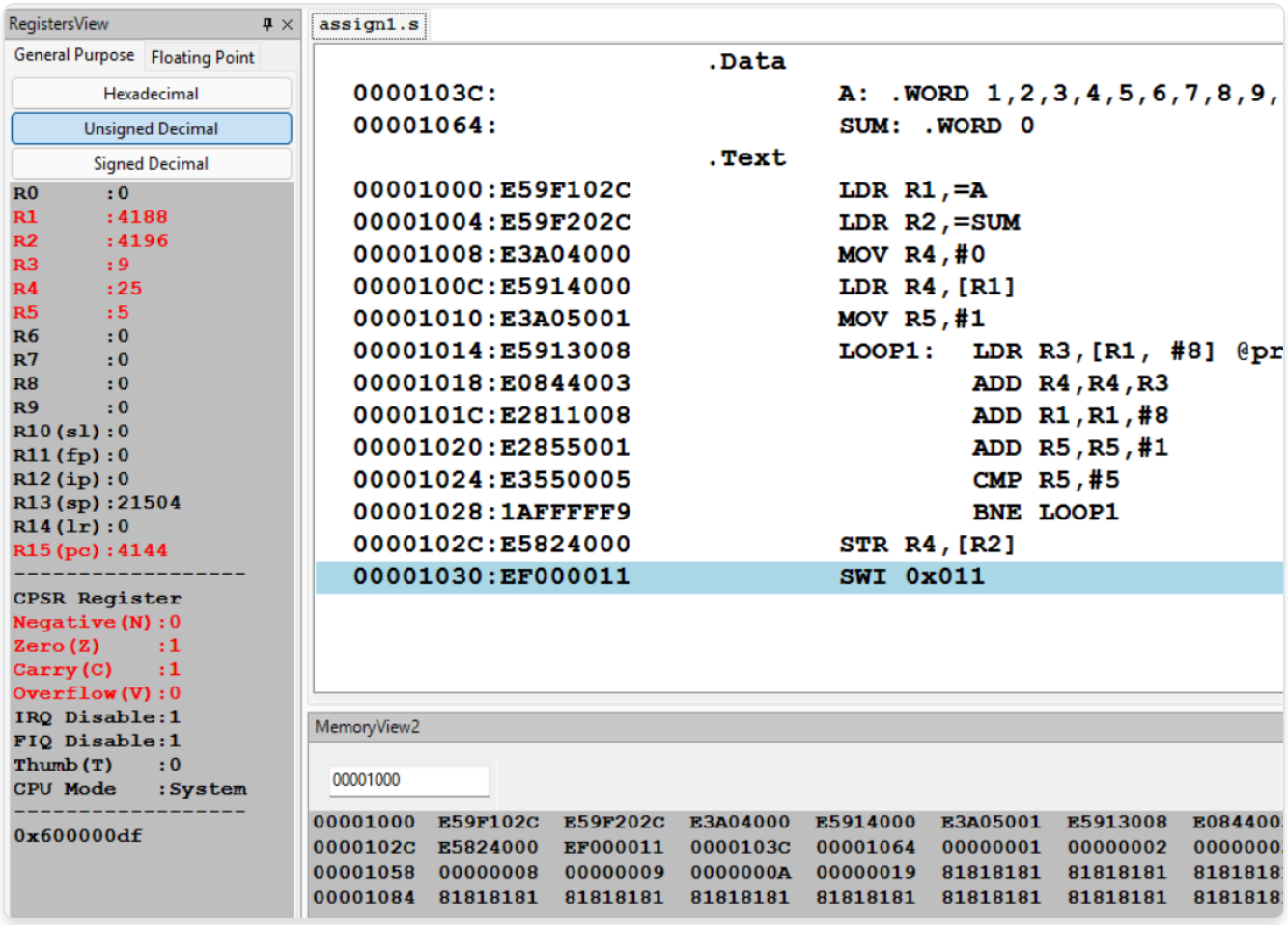

RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4164
R1 :1
R2 :4200
R3 :1
R4 :9
R5 :2
R6 :4236
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4124
CPSR Register
Negative(N):0
Zero(Z) :0
Carry(C) :0
Overflow(V):0
IRQ Disable:1
FIQ Disable:1
Thumb(T) :0
CPU Mode :System
0x000000df
matrixaddition.s
.TEXT
00001000:E59F0030 LDR R0,=A
00001004:E59F2030 LDR R2,=B
00001008:E59F6030 LDR R6,=C
0000100C:E3A04009 MOV R4,#9
00001010: LOOP:
00001010:E5923000 LDR R3,[R2]
00001014:E5901000 LDR R1,[R0]
00001018:E0835001 ADD R5,R3,R1;
0000101C:E5865000 STR R5,[R6]
00001020:E2800004 ADD R0,R0,#4
00001024:E2822004 ADD R2,R2,#4
00001028:E2866004 ADD R6,R6,#4
0000102C:E2544001 SUBS R4,R4,#1
00001030:1AFFFFF6 BNE LOOP
00001034:EF000011 SWI 0X11
MemoryView2
00001000
00001000 E59F0030 E59F2030 E59F6030 E3A04009 E5923000 E5901000 E08
0000102C E2544001 1AFFFFF6 EF000011 00001044 00001068 0000108C 000
00001058 00000001 00000001 00000001 00000001 00000001 00000002 000
00001084 00000008 00000009 00000000 00000000 00000000 00000000 000


Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4200
R1 :1
R2 :4236
R3 :9
R4 :0
R5 :10
R6 :4272
R7 :0
R8 :0
R9 :0
R10(sl):0
R11(fp):0
R12(ip):0
R13(sp):21504
R14(lr):0
R15(pc):4148
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
.TEXT
00001000:E59F0030 LDR R0,=A
00001004:E59F2030 LDR R2,=B
00001008:E59F6030 LDR R6,=C
0000100C:E3A04009 MOV R4,#9
00001010: LOOP:
00001010:E5923000 LDR R3,[R2]
00001014:E5901000 LDR R1,[R0]
00001018:E0835001 ADD R5,R3,R1;
0000101C:E5865000 STR R5,[R6]
00001020:E2800004 ADD R0,R0,#4
00001024:E2822004 ADD R2,R2,#4
00001028:E2866004 ADD R6,R6,#4
0000102C:E2544001 SUBS R4,R4,#1
00001030:1AFFFFF6 BNE LOOP
00001034:EF000011 SWI 0X11
MemoryView2
00001000

assignment 1: Write a program in ARM7TDMI-ISA to find the sum of N data items at alternate [odd or even positions] locations in the memory. Store the result in the memory location.

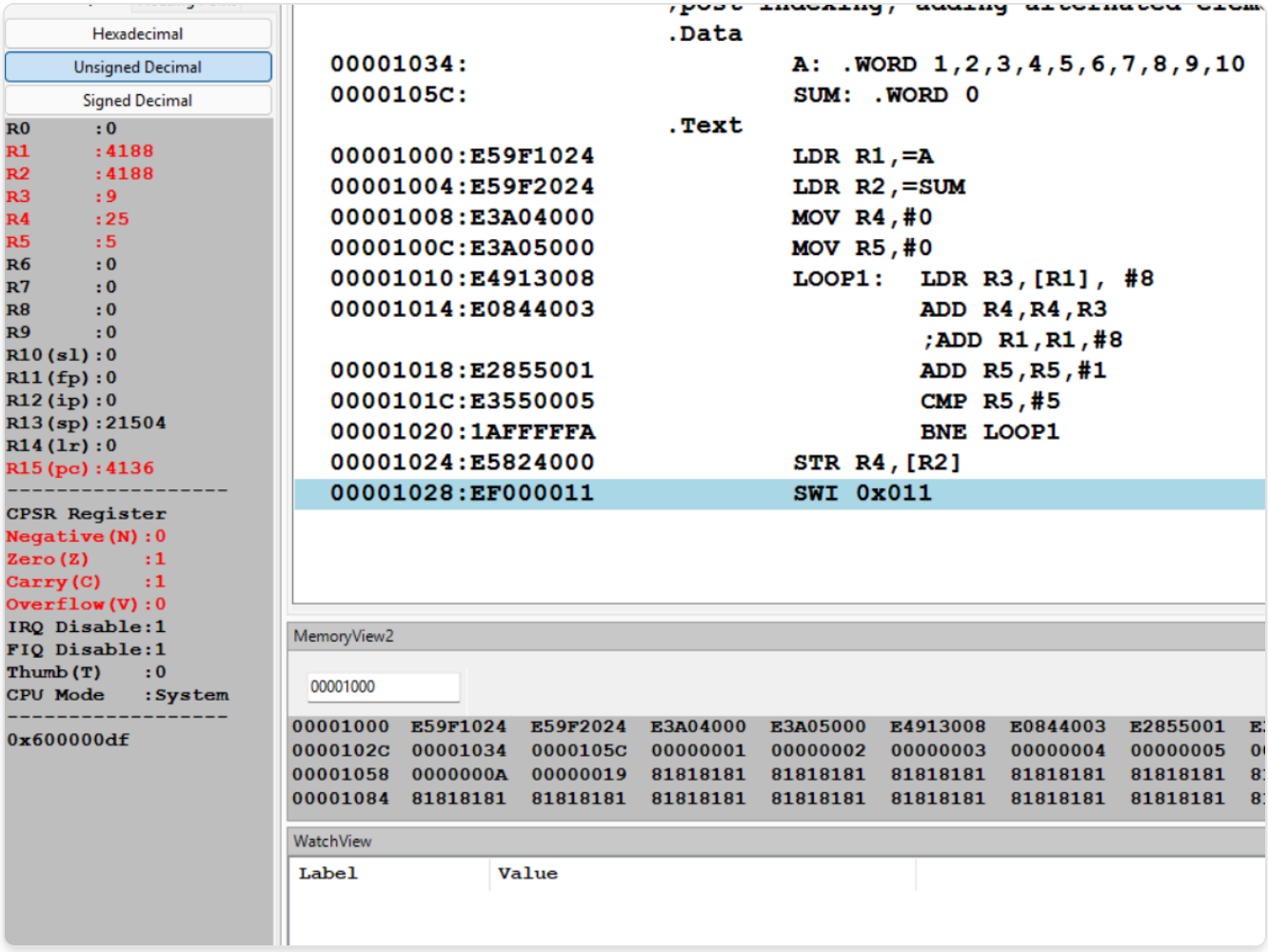
a. Use Pre-indexing addressing mode.

```
.Data
        A: .WORD 1,2,3,4,5,6,7,8,9,10
        SUM: .WORD 0
.Text
        LDR R1,=A
        LDR R2,=SUM
        MOV R4,#0
        LDR R4,[R1]
        MOV R5,#1
        LOOP1:  LDR R3,[R1, #8] @pre-indexing without write back
                ADD R4,R4,R3
                ADD R1,R1,#8
                ADD R5,R5,#1
                CMP R5,#5
                BNE LOOP1
        STR R4,[R2]
        SWI 0x011
~
```


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0
R1 :4188
R2 :4196
R3 :9
R4 :25
R5 :5
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):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
assign1.s
.Data
0000103C: A: .WORD 1,2,3,4,5,6,7,8,9,
00001064: SUM: .WORD 0
.Text
00001000:E59F102C LDR R1,=A
00001004:E59F202C LDR R2,=SUM
00001008:E3A04000 MOV R4,#0
0000100C:E5914000 LDR R4,[R1]
00001010:E3A05001 MOV R5,#1
00001014:E5913008 LOOP1: LDR R3,[R1, #8] @pr
00001018:E0844003 ADD R4,R4,R3
0000101C:E2811008 ADD R1,R1,#8
00001020:E2855001 ADD R5,R5,#1
00001024:E3550005 CMP R5,#5
00001028:1AFFFFF9 BNE LOOP1
0000102C:E5824000 STR R4,[R2]
00001030:EF000011 SWI 0x011
MemoryView2
00001000
00001000 E59F102C E59F202C E3A04000 E5914000 E3A05001 E5913008 E084400
0000102C E5824000 EF000011 0000103C 00001064 00000001 00000002 0000000
00001058 00000008 00000009 0000000A 00000019 81818181 81818181 8181818
00001084 81818181 81818181 81818181 81818181 81818181 81818181 8181818

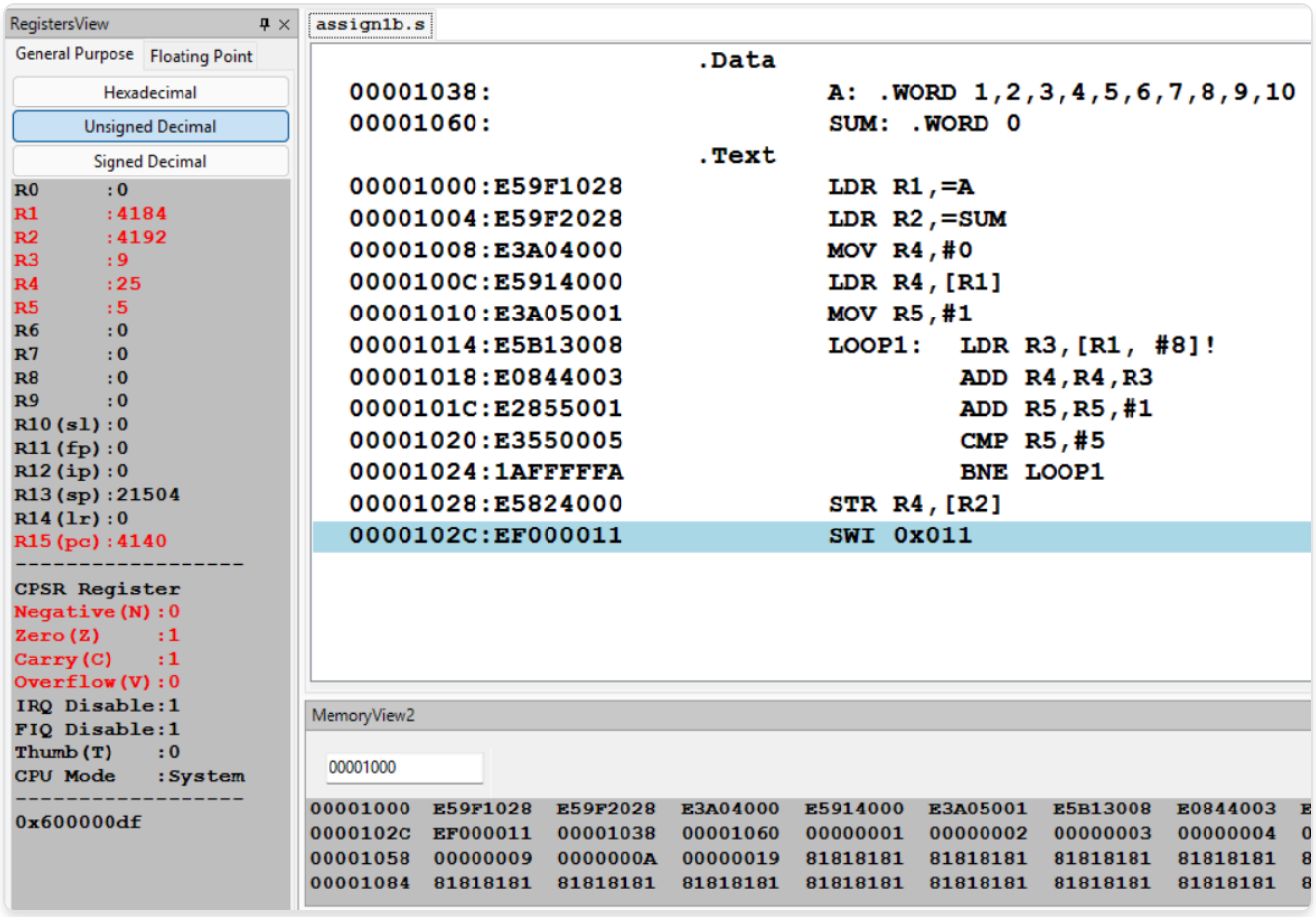
b.Use post-indexing addressing mode

```
;post indexing, adding alternate elements
.Data
        A: .WORD 1,2,3,4,5,6,7,8,9,10
        SUM: .WORD 0
.Text
        LDR R1,=A
        LDR R2,=SUM
        MOV R4,#0
        MOV R5,#0
        LOOP1:  LDR R3,[R1], #8
                ADD R4,R4,R3
                ;ADD R1,R1,#8
                ADD R5,R5,#1
                CMP R5,#5
                BNE LOOP1
        STR R4,[R2]
        SWI 0x011
```


Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0
R1 :4188
R2 :4188
R3 :9
R4 :25
R5 :5
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):4136
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
00001034: A: .WORD 1,2,3,4,5,6,7,8,9,10
0000105C: SUM: .WORD 0
.Text
00001000:E59F1024 LDR R1,=A
00001004:E59F2024 LDR R2,=SUM
00001008:E3A04000 MOV R4,#0
0000100C:E3A05000 MOV R5,#0
00001010:E4913008 LOOP1: LDR R3,[R1], #8
00001014:E0844003 ADD R4,R4,R3
;ADD R1,R1,#8
00001018:E2855001 ADD R5,R5,#1
0000101C:E3550005 CMP R5,#5
00001020:1AFFFFFA BNE LOOP1
00001024:E5824000 STR R4,[R2]
00001028:EF000011 SWI 0x011
MemoryView2
00001000
00001000 E59F1024 E59F2024 E3A04000 E3A05000 E4913008 E0844003 E2855001
0000102C 00001034 0000105C 00000001 00000002 00000003 00000004 00000005
00001058 0000000A 00000019 81818181 81818181 81818181 81818181 81818181
00001084 81818181 81818181 81818181 81818181 81818181 81818181 81818181
WatchView
Label
Value

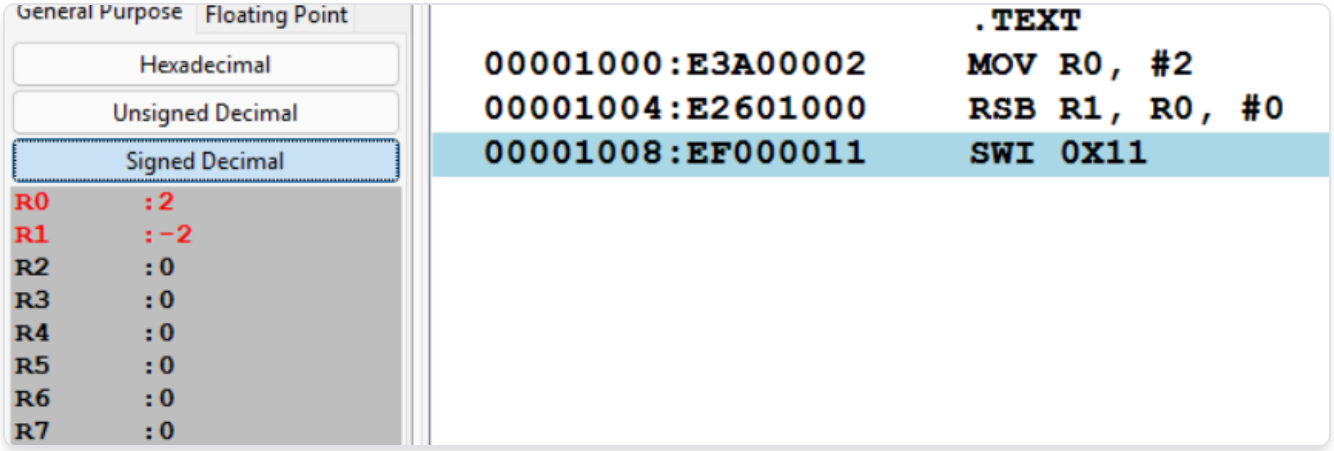
c. Use Auto-indexing addressing mode

```
.Data
        A: .WORD 1,2,3,4,5,6,7,8,9,10
        SUM: .WORD 0
.Text
        LDR R1,=A
        LDR R2,=SUM
        MOV R4,#0
        LDR R4,[R1]
        MOV R5,#1
        LOOP1:  LDR R3,[R1, #8]!
                ADD R4,R4,R3
                ADD R5,R5,#1
                CMP R5,#5
                BNE LOOP1
        STR R4,[R2]
        SWI 0x011
```



assignment 2: Write an ALP to perform 2's complement, Only using MOV and RSB instruction.

```
.TEXT
MOV R0, #2
RSB R1, R0, #0
SWI 0X11
```



- **DISCLAIMER:**

  The programs and output submitted is duly written, verified and executed by me.

  I have not copied from any of my peers nor from the external resource such as internet.

  If found plagiarized, I will abide with the disciplinary action of the University.