Microprocessor and Computer Architecture Laboratory

UE19CS256

4th Semester, Academic Year 2020-21

Date: 5/2/2021

Name: TUSHAR Y S	SRN: PES1UG19CS545	Section I

Week#____3_____ Program Number: __1___

Write an ALP to add two 64 bit numbers loaded from memory and store the result in memory.

ARM Assembly Code:

```
.text
        LDR R0,=A
        LDR R1,=B
        LDR R2,[R0],#4
        LDR R3,[R0]
        LDR R4,[R1],#4
        LDR R5,[R1]
        ADDS R7,R3,R5
        ADC R6,R2,R4
        LDR R8,=C
        STR R6,[R8],#4
```

```
STR R7,[R8]

LDR R9,=C

LDR R10,[R9,#4]

LDR R11,[R9]

SWI 0x11
```

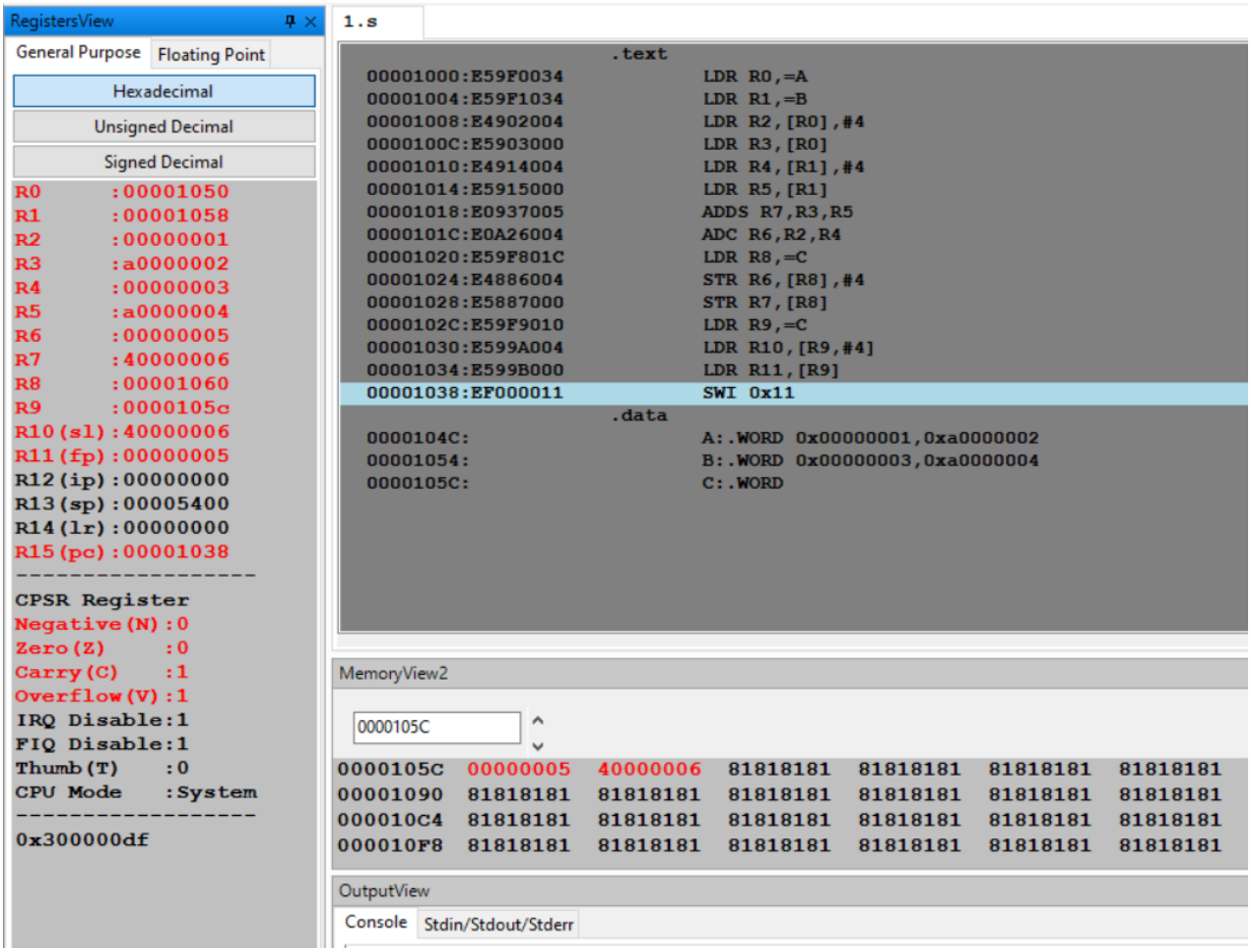
```
.data

A:.WORD 0x00000001,0xa0000002

B:.WORD 0x00000003,0xa0000004

C:.WORD
```

Output:



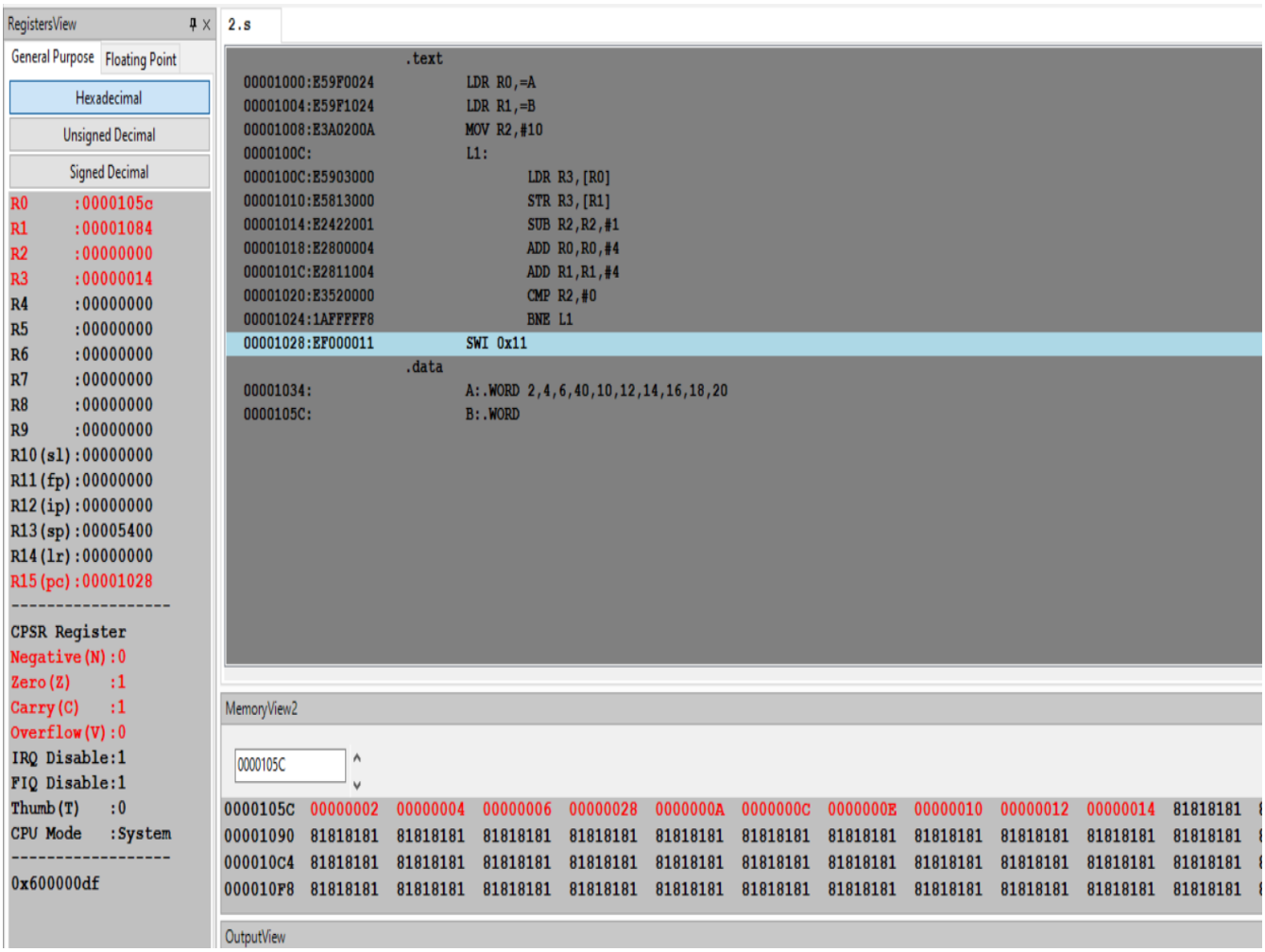
Week#____3_____ Program Number: __2___

Write an ALP to copy n numbers from Memory Location A to Memory Location B.

ARM Assembly Code:

```
.text
        LDR R0,=A
        LDR R1,=B
        MOV R2,#10
        L1:
                LDR R3,[R0]
                STR R3,[R1]
                SUB R2,R2,#1
                ADD R0,R0,#4
                ADD R1,R1,#4
                CMP R2,#0
                BNE L1
        SWI 0x11
.data
        A:.WORD 2,4,6,40,10,12,14,16,18,20
        B:.WORD
```

Output:



Week#____3_____ Program Number: __3___

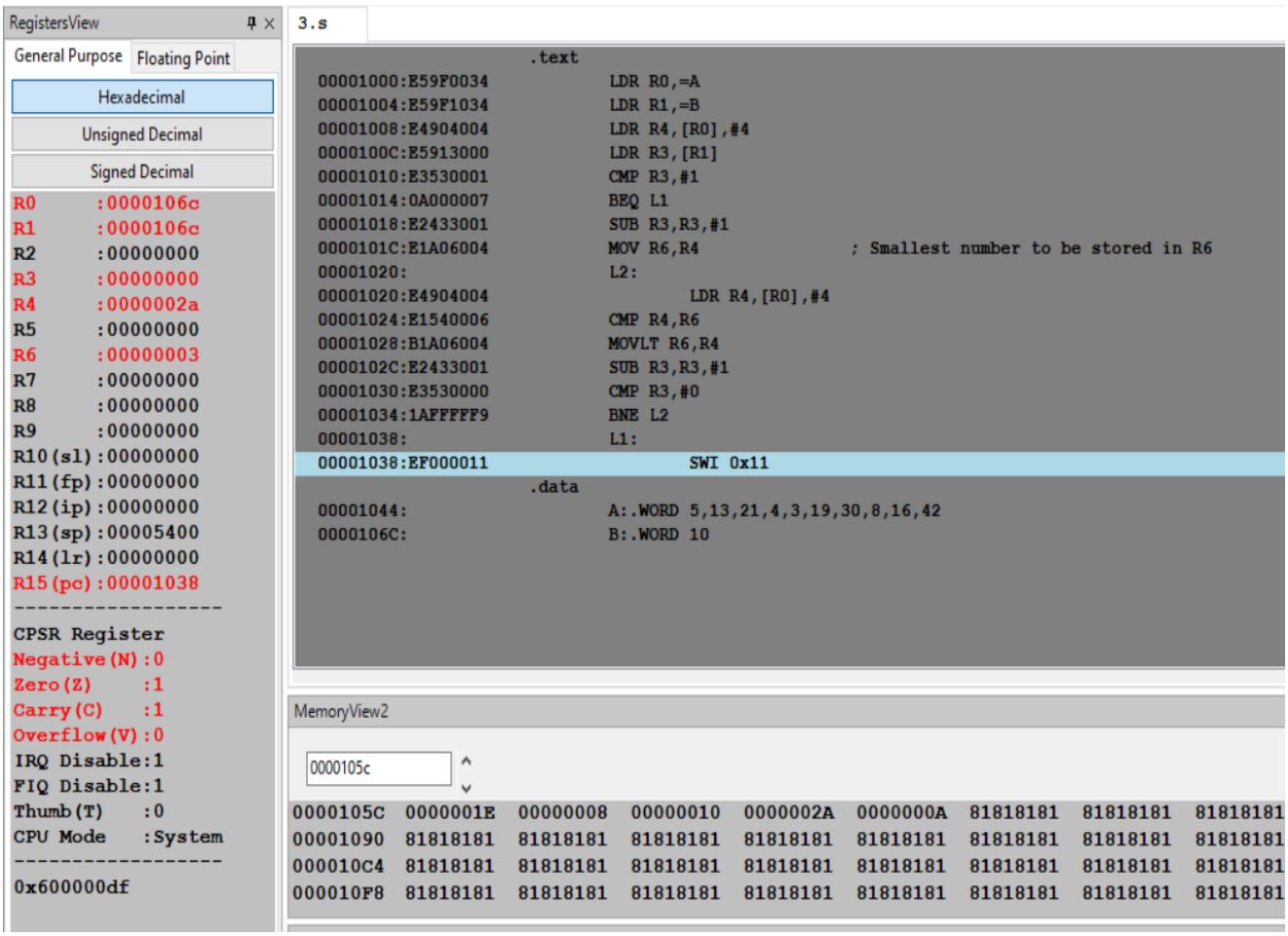
Write an ALP to find smallest number in an array of n – 32 bit numbers.

ARM Assembly Code:

```
.text
        LDR R0,=A
        LDR R1,=B
        LDR R4,[R0],#4
        LDR R3,[R1]
```

```
        CMP R3,#1
        BEQ L1
        SUB R3,R3,#1
        MOV R6,R4               ; Smallest number to be stored in R6
        L2:
                LDR R4,[R0],#4
        CMP R4,R6
        MOVLT R6,R4
        SUB R3,R3,#1
        CMP R3,#0
        BNE L2
        L1:
                SWI 0x11
.data
        A:.WORD 5,13,21,4,3,19,30,8,16,42
        B:.WORD 10
```

Output:



Week#____3_____ Program Number: __4a___

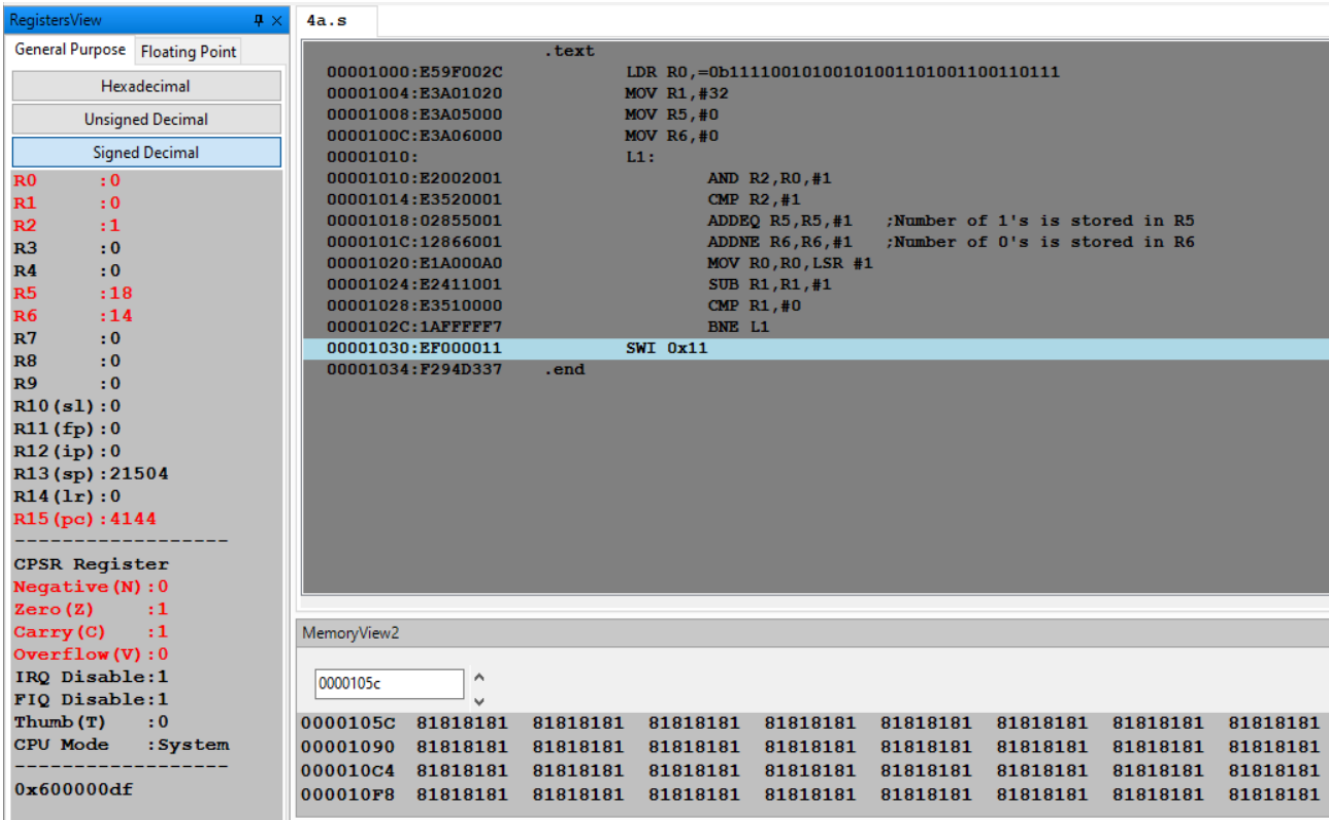
Write an ALP to count the number of 1's and 0's in a given 32 bit number.

ARM Assembly Code:

```
.text
        LDR R0,=0b11110010100101001101001100110111
        MOV R1,#32
        MOV R5,#0
        MOV R6,#0
```

```
L1:
        AND R2,R0,#1
        CMP R2,#1
        ADDEQ R5,R5,#1   ;Number of 1's is stored in R5
        ADDNE R6,R6,#1   ;Number of 0's is stored in R6
        MOV R0,R0,LSR #1
        SUB R1,R1,#1
        CMP R1,#0
        BNE L1
    SWI 0x11
.end
```

Output:



Week#____3_____ Program Number: __4b___

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

ARM Assembly Code:

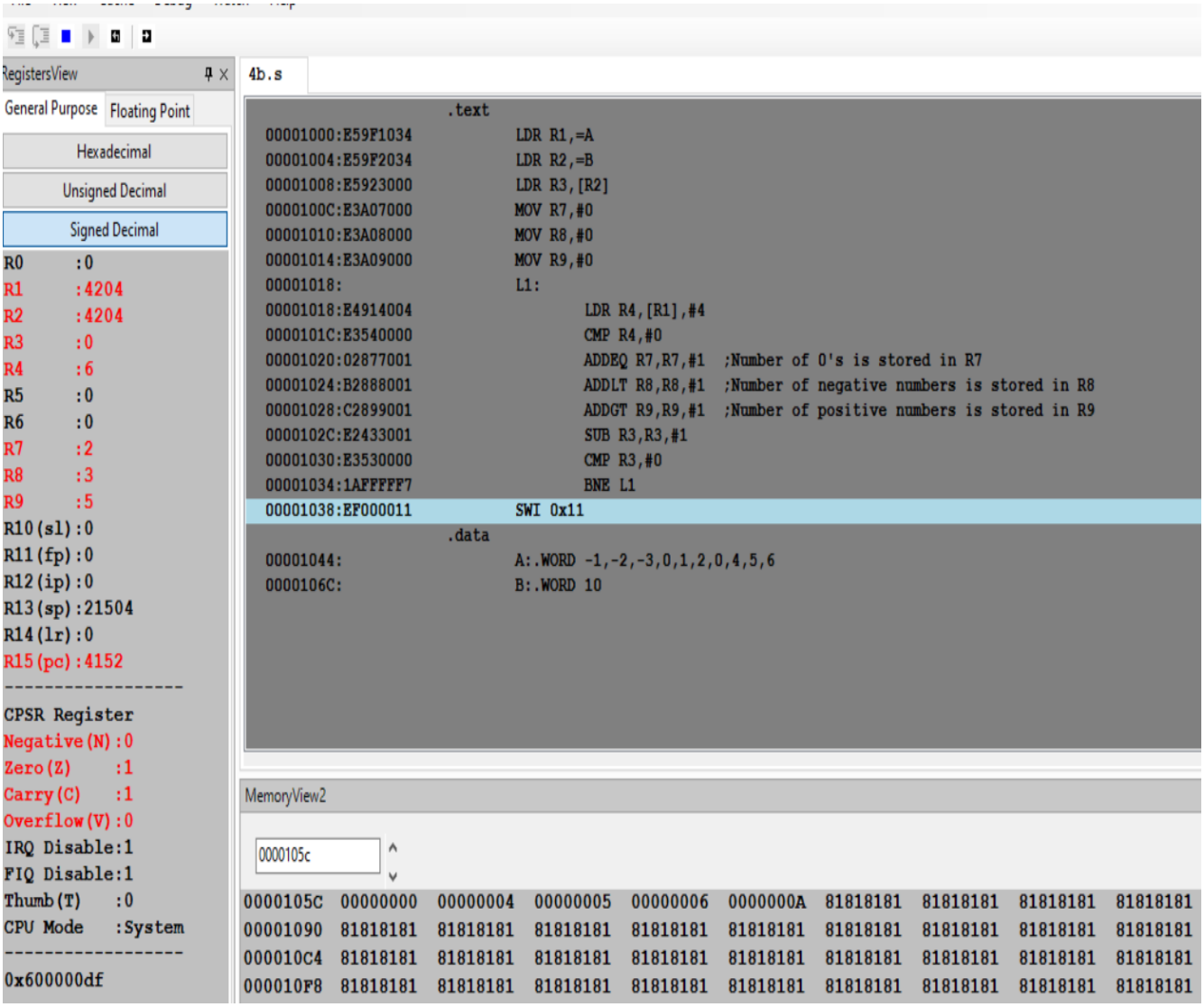
```
.text
        LDR R1,=A
        LDR R2,=B
        LDR R3,[R2]
        MOV R7,#0
        MOV R8,#0
        MOV R9,#0
        L1:
                LDR R4,[R1],#4
                CMP R4,#0
                ADDEQ R7,R7,#1  ;Number of 0's is stored in R7
                ADDLT R8,R8,#1  ;Number of negative numbers is stored in
R8
                ADDGT R9,R9,#1  ;Number of positive numbers is stored in
R9
                SUB R3,R3,#1
                CMP R3,#0
                BNE L1
        SWI 0x11
```

.data

A:.WORD -1,-2,-3,0,1,2,0,4,5,6

B:.WORD 10

Output:



Week#____3_____ Program Number: __5___

Write an ALP to check whether a given number is present in array using Linear Search (Without SWI 0x02), if found move +1 to R6 and key position to R7 else move -1 to R6 (if number not found).

Case 1(Element found):

ARM Assembly Code:

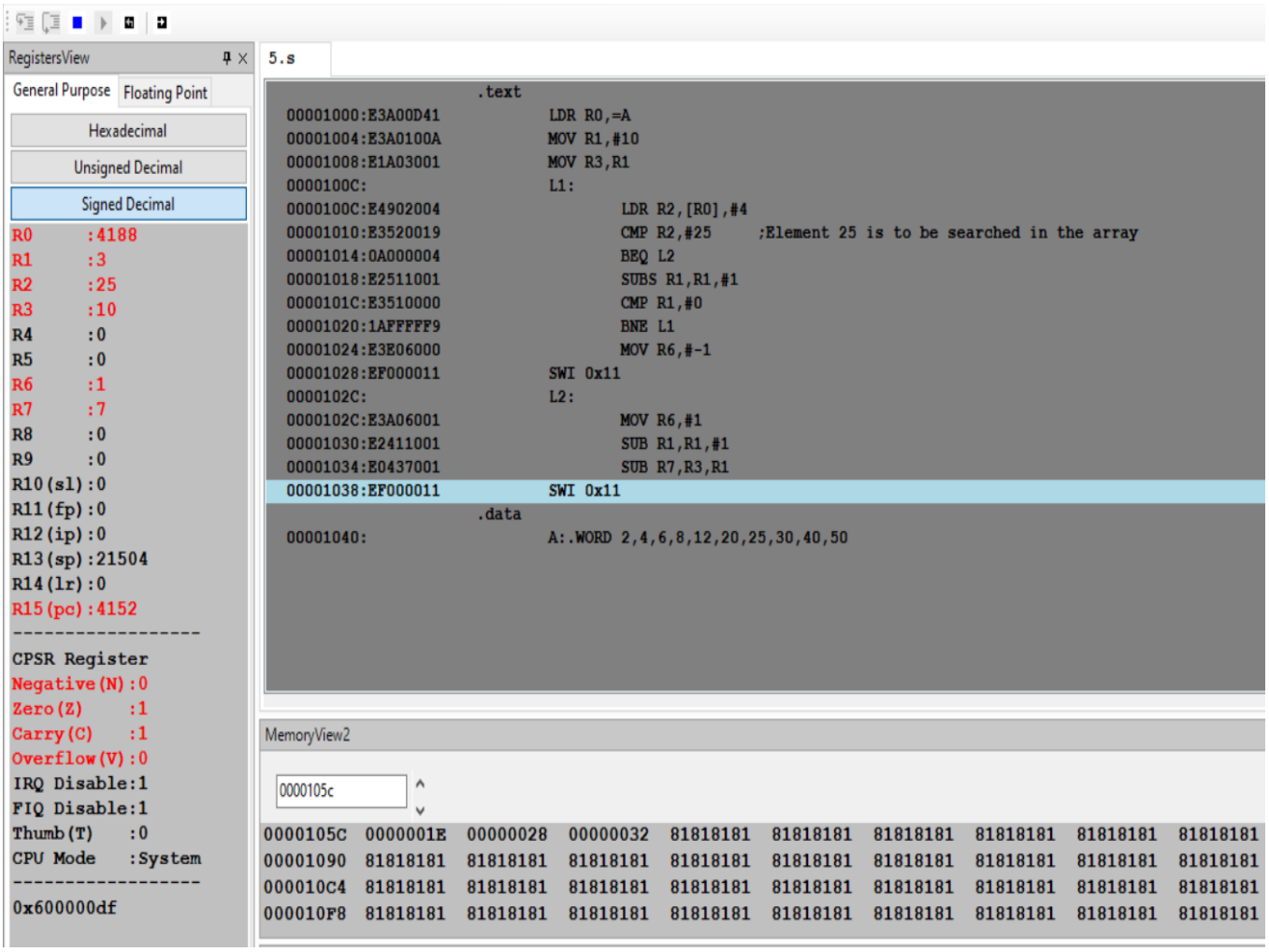
```
.text
        LDR R0,=A
        MOV R1,#10
        MOV R3,R1
        L1:
                LDR R2,[R0],#4
                CMP R2,#25     ;Element 25 is to be searched in the array
                BEQ L2
                SUBS R1,R1,#1
                CMP R1,#0
                BNE L1
                MOV R6,#-1
        SWI 0x11
        L2:
                MOV R6,#1
                SUB R1,R1,#1
                SUB R7,R3,R1
```

SWI 0x11

.data

A:.WORD 2,4,6,8,12,20,25,30,40,50

Output:



Case 2 (Element NOT found):

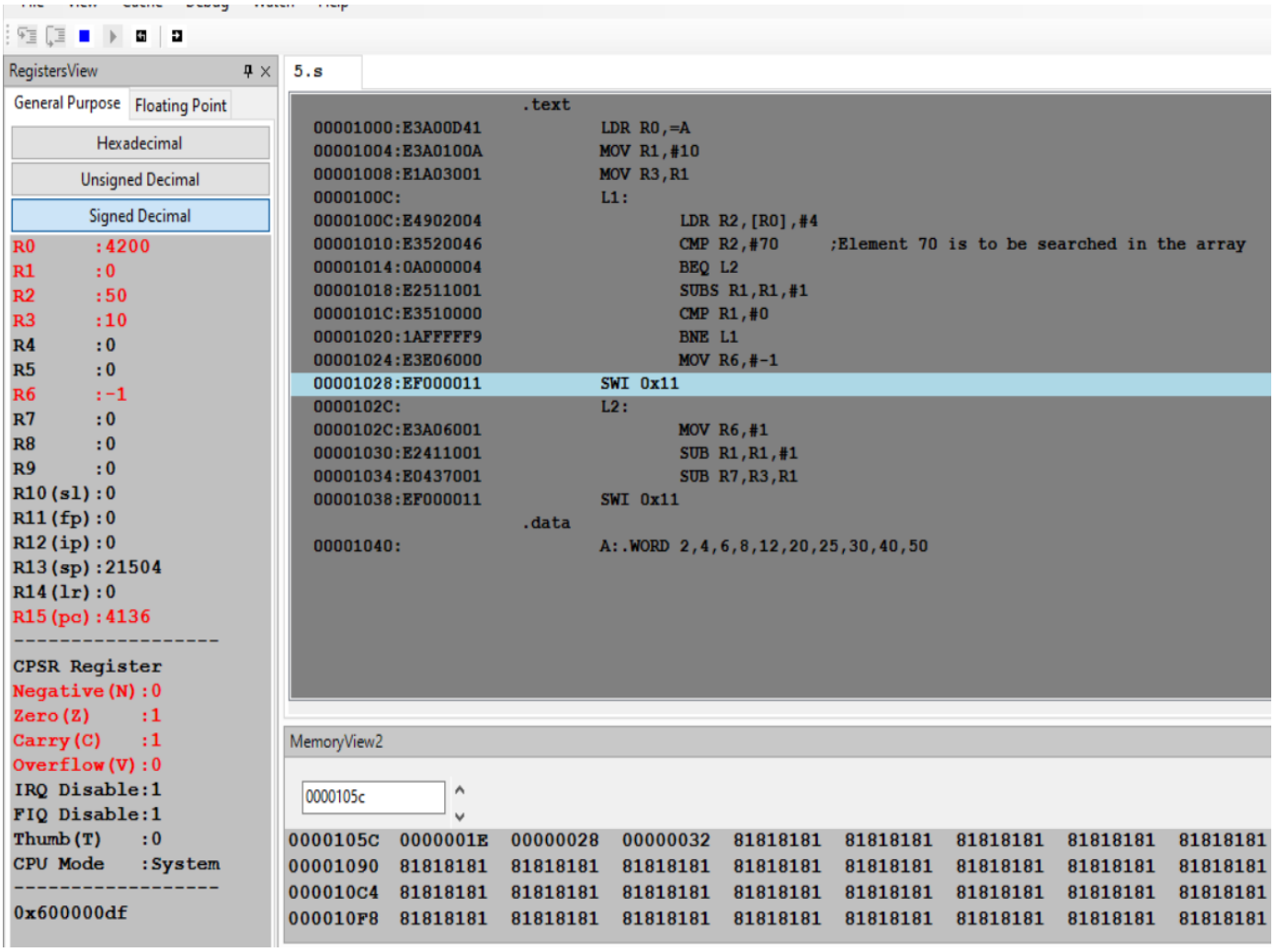
ARM Assembly Code:

.text

LDR R0,=A

```
        MOV R1,#10
        MOV R3,R1
        L1:
                LDR R2,[R0],#4
                CMP R2,#70     ;Element 70 is to be searched in the array
                BEQ L2
                SUBS R1,R1,#1
                CMP R1,#0
                BNE L1
                MOV R6,#-1
        SWI 0x11
        L2:
                MOV R6,#1
                SUB R1,R1,#1
                SUB R7,R3,R1
        SWI 0x11
.data
        A:.WORD 2,4,6,8,12,20,25,30,40,50
```

Output:


RegistersView
General Purpose
Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :4200
R1 :0
R2 :50
R3 :10
R4 :0
R5 :0
R6 :-1
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
5.s
.text
00001000:E3A00D41 LDR R0,=A
00001004:E3A0100A MOV R1,#10
00001008:E1A03001 MOV R3,R1
0000100C: L1:
0000100C:E4902004 LDR R2,[R0],#4
00001010:E3520046 CMP R2,#70 ;Element 70 is to be searched in the array
00001014:0A000004 BEQ L2
00001018:E2511001 SUBS R1,R1,#1
0000101C:E3510000 CMP R1,#0
00001020:1AFFFFF9 BNE L1
00001024:E3E06000 MOV R6,#-1
00001028:EF000011 SWI 0x11
0000102C: L2:
0000102C:E3A06001 MOV R6,#1
00001030:E2411001 SUB R1,R1,#1
00001034:E0437001 SUB R7,R3,R1
00001038:EF000011 SWI 0x11
.data
00001040: A:.WORD 2,4,6,8,12,20,25,30,40,50
MemoryView2
0000105c
0000105C 0000001E 00000028 00000032 81818181 81818181 81818181 81818181 81818181
00001090 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010C4 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010F8 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

Week#____3_____ Program Number: __6___

Write an ALP to generate Fibonacci Series and store them in an array.

ARM Assembly Code:

```
.text
        LDR R1,=A
        MOV R2,#0
        STR R2,[R1]
        ADD R1,R1,#4
        MOV R3,#1
        STR R3,[R1]
        MOV R5,#8  ; 8 fibonacci numbers will be stored after 0 and 1(So,total=10) in the array
        L1:
                ADD R4,R2,R3
                ADD R1,R1,#4
                STR R4,[R1]
                MOV R2,R3
                MOV R3,R4
                SUBS R5,R5,#1
                MOV R4,#0
                BNE L1
        SWI 0x11
```

.data

A:.WORD

Output: