Computer Architecture and Organization

UCS507

LAB ASSIGNMENT

Computer Science Engineering Department

THAPAR INSTITUTE OF ENGINEERING

AND TECHNOLOGY, PATIALA

Submitted By :

Yashwant

101803318

COE15

Submitted To :

Ms. Swati Sharma

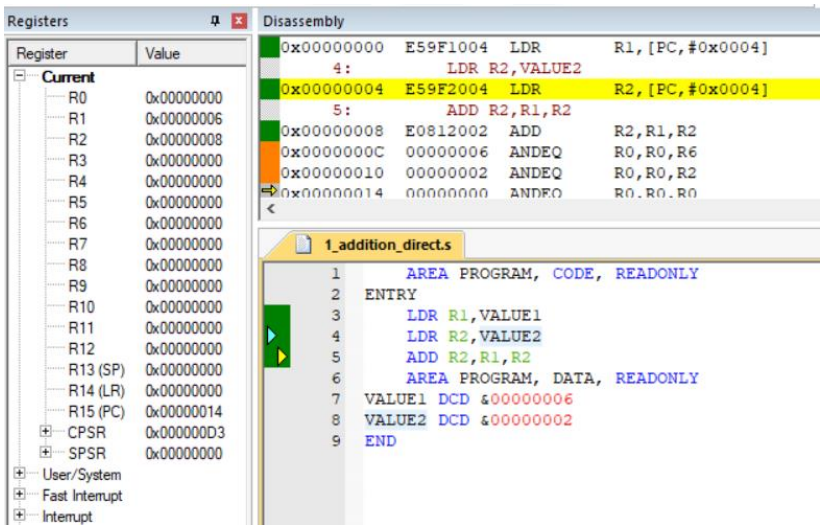
INDEX

1. Write a program in ARM assembly language to add and subtract two 32-bit numbers using: i) Direct addressing mode=2 ii) Indirect addressing mode=2 iii) Barrel shifter=4

i) Direct Addressing

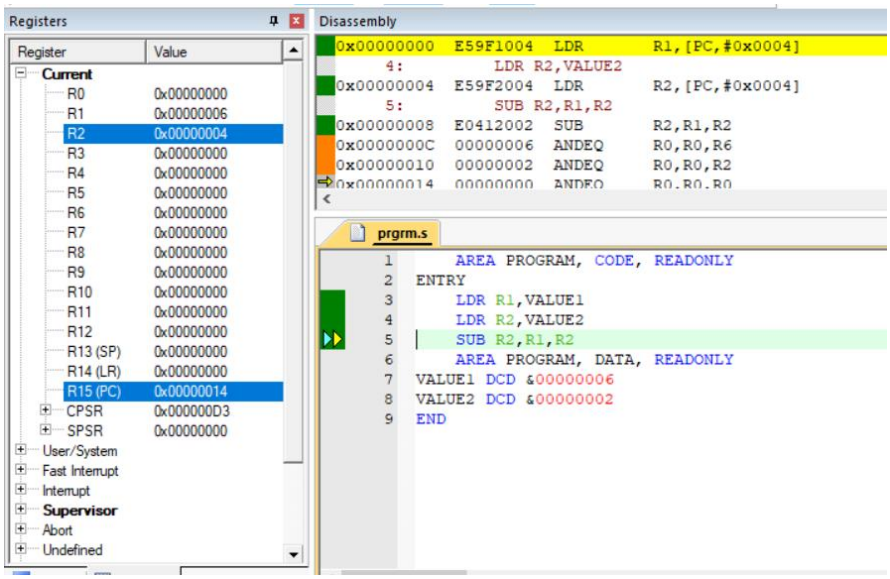
Addition :

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2
        ADD R2,R1,R2
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000006
VALUE2 DCD &00000002
END
```



Subtraction :

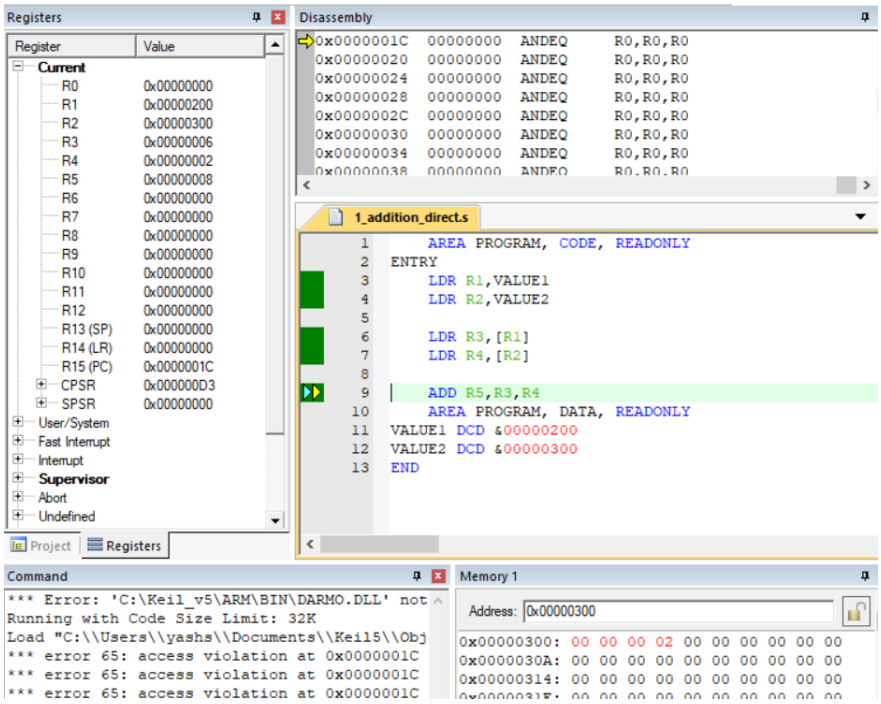
```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2
        SUB R2,R1,R2
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000006
VALUE2 DCD &00000002
END
```



ii) Indirect Addressing

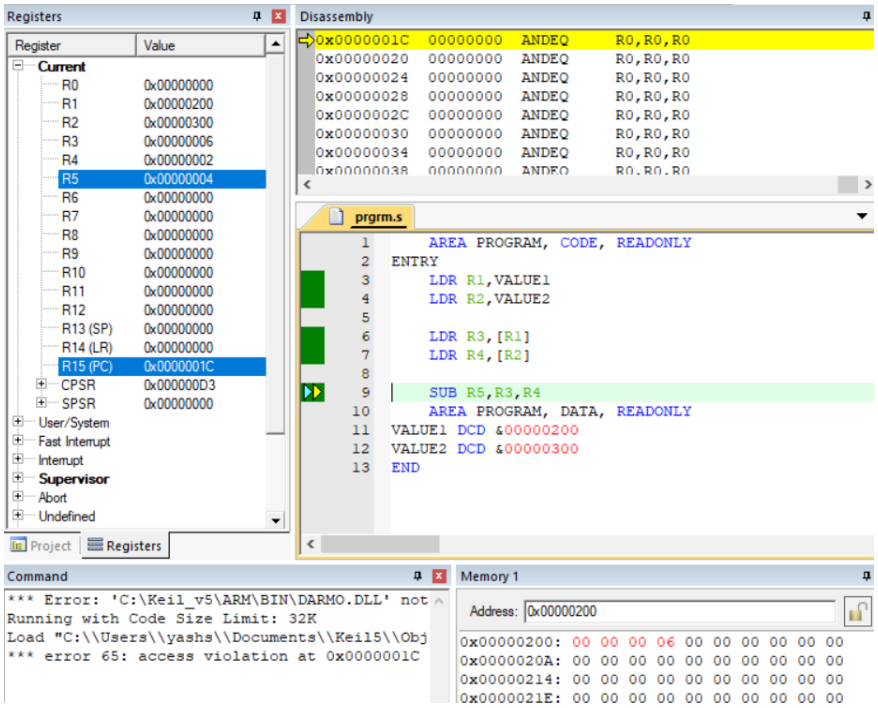
Addition :

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2

        LDR R3,[R1]
        LDR R4,[R2]

        ADD R5,R3,R4
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000200
VALUE2 DCD &00000300
END
```



Subtraction :

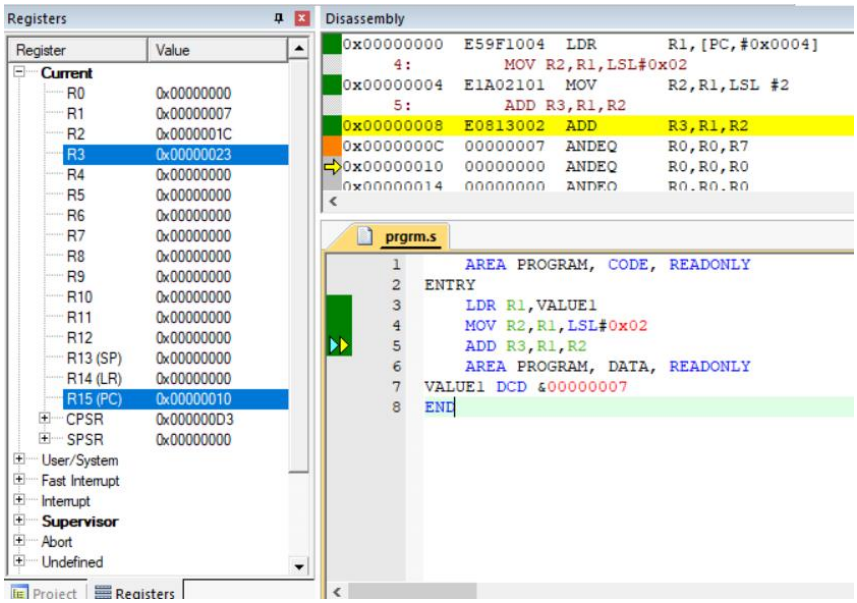
```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2

        LDR R3,[R1]
        LDR R4,[R2]

        SUB R5,R3,R4
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000200
VALUE2 DCD &00000300
END
```



iii) Barrel Shifter

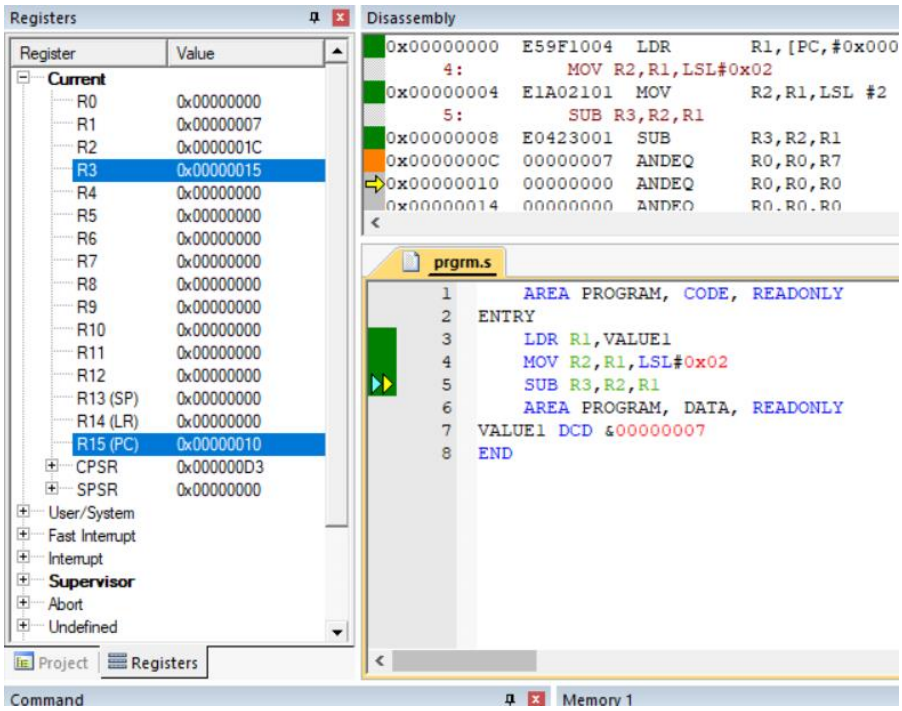
Addition :

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        MOV R2,R1,LSL#0x02
        ADD R3,R1,R2
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000001
END
```



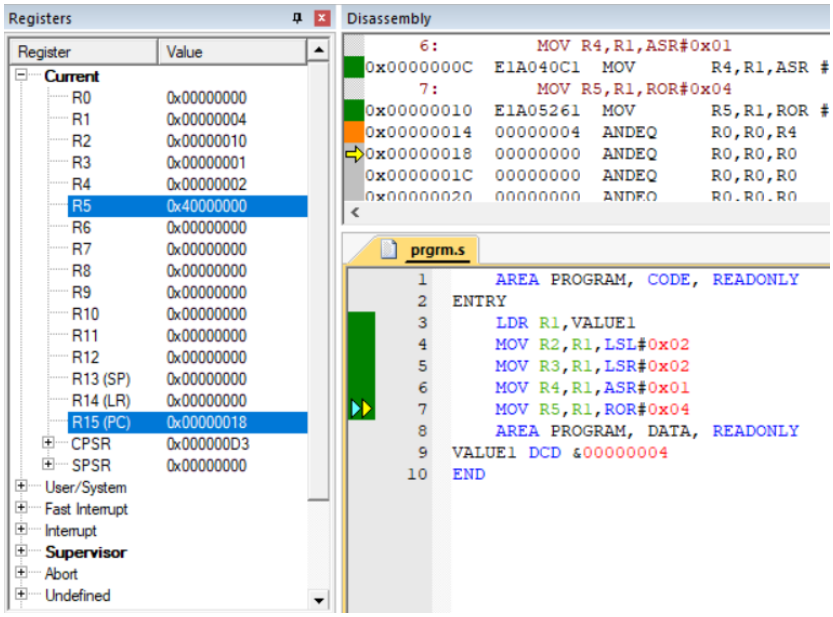
Subtraction :

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        MOV R2,R1,LSL#0x02
        SUB R3,R2,R1
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000007
END
```



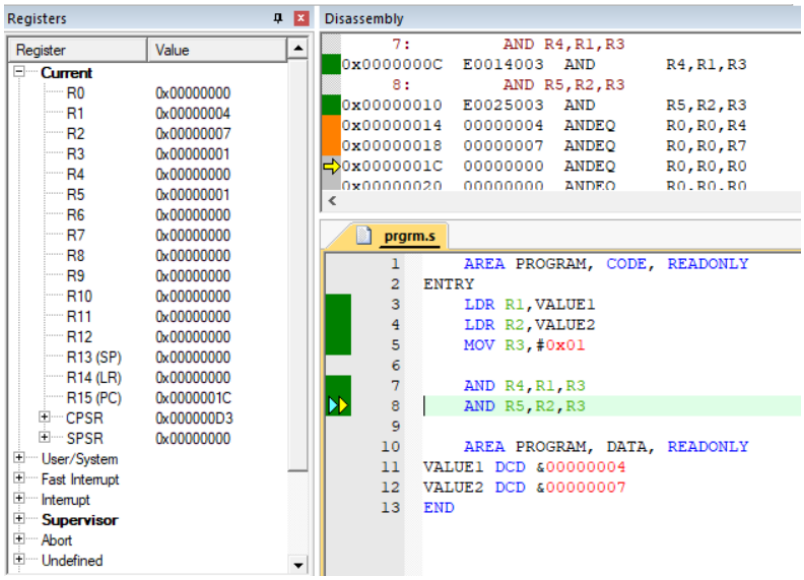
2. Write a program to perform left and right shift of a number. =4

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        MOV R2,R1,LSL#0x02
        MOV R3,R1,LSR#0x02
        MOV R4,R1,ASR#0x01
        MOV R5,R1,ROR#0x04
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000004
END
```



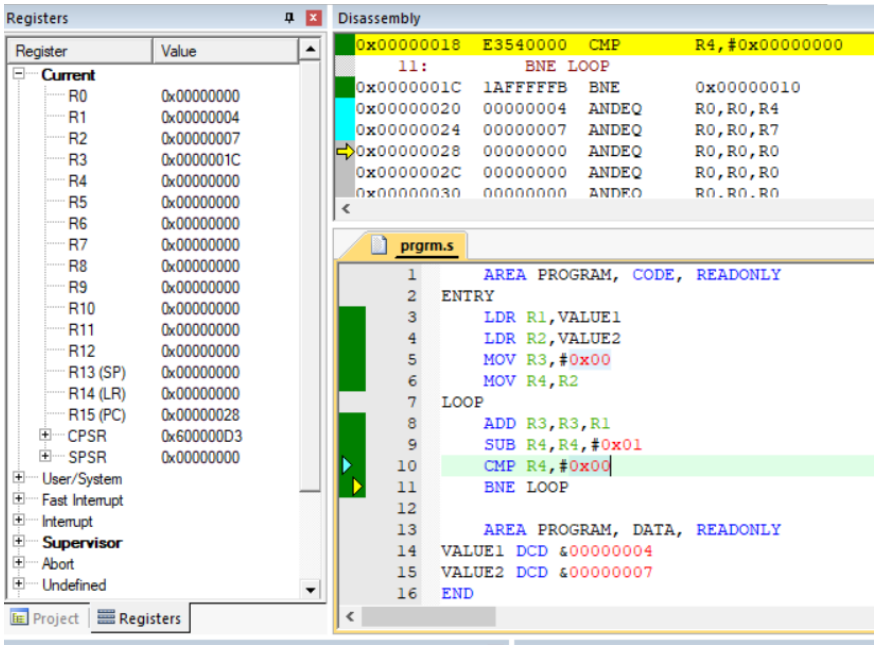
3. Write a program to find whether number is even or odd. =1

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2
        MOV R3,#0x01

        AND R4,R1,R3
        AND R5,R2,R3

        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000004
VALUE2 DCD &00000007
END
```



4. Write a program to perform Multiplication using addition. =1

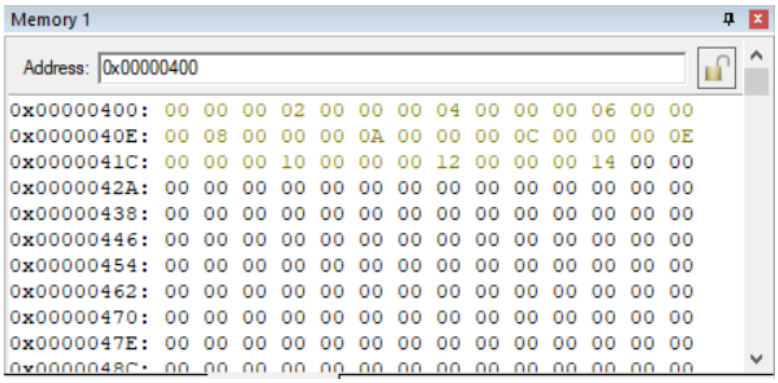
```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2
        MOV R3,#0x00
        MOV R4,R2
LOOP
        ADD R3,R3,R1
        SUB R4,R4,#0x01
        CMP R4,#0x00
        BNE LOOP

        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000004
VALUE2 DCD &00000007
END
```



5. Write a program to store Multiplication table of a number. =1

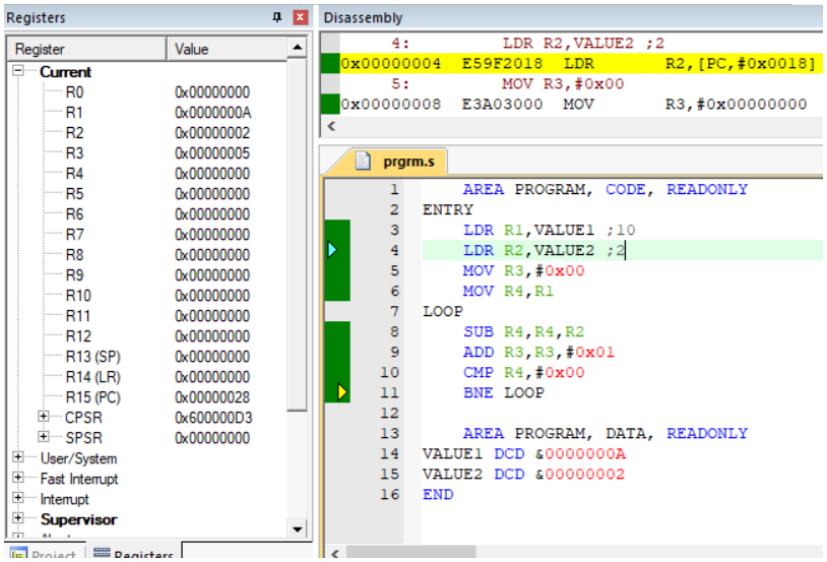
```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1
        LDR R2,VALUE2
        MOV R3,#0x0A
        MOV R4,R1
LOOP
        STR R1,[R2]
        ADD R1,R1,R4
        ADD R2,R2,#0x04
```

```
        SUB R3,R3,#0x01
        CMP R3,#0x00
        BNE LOOP

        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000002
VALUE2 DCD &00000400
END
```



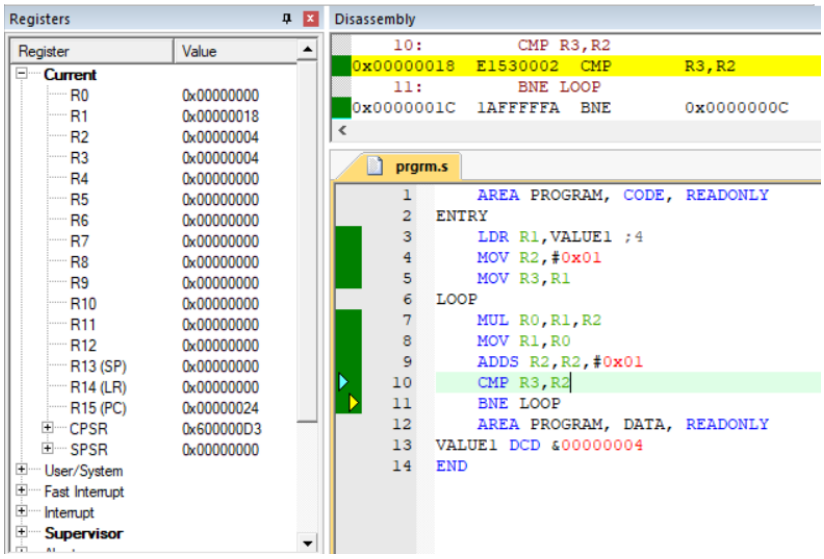
6. Write a program to perform Division using subtraction. =1

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1 ;10
        LDR R2,VALUE2 ;2
        MOV R3,#0x00
        MOV R4,R1
LOOP
        SUB R4,R4,R2
        ADD R3,R3,#0x01
        CMP R4,#0x00
        BNE LOOP

        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &0000000A
VALUE2 DCD &00000002
END
```



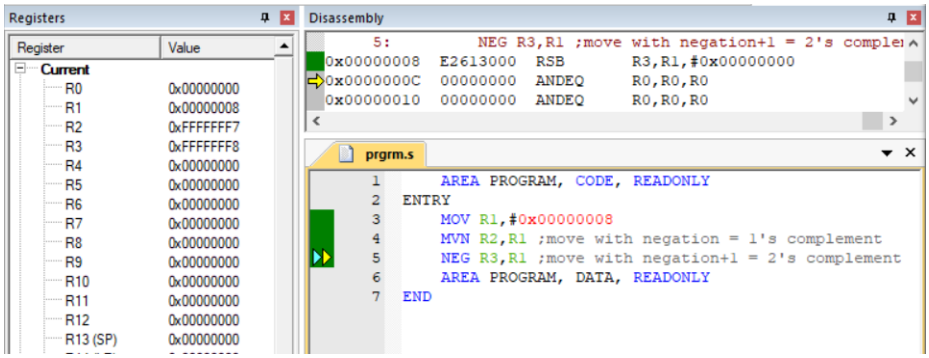
7. Write a program to find the factorial of a number.

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R1,VALUE1 ;4
        MOV R2,#0x01
        MOV R3,R1
LOOP
        MUL R0,R1,R2
        MOV R1,R0
        ADDS R2,R2,#0x01
        CMP R3,R2
        BNE LOOP
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &00000004
END
```



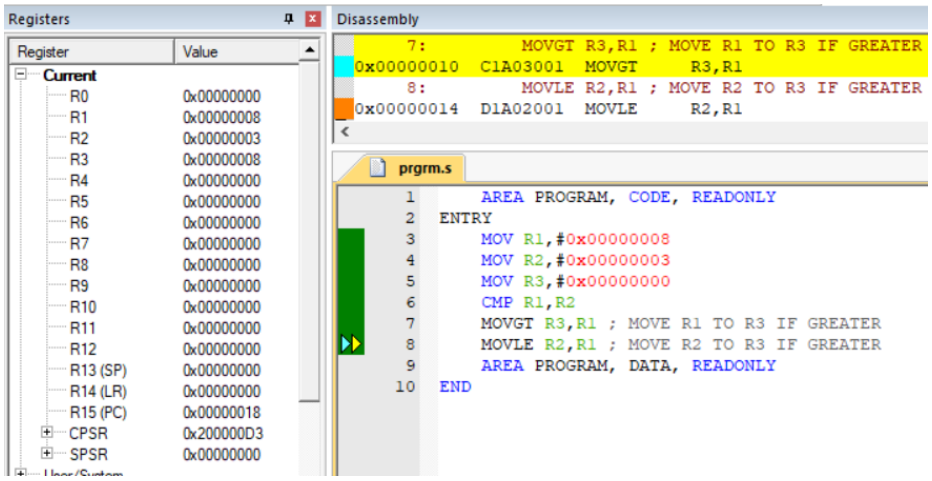
8. Write a program in ARM assembly language to find 1's and 2's compliment.

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        MOV R1,#0x00000008
        MVN R2,R1 ;move with negation = 1's complement
        NEG R3,R1 ;move with negation+1 = 2's complement
        AREA PROGRAM, DATA, READONLY
END
```



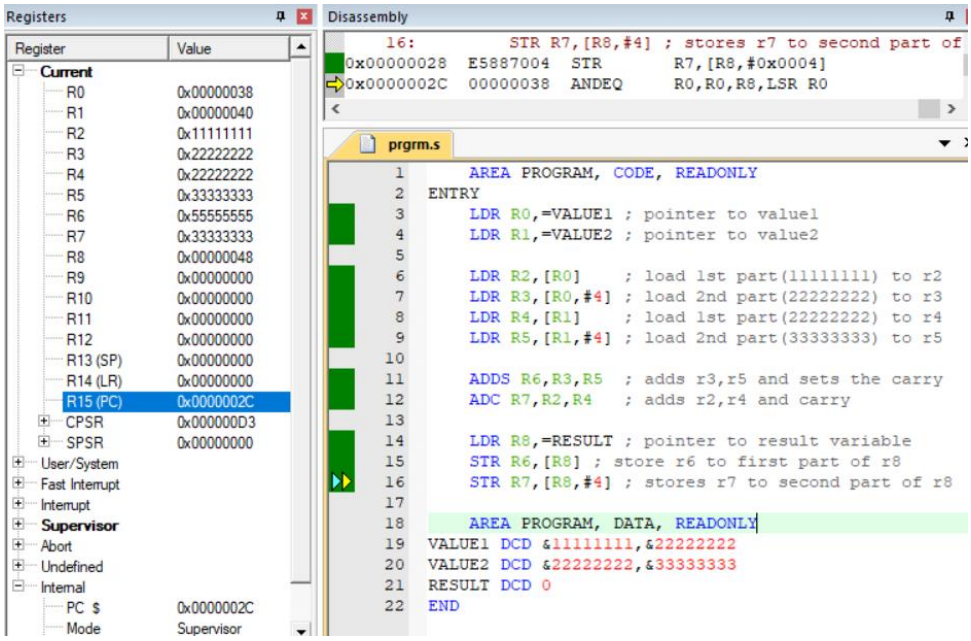
9. Write a program in ARM assembly language to find greater of two numbers.

```
        AREA PROGRAM, CODE, READONLY
ENTRY
        MOV R1,#0x00000008
        MOV R2,#0x00000003
        MOV R3,#0x00000000
        CMP R1,R2
        MOVGT R3,R1 ; MOVE R1 TO R3 IF GREATER
        MOVLE R2,R1 ; MOVE R2 TO R3 IF GREATER
        AREA PROGRAM, DATA, READONLY
END
```



10. Write a program to perform 64-bit addition of two 64-bit number.

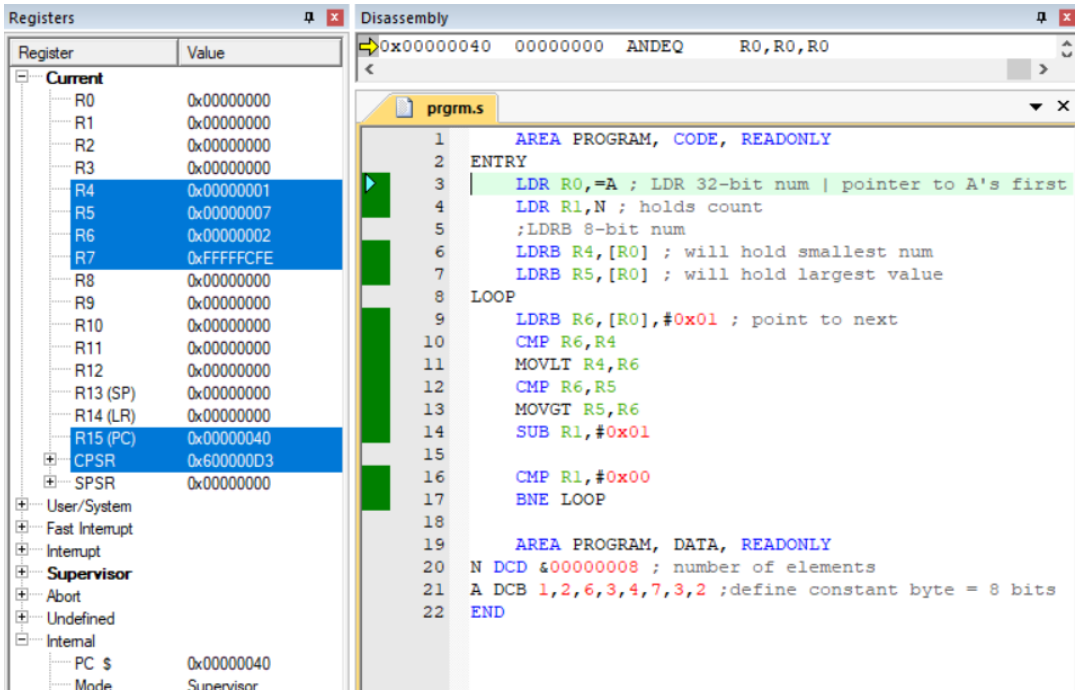
```
        AREA PROGRAM, CODE, READONLY
ENTRY
        LDR R0,=VALUE1 ; pointer to value1
        LDR R1,=VALUE2 ; pointer to value2

        LDR R2,[R0]    ; load 1st part(11111111) to r2
        LDR R3,[R0,#4] ; load 2nd part(22222222) to r3
        LDR R4,[R1]    ; load 1st part(22222222) to r4
        LDR R5,[R1,#4] ; load 2nd part(33333333) to r5

        ADDS R6,R3,R5  ; adds r3,r5 and sets the carry
        ADC R7,R2,R4   ; adds r2,r4 and carry

        LDR R8,=RESULT ; pointer to result variable
        STR R6,[R8] ; store r6 to first part of r8
        STR R7,[R8,#4] ; stores r7 to second part of r8

        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &11111111,&22222222
VALUE2 DCD &22222222,&33333333
RESULT DCD 0
END
```



11. Write a program to find the largest and smallest number in an array.

```
        AREA PROGRAM, CODE, READONLY
ENTRY
```

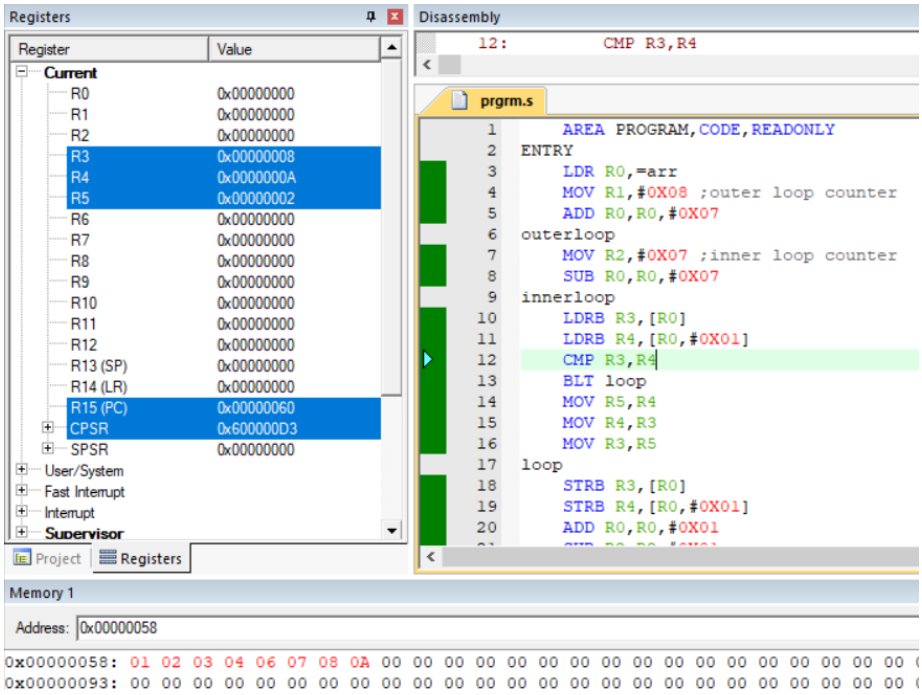
```
        LDR R0,=A ; LDR 32-bit num | pointer to A's first num
        LDR R1,N ; holds count
        ;LDRB 8-bit num
        LDRB R4,[R0] ; will hold smallest num
        LDRB R5,[R0] ; will hold largest value
LOOP
        LDRB R6,[R0],#0x01 ; point to next
        CMP R6,R4
        MOVLT R4,R6
        CMP R6,R5
        MOVGT R5,R6
        SUB R1,#0x01

        CMP R1,#0x00
        BNE LOOP

        AREA PROGRAM, DATA, READONLY
N DCD &00000008 ; number of elements
A DCB 1,2,6,3,4,7,3,2 ;define constant byte = 8 bits
END
```



12. Write a program to find the sorting in an array.

```
        AREA PROGRAM,CODE,READONLY
ENTRY
        LDR R0,=arr
        MOV R1,#0X08 ;outer loop counter
        ADD R0,R0,#0X07
outerloop
```

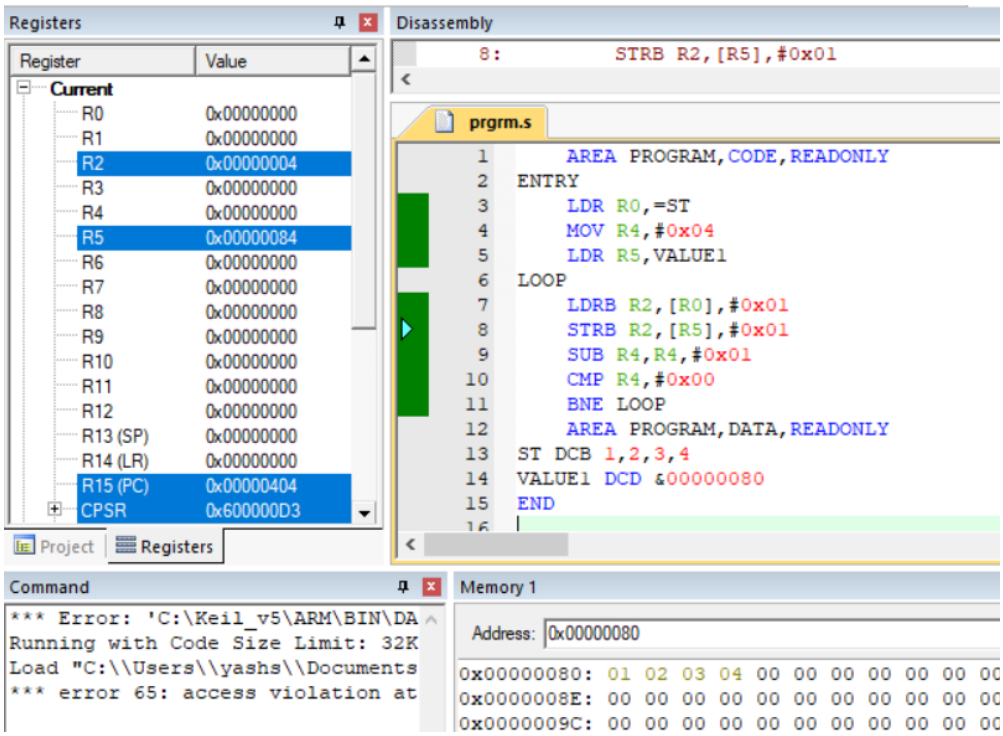
```
        MOV R2,#0X07 ;inner loop counter
        SUB R0,R0,#0X07
innerloop
        LDRB R3,[R0]
        LDRB R4,[R0,#0X01]
        CMP R3,R4
        BLT loop
        MOV R5,R4
        MOV R4,R3
        MOV R3,R5
loop
        STRB R3,[R0]
        STRB R4,[R0,#0X01]
        ADD R0,R0,#0X01
        SUB R2,R2,#0X01
        CMP R2,#0X00
        BNE innerloop
        SUB R1,R1,#0x01
        CMP R1,#0X00
        BNE outerloop
        AREA PROGRAM,DATA,READONLY
arr DCB 1,10,6,3,4,7,8,2 ;define constant byte=8bits
END
```



13. Write a program to copy an array.

```
        AREA PROGRAM,CODE,READONLY
ENTRY
        LDR R0,=ST
        MOV R4,#0x04
```

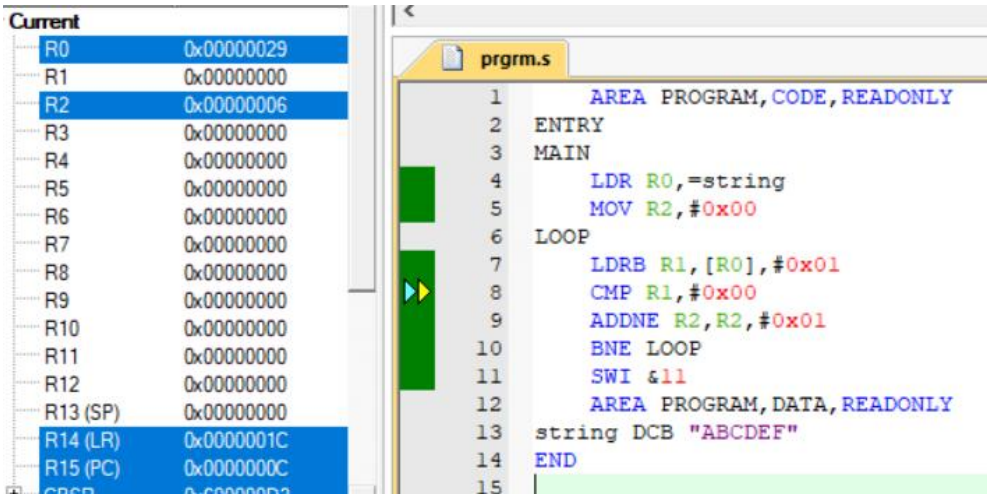
```
        LDR R5,VALUE1
LOOP
        LDRB R2,[R0],#0x01
        STRB R2,[R5],#0x01
        SUB R4,R4,#0x01
        CMP R4,#0x00
        BNE LOOP
        AREA PROGRAM,DATA,READONLY
ST DCB 1,2,3,4
VALUE1 DCD &00000080
END
```



14. Write a program to count the number of characters in a given string.

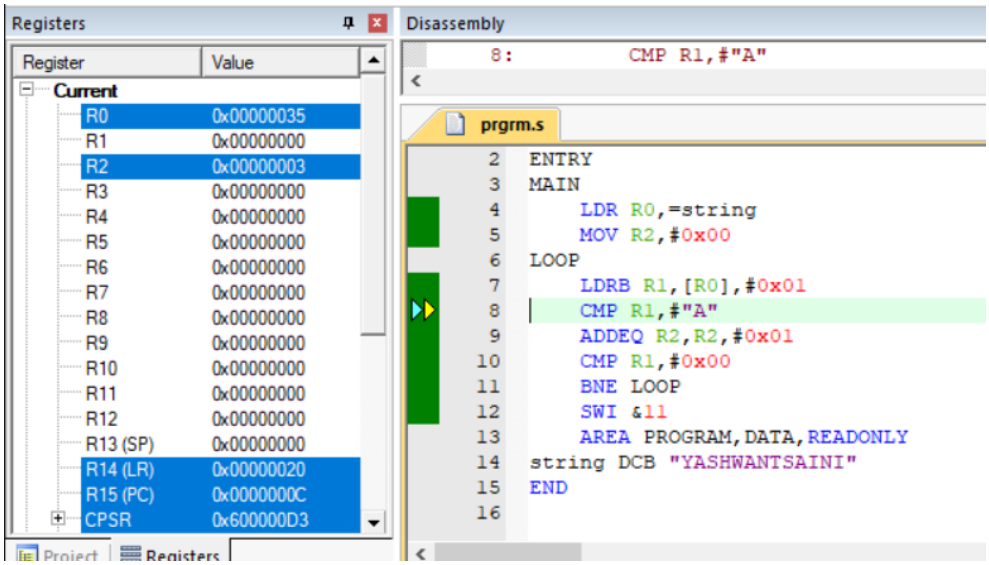
```
        AREA PROGRAM,CODE,READONLY
ENTRY
MAIN
        LDR R0,=string
        MOV R2,#0x00
LOOP
        LDRB R1,[R0],#0x01
        CMP R1,#0x00
        ADDNE R2,R2,#0x01
        BNE LOOP
        SWI &11
        AREA PROGRAM,DATA,READONLY
```

string DCB "ABCDEF"
END



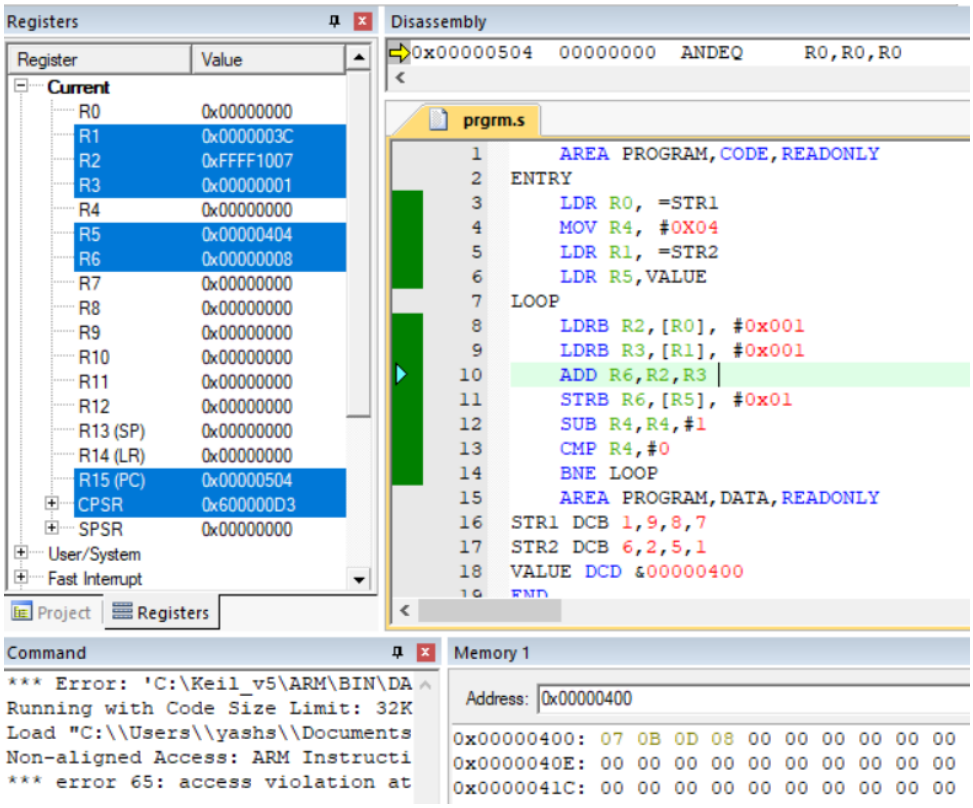
15. Write a program to find the number of occurrences of a particular character in a string.

```
        AREA PROGRAM,CODE,READONLY
ENTRY
MAIN
        LDR R0,=string
        MOV R2,#0x00
LOOP
        LDRB R1,[R0],#0x01
        CMP R1,#"A"
        ADDEQ R2,R2,#0x01
        CMP R1,#0x00
        BNE LOOP
        SWI &11
        AREA PROGRAM,DATA,READONLY
string DCB "YASHWANTSAINI"
END
```



16. Write a program to add two integer strings.

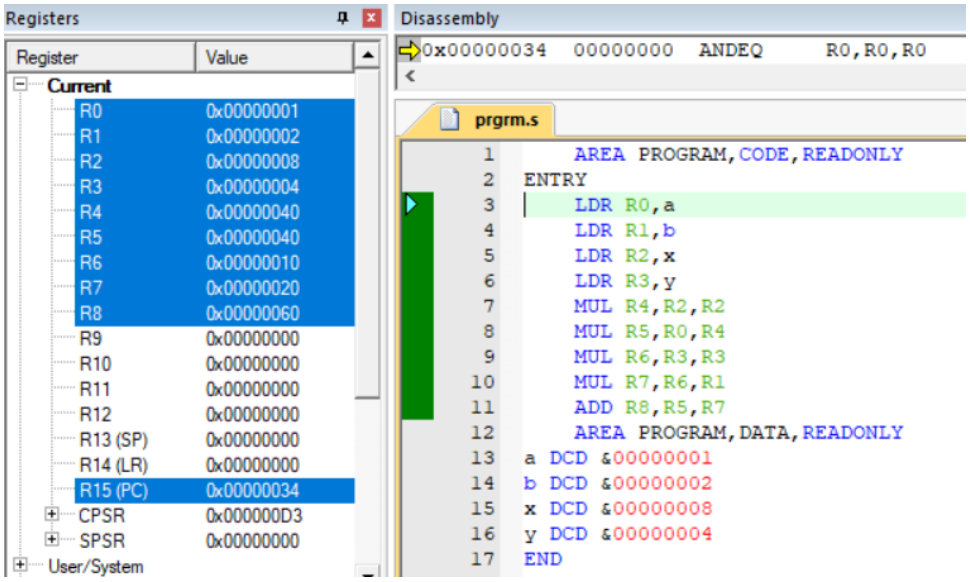
```
        AREA PROGRAM,CODE,READONLY
ENTRY
        LDR R0, =STR1
        MOV R4, #0X04
        LDR R1, =STR2
        LDR R5,VALUE
LOOP
        LDRB R2,[R0], #0x001
        LDRB R3,[R1], #0x001
        ADD R6,R2,R3
        STRB R6,[R5], #0x01
        SUB R4,R4,#1
        CMP R4,#0
        BNE LOOP
        AREA PROGRAM,DATA,READONLY
STR1 DCB 1,9,8,7
STR2 DCB 6,2,5,1
VALUE DCD &00000400
END
```



17. Write a program in ARM assembly language to implement the following equation: i) ax2+by2 ii) 6(x + y) +2z+4.

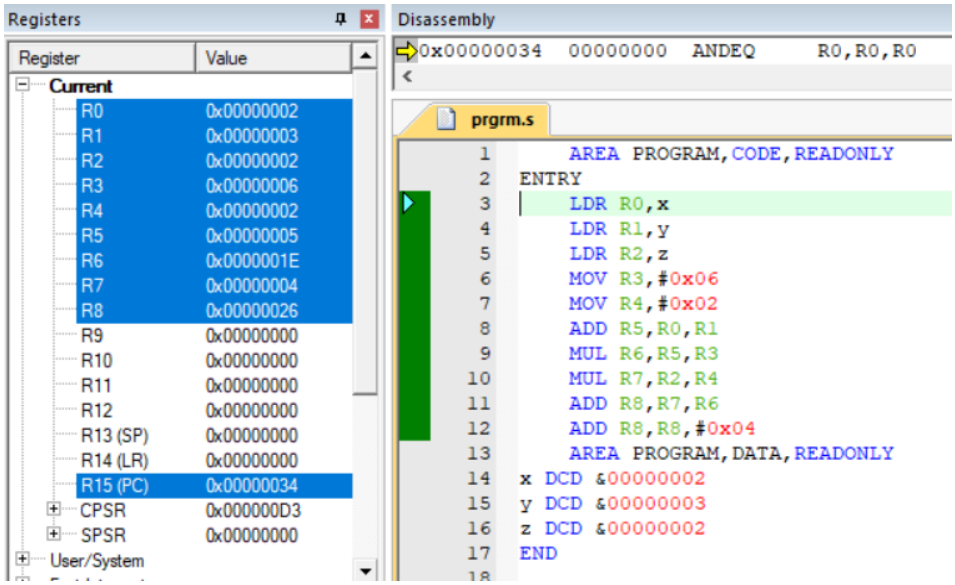
i)

```
   AREA PROGRAM,CODE,READONLY
ENTRY
   LDR R0,a
   LDR R1,b
   LDR R2,x
   LDR R3,y
   MUL R4,R2,R2
   MUL R5,R0,R4
   MUL R6,R3,R3
   MUL R7,R6,R1
   ADD R8,R5,R7
   AREA PROGRAM,DATA,READONLY
a DCD &00000001
b DCD &00000002
x DCD &00000008
y DCD &00000004
END
```



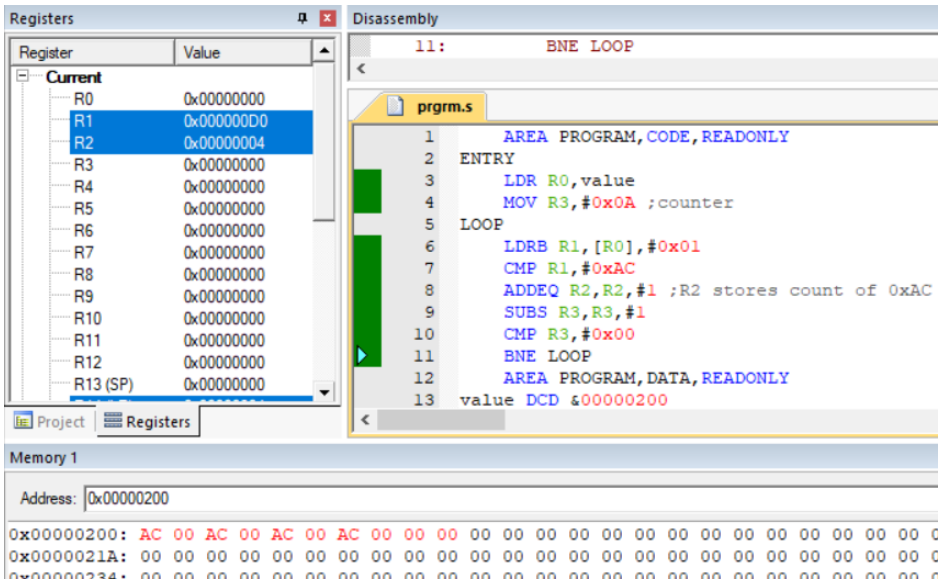
ii)

```
        AREA PROGRAM,CODE,READONLY
ENTRY
        LDR R0,x
        LDR R1,y
        LDR R2,z
        MOV R3,#0x06
        MOV R4,#0x02
        ADD R5,R0,R1
        MUL R6,R5,R3
        MUL R7,R2,R4
        ADD R8,R7,R6
        ADD R8,R8,#0x04
        AREA PROGRAM,DATA,READONLY
x DCD &00000002
y DCD &00000003
z DCD &00000002
END
```



18. Write a program in ARM assembly language to verify how many bytes are present in a given set which resemble 0xAC.

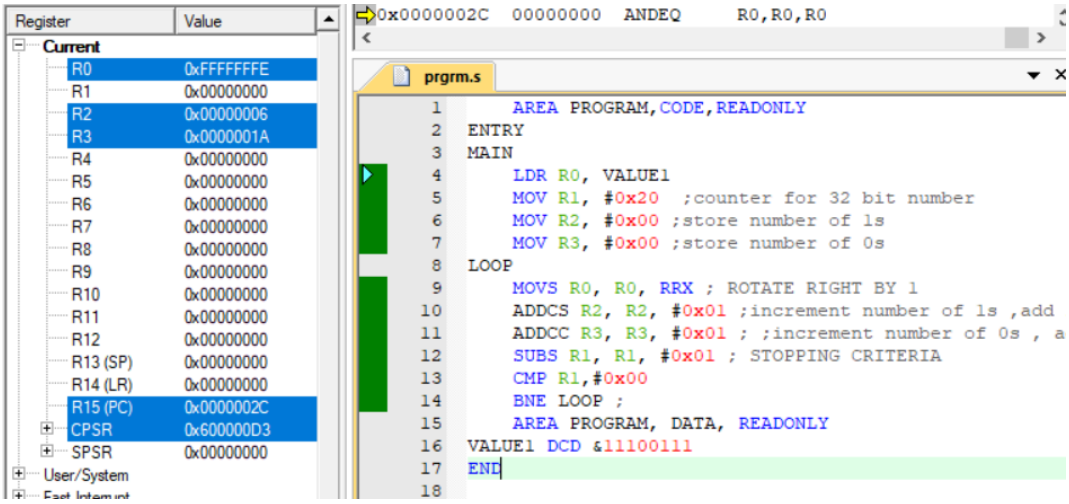
```
        AREA PROGRAM,CODE,READONLY
ENTRY
        LDR R0,value
        MOV R3,#0x0A ;counter
LOOP
        LDRB R1,[R0],#0x01
        CMP R1,#0xAC
        ADDEQ R2,R2,#1 ;R2 stores count of 0xAC
        SUBS R3,R3,#1
        CMP R3,#0x00
        BNE LOOP
        AREA PROGRAM,DATA,READONLY
value DCD &00000200
END
```



19. Write a program in ARM assembly language to count the number of 1s and 0s in a given byte and verify the result.

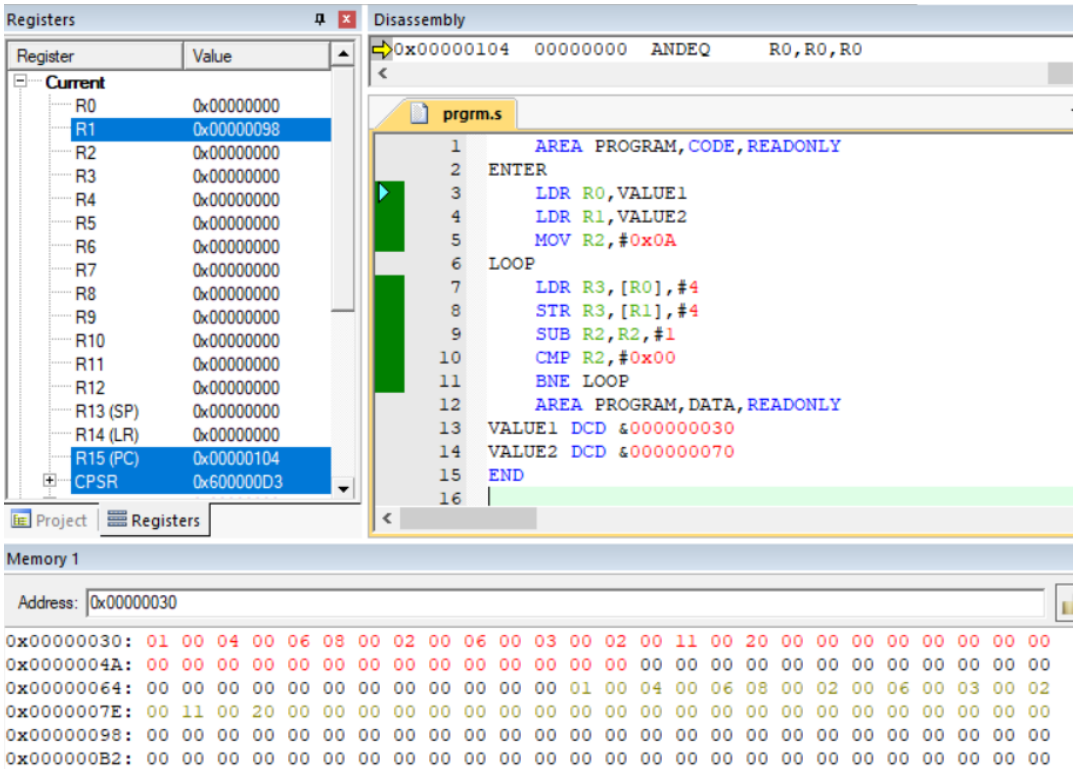
```
        AREA PROGRAM,CODE,READONLY
ENTRY
MAIN
        LDR R0, VALUE1
        MOV R1, #0x20  ;counter for 32 bit number
        MOV R2, #0x00 ;store number of 1s
        MOV R3, #0x00 ;store number of 0s
LOOP
        MOVS R0, R0, RRX ; ROTATE RIGHT BY 1
        ADDCS R2, R2, #0x01 ;increment number of 1s ,add if carry is set
```

```
        ADDCC R3, R3, #0x01 ; ;increment number of 0s , add if carry is clear
        SUBS R1, R1, #0x01 ; STOPPING CRITERIA
        CMP R1,#0x00
        BNE LOOP ;
        AREA PROGRAM, DATA, READONLY
VALUE1 DCD &11100111
END
```



20. Write a program in ARM assembly language for transferring of block of data (e.g. block transfer of 10 numbers from one memory location to another e.g. 0x00000030 to 0x00000300.)

```
        AREA PROGRAM,CODE,READONLY
ENTER
        LDR R0,VALUE1
        LDR R1,VALUE2
        MOV R2,#0x0A
LOOP
        LDR R3,[R0],#4
        STR R3,[R1],#4
        SUB R2,R2,#1
        CMP R2,#0x00
        BNE LOOP
        AREA PROGRAM,DATA,READONLY
VALUE1 DCD &000000030
VALUE2 DCD &000000070
END
```


Registers
Register
Value
Current
R0 0x00000000
R1 0x00000098
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000104
CPSR 0x600000D3
Project
Registers
Disassembly
0x00000104 00000000 ANDEQ R0,R0,R0
prgrm.s
1 AREA PROGRAM,CODE,READONLY
2 ENTER
3 LDR R0,VALUE1
4 LDR R1,VALUE2
5 MOV R2,#0x0A
6 LOOP
7 LDR R3,[R0],#4
8 STR R3,[R1],#4
9 SUB R2,R2,#1
10 CMP R2,#0x00
11 BNE LOOP
12 AREA PROGRAM,DATA,READONLY
13 VALUE1 DCD &000000030
14 VALUE2 DCD &000000070
15 END
16
Memory 1
Address: 0x00000030
0x00000030: 01 00 04 00 06 08 00 02 00 06 00 03 00 02 00 11 00 20 00 00 00 00 00 00 00 00
0x0000004A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000064: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 04 00 06 08 00 02 00 06 00 03 00 02
0x0000007E: 00 11 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x00000098: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x000000B2: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00