1. Write a program in ARM assembly language to add and subtract two 32-bit numbers using:
   1. Direct addressing mode

**ADDITION**

**CODE:**

area program,code,readonly

entry

main

LDR R1,value1

LDR R2,value2

ADD R3,R1,R2

area program,data,readonly

value1 DCD &02000005

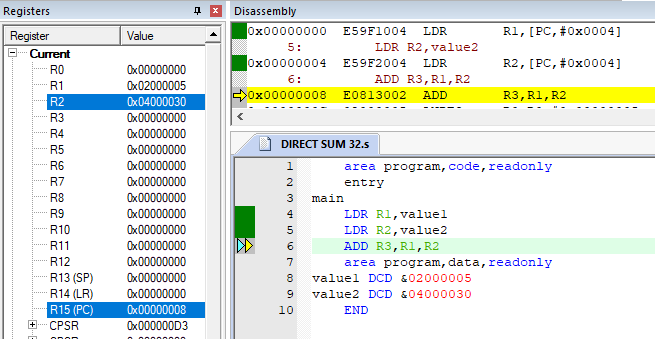
value2 DCD &04000030

END

**INPUT:**

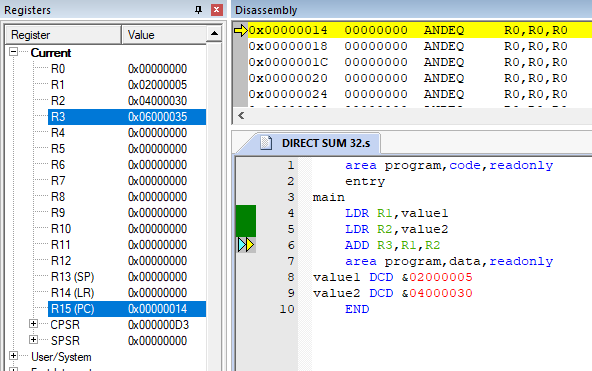
R1 <- 0x02000005

R2 <- 0x04000030

****

**OUTPUT:**

R3 <- R1+R2=0x06000035

****

**SUBTRACTION**

**CODE:**

area program,code,readonly

entry

main

LDR R1,value1

LDR R2,value2

SUB R3,R2,R1

area program,data,readonly

value1 DCD &02000005

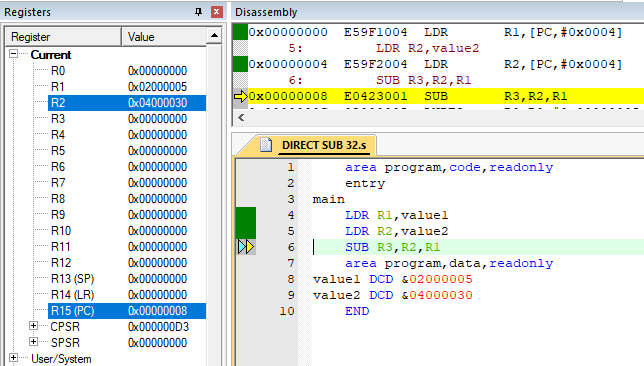
value2 DCD &04000030

END

**INPUT:**

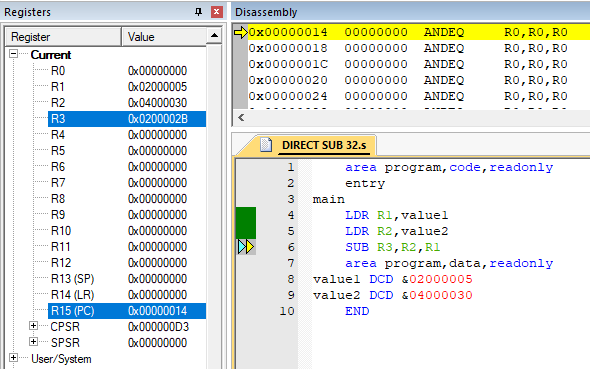
R1 <- 0x02000005

R2 <- 0x04000030

****

**OUTPUT:**

R3 <- R2-R1

****

* 1. Indirect addressing mode

**ADDITION**

**CODE:**

area program,code,readonly

entry

main

LDR R0,value1

LDR R1,value2

LDR R2,[R0]

LDR R3,[R1]

ADD R4,R2,R3

area program,data,readonly

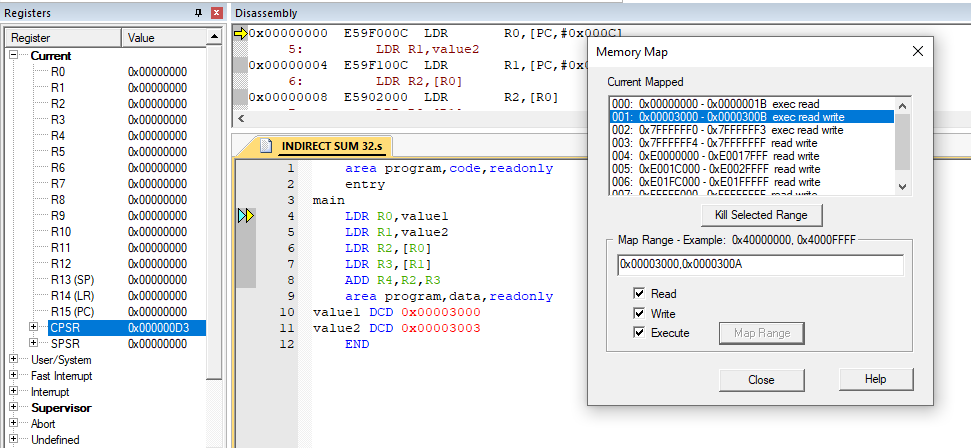
value1 DCD 0x00003000

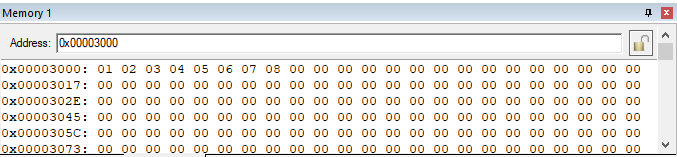
value2 DCD 0x00003004

END

INPUT:

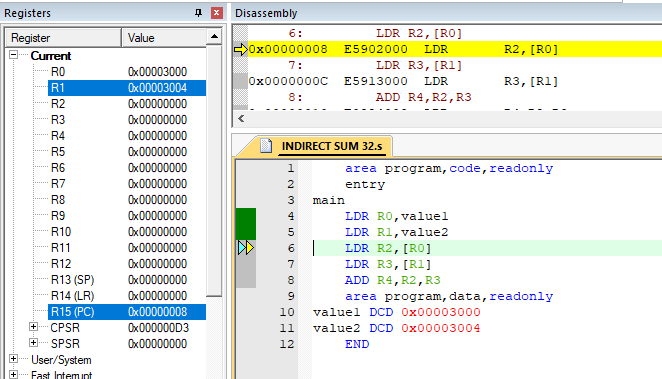
MAP RANGE (0x00003000,0x0000300A)





R0 <- 0x00003000

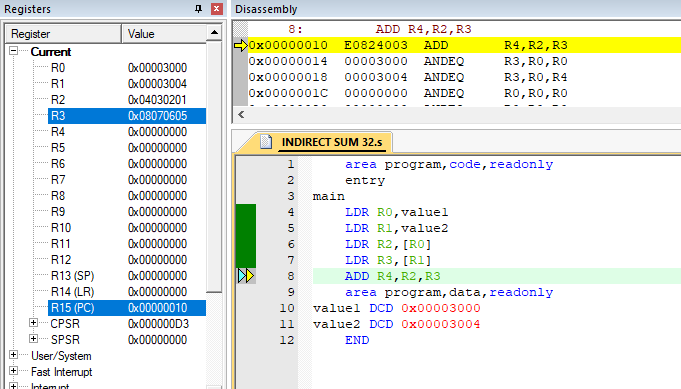
R1 <- 0x00003004



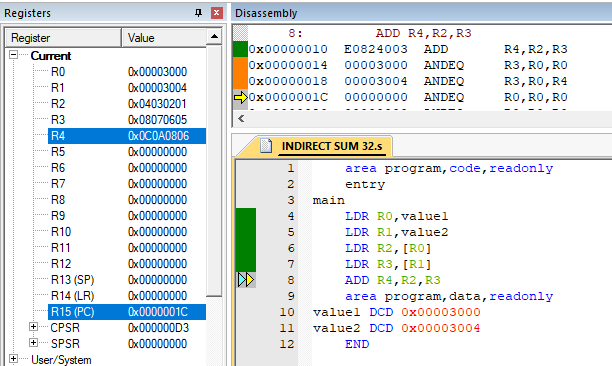
R2 <- [R0] = 0x04030201

R3 <- [R1] = 0x08070605

(Little Endian Format)



R4 <- R2+R3



**SUBTRACTION**

**CODE:**

area program,code,readonly

entry

main

LDR R0,value1

LDR R1,value2

LDR R2,[R0]

LDR R3,[R1]

SUB R4,R3,R2

area program,data,readonly

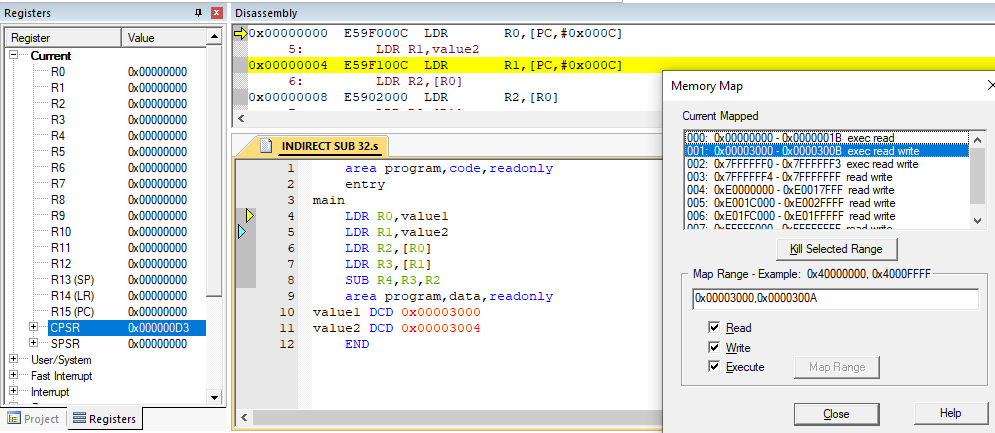
value1 DCD 0x00003000

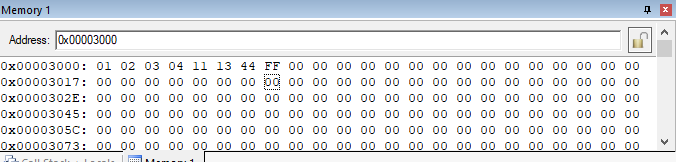
value2 DCD 0x00003004

END

**INPUT:**

MAP RANGE (0x00003000,0x0000300A)





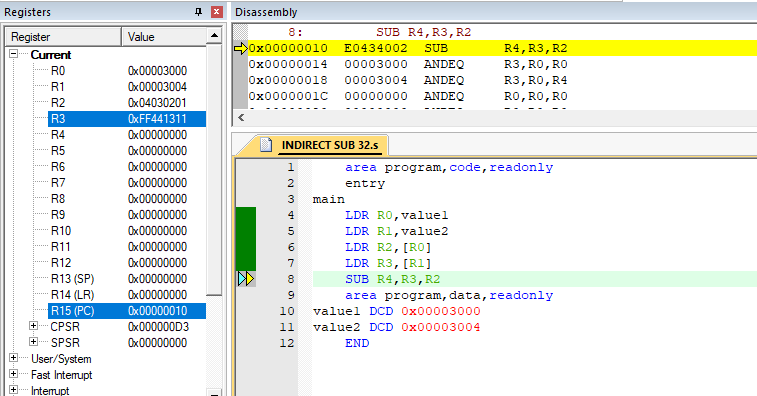
**OUTPUT:**

R0 <- 0x00003000

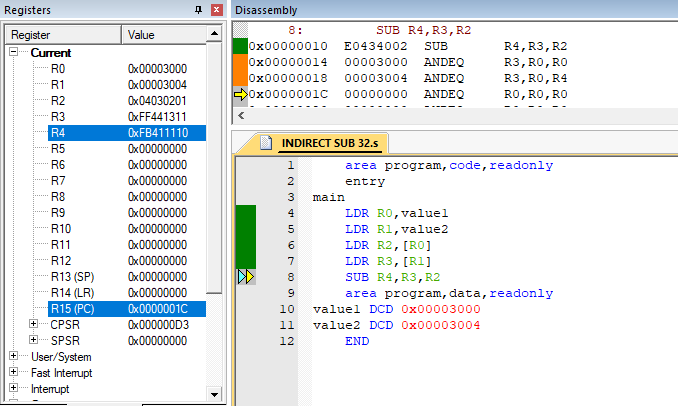
R1 <- 0x00003004

R2 <- [R0]

R3 <- [R1]



R4 <- R3-R2



* 1. Barrel shifter

**ADDITION**

**CODE:**

area program,code,readonly

entry

main

LDR R1,value

MOV R2,R1,LSL#0x04

ADD R3,R1,R2

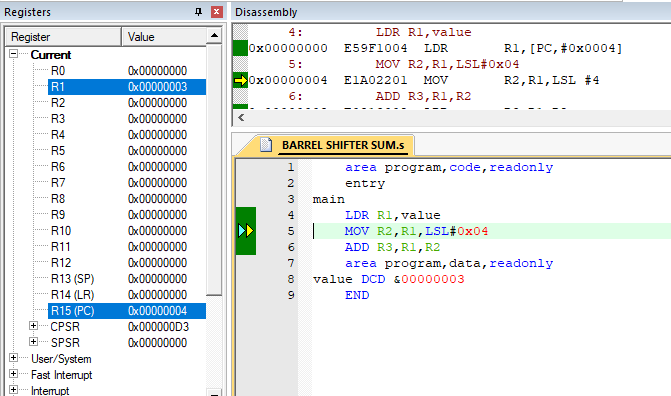
area program,data,readonly

value DCD &00000003

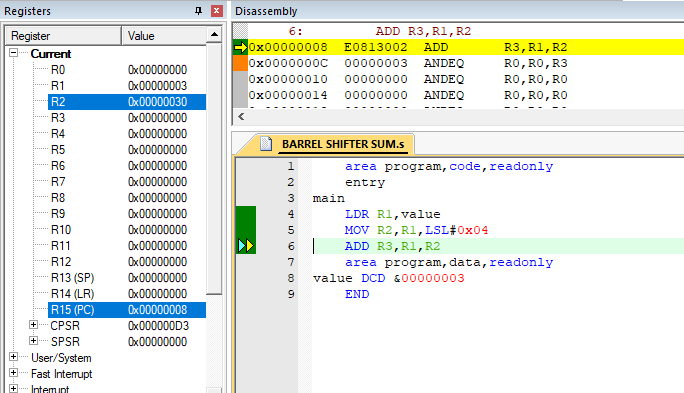
END

**OUTPUT:**

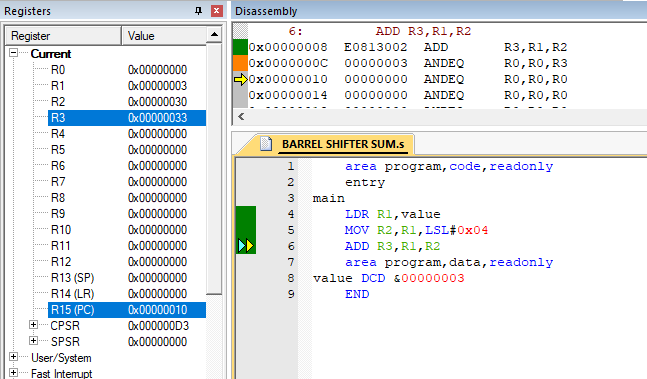
R1 <- 0x00000003



R2 <- R1, LSL 0x04 = 0x00000030

****

R3 <- R1+R2



**SUBSTRACTION**

**CODE:**

area program,code,readonly

entry

main

LDR R1,value

MOV R2,R1,LSL#0x04

SUB R3,R2,R1

area program,data,readonly

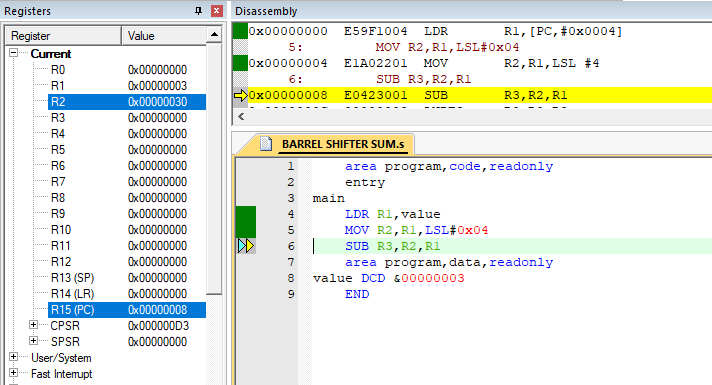
value DCD &00000003

END

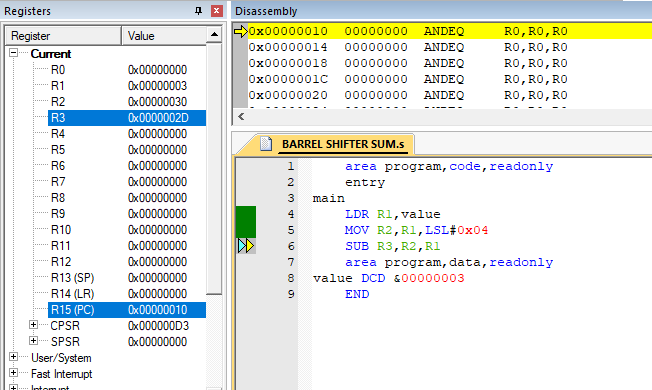
**OUTPUT:**

R1 <- 0x00000003

R2 <- R1, LSL 0x04 = 0x00000030



R3 <- R2-R1



1. Write a program to perform left and right shift of a number.

**CODE:**

area program,code,readonly

entry

main

LDR R1,value

MOV R2,R1,LSL#0x04

MOV R3,R1,LSR#0x04

MOV R4,R1,ASR#0x04

MOV R5,R1,ROR#0x04

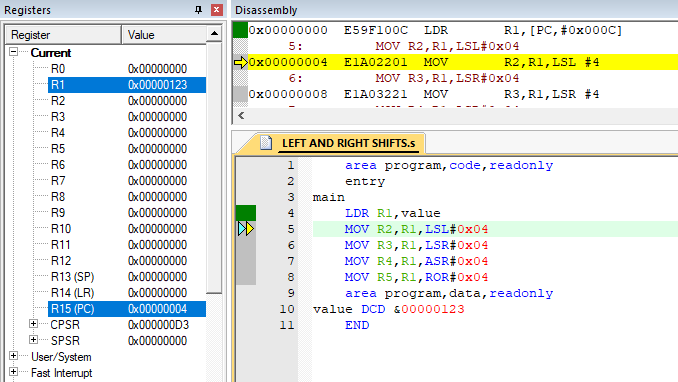
area program,data,readonly

value DCD &00000123

END

**OUTPUT:**

R1 <- 0x00000123

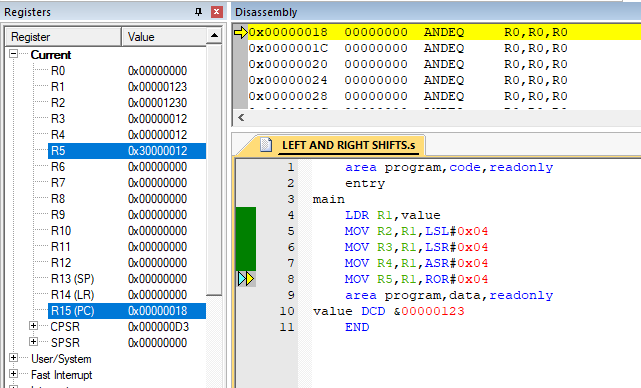
****

R2 <- R1,LSL#0x04

R3 <- R1,LSR#0x04

R4 <- R1,ASR#0x04

R5 <- R1,ROR#0x04

****

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

**CODE:**

area program,code,readonly

entry

main

LDR R1,value1

LDR R2,value2

AND R1,#0x01

AND R2,#0x01

area program,data,readonly

value1 DCD &00000043

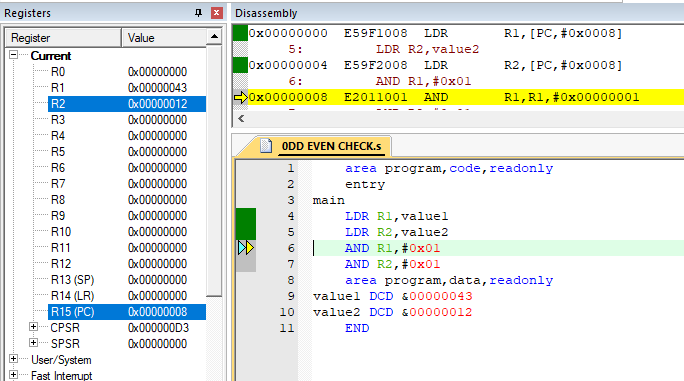
value2 DCD &00000012

END

**OUTPUT:**

R1 <- 0x00000043

R2 <- 0x00000012

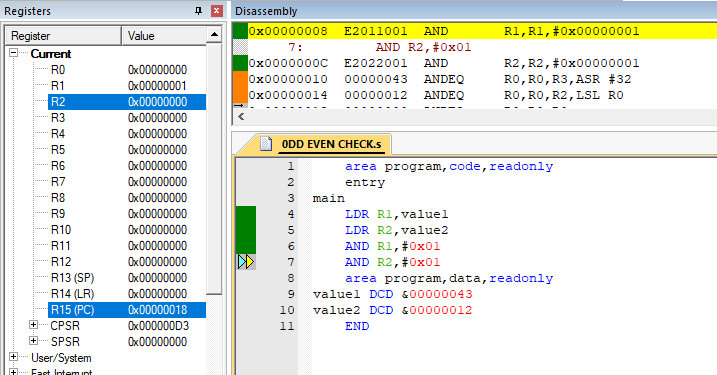


If number is EVEN, Result will be 0x00

Else 0x01

R1 <- 0x01 (ODD)

R2 <- 0x00 (EVEN)



1. Write a program to perform Multiplication using addition.

**CODE:**

area program,code,readonly

entry

main

LDR R0,value1

LDR R1,value2

MOV R2,R0

MOV R3,#0x01

LOOP

ADD R3,R3,#0x01

ADD R2,R0,R2

CMP R1,R3

BNE LOOP

area program,data,readonly

value1 DCD &00000002

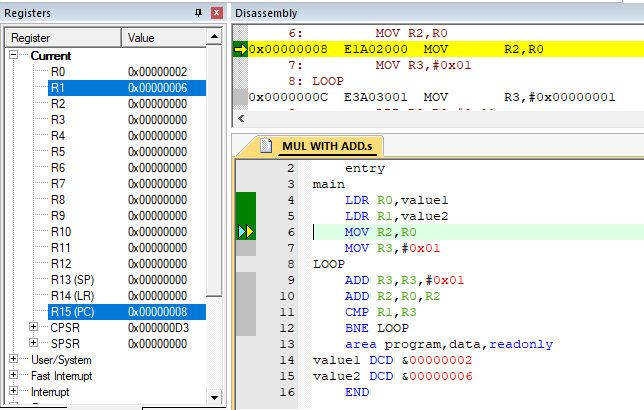
value2 DCD &00000006

END

OUTPUT:

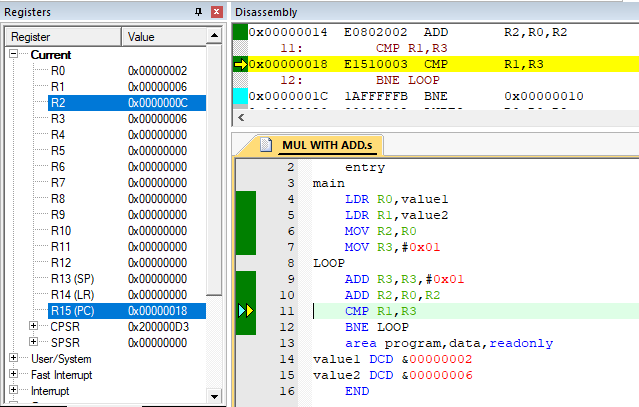
R0 <- 0x00000002

R1 <- 0x00000006



After the loop ends,

R3 <- 0x000000C (6X2)



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

**CODE**

area program,code,readonly

entry

main

LDR R0,value1

LDR R1,value2

MOV R2,#0x0A

MOV R3,R0

LOOP

STRB R0,[R1]

ADD R0,R0,R3

SUB R2,R2,#0x01

ADD R1,#0x01

CMP R2,#0x00

BNE LOOP

area program,data,readonly

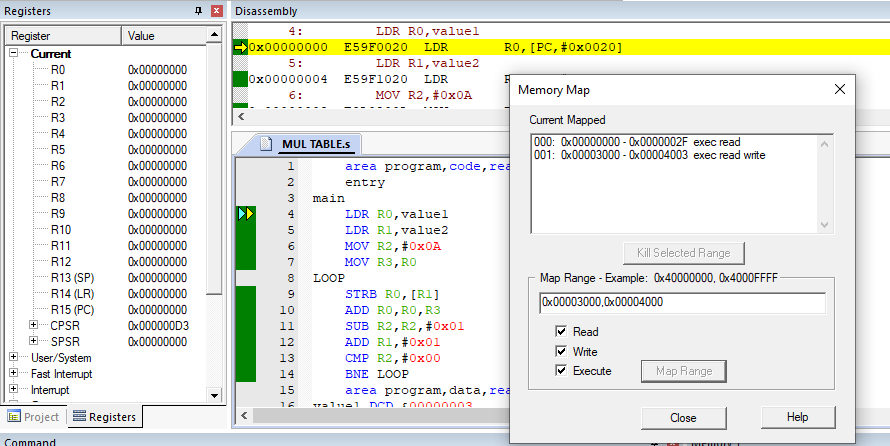
value1 DCD &00000003

value2 DCD &00003000

END

**OUTPUT:**

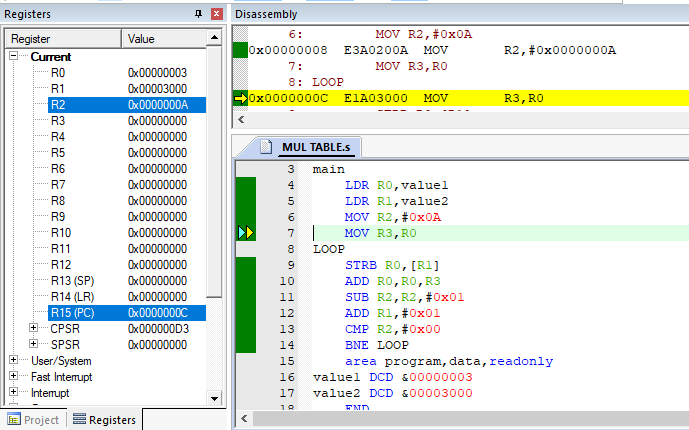
MAP MEMORY (0x00003000,0x00004000)



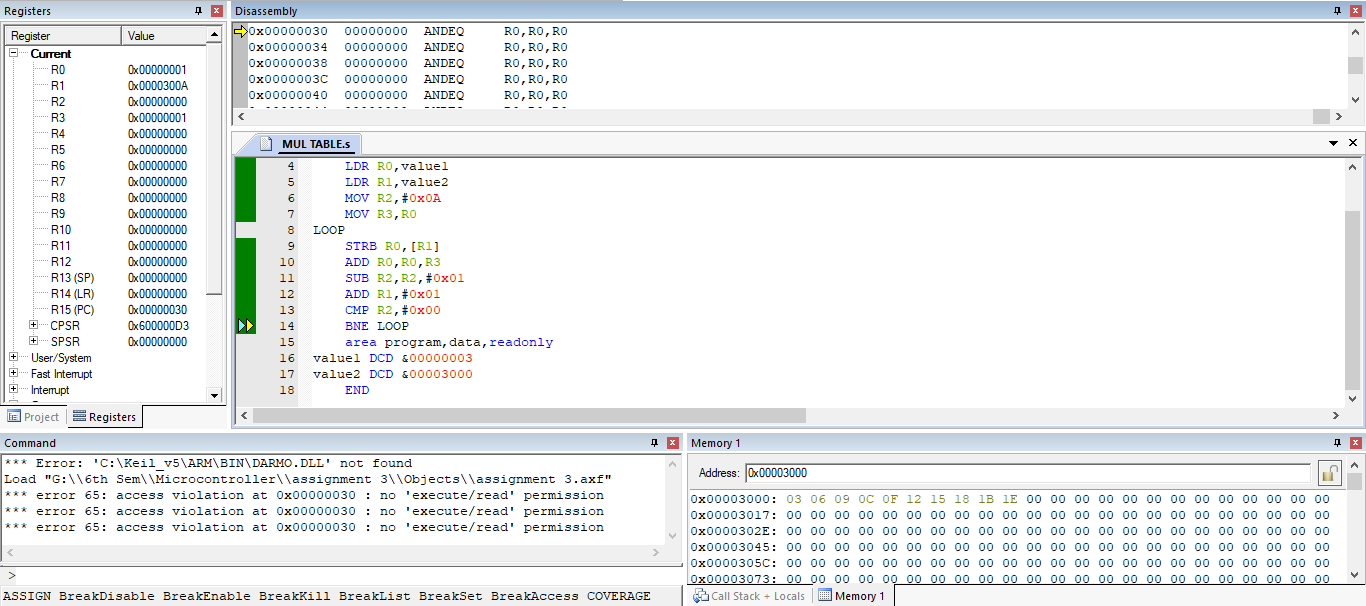
R0 <- 0x00000003 (TABLE OF 3)

R1 <- 0x00003000 (RESULT TO BE STORED AT)

R2 <- 0x0A



After Running Complete Loop



1. Write a program to perform Division using subtraction.

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

MOV R0,# 4

MOV R1,# 17

MOV R3,#0x000

MOV R4,#0x000

loop

ADD R3, #0x01

SUBS R1,R1,R0

MOV R4,R1

CMP R4,R0

BGT loop

SUB R3,#0x01

AREA program, code, readonly

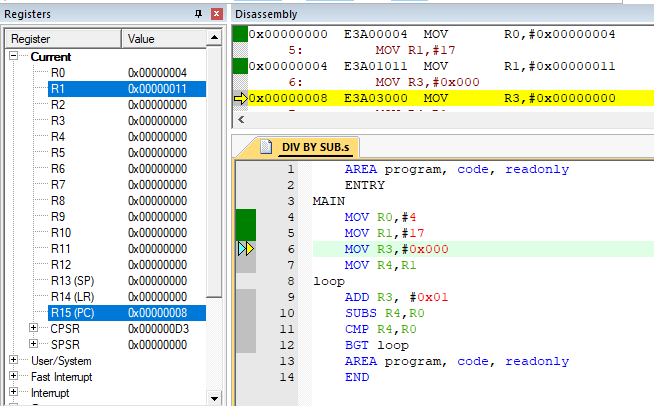
END

**OUTPUT:**

R0 <- 0x04

R1 <- 0x11

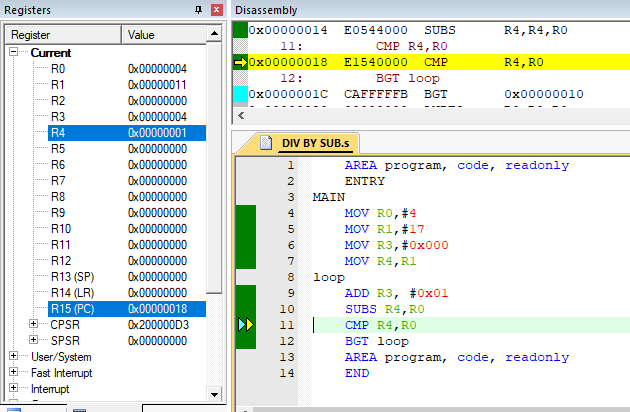
(WE ARE PERFORMING 17/4)

****

AFTER THE LOOP ENDS,

R3 <- QUOTIENT

R4 <- REMAINDER



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

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =string

MOV R2, #0x00

MOV R4, #0x00

MOV R5, #0x00

loop

LDRB R1,[R0],#0x01

MOV R3, R1

CMP R1,#0x20

ADDEQ R4,#0x01

CMP R1,#0x00

ADDNE R2,#0x01

BNE loop

SUB R5,R2,R4

AREA program, data, readonly

string DCB "Gourish Singla"

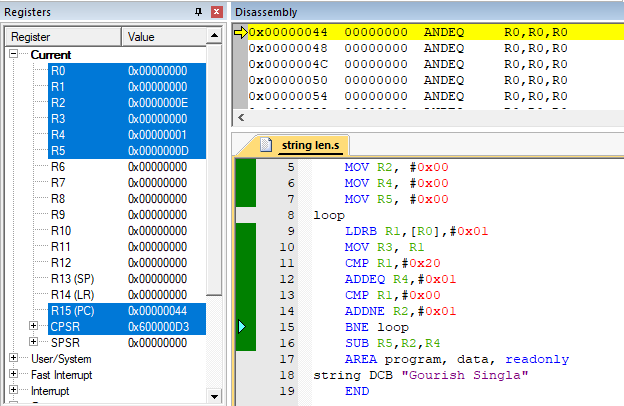
END

**OUTPUT:**

R5 <- NUMBER OF CHARACTERS WITHOUT SPACE

R4 <- NUMBER OF SPACES

R2 <- NUMBER OF CHARACTERS WITH SPACE

****

1. Write a program to find the number of occurrence of a particular character in a string.

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =string

MOV R2, #0x00

MOV R4, #0x00

loop

LDRB R1,[R0],#0x01

MOV R3, R1

CMP R1,#0x69

ADDEQ R4,#0x01

CMP R1,#0x00

BNE loop

AREA program, data, readonly

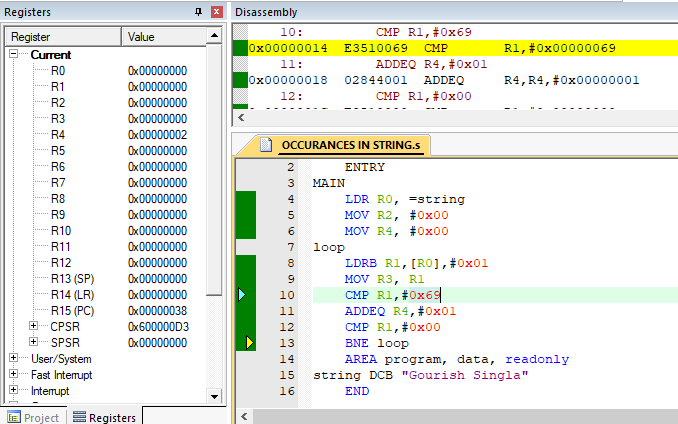
string DCB "Gourish Singla"

END

**OUTPUT:**

After Running the Loop,

R4 <- NUMBER OF OCCURANCES OF CHARANTER ‘i’ (0x69)

****

1. Write a program to add two integer strings.

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =value1

LDR R1, =value2

LDR R3, val3

MOV R4,#0x00

LDRB R5,[R0],#001

MOV R7,#010

loop

SUB R5,#0x30

MUL R6,R4,R7

MOV R4,R6

ADD R4,R5

LDRB R5,[R0],#001

CMP R5,#0x00

BNE loop

MOV R8,R4

LDRB R5,[R1],#001

MOV R4,#0x00

loop2

SUB R5,#0x30

MUL R6,R4,R7

MOV R4,R6

ADD R4,R5

LDRB R5,[R1],#001

CMP R5,#0x00

BNE loop2

ADDS R8,R4

STR R8,[R3]

AREA program, data, readonly

value1 DCB "123"

EOS1 DCB 0x00

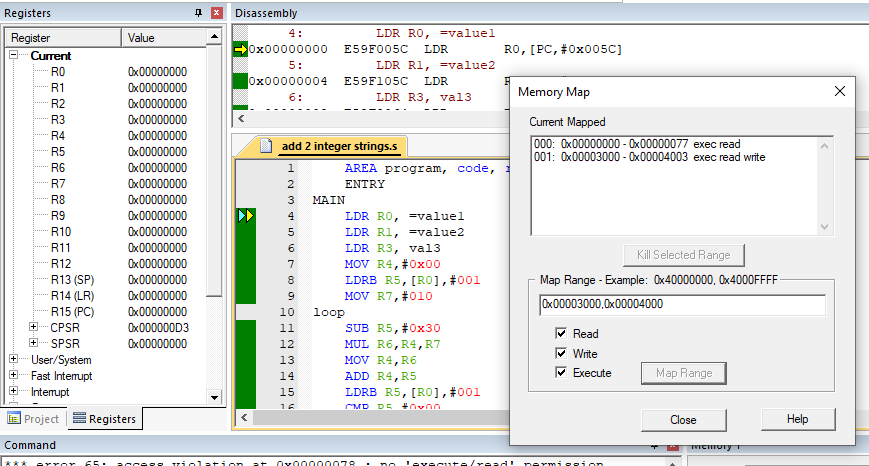
value2 DCB "245"

EOS2 DCB 0x00

val3 DCD &00003000

END

**OUTPUT:**

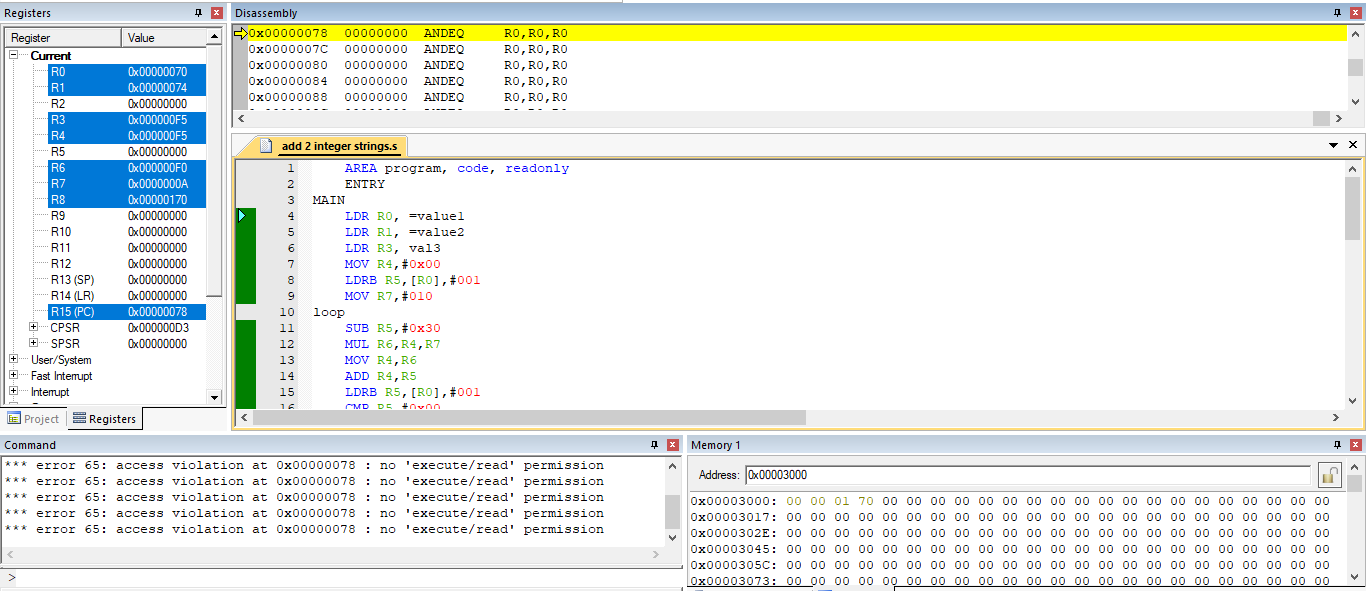
**** MAP MEMORY (0x00003000,0x00004000)

At the END of Execution

(value1:=”123” OR value1:=0x7B)

(value2:=”245” OR value2:=0XF5)

RESULT:=0x7B+0XF5 = 170 (AS STORED IN MEMORY LOCATION 0x00003000)



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

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0,val1

LDR R2,val2

LDR R6,val3

MOV R1,#0x000

loop

MUL R5,R2,R0

SUB R0,#0x01

MOV R2,R5

CMP R0,R1

BGT loop

STR R2,[R6]

AREA program, code, readonly

val1 DCD &00000005

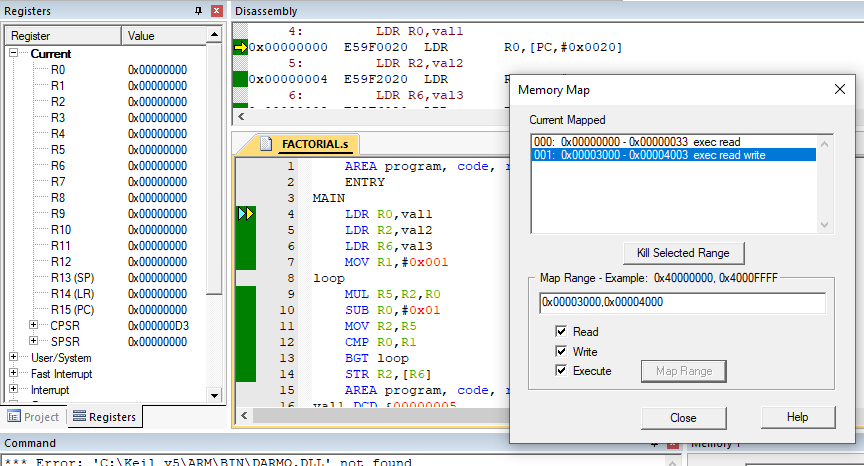
val2 DCD &00000001

val3 DCD &00003000

END

**OUTPUT:**

MAP MEMORY (0x00003000,0x00004000)

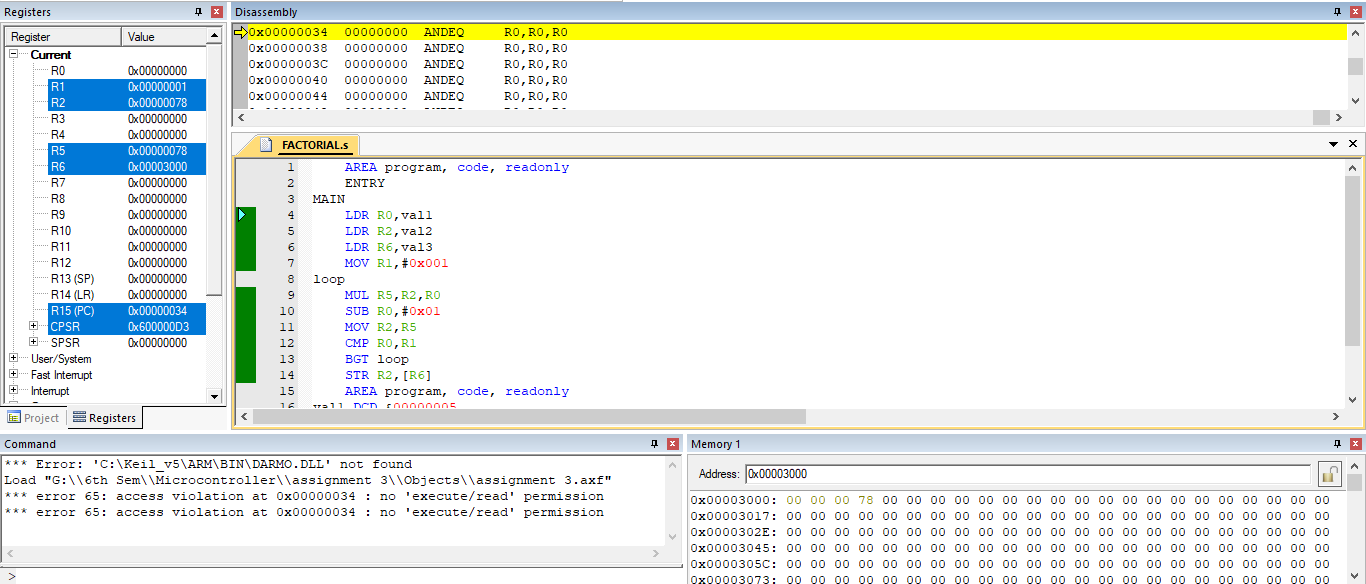


R0 <- 0x05 (Factorial Value to be found)

R6 <- 0x00003000 (where to store result)

FACTORIAL(5) = 120

FACTORIAL(0x05) = 0x78



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

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0,=val1

LDR R1,[R0]

LDR R2,[R0,#0x004]

LDR R0,=val2

LDR R3,[R0]

LDR R4,[R0,#0x004]

ADDS R6,R2,R4

ADC R5,R1,R3

LDR R0,val3

STR R5,[R0]

STR R6,[R0,#0x004]

AREA program, code, readonly

val1 DCD &00000001, &00000002

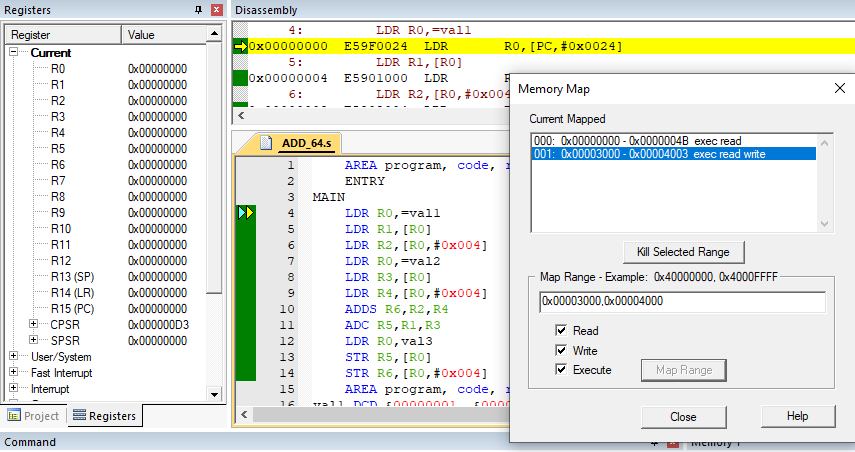
val2 DCD &00000003, &00000004

val3 DCD &00003000, &00003004

END

**OUTPUT:**

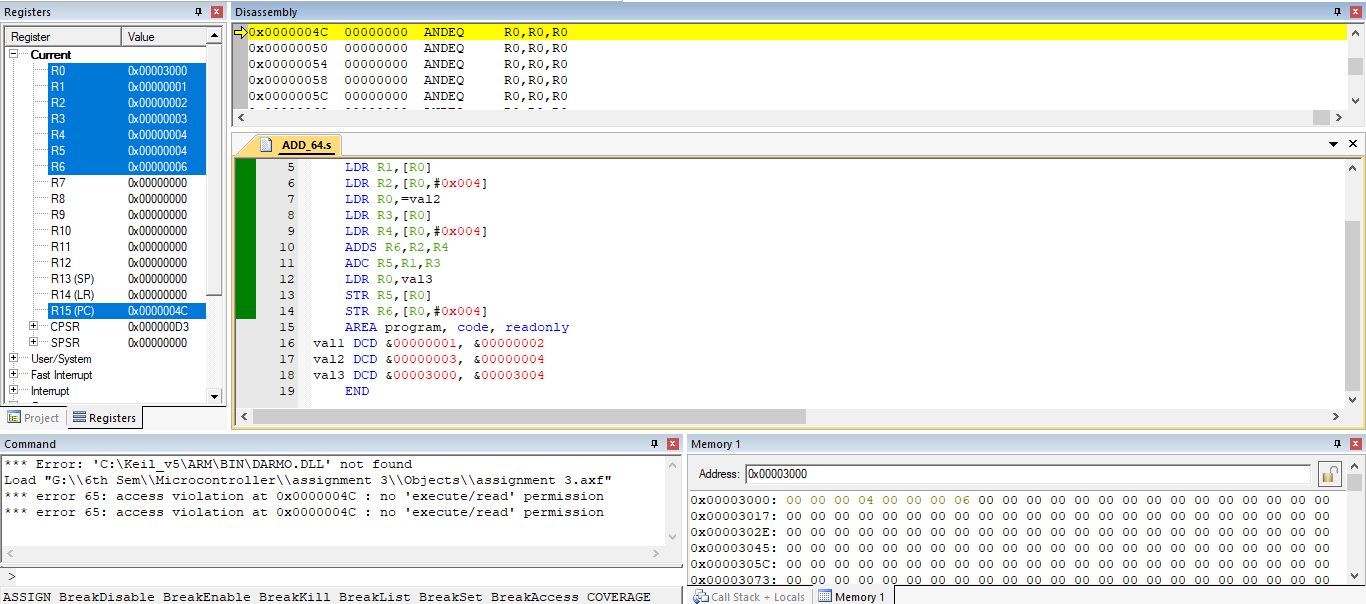
MAP MEMORY (0x00003000,0x00004000)

****

val 1 := 0x00000001 00000002

val 2 := 0x00000003 00000004

result := val 1 + val 2 = 0x00000004 00000006

****

1. Write a program to find the largest number in an array.

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =value1

LDRB R1, [R0], #0x01

LDRB R2, [R0], #0x01

LDR R3, size

LDR R4,val3

loop

CMP R2, R1

MOVGT R1, R2

SUB R3, #0x01

LDRB R2, [R0], #0x01

CMP R3, #0x01

BGT loop

STRB R1, [R4]

AREA program, data, readonly

value1 DCB 5, 10, 6, 18, 54, 28, 32, 14

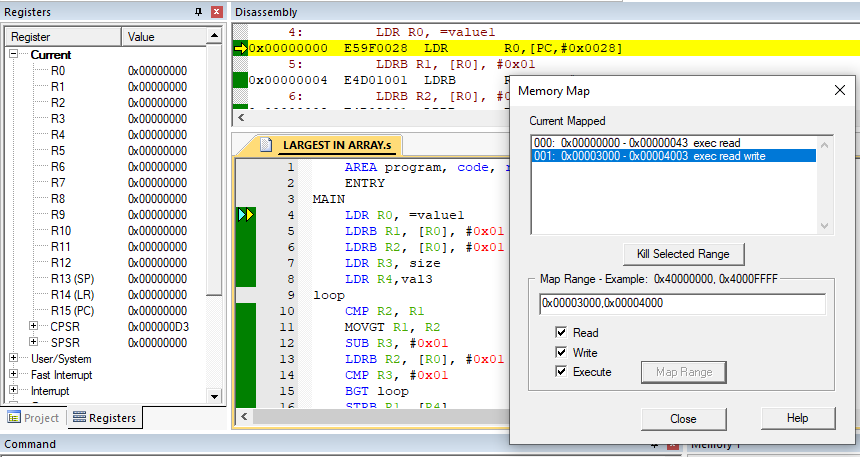
size DCD &00000008

val3 DCD &00003000

END

**OUTPUT:**

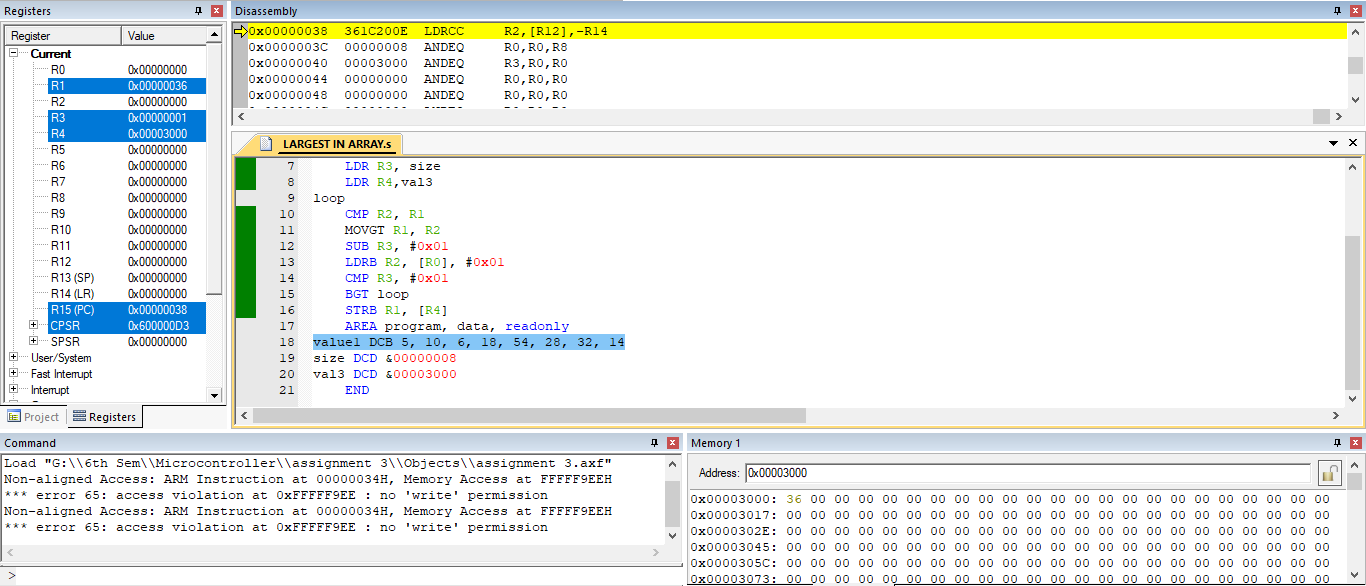
MAP MEMORY (0x00003000,0x00004000)

****

value1 := {5, 10, 6, 18, 54, 28, 32, 14}

LARGEST ELEMENT := 54 = 0x36

RESULT := 36



1. Write a program to copy an array.

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =value1

LDR R3, size

LDR R4,val

LDR R5,val2

loop

LDRB R1, [R0], #0x01

STRB R1, [R4], #0x01

SUB R3, #0x01

CMP R3, #0x01

BGE loop

LDR R0, =0x00000090

LDR R3, size

LDR R5,val2

loop1

LDR R1, [R0], #0x04

STR R1, [R5], #0x04

SUB R3, #0x04

CMP R3, #0x01

BGE loop1

AREA program, data, readonly

value1 DCB 1, 2, 4, 172, 6, 7, 2, 4, 2, 3

size DCD &0000000A

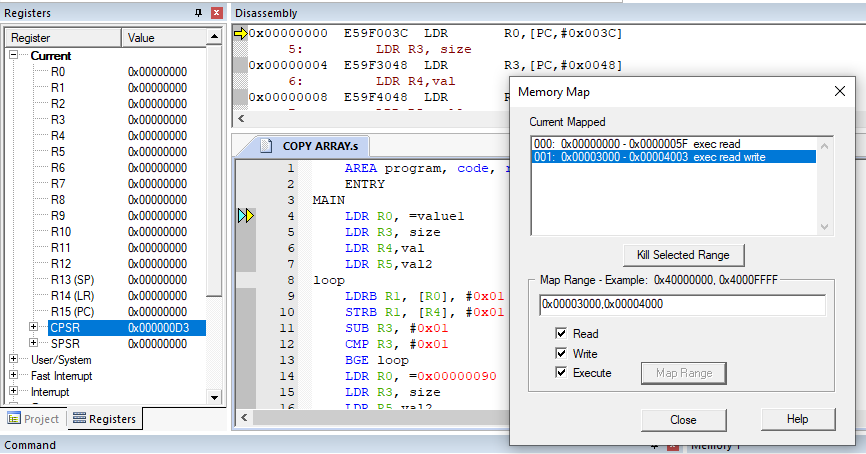
val DCD &00003000

val2 DCD &00000200

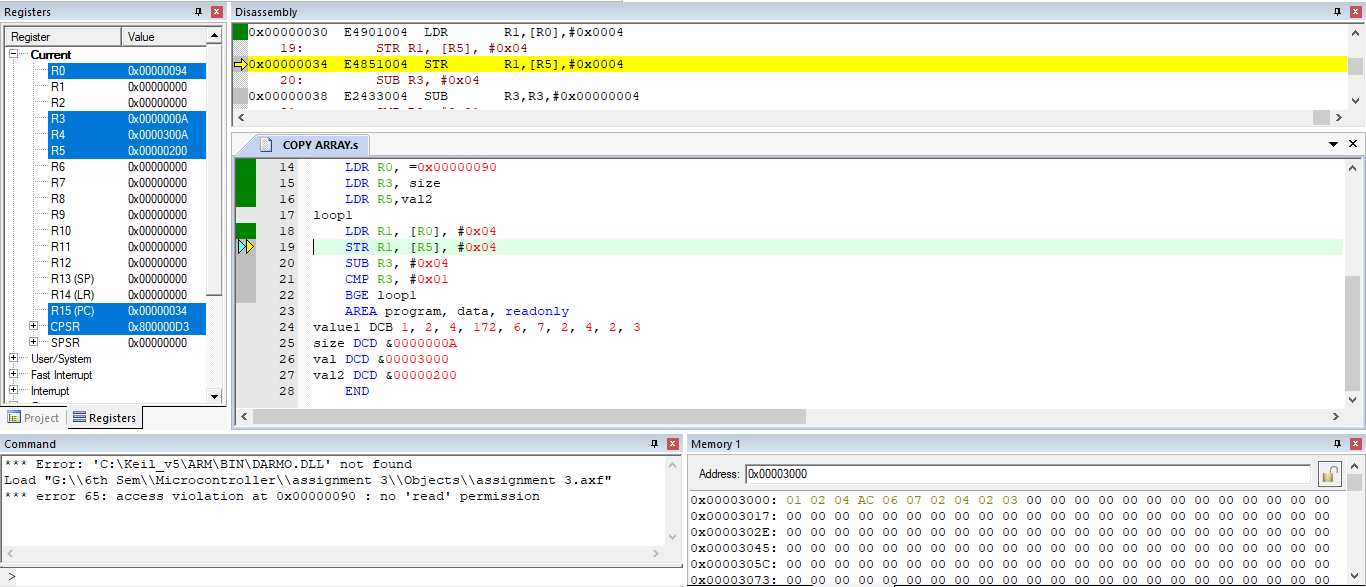
END

**OUTPUT:**

MAP MEMORY (0x00003000,0x00004000)



COPY ARRAY AT GIVEN MEMORY LOCATION (0x00003000)

****

1. Write a program in ARM assembly language to implement the following equation:
   1. ax2+by2

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R1,avar

LDR R2,bvar

LDR R3,xvar

LDR R4,yvar

LDR R8,val

MUL R5,R3,R3

MUL R6,R4,R4

MUL R5,R1,R5

MUL R6,R2,R6

ADDS R7,R5,R6

STR R7,[R8]

AREA program, data, readonly

avar DCD &00000004

bvar DCD &00000005

xvar DCD &00000002

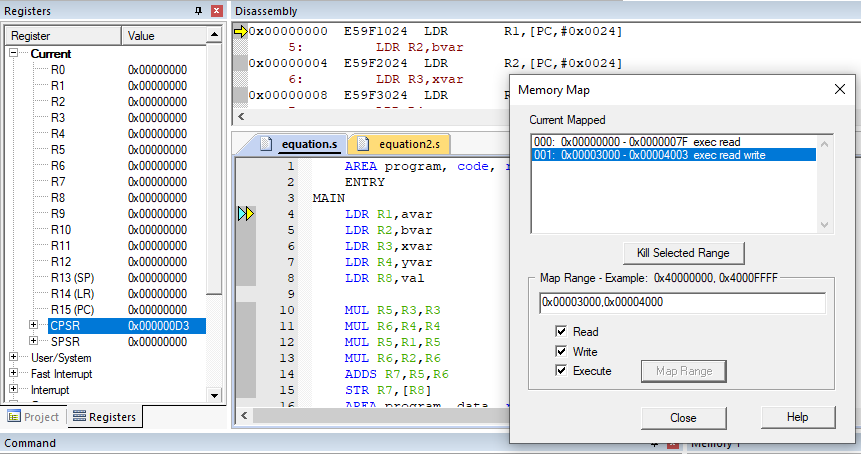
yvar DCD &00000003

val DCD &00003000

END

**OUTPUT**:

MAP MEMORY (0x00003000,0x00004000)



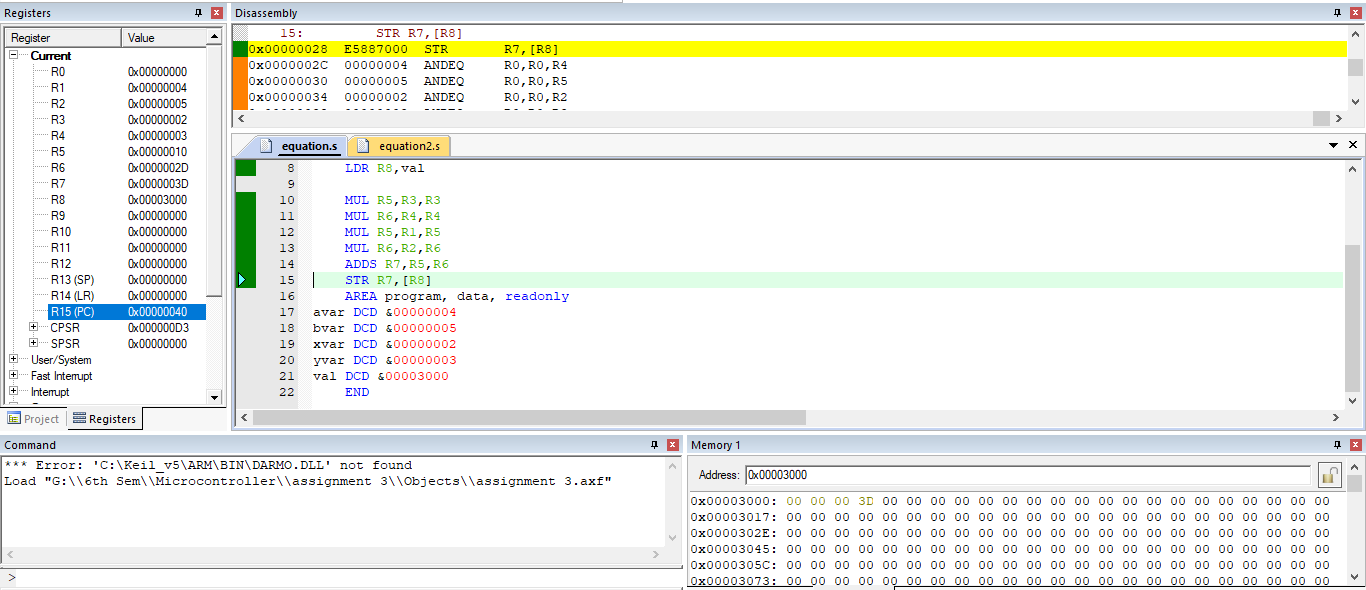
a = avar := 0x04

b = bvar := 0x05

x = xvar := 0x02

y = yvar := 0x03

RESULT := a\*x^2 + b\*y^2 = 61 = 0x3D



* 1. 6(x+y) +2z+4

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R1,xvar

LDR R2,yvar

LDR R3,zvar

LDR R8,val

ADD R4,R1,R2

MOV R0,#06

MUL R5,R4,R0

MOV R0,#02

MUL R6,R3,R0

ADD R7,R5,R6

ADD R7,#04

STR R7,[R8]

AREA program, data, readonly

xvar DCD &00000002

yvar DCD &00000003

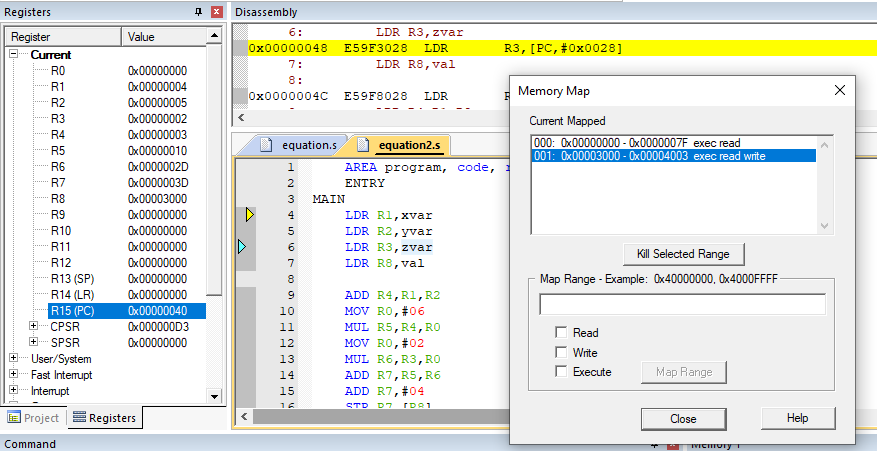
zvar DCD &00000004

val DCD &00003000

END

**OUTPUT:**

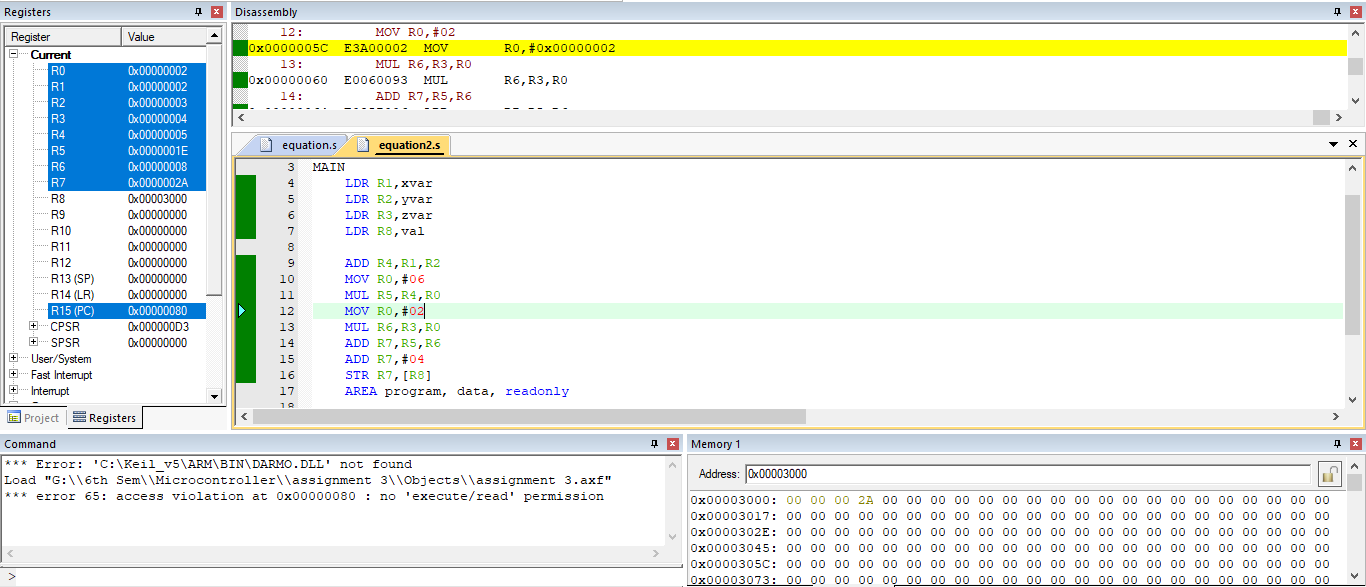
MAP MEMORY (0x00003000,0x00004000)



x := xvar = 0x02

y := yvar = 0x03

z := zvar = 0x04

RESULT := 6(x+y) +2z+4 := 42 = 0x2A

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

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

LDR R0, =value1

MOV R2,#0x00

LDR R3, size

LDR R4,val3

loop

LDRB R1, [R0], #0x01

CMP R1, #0xAC

ADDEQ R2,#0x01

SUB R3, #0x01

CMP R3, #0x01

BGE loop

STR R2,[R4]

AREA program, data, readonly

value1 DCB 1, 2, 172, 2, 6, 172, 2, 4, 172, 8

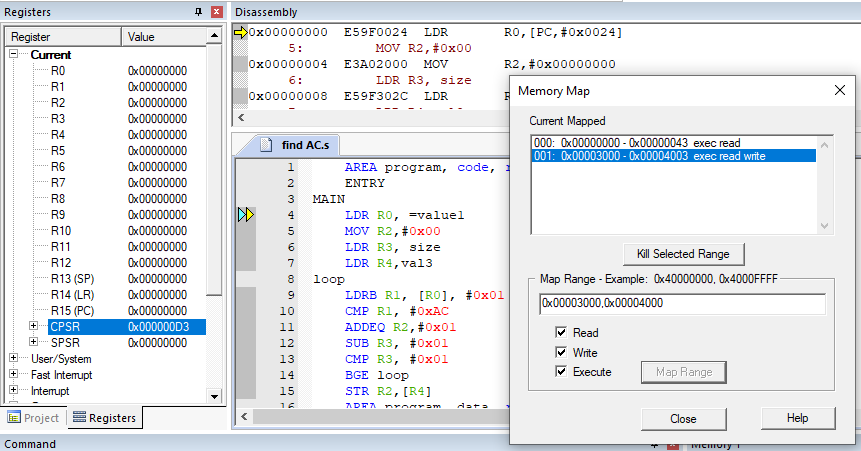
size DCD &00000010

val3 DCD &00003000

END

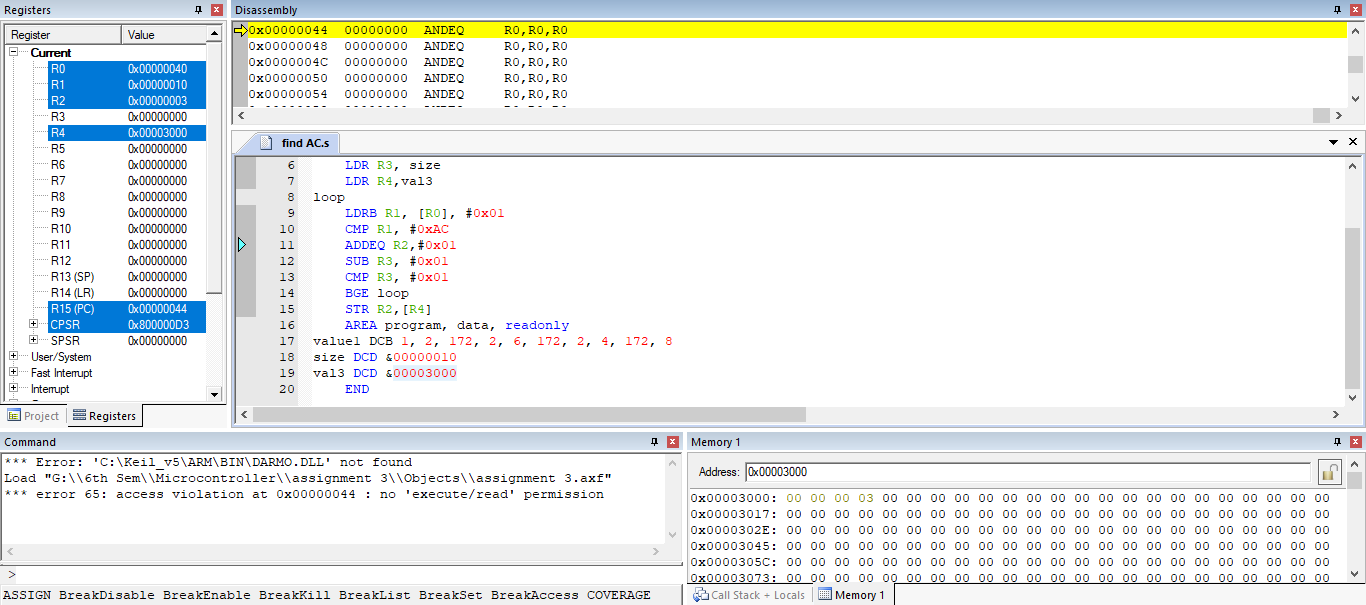
**OUTPUT:**

MAP MEMORY (0x00003000,0x00004000)

****

value1 := {1, 2, 172, 2, 6, 172, 2, 4, 172, 8}

FREQUENCY OF 0xAC (172) in value1 := 0x03

****

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

**CODE:**

AREA program, code, readonly

ENTRY

MAIN

MOVS R0, #0xAD

MOV R2, #0x8

LDR R5,val

loop

MOVS R0,R0,LSR #01

ADDCS R3,#0x01

ADDCC R4,#0x01

SUB R2,#0x01

CMP R2,#0x01

BGE loop

STR R3,[R5]

STR R4,[R5,#0x04]

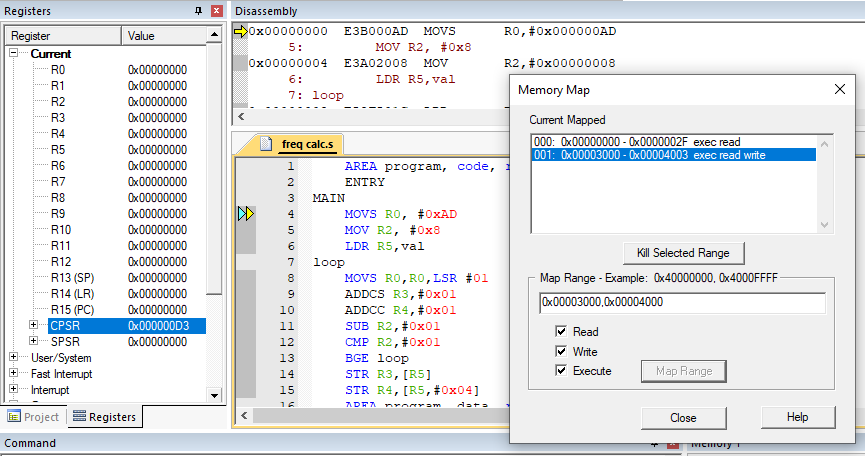
AREA program, data, readonly

val DCD &00003000

END

**OUTPUT:**

MAP MEMORY (0x00003000,0x00004000)

****

0xAD = 10101101

NUMBER OF 1’s = 5

NUMBER OF 0’s = 3

