

List of Experiments

1. Write a program in ARM assembly language to load any register with 32-bit data and perform the following:
 - A) Shift left by 2 bits
 - B) Shift right by the number of bits stored in register R2
 - C) Shift left 5 bits conditionally when zero flag is set
 - D) Arithmetic shift right by the value contained in register R2
2. Write a program in ARM assembly language to add two 32 bit numbers using:
 - A) Direct addressing mode
 - B) Indirect addressing mode (2 consecutive values)
 - C) Offset Addressing Mode, Immediate Addressing Mode, Barrel Shift.
3. Write a program in ARM assembly language to copy & consecutive words from source to destination in memory using:
 - a) Multiple register transfer instructions
 - b) Load and store instructions in a loop
4. Write a program to verify how many bytes are present in a given set which resemble 0xAC.
5. Write a program to count the number of 1's and 0's in a given bytes and verify the result.
6. Write a program in ARM assembly language to perform multiplication using repeated addition.
7. Write a program in ARM assembly language to perform division using repeated subtraction.
8. Write a program in ARM assembly language to find the factorial of a number using lookup table.
9. Write a program in ARM assembly language to find the Fibonacci of a number using lookup table.
10. Write a program in ARM assembly language to find the number of occurrences of a letter in a given string.
11. Write a program in ARM assembly language to implement the equation:
 - A) $ax^2 + by^2$
 - B) $6(x+y) + 2z+4$
12. Write a program in ARM assembly language to find the length of a given string.
13. Write a program in ARM assembly language to construct STACK.
14. Write a program in ARM assembly language to add two 64 bit registers.

1. Write a program in ARM assembly language to load any register with 32 bit data and perform the following:

A) Shift left by 2 bits

area program,code,readonly

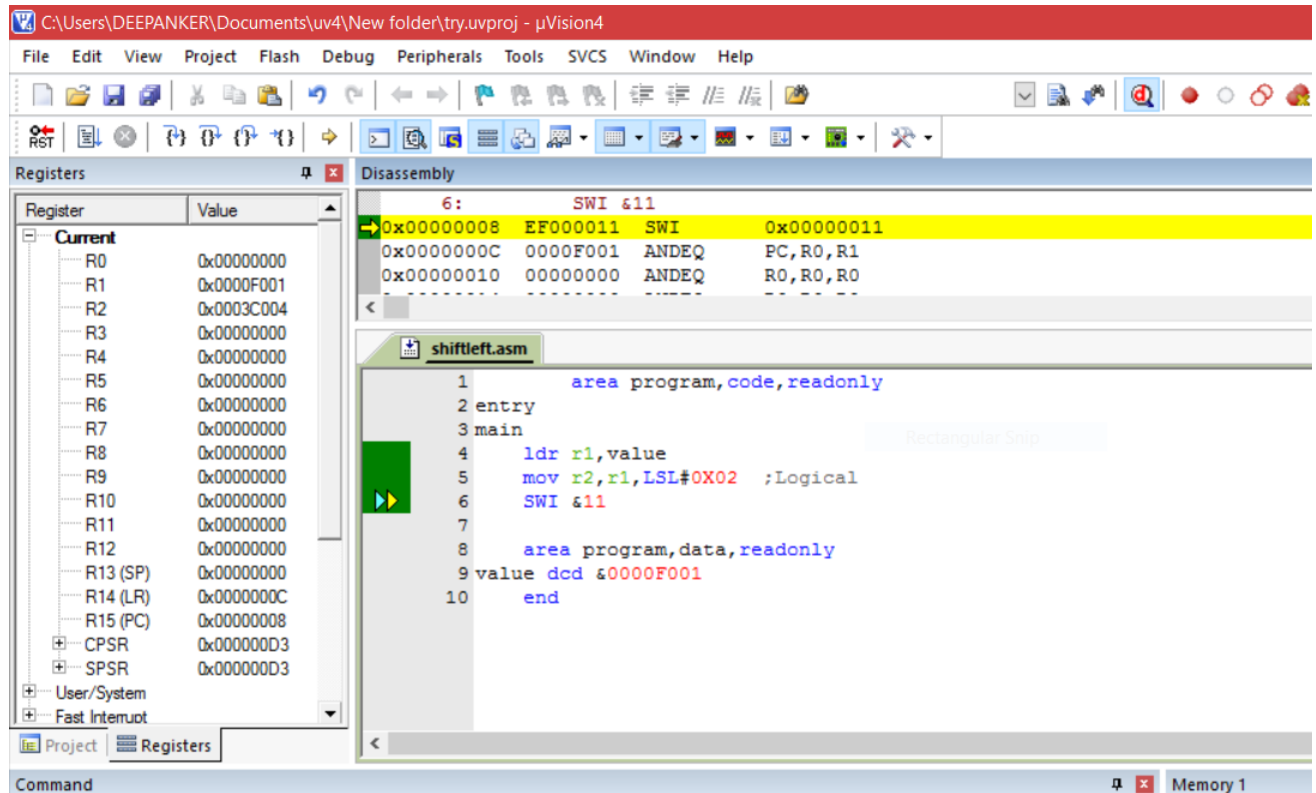
entry

main

ldr r1,value

mov r1,r1,LSL#0X02 ;Logical

SWI &11



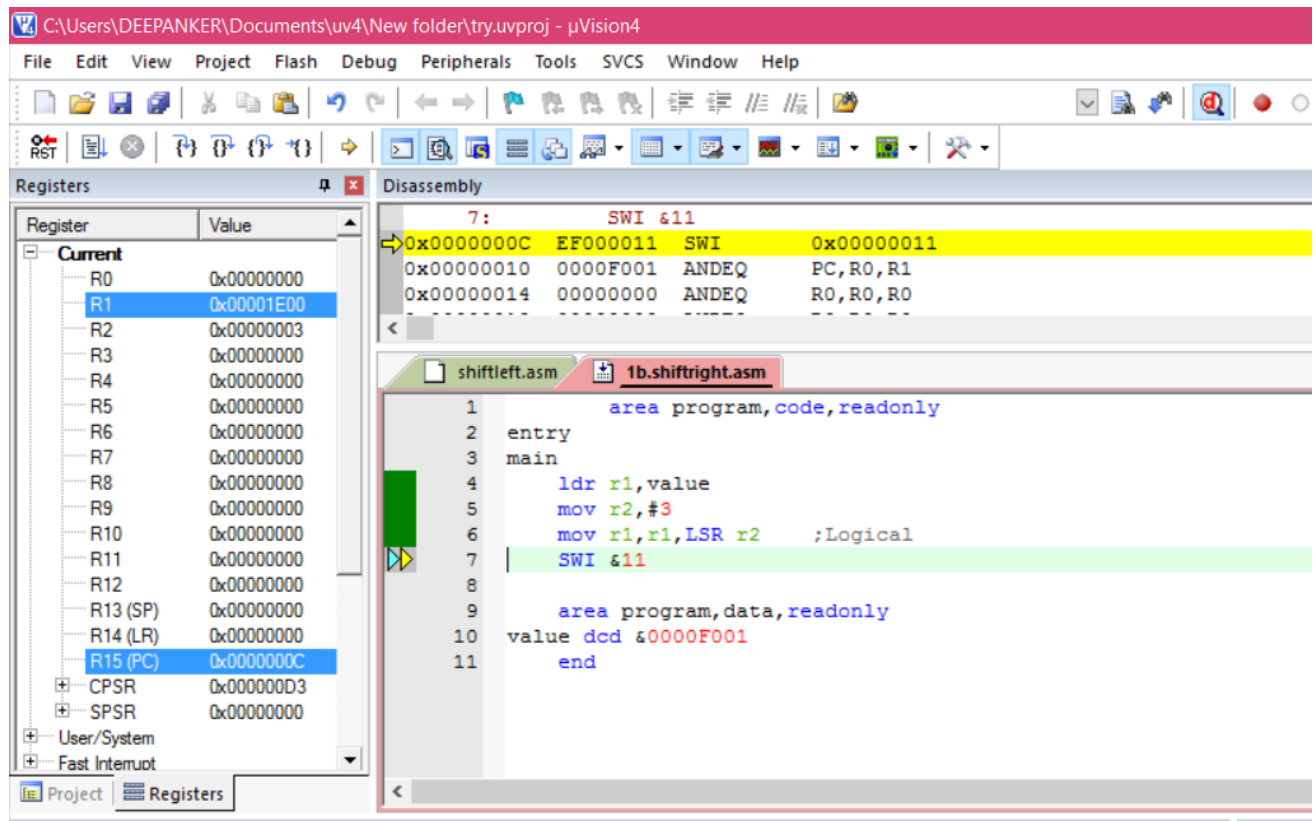
B) Shift right by the number of bits stored in register R2

area program,code,readonly

entry
main

```
ldr r1,value  
mov r2,#3  
mov r1,r1,LSR r2      ;Logical  
SWI &11
```

```
area program,data,readonly  
value dcd &0000F001  
end
```

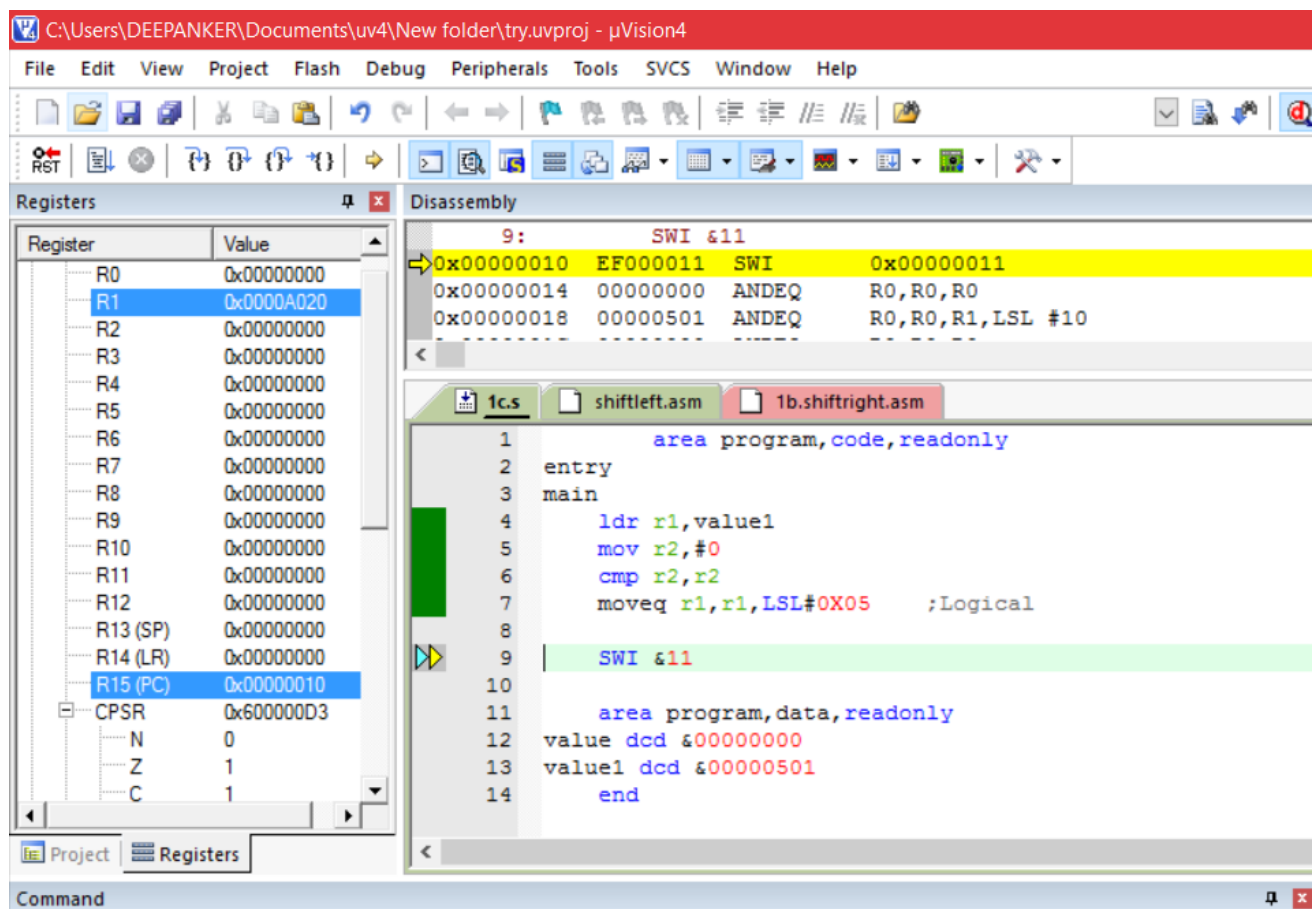


C) Shift left 5 bits conditionally when zero flag is set

```
                area program,code,readonly
entry
main
    ldr r1,value1
    mov r2,#0
    cmp r2,r2
    moveq r1,r1,LSL#0X05 ;Logical

    SWI &11

                area program,data,readonly
value dcd &00000000
value1 dcd &00000501
end
```



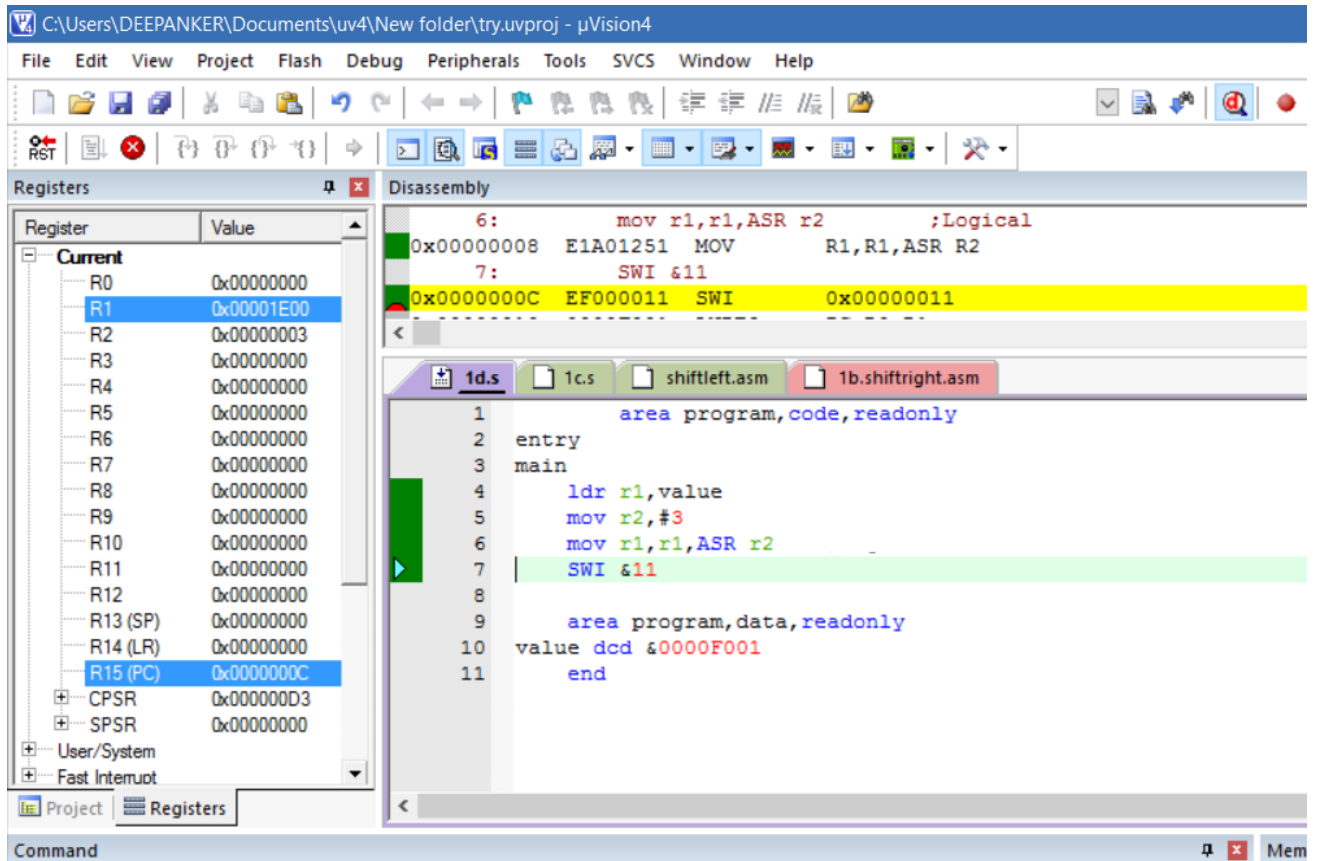
D) Arithmetic shift right by the value contained in register R2

area program,code,readonly

entry
main

```
ldr r1,value  
mov r2,#3  
mov r1,r1,ASR r2  
SWI &11
```

```
area program,data,readonly  
value dcd &0000F001  
end
```

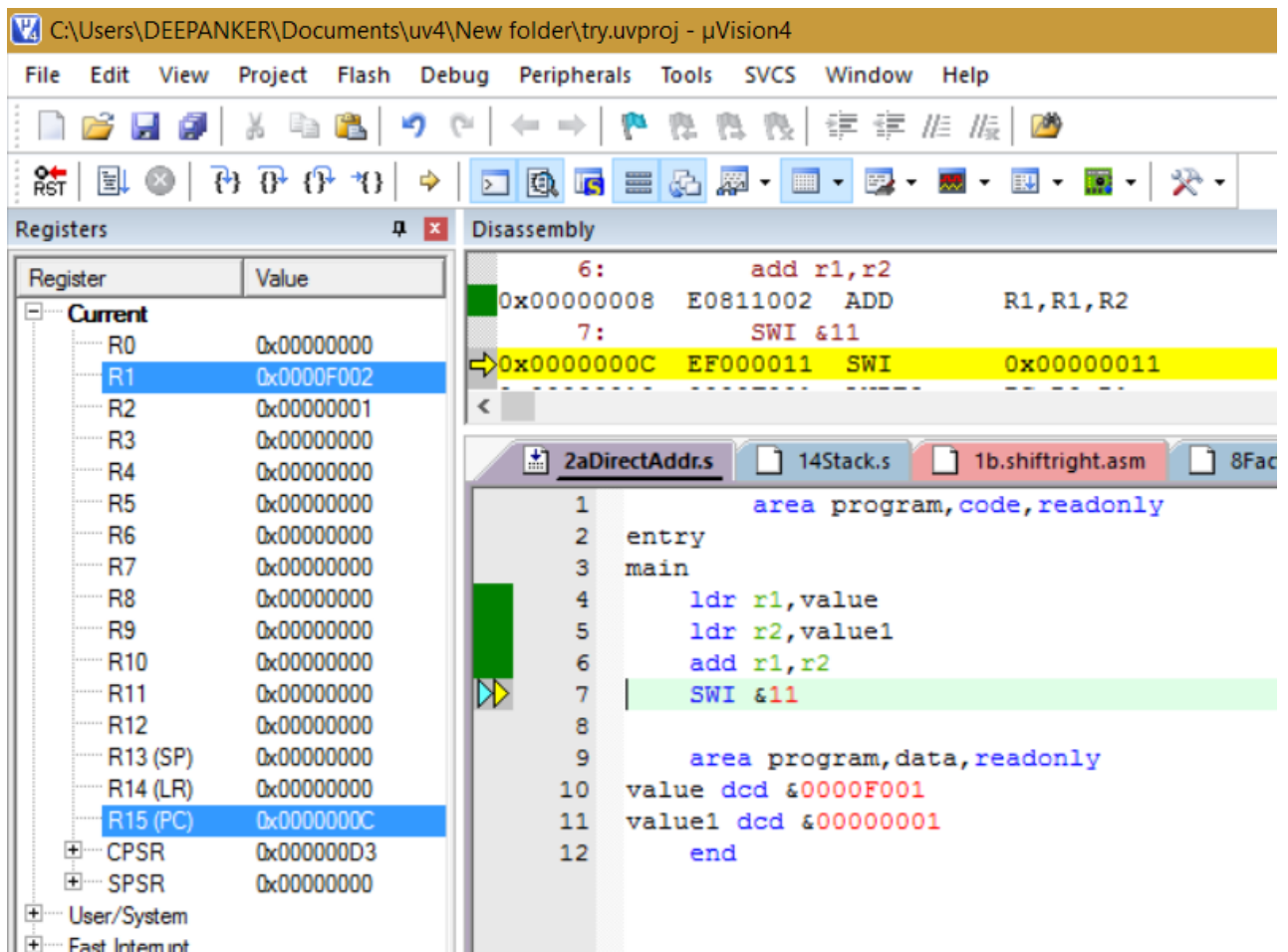


2. Write a program in ARM assembly language to add two 32 bit numbers using:

A) Direct addressing mode

```
                area program,code,readonly
entry
main
    ldr r1,value
    ldr r2,value1
    add r1,r2
    SWI &11

                area program,data,readonly
value dcd &0000F001
value1 dcd &00000001
end
```



B) Indirect addressing mode (2 consecutive values)

area program,code,readonly

entry

main

ldr r1,addr1 ;direct am

ldr r2,[r1]

ldr r3,[r1,#4]

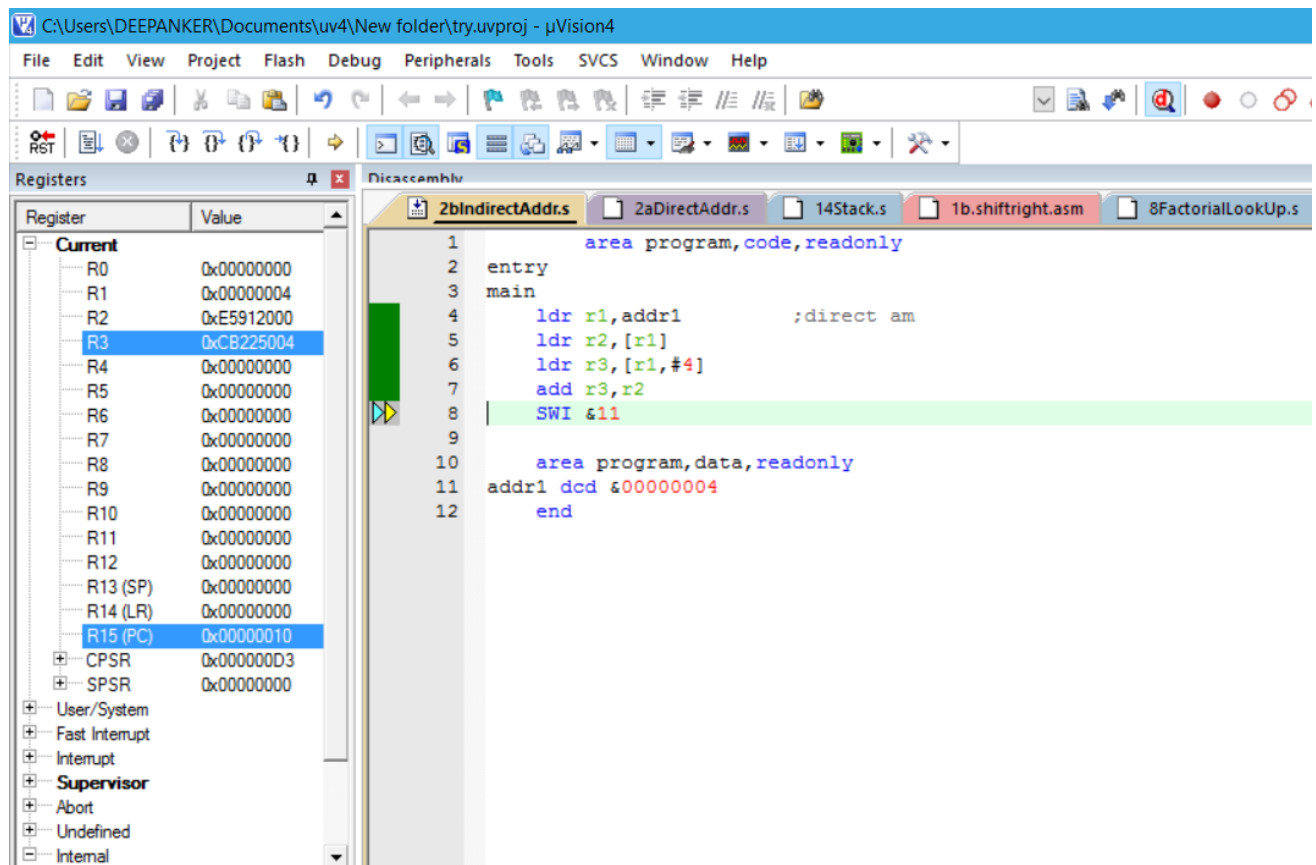
add r3,r2

SWI &11

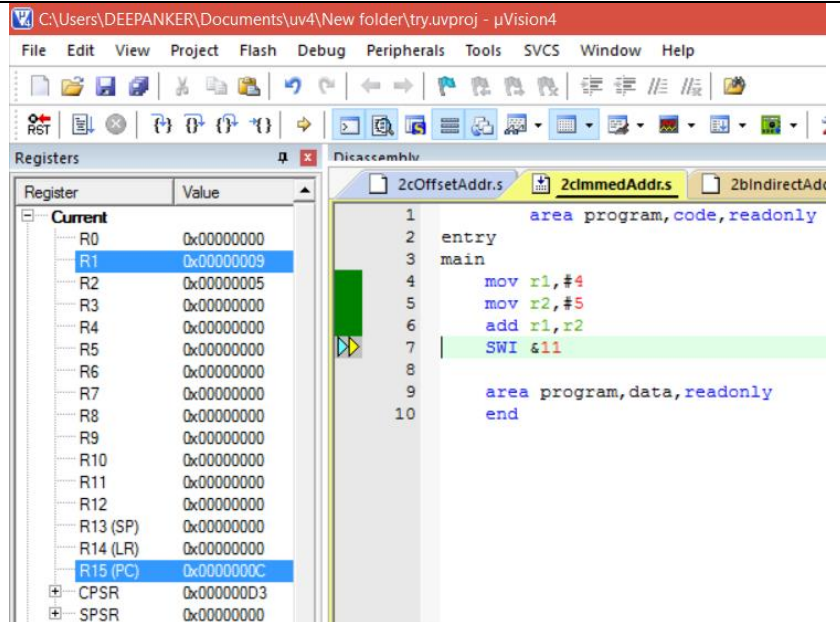
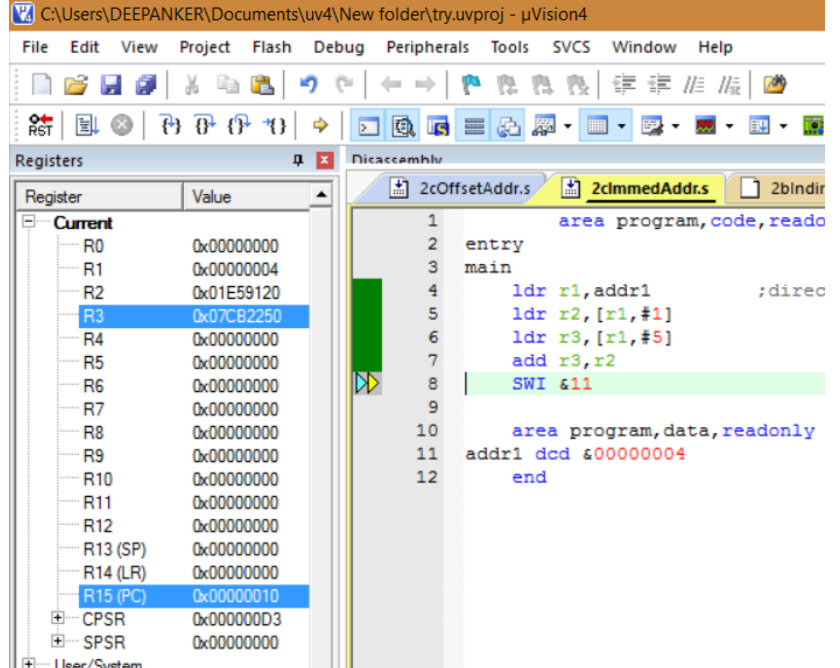
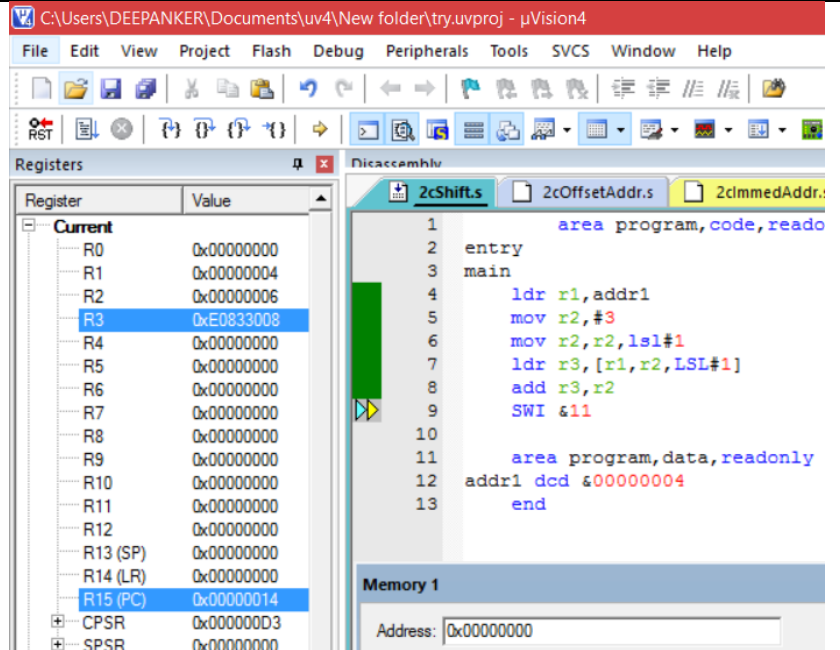
area program,data,readonly

addr1 dcd &00000004

end



C) Offset Addressing Mode, Immediate Addressing Mode, Barrel Shift.

<pre> entry main ;Immediate mov r1,#4 mov r2,#5 add r1,r2 SWI &11 area program,data,readonly end </pre>	
<pre> entry main ;Offset ldr r1,addr1 ldr r2,[r1,#1] ldr r3,[r1,#5] add r3,r2 SWI &11 area program,data,readonly addr1 dcd &00000004 end </pre>	
<pre> entry main ldr r1,addr1 mov r2,#3 mov r2,r2,ls1#1 ldr r3,[r1,r2,LSL#1] add r3,r2 SWI &11 area program,data,readonly addr1 dcd &00000004 end </pre>	

3. Write a program in ARM assembly language to copy consecutive words from source to destination in memory using:

a) Load and store instructions in a loop

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

```
ENTRY
```

```
main
```

```
LDR r0,value1
```

```
LDR r1,value2
```

```
MOV r2,#0x04
```

```
MOV r6,#0x00
```

```
loop
```

```
LDR r3,[r0]
```

```
STR r3,[r1]
```

```
ADD r0,r0,#0x04
```

```
ADD r1,r1,#0x04
```

```
ADD r6,r6,#0x01
```

```
cmp r6,r2
```

```
BNE loop
```

```
end
```

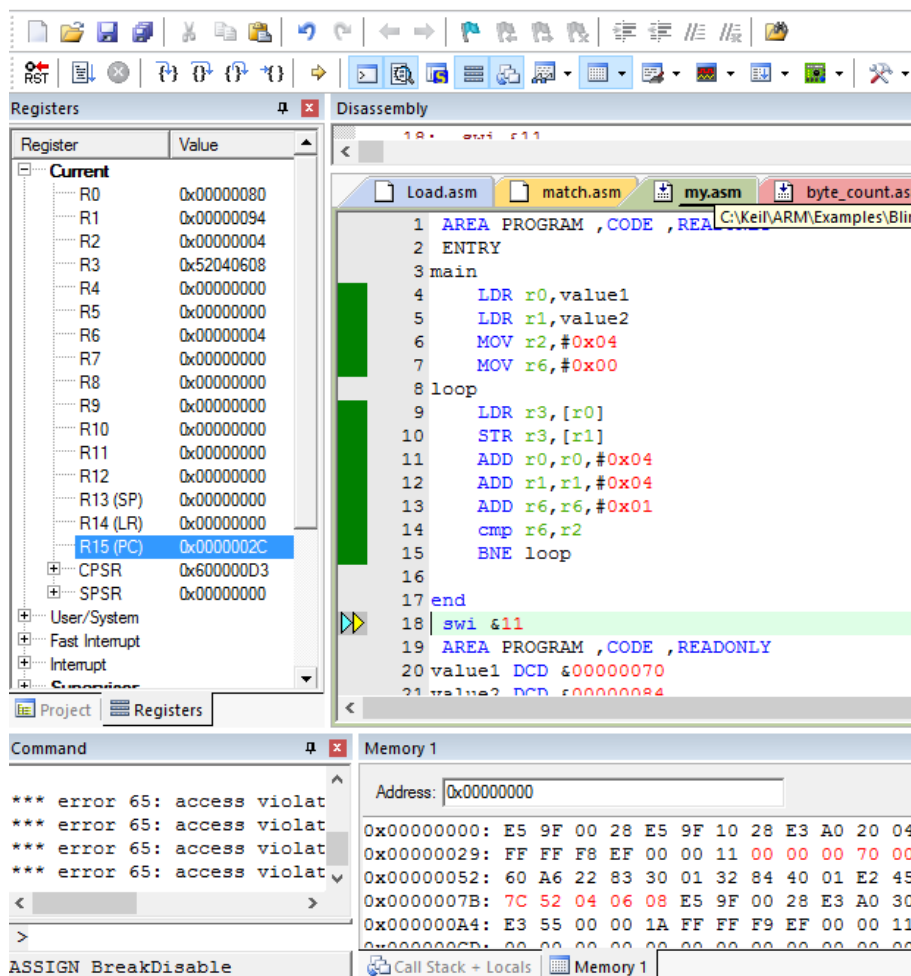
```
swi &11
```

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

```
value1 DCD &00000070
```

```
value2 DCD &00000084
```

```
END
```



4. Write a program to verify how many bytes are present in a given set which resemble 0xAC.

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

```
ENTRY
```

```
main
```

```
    LDR R0, =value
```

```
    MOV R1, #2                ; Size of the array
```

```
    MOV R2, #4                ; No of bytes in one cell of array. Every memory word is of 32 bits, so 4 bytes
```

```
    MUL R3, R2, R1
```

```
    MOV R4, #0
```

```
loop
```

```
    LDRB R5, [R0], #1
```

```
    CMP R5, #0xAC
```

```
    ADDEQ R4, #1
```

```
    SUB R3, #1
```

```
    CMP R3, #0
```

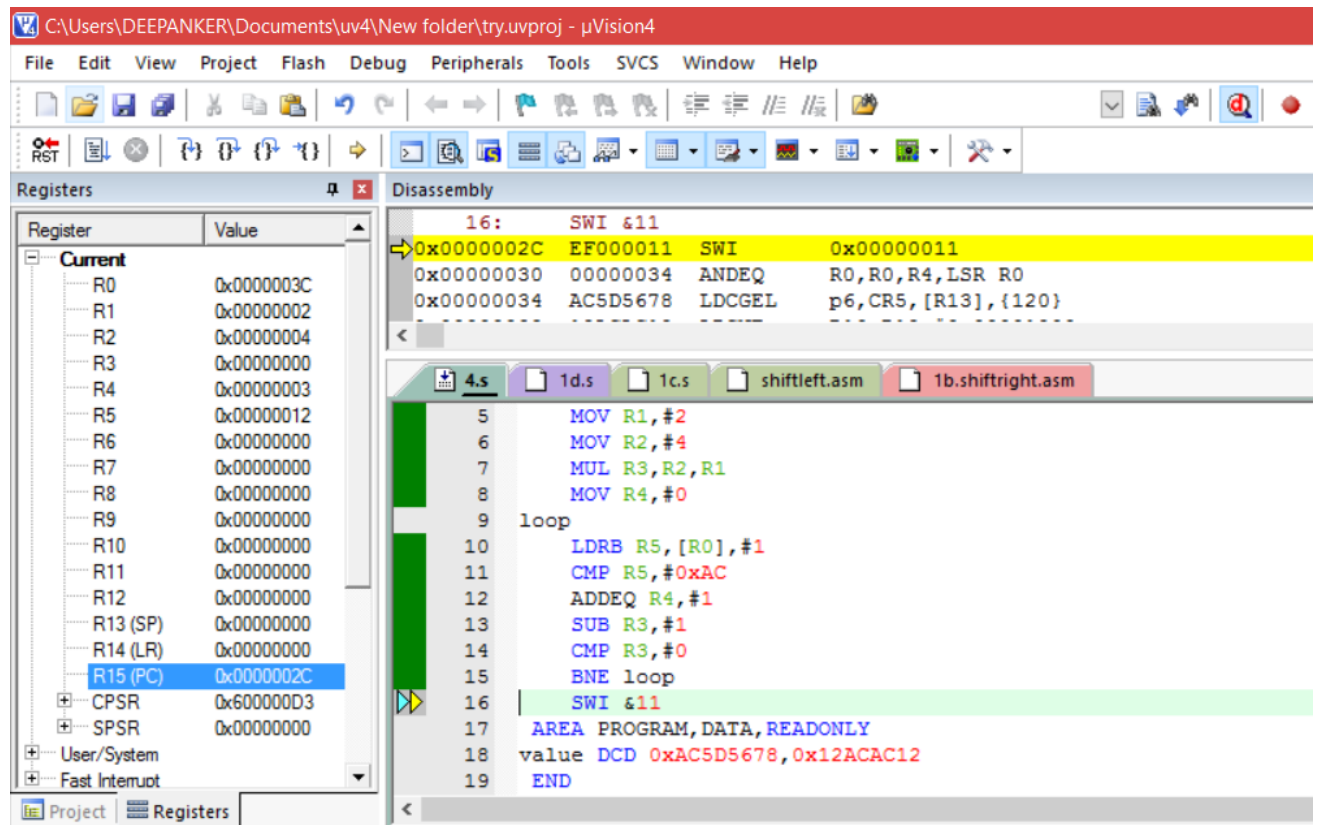
```
    BNE loop
```

```
    SWI &11
```

```
AREA PROGRAM, DATA, READONLY
```

```
value DCD 0xAC5D5678, 0x12ACAC12
```

```
END
```



5. Write a program to count the number of 1's and 0's in a given bytes and verify the result.

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

```
ENTRY
```

```
main
```

```
    LDRB R0, value
```

```
    MOV R1, #01
```

```
    MOV R3, #8
```

```
loop
```

```
    AND R2, R0, R1
```

```
    CMP R2, #1
```

```
    ADDEQ R7, #1           ;Count of 1s in R7
```

```
    CMP R2, #0
```

```
    ADDEQ R6, #1           ;Count of 0s in R6
```

```
    MOV R0, R0, LSR #1
```

```
    SUB R3, #1
```

```
    CMP R3, #0
```

```
    BNE loop
```

```
    SWI &11
```

```
AREA PROGRAM, DATA, READONLY
```

```
value dcb 0xA7
```

```
END
```

The screenshot displays the Keil uVision IDE interface. On the left, the 'Registers' window shows the current state of registers, with R15 (PC) highlighted at address 0x00000030. The main window is split into 'Disassembly' and 'Source Code' panes. The 'Disassembly' pane shows the instruction SWI &11 at address 0x00000030. The 'Source Code' pane shows the assembly program being assembled, which matches the code provided in the text above.

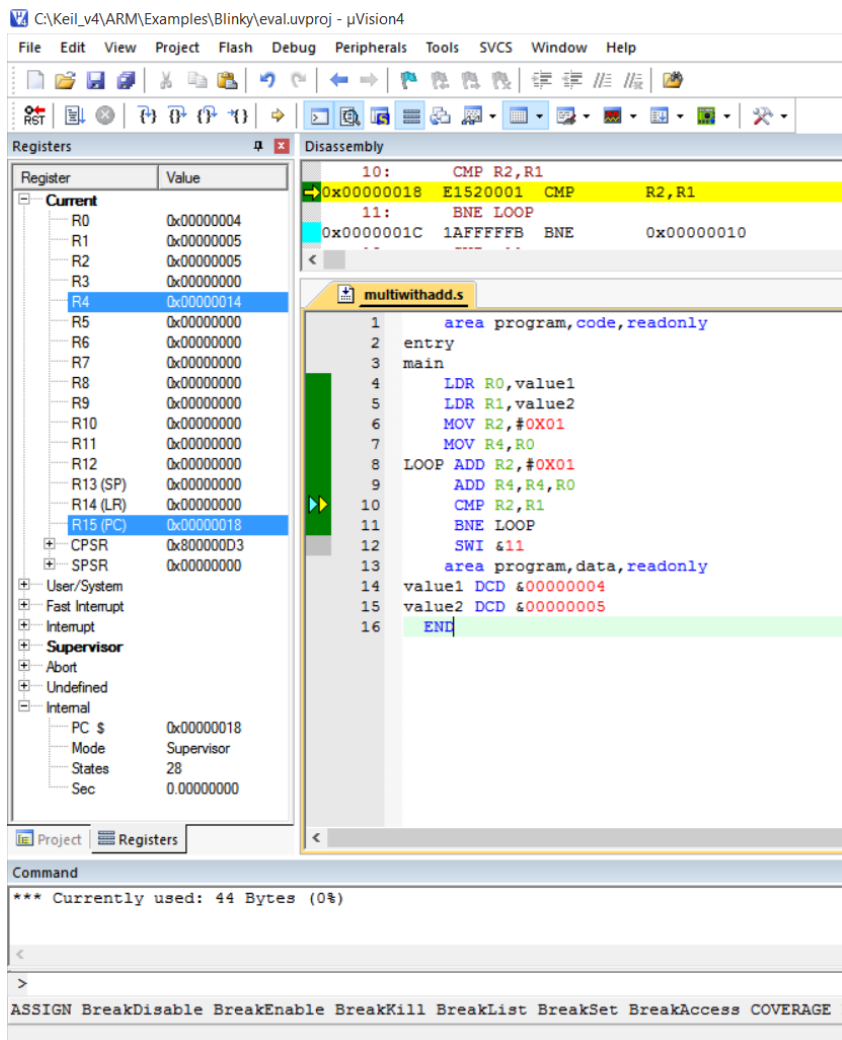
```

1  AREA PROGRAM, CODE, READONLY
2  ENTRY
3  main
4      LDRB R0, value
5      MOV R1, #01
6      MOV R3, #8
7  loop
8      AND R2, R0, R1
9      CMP R2, #1
10     ADDEQ R7, #1           ;Count of 1s in R7
11     CMP R2, #0
12     ADDEQ R6, #1           ;Count of 0s in R6
13     MOV R0, R0, LSR #1
14     SUB R3, #1
15     CMP R3, #0
16     BNE loop
17     SWI &11
18  AREA PROGRAM, DATA, READONLY
19  value dcb 0xA7
20  END
    
```

6. Write a program in ARM assembly language to perform multiplication using repeated addition.

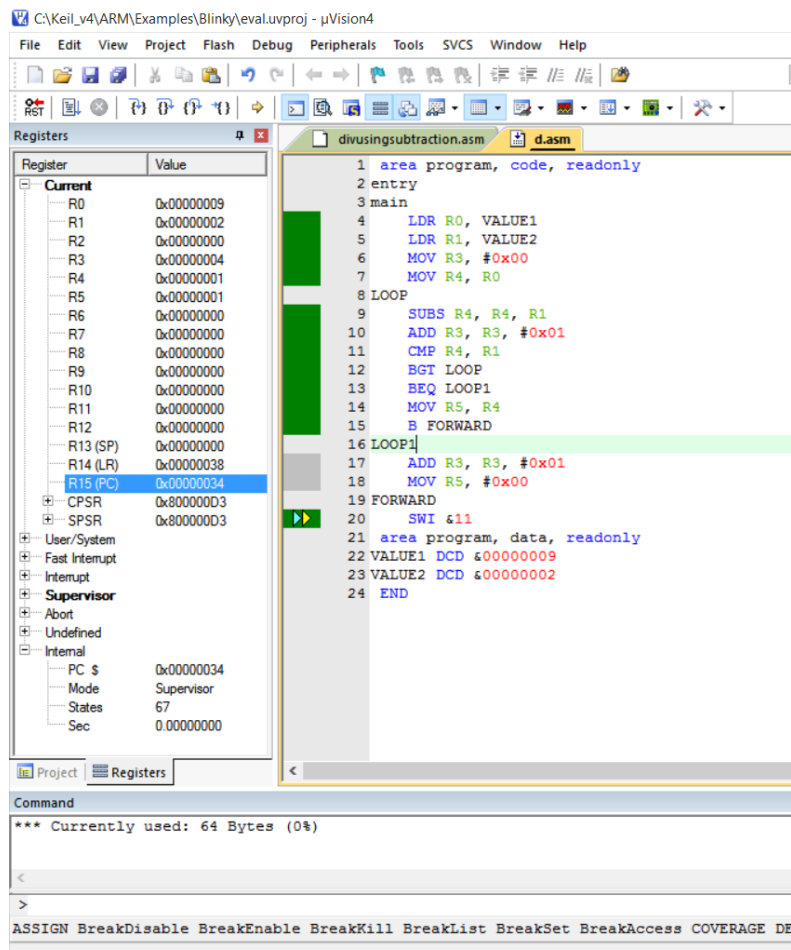
```
                area program,code,readonly
entry
main
    LDR R0,value1
    LDR R1,value2
    MOV R2,#0X01
    MOV R4,R0
LOOP ADD R2,#0X01
    ADD R4,R4,R0
    CMP R2,R1
    BNE LOOP
    SWI &11

                area program,data,readonly
value1 DCD &00000004
value2 DCD &00000005
END
```



7. Write a program in ARM assembly language to perform Division using repeated subtraction.

```
area program, code, readonly
entry
main
    LDR R0, VALUE1
    LDR R1, VALUE2
    MOV R3, #0x00
    MOV R4, R0
LOOP
    SUBS R4, R4, R1
    ADD R3, R3, #0x01
    CMP R4, R1
    BGT LOOP
    BEQ LOOP1
    MOV R5, R4
    B FORWARD
LOOP1
    ADD R3, R3, #0x01
    MOV R5, #0x00
FORWARD
    SWI &11
area program, data, readonly
VALUE1 DCD &00000009
VALUE2 DCD &00000002
END
```



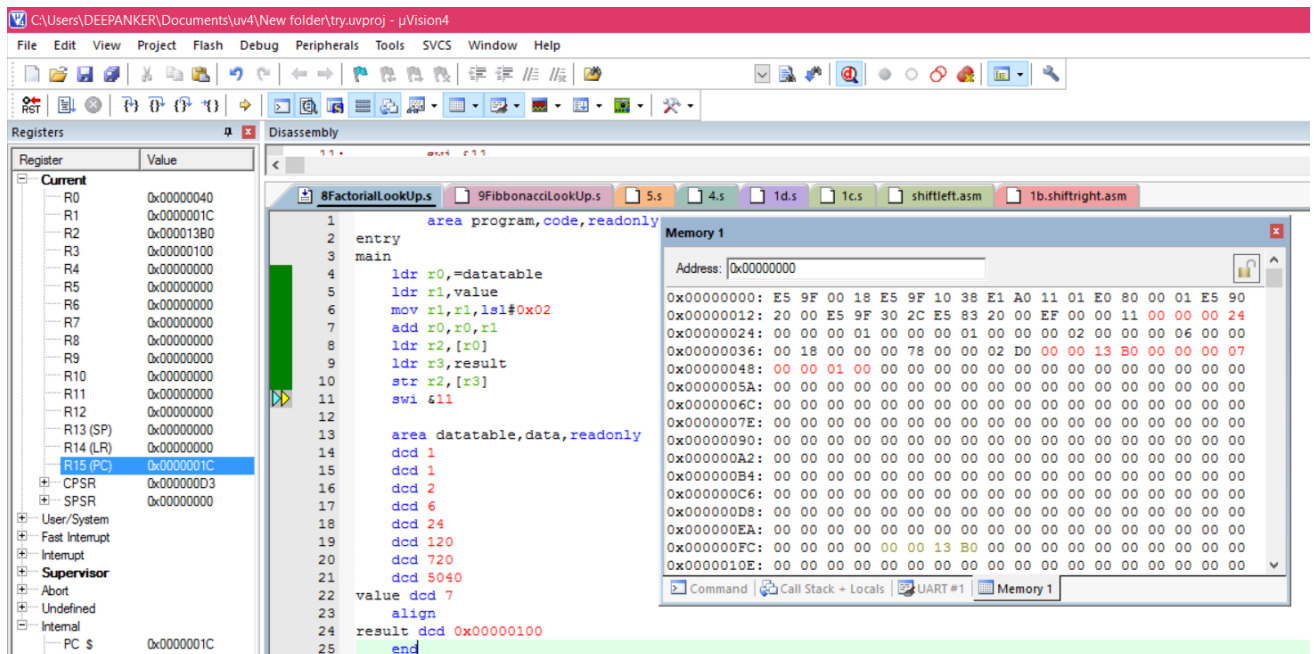
8. Write a program in ARM assembly language to find the factorial of a number using lookup table.

```

                area program,code,readonly
entry
main
    ldr r0,=datatable
    ldr r1,value
    mov r1,r1,lsl#0x02
    add r0,r0,r1
    ldr r2,[r0]
    ldr r3,result
    str r2,[r3]
    swi &11

    area datatable,data,readonly
    dcd 1
    dcd 1
    dcd 2
    dcd 6
    dcd 24
    dcd 120
    dcd 720
    dcd 5040
value dcd 7
    align
result dcd 0x00000100
end

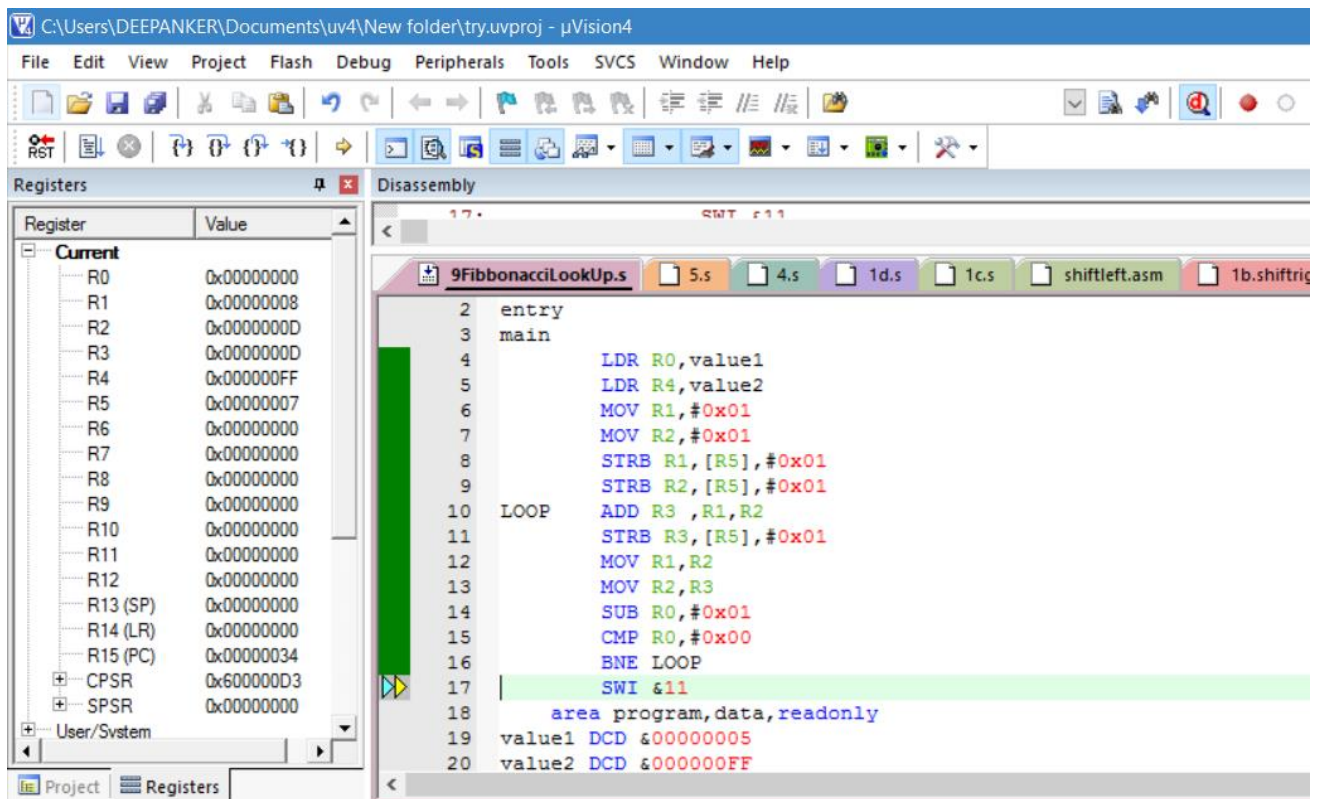
```



9. Write a program in ARM assembly language to find the Fibonacci of a number using lookup table.

```
                area program,code,readonly
entry
main
                LDR R0,value1
                LDR R4,value2
                MOV R1,#0x01
                MOV R2,#0x01
                STRB R1,[R5],#0x01
                STRB R2,[R5],#0x01
LOOP  ADD R3 ,R1,R2
                STRB R3,[R5],#0x01
                MOV R1,R2
                MOV R2,R3
                SUB R0,#0x01
                CMP R0,#0x00
                BNE LOOP
                SWI &11

                area program,data,readonly
value1 DCD &00000005
value2 DCD &000000FF
END
```



10. Write a program in ARM assembly language to find the number of occurrences of a letter in a given string.

```

area program ,code ,readonly
entry
main
    LDR r0,value1
    MOV r2,#0x00
    MOV r3,'#L'

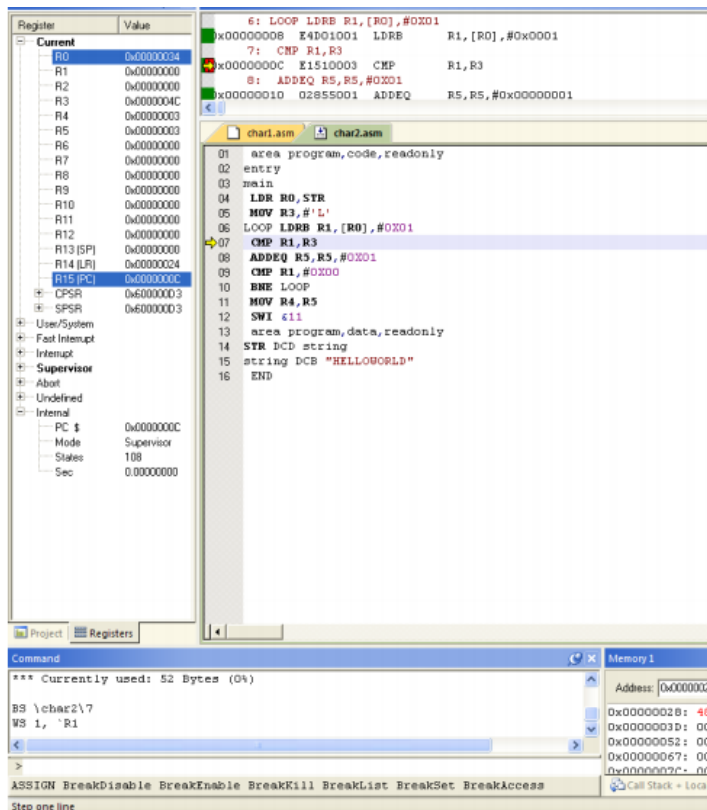
loop
    LDRB r1,[r0],#0x01
    cmp r1,r3
    ADDEQ r2,r2,#0x01
    BNE loop

end

swi &11

area program ,data ,readonly
value1 DCD value2
value2 DCB "HELLOWORLD"
end

```



11. Write a program in ARM assembly language to implement the equation:

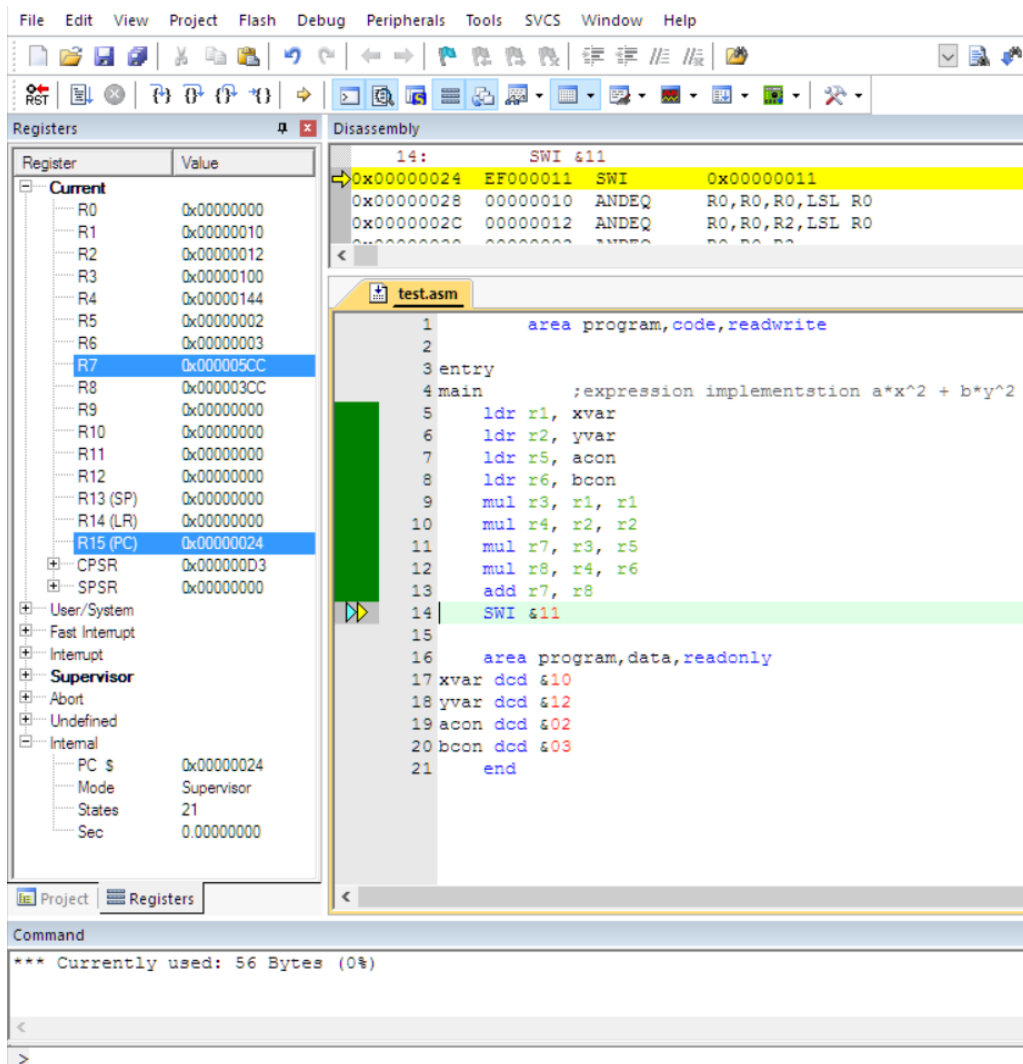
A) $ax^2 + by^2$

```

                area program,code,readwrite
entry
main            ;expression implementation a*x^2 + b*y^2
    ldr r1, xvar
    ldr r2, yvar
    ldr r5, acon
    ldr r6, bcon
    mul r3, r1, r1
    mul r4, r2, r2
    mul r7, r3, r5
    mul r8, r4, r6
    add r7, r8
    SWI &11

                area program,data,readonly
xvar dcd &10
yvar dcd &12
acon dcd &02
bcon dcd &03
end

```



B) $6(x+y)+2z+4$

area program,code,readwrite

entry

main ;expression implementstion $6(x+y)+2z+4$

```
ldr r1, xvar
ldr r2, yvar
ldr r3, zvar
mov r11, #6
mov r12, #2
mov r13, #4
add r4, r1, r2
mul r5, r4, r11
mul r6, r3, r12
add r7, r5, r6
add r8, r7, r13
SWI &11
```

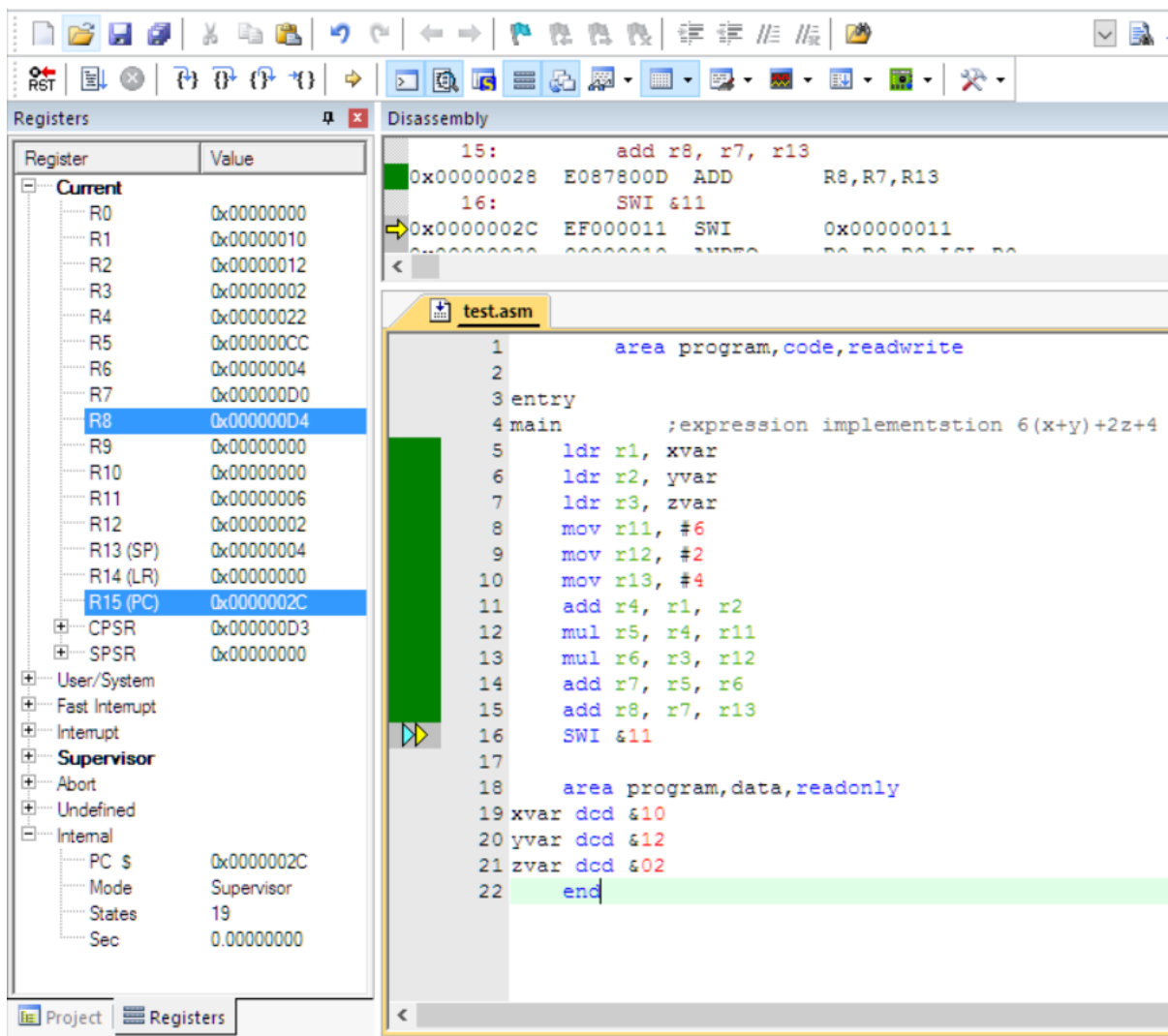
area program,data,readonly

xvar dcd &10

yvar dcd &12

zvar dcd &02

end



12. Write a program in ARM assembly language to find the length of a given string

```
area program,code,readonly
```

```
entry
```

```
main
```

```
LDR R0,STR
```

```
LOOP
```

```
LDRB R1,[R0],#0x01
```

```
CMP R1,#0x00
```

```
ADDNE R2,R2,#0x01
```

```
BNE LOOP
```

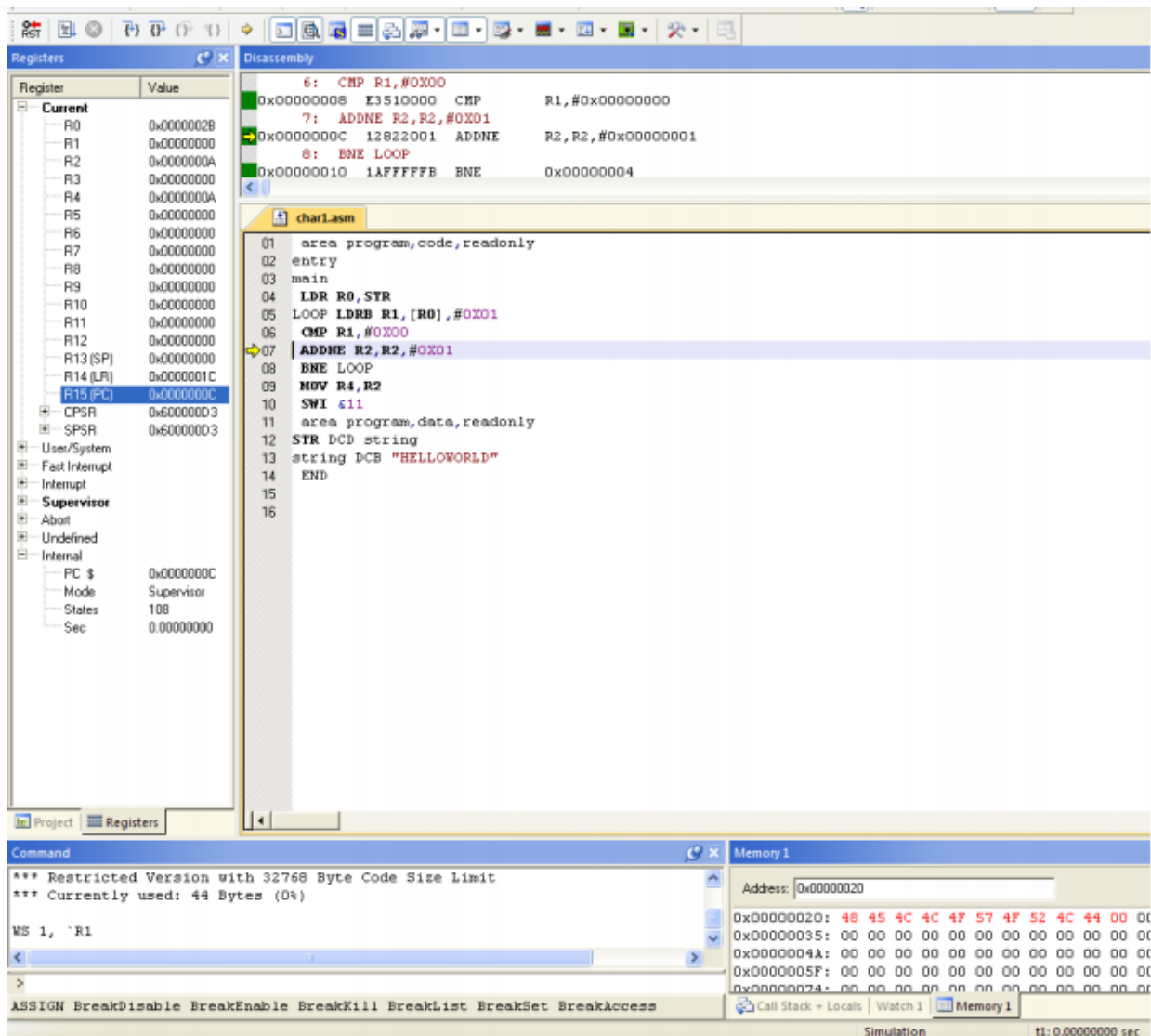
```
SWI &11
```

```
area program, data, readonly
```

```
STR DCD string
```

```
string DCB "HELLOWORLD"
```

```
END
```



13. Write a program in ARM assembly language to construct STACK.

```

        area program,code,readonly
entry
main    ;stack implementation

        ldr r1, value1
        ldr r2, value2
        ldr r3, value3
        ldr r4, value4
        ldr r7, =data2
        stmfd r7, {r1-r4}
        SWI &11

        area stack1,data

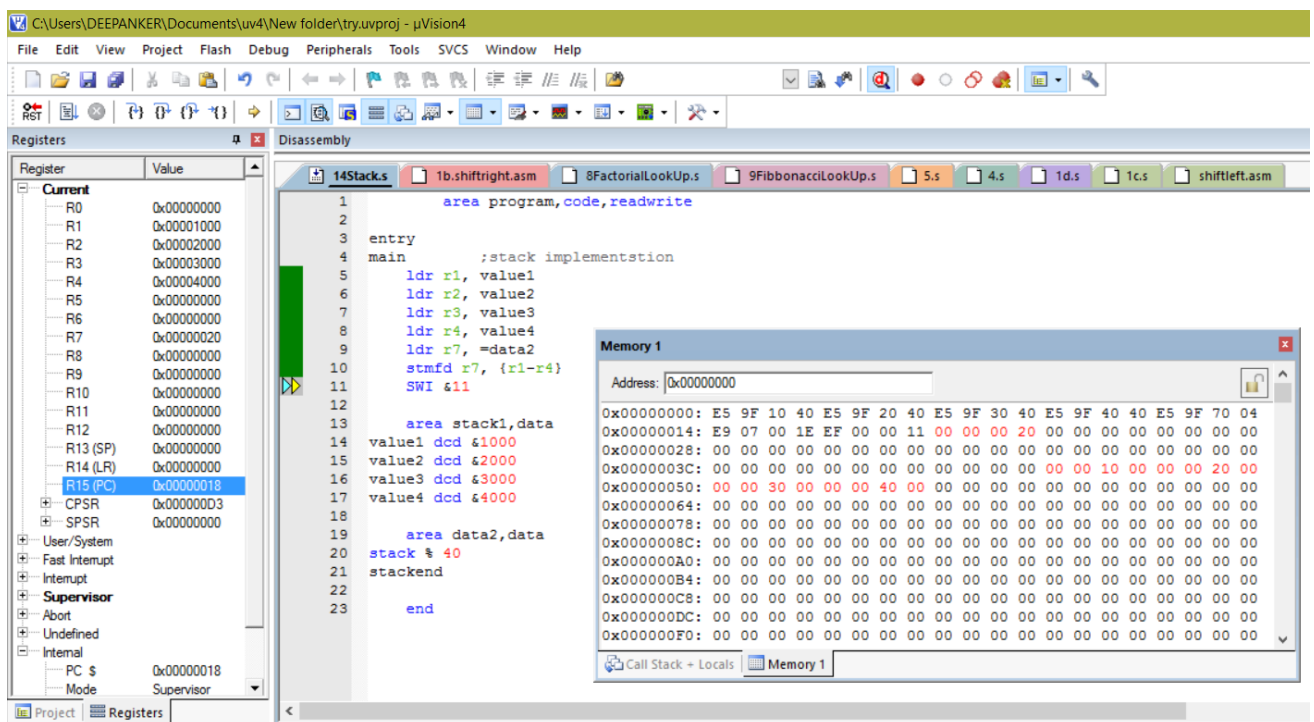
value1 dcd &1000
value2 dcd &2000
value3 dcd &3000
value4 dcd &4000

        area data2,data

stack % 40
stackend

        end

```



14. Write a program in ARM assembly language to add two 64 bit registers.

```

                area program,code,readwrite

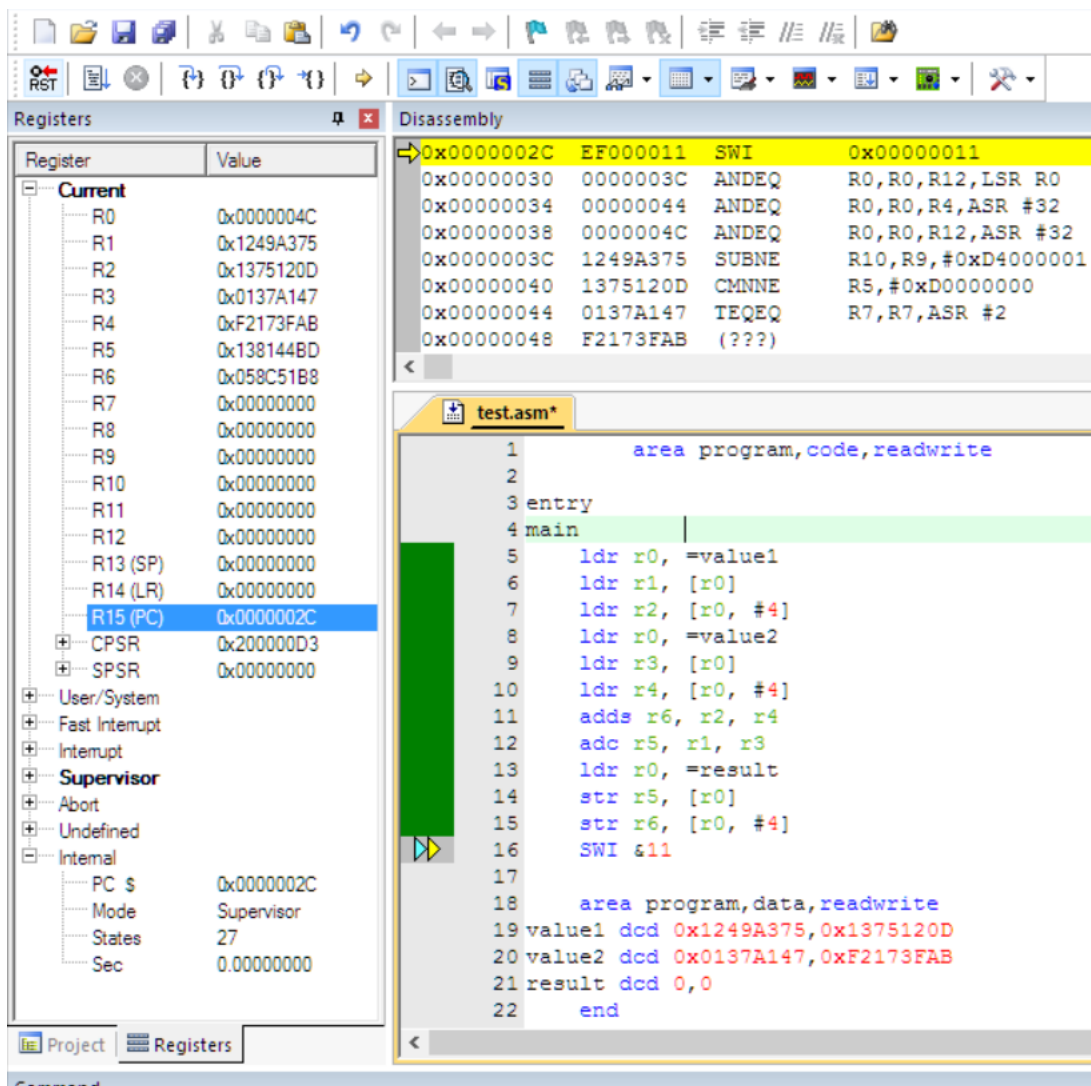
entry
main
    ldr r0, =value1
    ldr r1, [r0]
    ldr r2, [r0, #4]
    ldr r0, =value2
    ldr r3, [r0]
    ldr r4, [r0, #4]
    adds r6, r2, r4
    adc r5, r1, r3
    ldr r0, =result
    str r5, [r0]
    str r6, [r0, #4]
    SWI &11

```

```

                area program,data,readwrite
value1 dcd 0x1249A375,0x1375120D
value2 dcd 0x0137A147,0xF2173FAB
result dcd 0,0
end

```



The screenshot displays the ARM assembly development environment. The **Registers** window on the left shows the current state of the registers. The **Disassembly** window on the right shows the assembly code being executed. The **test.asm*** window at the bottom shows the source code.

Register	Value
R0	0x0000004C
R1	0x1249A375
R2	0x1375120D
R3	0x0137A147
R4	0xF2173FAB
R5	0x138144BD
R6	0x058C51B8
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000002C
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x0000002C
Mode	Supervisor
States	27
Sec	0.00000000

Disassembly:

```

0x0000002C EF000011 SWI 0x00000011
0x00000030 0000003C ANDEQ R0,R0,R12,LSR R0
0x00000034 00000044 ANDEQ R0,R0,R4,ASR #32
0x00000038 0000004C ANDEQ R0,R0,R12,ASR #32
0x0000003C 1249A375 SUBNE R10,R9,#0xD4000001
0x00000040 1375120D CMNNE R5,#0xD0000000
0x00000044 0137A147 TEQEQ R7,R7,ASR #2
0x00000048 F2173FAB (???)

```

test.asm*:

```

1      area program,code,readwrite
2
3  entry
4  main
5      ldr r0, =value1
6      ldr r1, [r0]
7      ldr r2, [r0, #4]
8      ldr r0, =value2
9      ldr r3, [r0]
10     ldr r4, [r0, #4]
11     adds r6, r2, r4
12     adc r5, r1, r3
13     ldr r0, =result
14     str r5, [r0]
15     str r6, [r0, #4]
16     SWI &11
17
18     area program,data,readwrite
19 value1 dcd 0x1249A375,0x1375120D
20 value2 dcd 0x0137A147,0xF2173FAB
21 result dcd 0,0
22     end

```