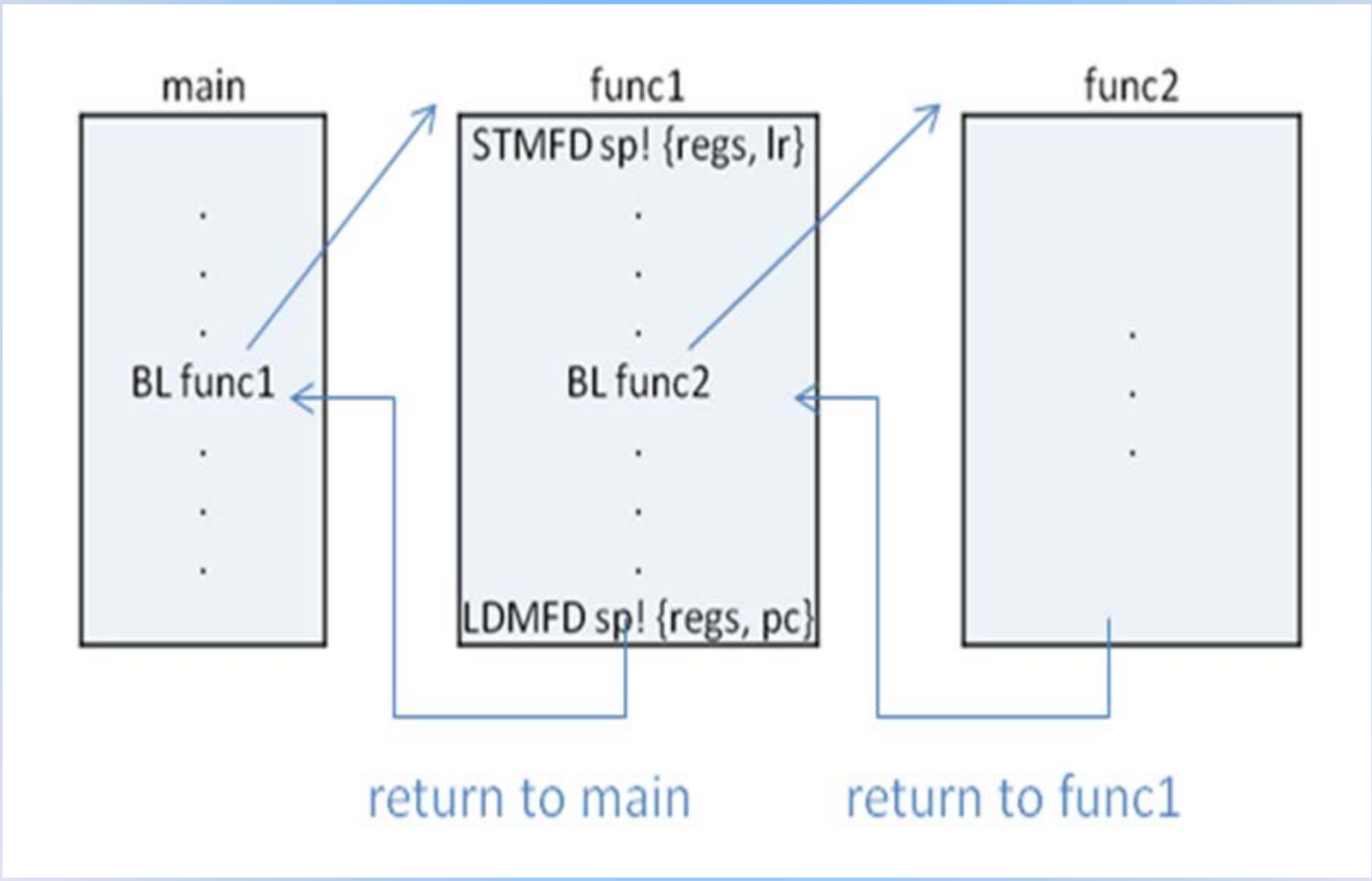
EE3002/IM2002

Microprocessors

Tutorial 9

1. Why is it that we may get into an infinite loop when one subroutine calls another subroutine?

- Subroutines can call other subroutines
- Next figure shows two subroutines, func1 and func2



- If not careful we can get stuck in an infinite loop
- If func1 does not save the value of the link register (lr) at the start, lr will be overwritten when BL func2 is executed (call to func2) within func1
- Upon return from func2 to func1, value in lr is used as return address (address after the BL func2 instruction)
- When func1 is finished and wants to return back to main, it cannot return properly!
- Instead it will move to the instruction after BL func2 – results in an infinite loop
- Solve easily by stacking the link register lr at the start of subroutine func1

2. Write ARM subroutines in assembly to implement the following stack operations without using LDM or STM instructions.

   Assume that the stack is a Full Descending (FD) stack.

   a. myPUSH ;push content of register r0
      ;to stack
   b. myPOP ;pop content of stack to
      ;register r0

a. myPUSH

```
SUB r13, r13, #4        ;decrement sp (r13)
STR r0, [r13]           ;stores a word to stack
BX   lr                 ;return to calling program
```

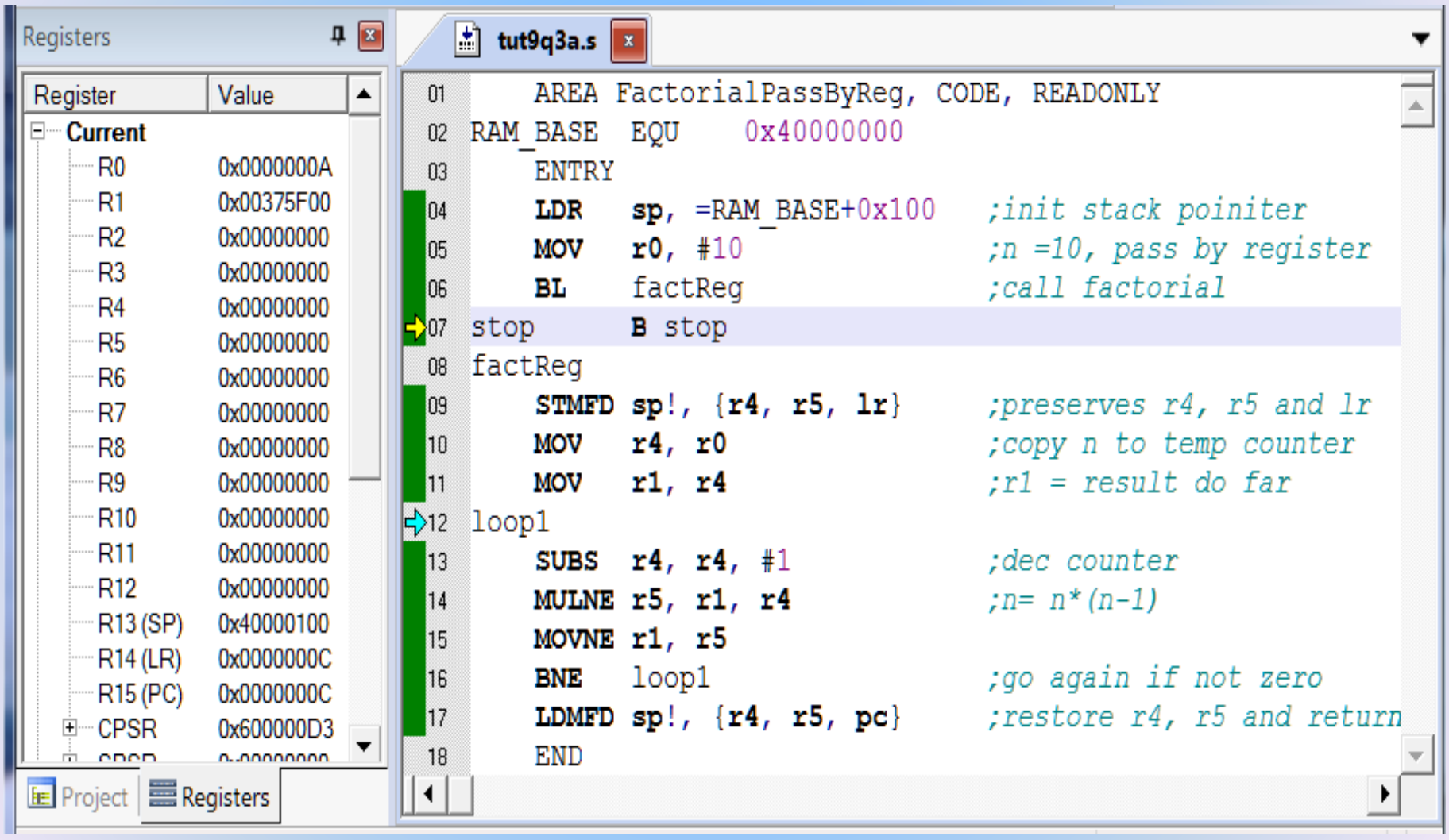
b. myPOP

```
LDR r0, [r13]           ;loads a word from stack
ADD r13, r13, #4        ;increment sp (r13)
BX    lr                ;return to calling program
```

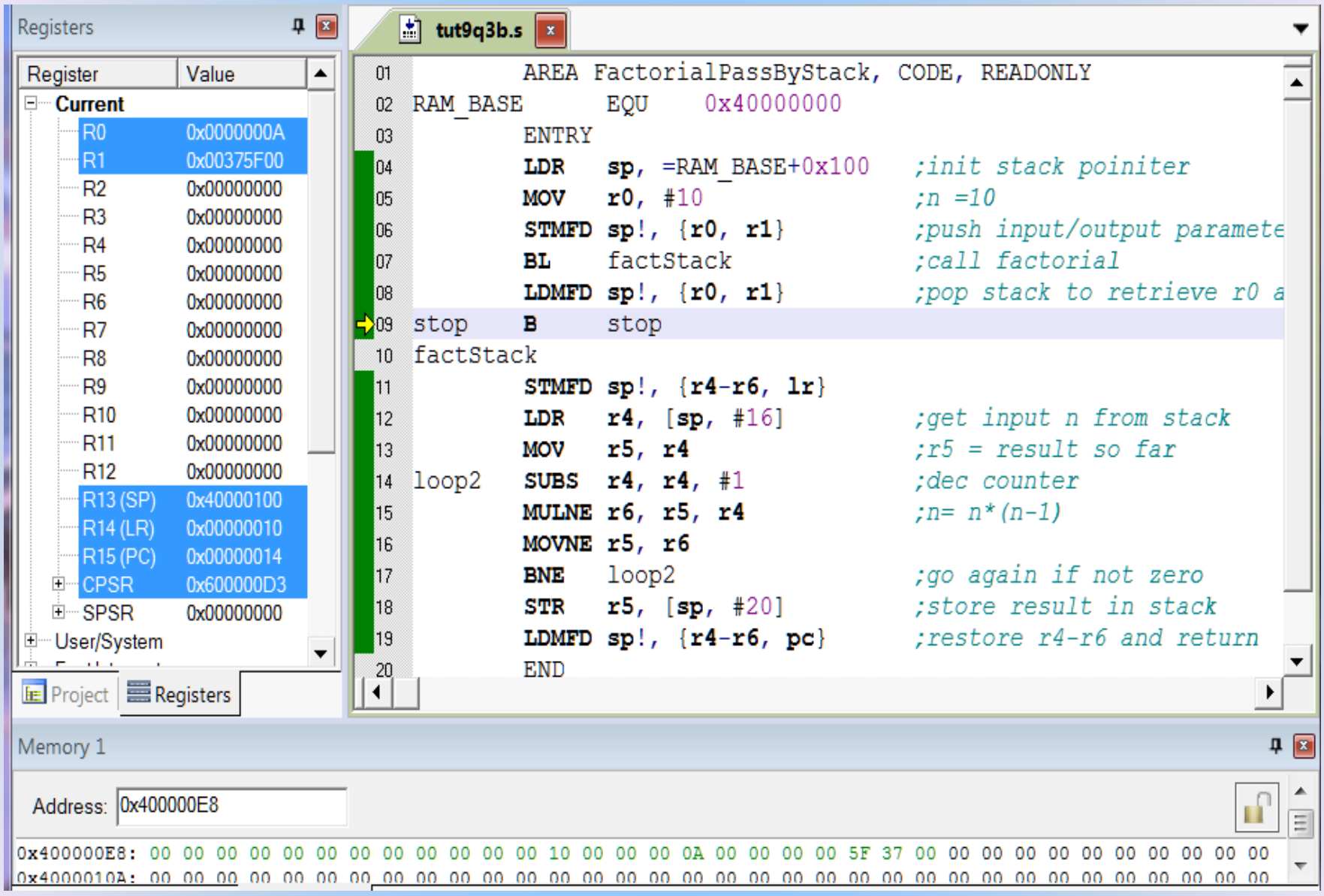
3. Write an ARM subroutine in assembly to compute factorial of n, using a full descending stack, for each of the following methods of passing parameters. Assume that register r0 contains n (=10) and the result will be stored in register r1.

   a. Passing parameters to/from the subroutine using pass-by-register.
   b. Passing parameters to/from the subroutine using pass-by-stack.

```
a)        AREA FactorialPassByReg, CODE, READONLY
RAM_BASE     EQU      0x40000000
          ENTRY
          LDR      sp, =RAM_BASE+0x100          ;init stack poiniter
          MOV     r0,  #10                      ;n =10, pass by reg
          BL  factReg                           ;call factorial
stop      B     stop
factReg
          STMFD  sp!, {r4, r5, lr}          ;preserves r4, r5 and lr
          MOV       r4,  r0                 ;copy n to temp counter
          MOV       r1,  r4                 ;r1 = result do far
loop1     SUBS      r4,  r4, #1             ;dec counter
          MULNE    r5,  r1, r4              ;n= n*(n-1)
          MOVNE   r1,  r5
          BNE       loop1                   ;go again if not zero
          LDMFD    sp!, {r4, r5, pc}        ;restore r4, r5 and return
          END
```


Registers
Register
Value
Current
R0 0x0000000A
R1 0x00375F00
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x40000100
R14 (LR) 0x0000000C
R15 (PC) 0x0000000C
CPSR 0x600000D3
Project
Registers
tut9q3a.s
01 AREA FactorialPassByReg, CODE, READONLY
02 RAM_BASE EQU 0x40000000
03 ENTRY
04 LDR sp, =RAM_BASE+0x100 ;init stack poiniter
05 MOV r0, #10 ;n =10, pass by register
06 BL factReg ;call factorial
07 stop B stop
08 factReg
09 STMFD sp!, {r4, r5, lr} ;preserves r4, r5 and lr
10 MOV r4, r0 ;copy n to temp counter
11 MOV r1, r4 ;r1 = result do far
12 loop1
13 SUBS r4, r4, #1 ;dec counter
14 MULNE r5, r1, r4 ;n= n*(n-1)
15 MOVNE r1, r5
16 BNE loop1 ;go again if not zero
17 LDMFD sp!, {r4, r5, pc} ;restore r4, r5 and return
18 END

```
b.          AREA FactorialPassByStack, CODE, READONLY
RAM_BASE    EQU     0x40000000
            ENTRY
            LDR     sp, =RAM_BASE+0x100   ;init stack pointer
            MOV     r0, #10               ;n =10
            STMFD sp!, {r0, r1}           ;push input/output parameters to stack
            BL      factStack             ;call factorial
            LDMFD sp!, {r0, r1}           ;pop stack to retrieve r0 and result
                                          ;in r1
stop        B   stop
factStack
            STMFD  sp!, {r4-r6, lr}
            LDR     r4, [sp, #16]         ;get input n from stack
            MOV     r5,  r4               ;r5 = result so far
loop2       SUBS    r4,  r4, #1           ;dec counter
            MULNE  r6,  r5, r4            ;n= n*(n-1)
            MOVNE r5,  r6
            BNE     loop2                 ;go again if not zero
            STR     r5, [sp, #20]         ;store result in stack
            LDMFD  sp!, {r4-r6, pc}       ;restore r4, r5 and return
            END
```


Registers
Register
Value
Current
R0 0x0000000A
R1 0x00375F00
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x40000100
R14 (LR) 0x00000010
R15 (PC) 0x00000014
CPSR 0x600000D3
SPSR 0x00000000
User/System
Project
Registers
tut9q3b.s
01 AREA FactorialPassByStack, CODE, READONLY
02 RAM_BASE EQU 0x40000000
03 ENTRY
04 LDR sp, =RAM_BASE+0x100 ;init stack poiniter
05 MOV r0, #10 ;n =10
06 STMFD sp!, {r0, r1} ;push input/output paramete
07 BL factStack ;call factorial
08 LDMFD sp!, {r0, r1} ;pop stack to retrieve r0 a
09 stop B stop
10 factStack
11 STMFD sp!, {r4-r6, lr}
12 LDR r4, [sp, #16] ;get input n from stack
13 MOV r5, r4 ;r5 = result so far
14 loop2 SUBS r4, r4, #1 ;dec counter
15 MULNE r6, r5, r4 ;n= n*(n-1)
16 MOVNE r5, r6
17 BNE loop2 ;go again if not zero
18 STR r5, [sp, #20] ;store result in stack
19 LDMFD sp!, {r4-r6, pc} ;restore r4-r6 and return
20 END
Memory 1
Address: 0x400000E8
0x400000E8: 00 00 00 00 00 00 00 00 00 00 00 00 10 00 00 00 0A 00 00 00 00 5F 37 00 00 00 00 00 00 00 00 00 00 00
0x4000010A: 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 00 00 00 00 00 00

4. Write an ARM assembly program to sum the square of 4 numbers
(sum = x1*x1 + x2*x2 + x3*x3 + x4*x4).

The main program should call a subroutine to add a list of number.

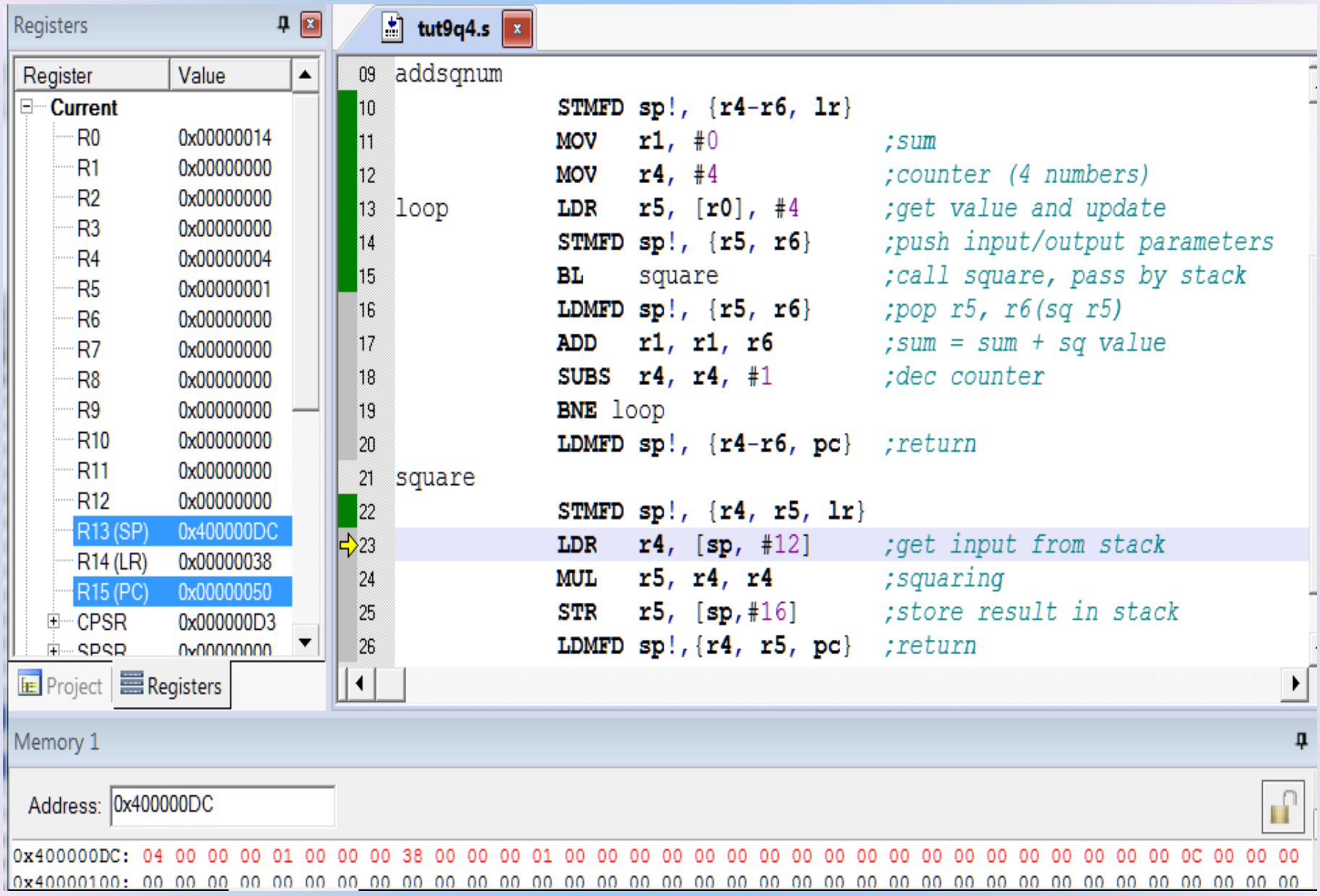
This subroutine would then call another subroutine to perform the square operation.

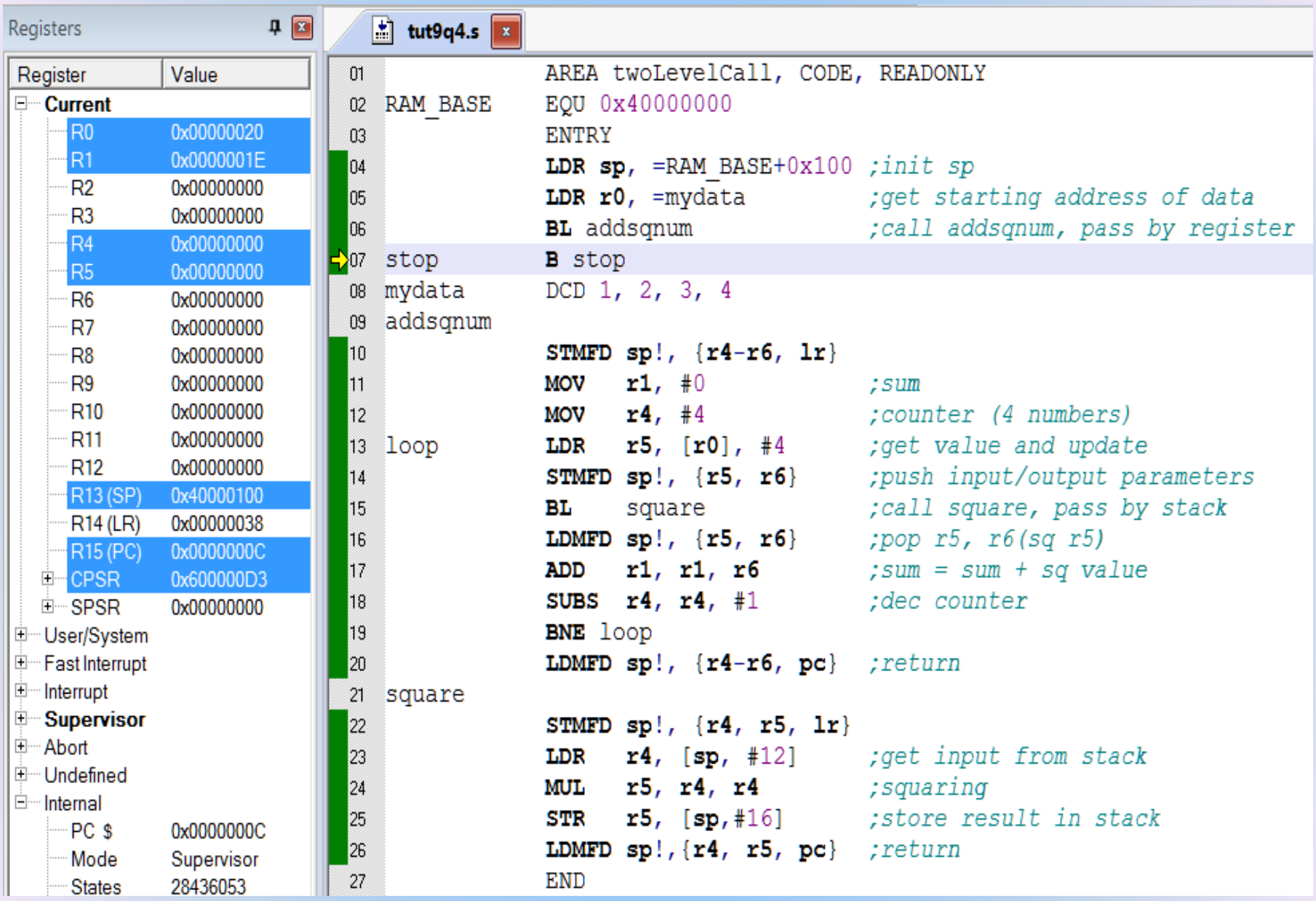
Test it out using Keil uVision and observe the content of registers sp, lr and pc during subroutine calls and returns.

Let register r0 be the pointer to the numbers and register r1 contains the final result.

```
STACK_BASE   EQU 0x40000000
             AREA   twoLevelCall,  CODE,  READONLY
             ENTRY

             LDR     sp, =STACK_BASE+0x100       ;init sp
             ADR     r0, mydata                  ;get starting addr of data

             BL      addsqnum                    ;call addsqnum
                                                 ;pass by register
stop         B stop

mydata
             DCD 1, 2, 3, 4
```

```
addsqnum
        STMFD sp!, {r4-r6, lr}
        MOV   r1, #0            ;sum
        MOV   r4, #4            ;counter (4 numbers)
loop    LDR   r5, [r0], #4      ;get value and update
        STMFD sp!, {r5, r6}     ;push input/output parameters
        BL    square            ;call square, pass by stack
        LDMFD sp!, {r5, r6}     ;pop r5, r6(sq r5)
        ADD   r1, r1, r6        ;sum = sum + sq value
        SUBS  r4, r4, #1        ;dec counter
        BNE   loop
        LDMFD sp!, {r4-r6, pc}          ;return
square
        STMFD sp!, {r4, r5, lr}
        LDR   r4, [sp, #12]             ;get input from stack
        MUL   r5, r4, r4                ;squaring
        STR   r5, [sp,#16]              ;store result in stack
        LDMFD sp!,{r4, r5, pc}          ;return
        END
```


Registers
Register
Value
Current
R0 0x00000014
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000004
R5 0x00000001
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x400000DC
R14 (LR) 0x00000038
R15 (PC) 0x00000050
CPSR 0x000000D3
Project
Registers
tut9q4.s
09 addsqnum
10 STMFD sp!, {r4-r6, lr}
11 MOV r1, #0 ;sum
12 MOV r4, #4 ;counter (4 numbers)
13 loop LDR r5, [r0], #4 ;get value and update
14 STMFD sp!, {r5, r6} ;push input/output parameters
15 BL square ;call square, pass by stack
16 LDMFD sp!, {r5, r6} ;pop r5, r6(sq r5)
17 ADD r1, r1, r6 ;sum = sum + sq value
18 SUBS r4, r4, #1 ;dec counter
19 BNE loop
20 LDMFD sp!, {r4-r6, pc} ;return
21 square
22 STMFD sp!, {r4, r5, lr}
23 LDR r4, [sp, #12] ;get input from stack
24 MUL r5, r4, r4 ;squaring
25 STR r5, [sp,#16] ;store result in stack
26 LDMFD sp!,{r4, r5, pc} ;return
Memory 1
Address: 0x400000DC
0x400000DC: 04 00 00 00 01 00 00 00 38 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0C 00 00 00
0x40000100: 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 00 00 00 00 00 00


Registers
Register
Value
Current
R0 0x00000020
R1 0x0000001E
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x40000100
R14 (LR) 0x00000038
R15 (PC) 0x0000000C
CPSR 0x600000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x0000000C
Mode Supervisor
States 28436053
tut9q4.s
01 AREA twoLevelCall, CODE, READONLY
02 RAM_BASE EQU 0x40000000
03 ENTRY
04 LDR sp, =RAM_BASE+0x100 ;init sp
05 LDR r0, =mydata ;get starting address of data
06 BL addsqnum ;call addsqnum, pass by register
07 stop B stop
08 mydata DCD 1, 2, 3, 4
09 addsqnum
10 STMFD sp!, {r4-r6, lr}
11 MOV r1, #0 ;sum
12 MOV r4, #4 ;counter (4 numbers)
13 loop LDR r5, [r0], #4 ;get value and update
14 STMFD sp!, {r5, r6} ;push input/output parameters
15 BL square ;call square, pass by stack
16 LDMFD sp!, {r5, r6} ;pop r5, r6(sq r5)
17 ADD r1, r1, r6 ;sum = sum + sq value
18 SUBS r4, r4, #1 ;dec counter
19 BNE loop
20 LDMFD sp!, {r4-r6, pc} ;return
21 square
22 STMFD sp!, {r4, r5, lr}
23 LDR r4, [sp, #12] ;get input from stack
24 MUL r5, r4, r4 ;squaring
25 STR r5, [sp,#16] ;store result in stack
26 LDMFD sp!,{r4, r5, pc} ;return
27 END