Microprocessors LAB

EC210

Assignment 1

Submitted By:

Suhas S G

201EC263

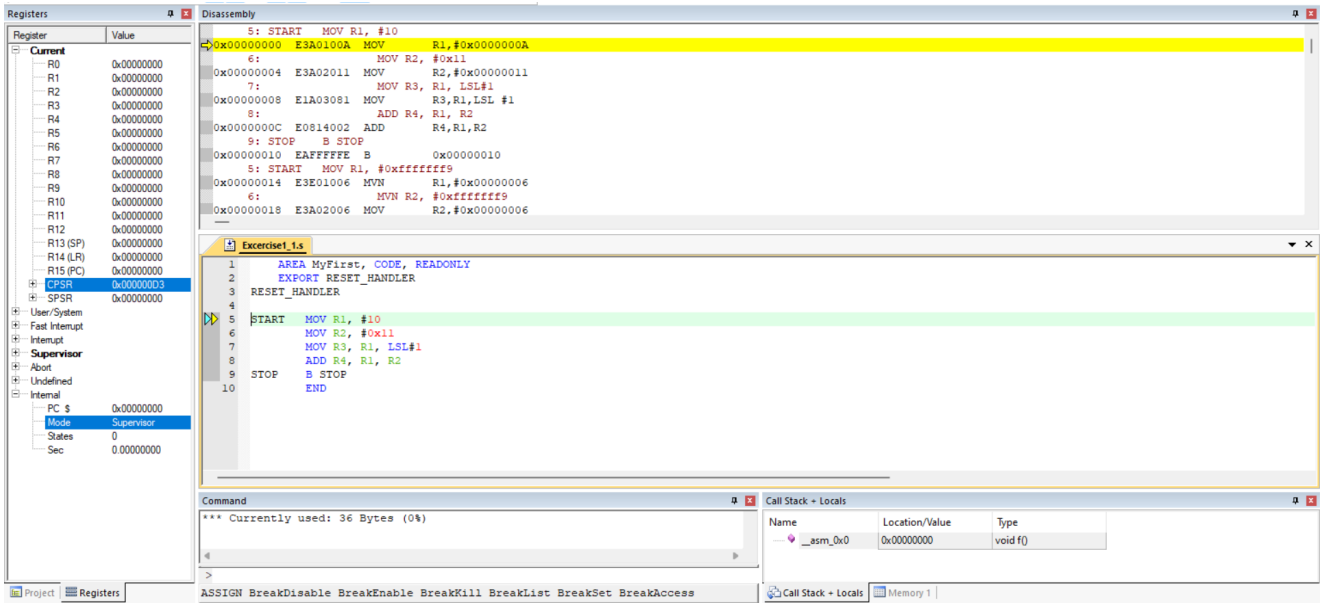
Nikhil P Reddy

201EC241

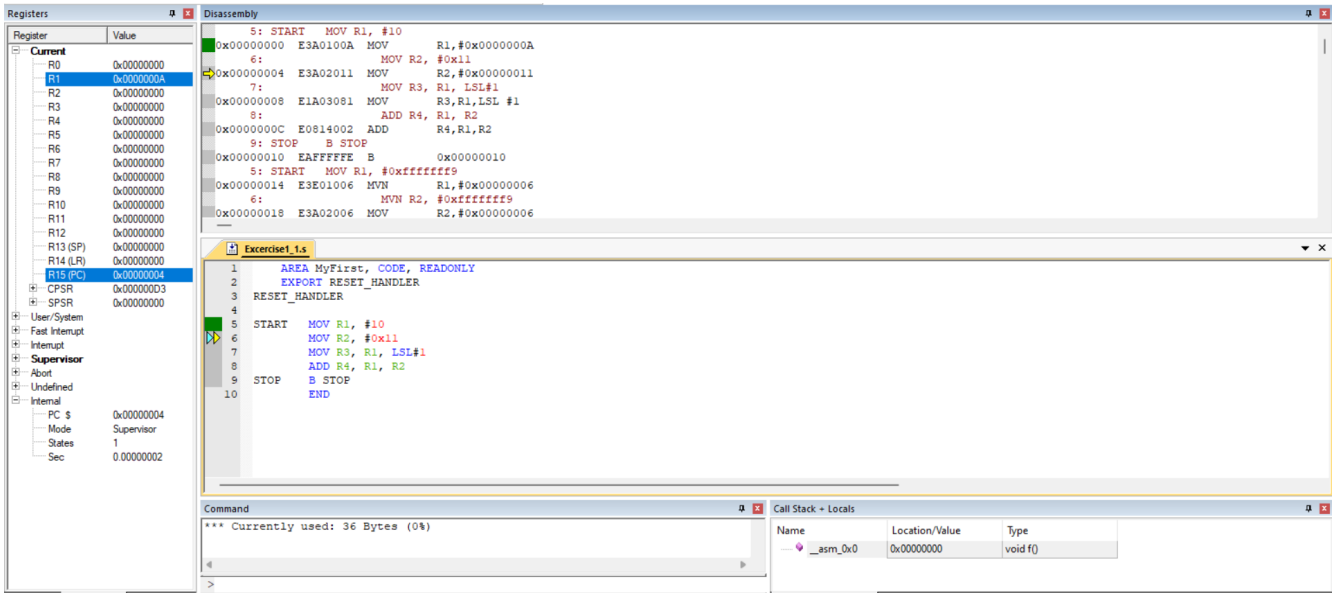
Program 1.1

```
                AREA        MyFirst, CODE, READONLY
                EXPORT      Reset_Handler

Reset_Handler

Start           MOV   R1, #10           ; Load a decimal Value
                MOV   R2, #0x11         ; Load an initial Hex Value
                MOV   R3, R1, LSL#1     ; Shift left by 1bit
                ADD   R4, R1, R2

Stop            B     Stop              ; Stop program
                END
```

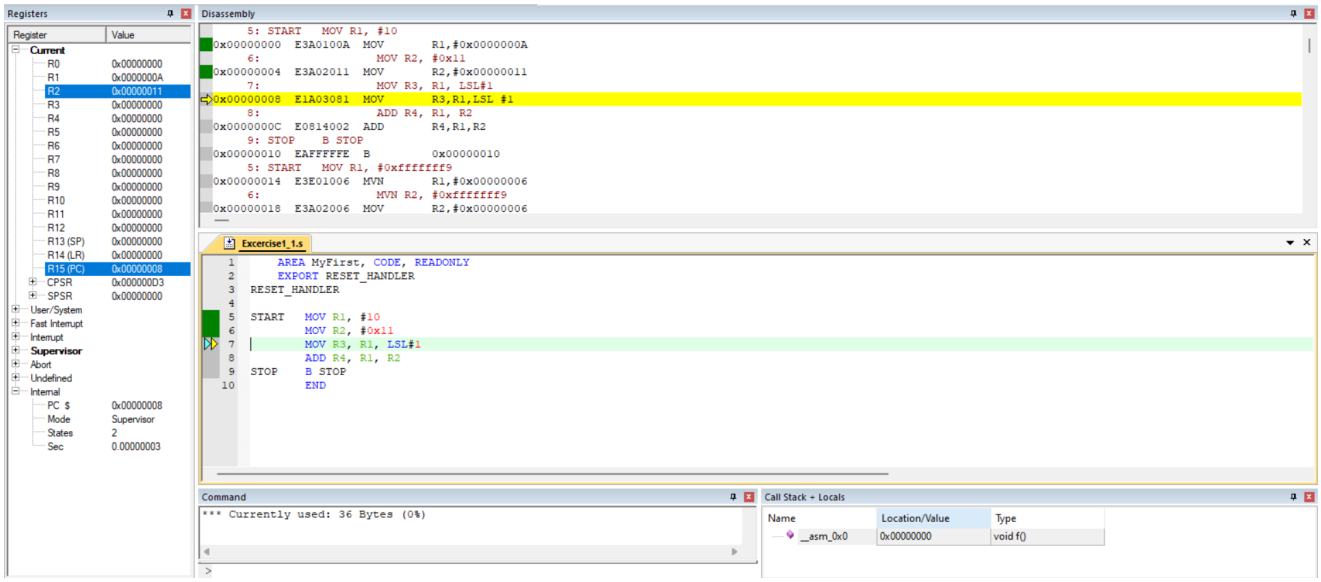
Step-by-Step Execution



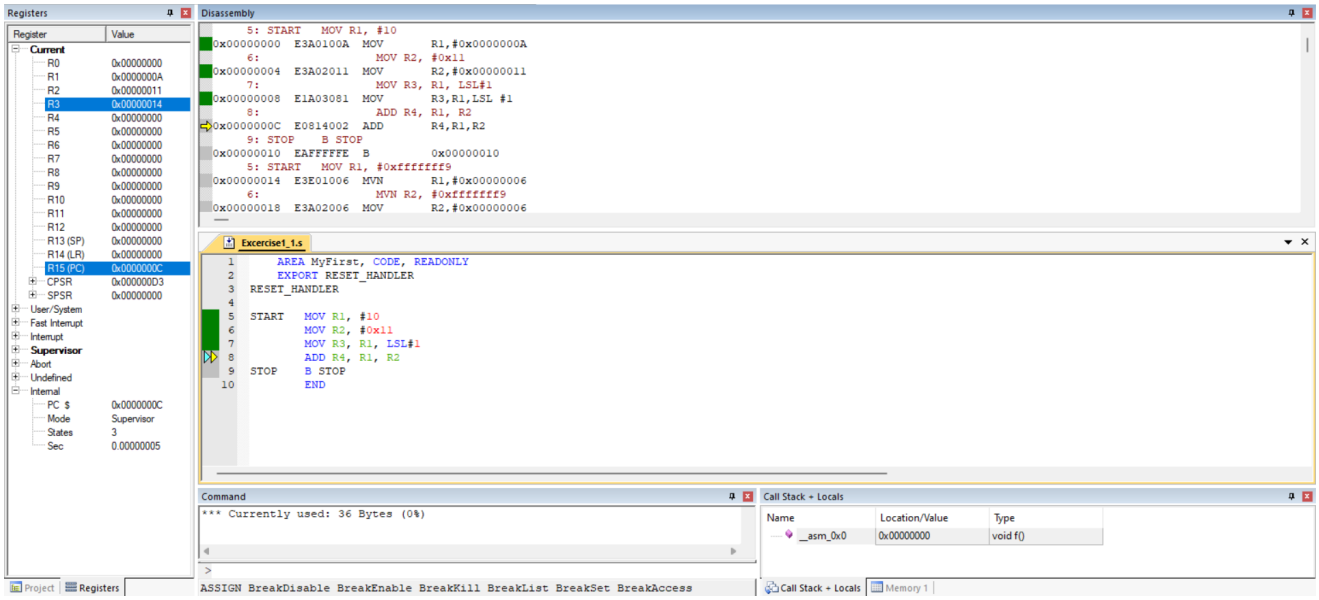
Program is compiled/translated and set to debug mode.



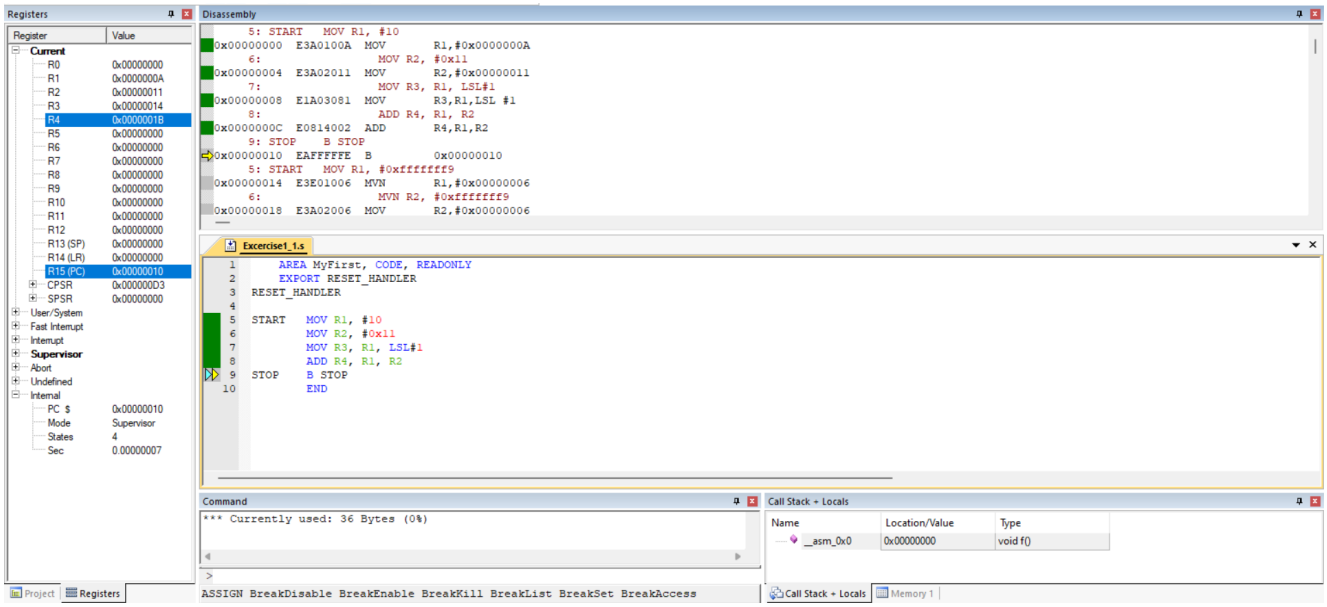
The decimal value 10 (0x0000000A) is loaded into register R1.



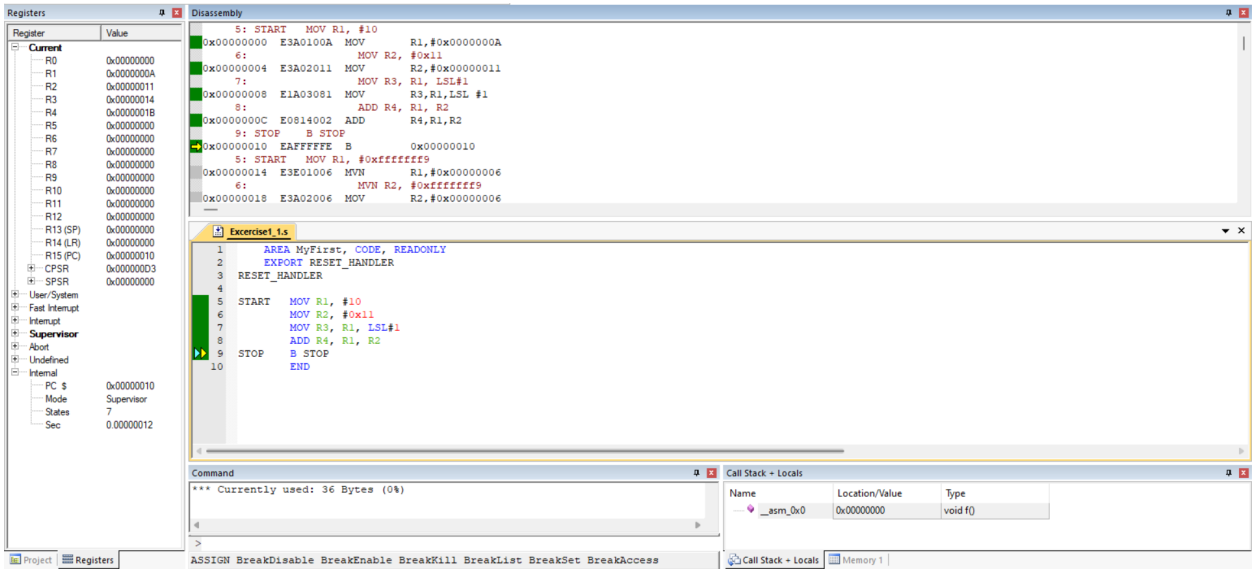
The hexadecimal value 0x11 is loaded into register R2.



The data/value of the register R1(0x0000000A) Logical Left Shifted by 1 unit = 0x00000014 and then is copied into the register R3.



The data/value of the register R1 (0x0000000A) is ADDed to the data present in the register R2 = 0x0000001B , after which it is stored in the register R4.



The program stops.

EXERCISE:

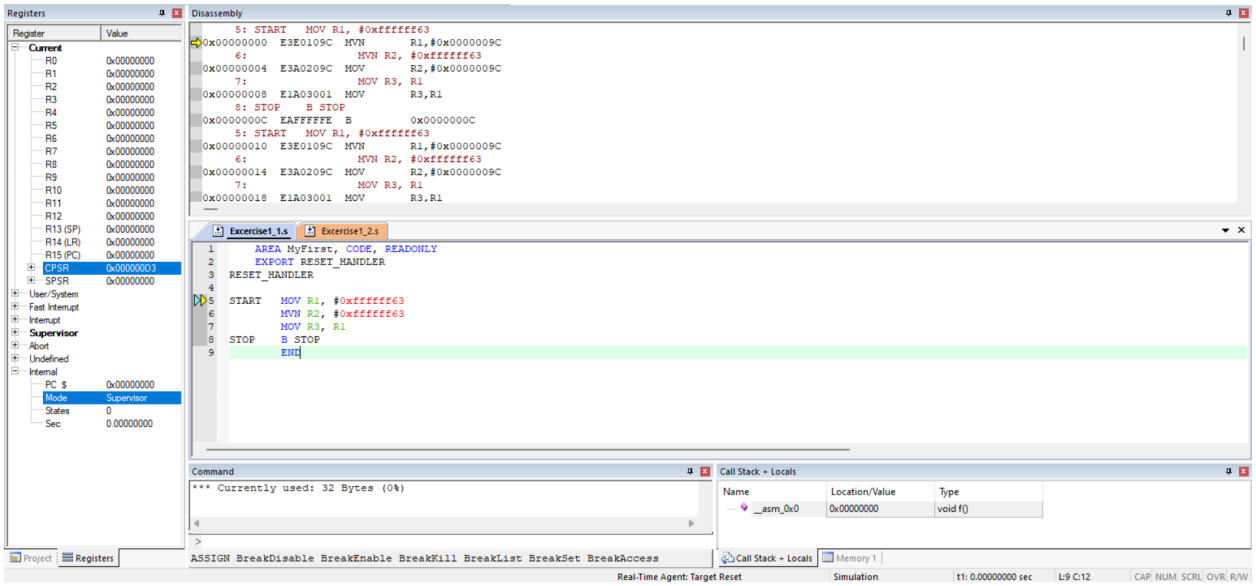
1.2. Observe the use of MOV and MVN instructions in loading a 32 bit immediate data into the registers.

Code

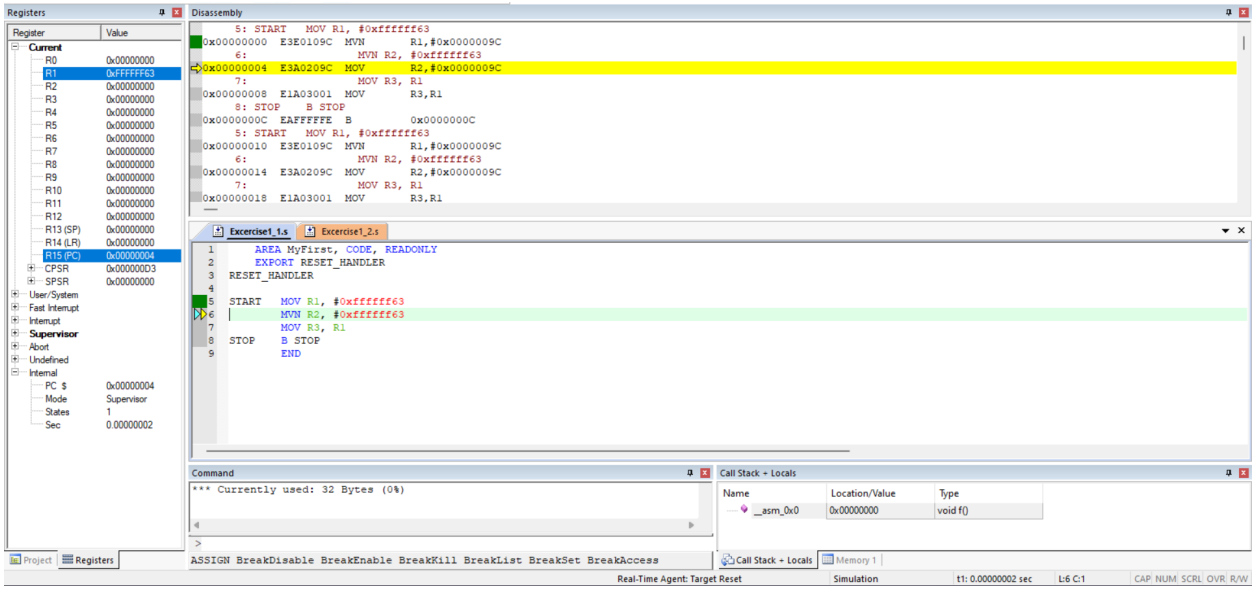
```
        AREA MyFirst, CODE, READONLY
        EXPORT RESET_HANDLER
RESET_HANDLER


START   MOV R1, #0xffffff63
        MVN R2, #0xffffff63
```

```
        MOV R3, R1
STOP    B STOP
        END
```

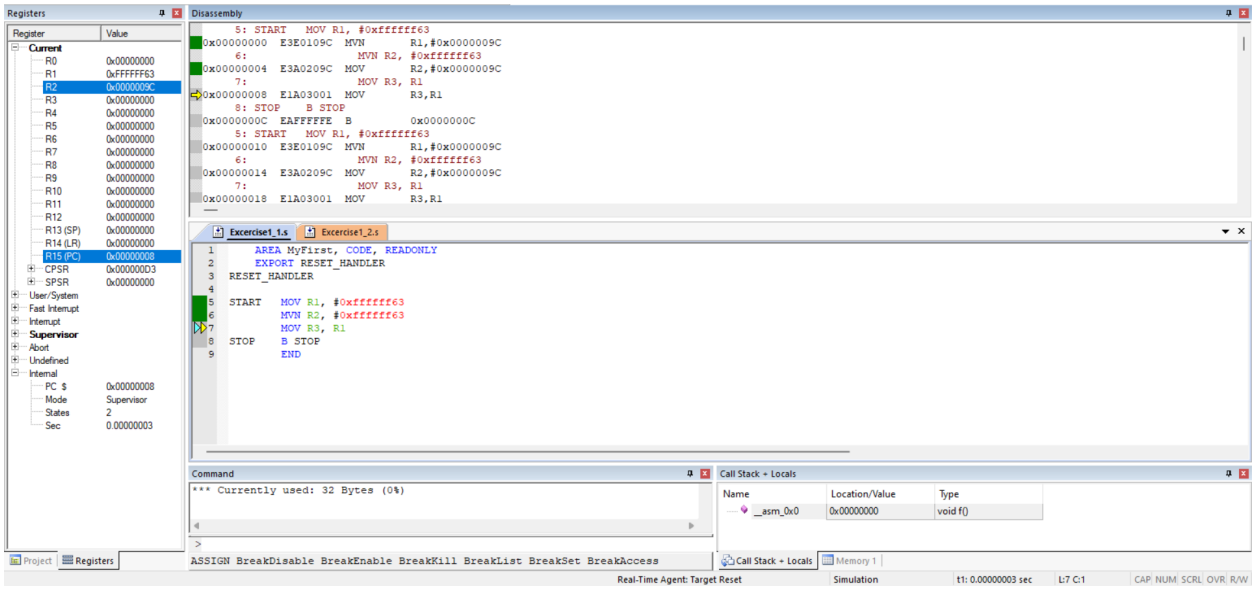
Step-by-Step Execution



Program is compiled/translated and set to debug mode.

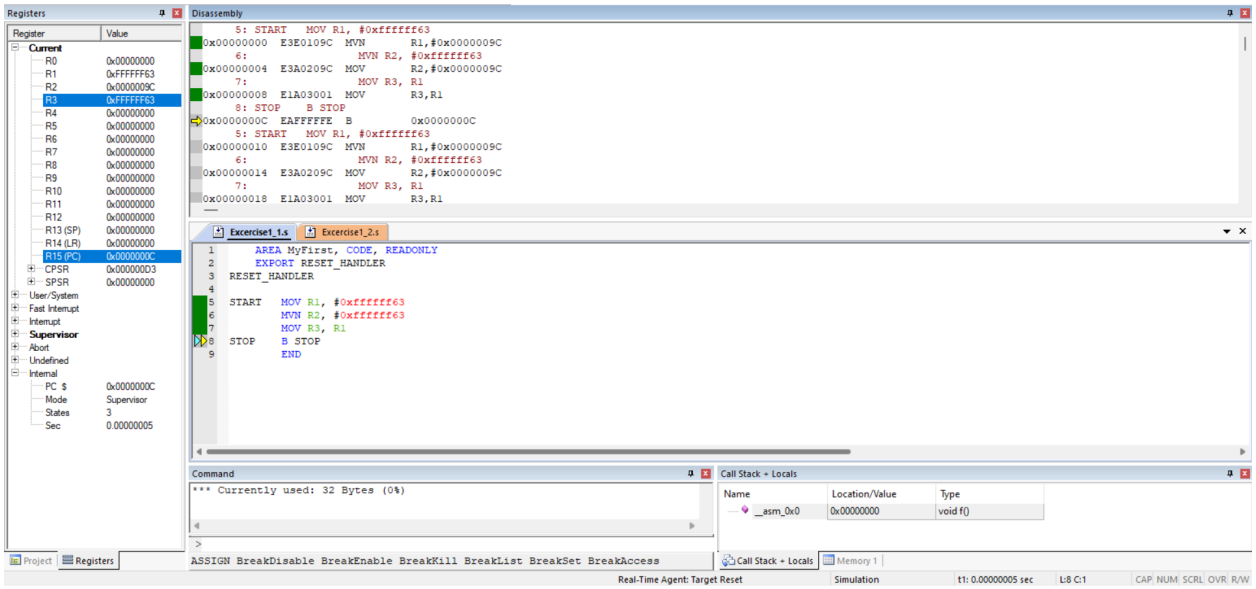


The hexadecimal value 0xffffff63 is loaded into register R1 using MOV instruction.



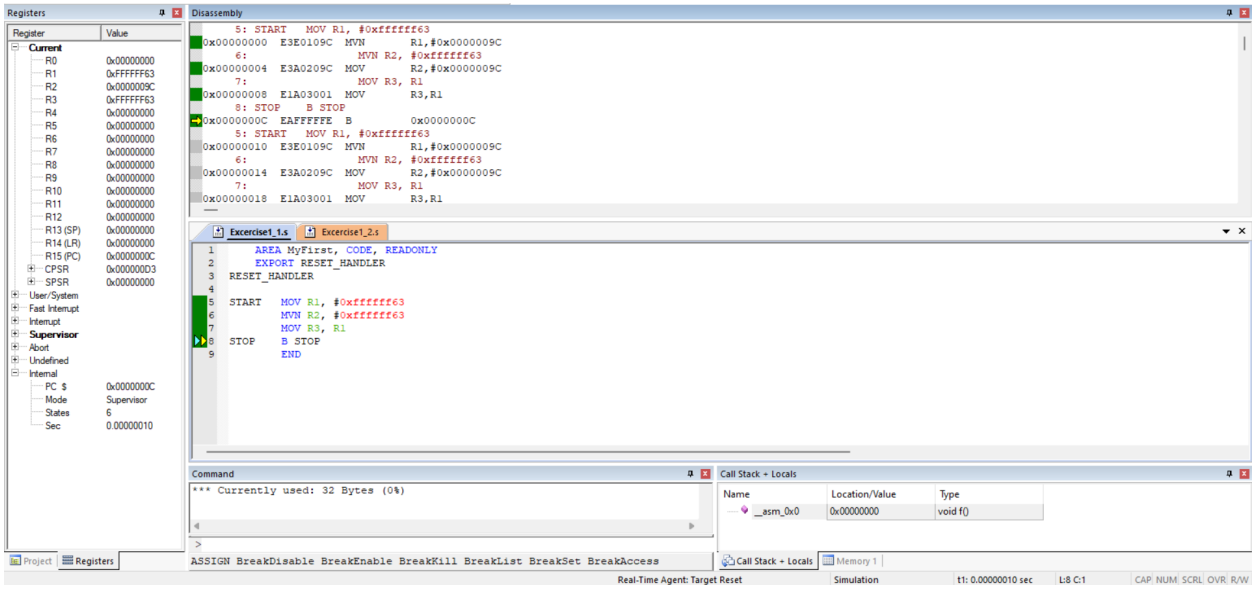
The hexadecimal value 0xffffff63 is loaded into register R2 using MVN opcode.

Notice that the bitwise negated value = 0xffffff9c is stored in the register R1.



The MOV instruction can also be used to copy values from one register to another.

In the above example, the value of register R1(0xffffff63) is copied into register R3.



The program stops.

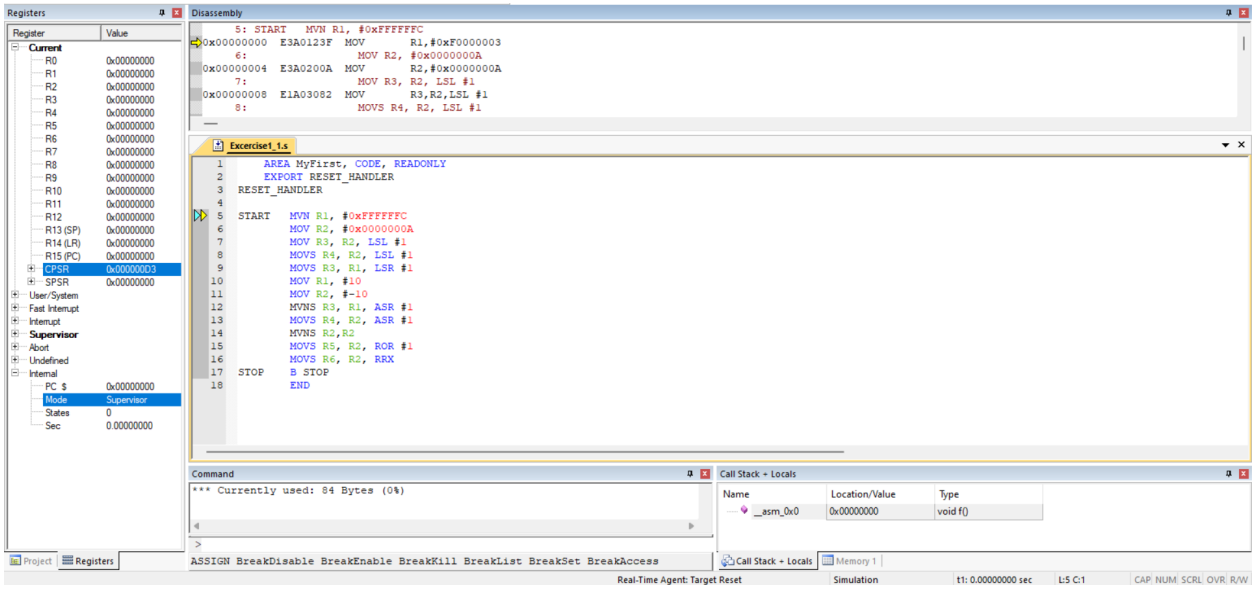
1.3. Demonstrate the use of LSL, LSR, ASR, ROR, RRX with MOV, MVN, MOVS and MVNS for different data. Observe the results and conditional code flags in CPSR.

Code

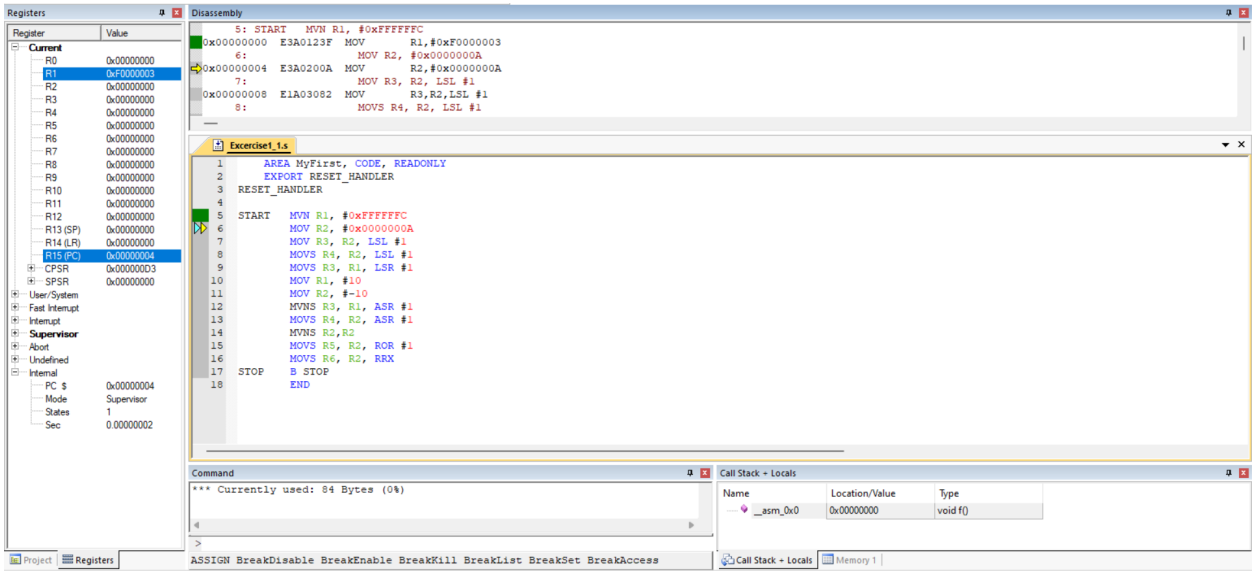
```
_____AREA MyFirst, CODE, READONLY

        EXPORT RESET_HANDLER

RESET_HANDLER


START       MVN R1, #0xFFFFFFC

            MOV R2, #0x0000000A

            MOV R3, R2, LSL #1

            MOVS R4, R2, LSL #1

            MOVS R3, R1, LSR #1

            MOV R1, #10

            MOV R2, #-10

            MVNS R3, R1, ASR #1

            MOVS R4, R2, ASR #1

            MVNS R2, R2

            MOVS R5, R2, ROR #1
```

```
        MOVS R6, R2, RRX

STOP    B STOP

        END
```

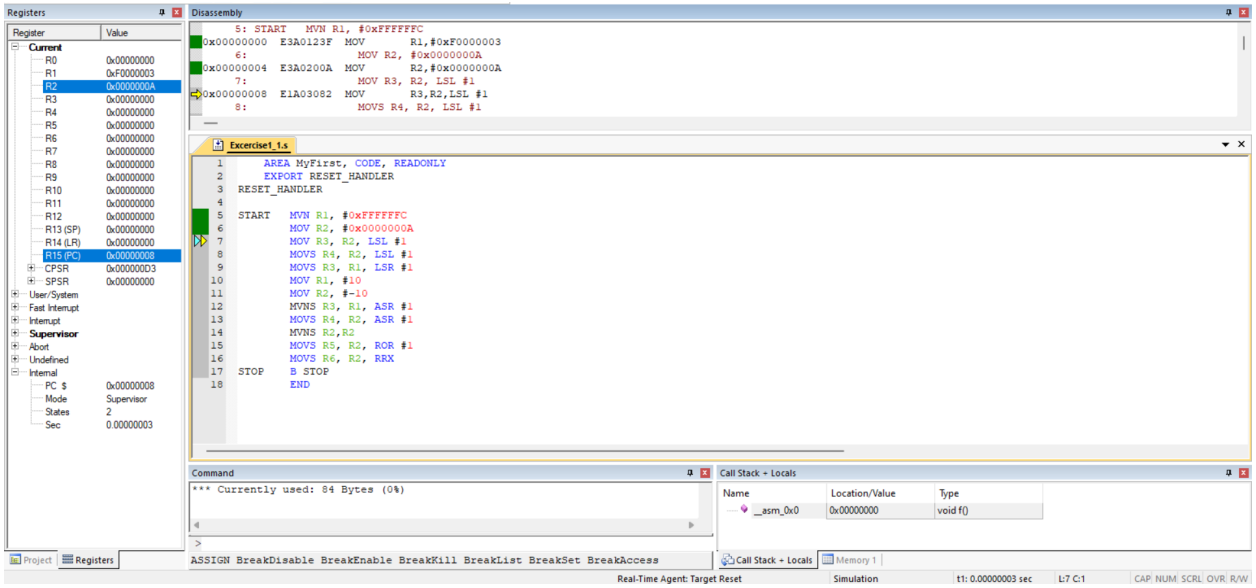
Step-by-Step Execution



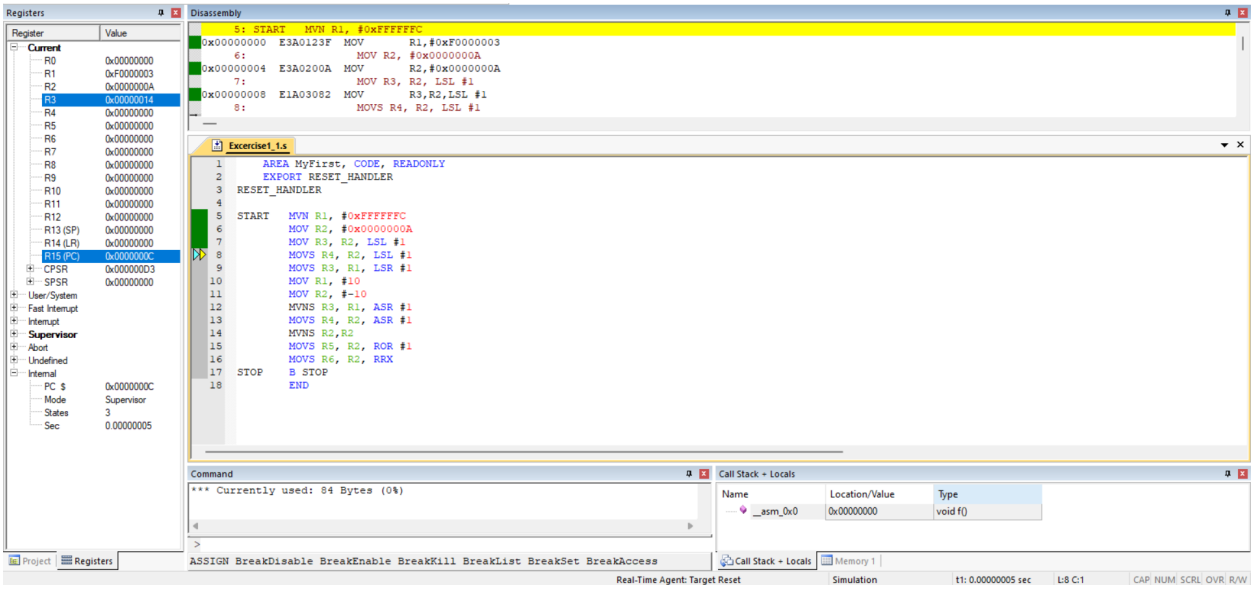
Program is compiled/translated and set to debug mode.



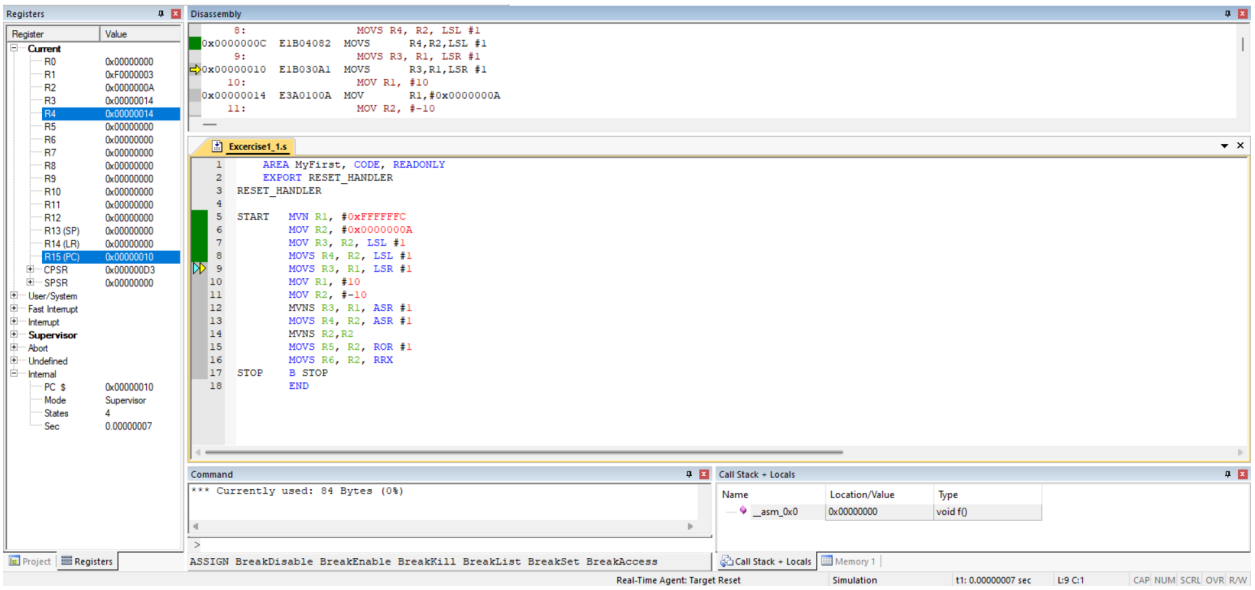
The hexadecimal value 0x0FFFFFFC is loaded into register R1 using MVN instruction; therefore the value of R1 is set to 0xF0000003, which is the bitwise not of 0x0FFFFFFC.



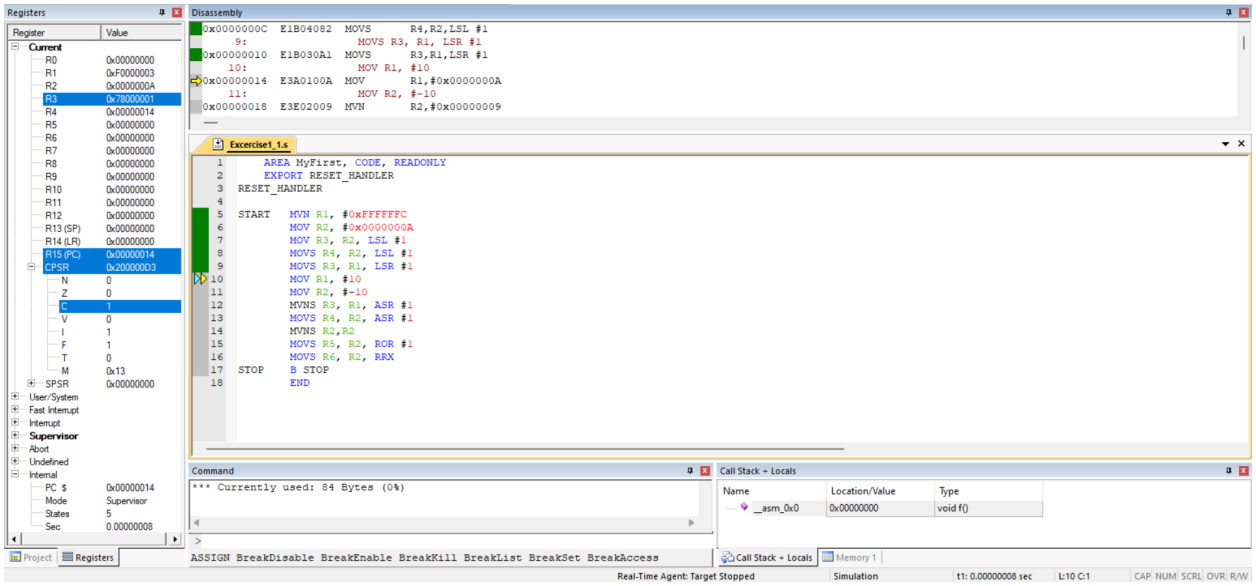
The hexadecimal value 0x0000000A is loaded into register R2 using MOV.



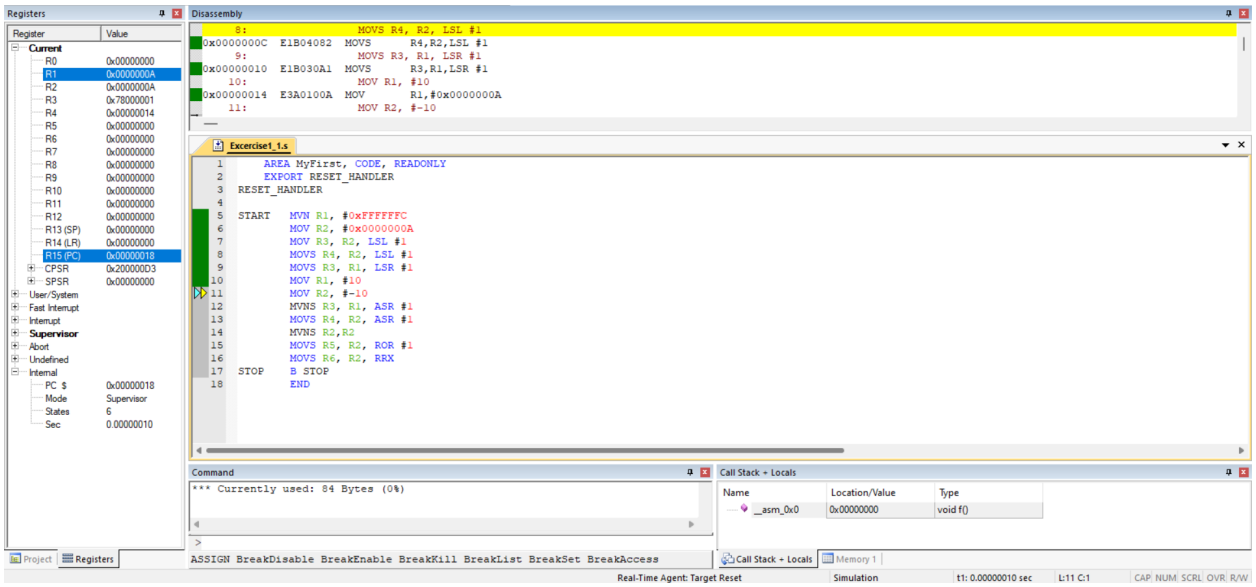
The hexadecimal value 0x0000000A , which is present in the register R2 is logical left shifted by 1 unit (multiply by 2) = 0x00000014, and then stored in register R3.



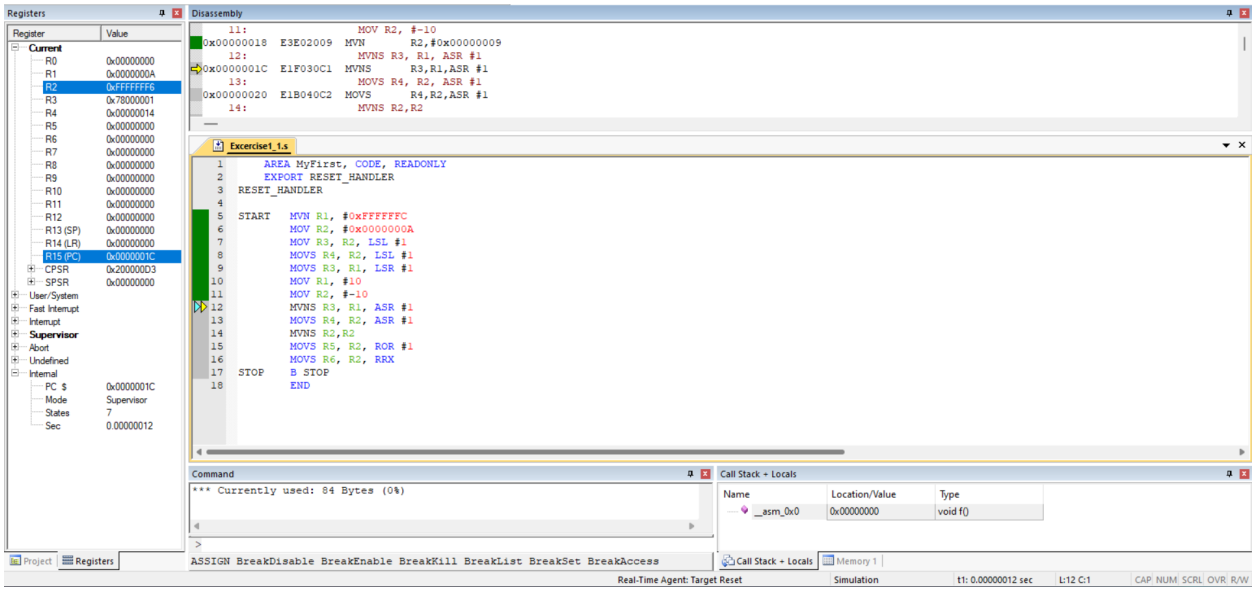
We performed the same instruction however on register R4 instead of R3 and using the MOVS instead of MOV (not the CPSR does not change as there is no carry).



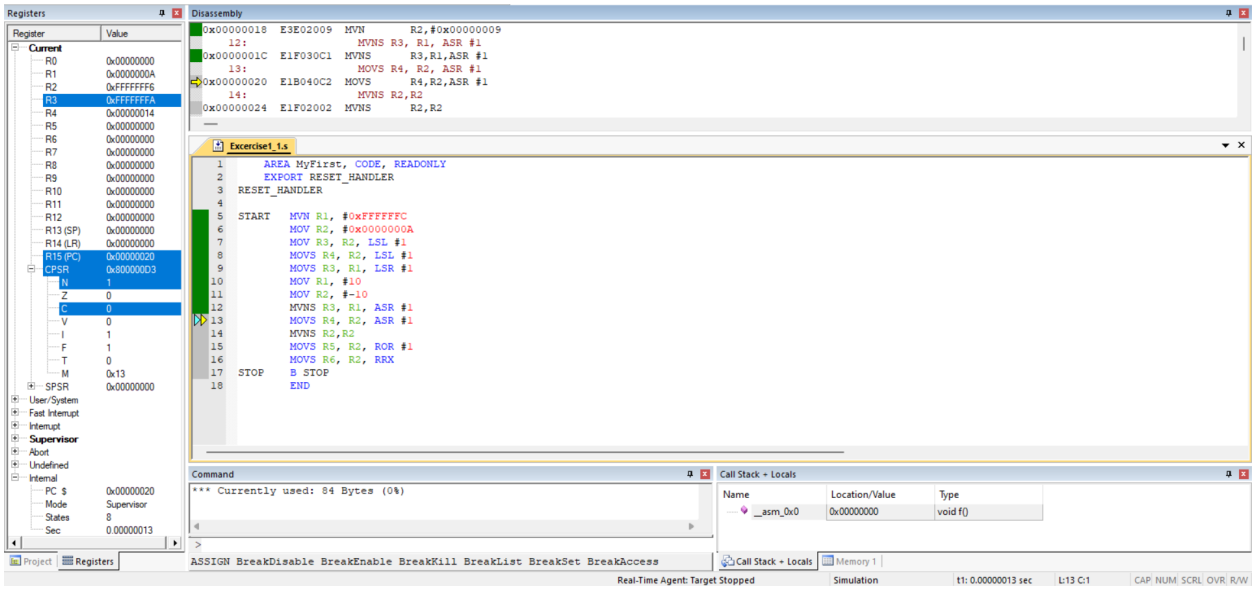
The hexadecimal value 0xF0000003, which is present in the register R1 is logical right shifted by 1 unit (divided by 2) = 0x78000001, and then stored in register R3. Note the C flag (Carry) is incremented to 1 as the LSB of 0xF0000003 is one.



The decimal value 10 = 0xFFFFFFFA is loaded into register R1 using MOV.



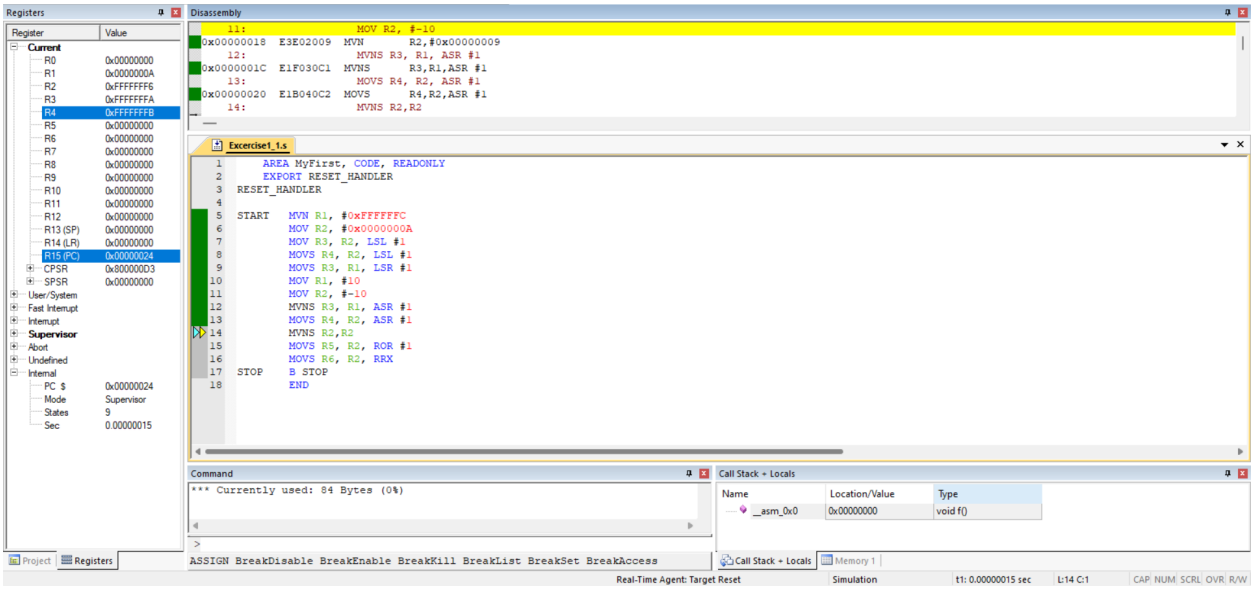
The decimal value -10 is loaded into register R2 using MOV. Hence R2 is set to 0xFFFFFFF6.



The hexadecimal value 0x0000000A, which is present in the register R1 is arithmetic right shifted by 1 unit (divided by 2) = 0x00000005, and then is bitwise

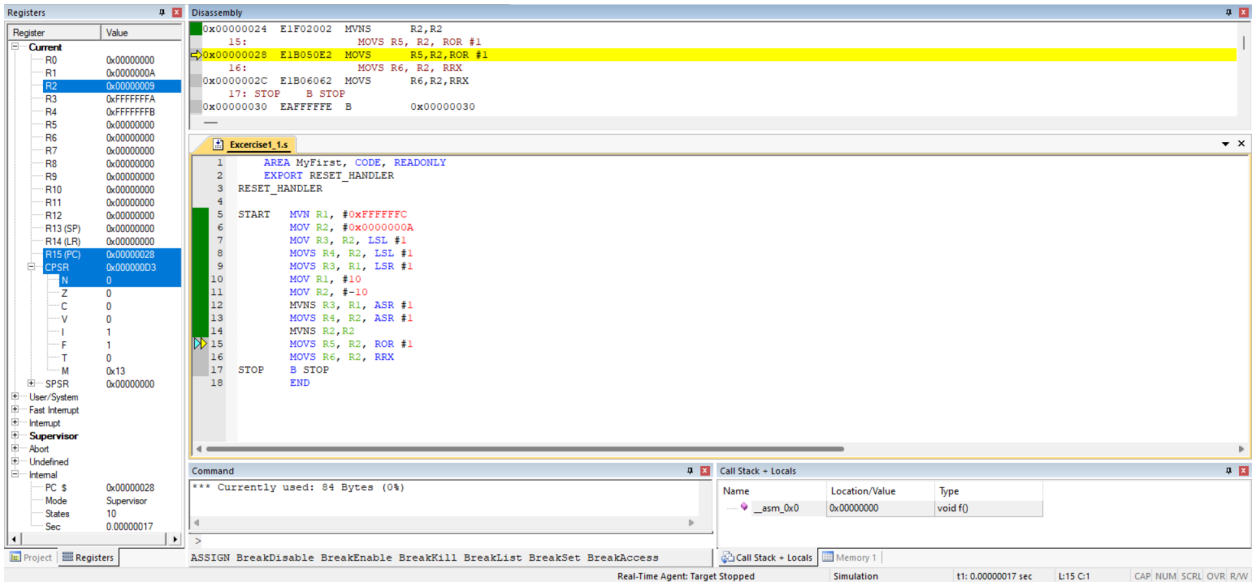
not is applied = 0xFFFFFFFA as we are using MVNS after which we are storing it in the register R3. Note the C flag (Carry) is set to 0 as the LSB of 0x00000005 is 0, also since the output is negative the N flag (Negative) is set to 1 .

*The ASR copies the MSB (the sign bit) into the empty slot which the LSR does not.

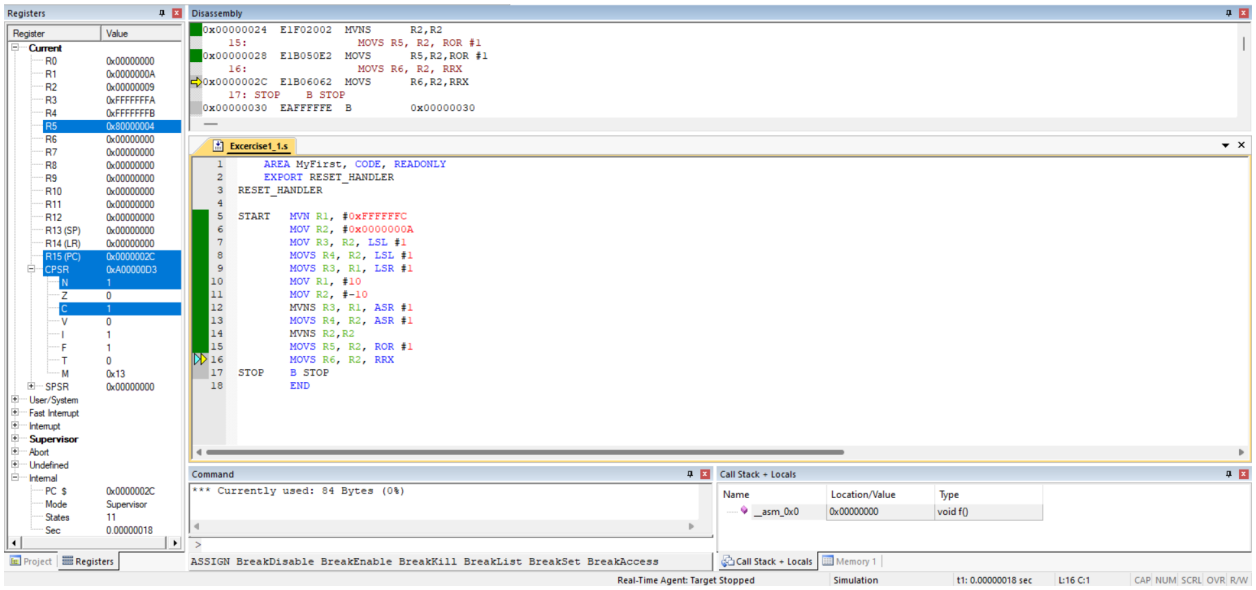


The hexadecimal value 0xFFFFFFF6, which is present in the register R2 is arithmetic right shifted by 1 unit (divided by 2) = 0xFFFFFFFB, and then is stored in the register R4. CPSR remains the same as the carry is 0.

*The ASR copies the MSB (the sign bit) into the empty slot which the LSR does not.



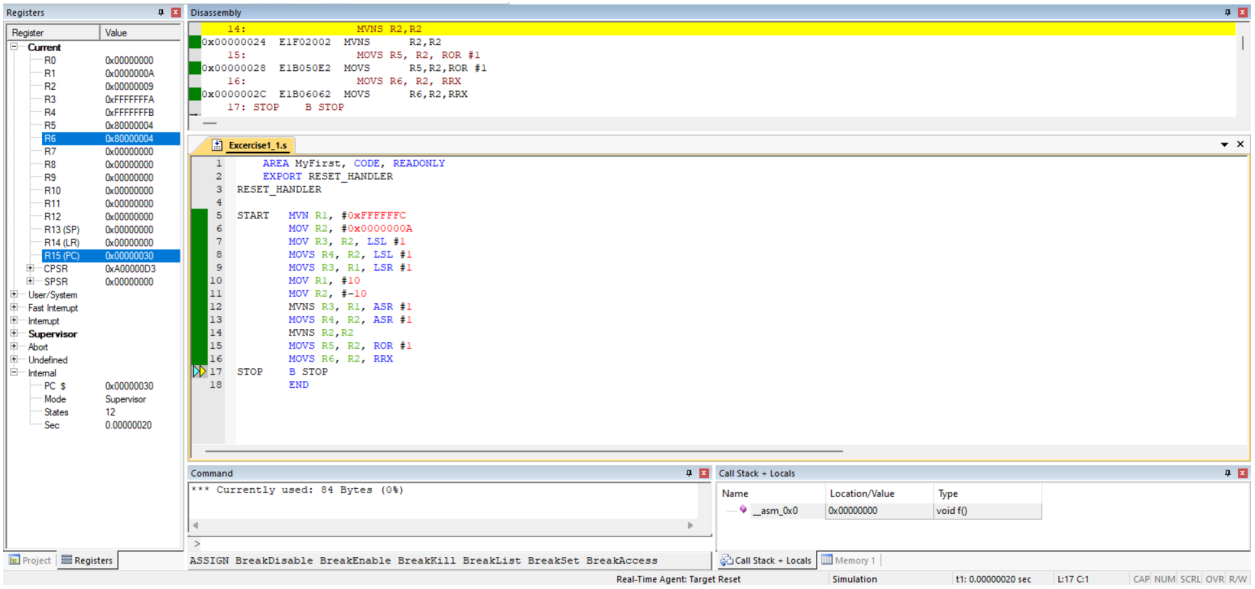
The hexadecimal value 0xFFFFFFF6, which is present in the register R2, is bitwise not as we are using MVNS instruction = 0x00000009 and stored in R2 itself. Note here the N flag (negative flag) is toggled to 0 as the result is positive.



The hexadecimal value 0x00000009, which is present in the register R2, is rotated

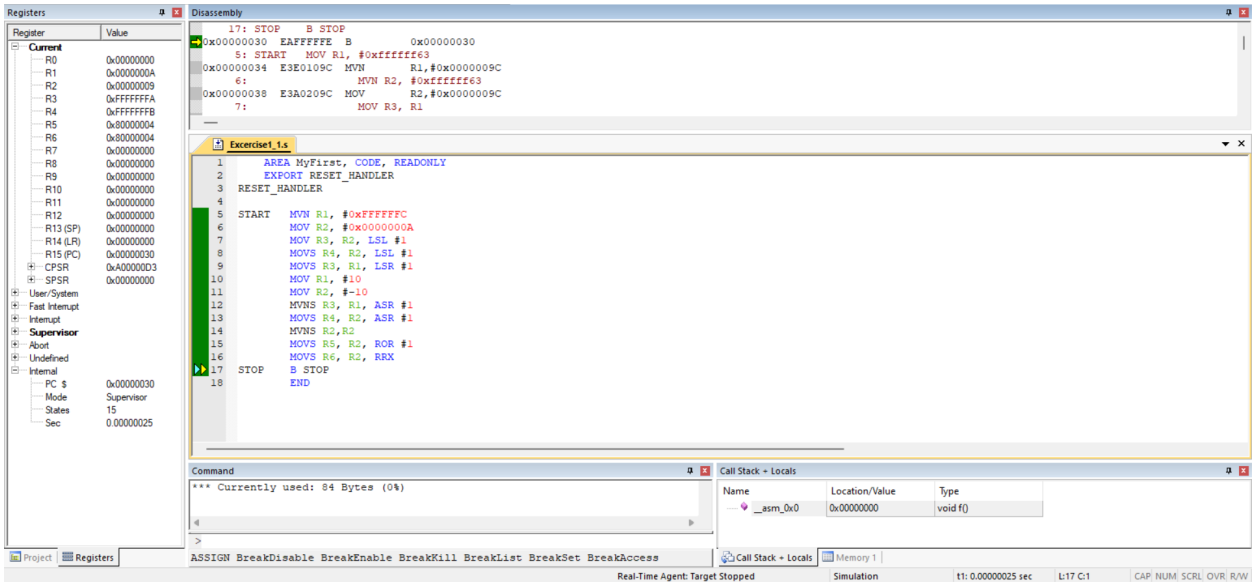
right by 1 unit using ROR = 0x80000004 and stored in R5. Note here the N flag (negative flag) is toggled to 1 as the result is negative and also the C flag (carry flag) is also set to 1 as LSB (of 0x00000009)is 1 .

*The ROR instruction gives the LSB as carry flag and n units are logically shifted by right and copied into the empty spaces.



The hexadecimal value 0x00000009, which is present in the register R2, is rotated right using RRX by 1 unit and stored in R5 = 0x80000004. Note here the N flag (negative flag) remains 1 as the result is negative and also the C flag (carry flag) remains as LSB (of 0x00000009)is 1 .

*The RXX is a 33 bit (carry is included as LSB of the value) rotation which is different from the ROR which is a 32 bit rotation.



The program stops.