Copying 10 bytes of data from source address to destination address

- LDR r0,=0x20000000 ; Source address
- LDR r1,=0x20000050 ; Destination address
- LDR r2,=10 ; number of bytes to copy

copy_loop

- LDRB r3, [r0] ; read 1 byte
- ADDS r0, r0, #1 ; increment source pointer
- STRB r3, [r1] ; write 1 byte
- ADDS r1, r1, #1 ; increment destination pointer
  SUBS r2, r2, #1 ; decrement loop counter
  BNE copy_loop ; loop until all data copied

Moving the contents from Source to Destination Address

- LDR r0,=0x20000000 ; Source address
- LDR r1,=0x20000100 ; Destination address
- LDR r2,=50 ; number of bytes to copy, also

copy_loop ; acts as loop counter

SUBS r2, r2, #1 ; decrement offset and loop counter

LDRB r4,[r0, r2] ; read 1 byte

- STRB r4,[r1, r2] ; write 1 byte
- BNE copy_loop ; loop until all data copied
- By using the loop counter as a memory offset, we have reduced usage of more no of instructions

Read and store significance of !

- LDMIA R0!, {R1, R2, R5 & R7} ; Read multiple registers, R0 update to address after last read operation.
- STMIA R0!, {R1, R2, R5 & R7} ; Store R1, R2, R5, R6, and R7 to memory; and update R0 to address after where R7 stored

128 Bytes to be copied

- LDR r0,=0x20000000 ; Source address
- LDR r1,=0x20000100 ; Destination address
- LDR r2,=128 ; number of bytes to copy, also
- copy_loop ; acts as loop counter
- LDMIA r0!,{r4-r7} ; Read 4 words and increment r0
- STMIA r1!,{r4-r7} ; Store 4 words and increment r1
- LDMIA r0!,{r4-r7} ; Read 4 words and increment r0
- STMIA r1!,{r4-r7} ; Store 4 words and increment r1
- LDMIA r0!,{r4-r7} ; Read 4 words and increment r0
- STMIA r1!,{r4-r7} ; Store 4 words and increment r1
- LDMIA r0!,{r4-r7} ; Read 4 words and increment r0
- STMIA r1!,{r4-r7} ; Store 4 words and increment r1
- SUBS r2, r2, #64 ; Each time 64 bytes are copied
- BNE copy_loop ; loop until all data copied

64 Bit Addition

- LDR r0,=0xFFFFFFFF ; X_Low (X ¼=0x3333FFFFFFFFFFFF)
- LDR r1,=0x3333FFFF ; X_High
- LDR r2,=0x00000001 ; Y_Low (Y ¼=0x3333000000000001)
- LDR r3,=0x33330000 ; Y_High
- ADDS r0,r0,r2 ; lower 32-bit
- ADCS r1,r1,r3 ; upper 32-bit

Implementation of FOR LOOP

- **Total = 0;**
- **for (i=0;i<5;i=i+1)**
- **Total = Total + i;**
- //Assume “Total” is R0 and “i” is R1; the program can be implemented as
- MOVS R0, #0 ; Total = 0
- MOVS R1, #0 ; i = 0
- loop
- ADDS R0, R0, R1 ; Total = Total + i
- ADDS R1, R1, #1 ; i = i + 1
- CMP R1, #5 ; compare i to 5
- BLT loop ; if less than then branch to loop

Function

```
function1
    SUB SP, SP, #0x8 ; Reserve 2 words of stack ;(8 bytes) for local variables ;Data
     processing in function
     MOVS r0, #0x12 ; set a dummy value
      STR r0, [sp, #0] ; Store 0x12 in 1st local variable
      STR r0, [sp, #4] ; Store 0x12 in 2nd local variable
      LDR r1, [sp, #0] ; Read from 1st local variable
     LDR r2, [sp, #4] ; Read from 2nd local variable
     ADD SP, SP, #0x8; Restore SP to original position
     BX LR

__main
BL function1
stop B stop
```

Calling Function from other file

```
PRESERVE8 ; Indicate the code here preserve
; 8 byte stack alignment
                THUMB    ; Indicate THUMB code is used
               AREA    |.text|, CODE, READONLY

            EXPORT __main
            EXTERN func
; Start of CODE area

__main
     LDR r0,=0x10;
 BL func
stop B stop
  END
```

Called Function-(2x+9)

```
PRESERVE8
        THUMB
        AREA |.text|, CODE, READONLY
        EXPORT func

func
        push {r0}
        LDR R2,[r13];
        MOVS R3,#2
        MULS R2,R3,R2
        ADDS r2,r2,#9
        STR r2,[R13]
        pop {r1}
        BX LR
        END
```

Nested Function

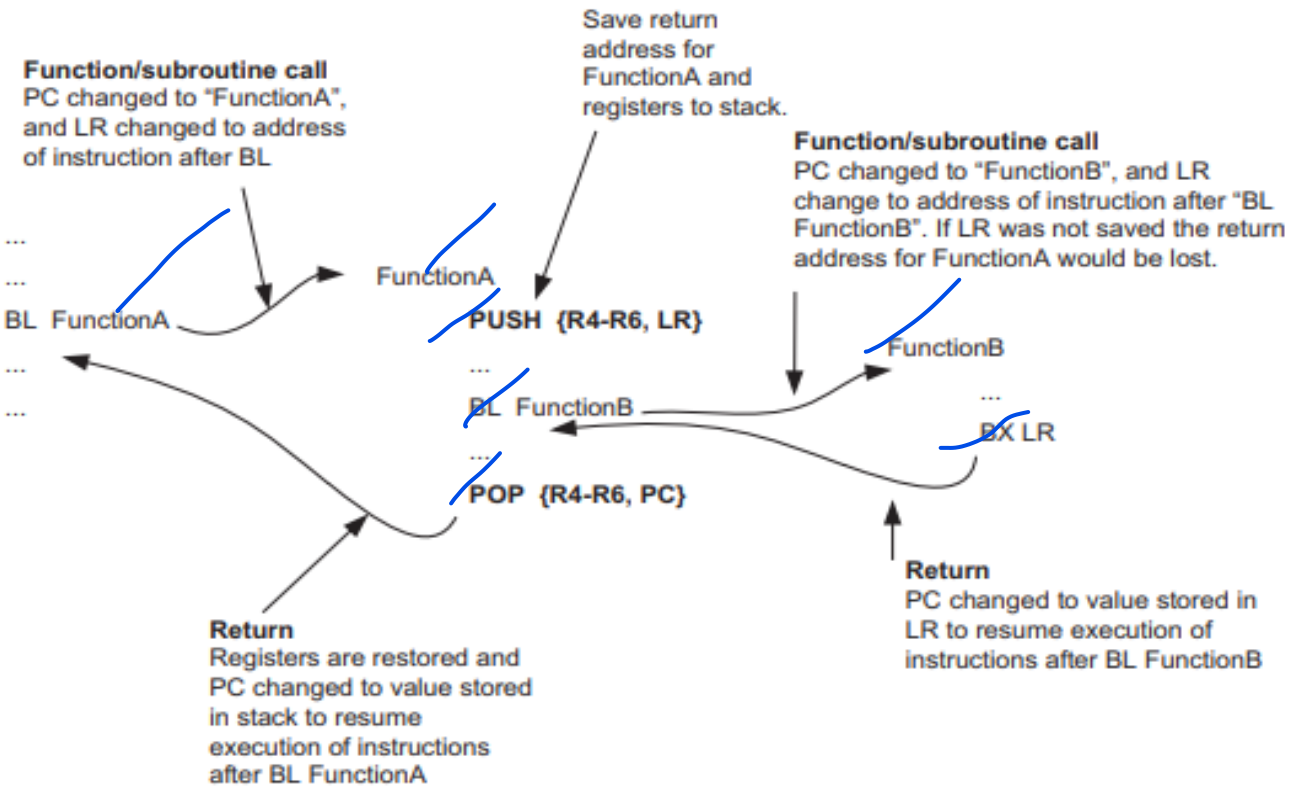


Figure 6.3:
Using push and pop of multiple registers in functions.

Main Function

```
 PRESERVE8 ; Indicate the code here preserve
; 8 byte stack alignment
                THUMB     ; Indicate THUMB code is used
              AREA    |.text|, CODE, READONLY

                                EXPORT __main
                                 EXTERN  func
; Start of CODE area

__main
        LDR r0,=0x10;
 BL func
stop B stop
  END
```

Code for calling func

```
PRESERVE8
        THUMB
        AREA |.text|, CODE, READONLY
        EXPORT func
        EXTERN func2

func
        push{LR}
        MOVS R1,#08
        BL func2
        pop{PC}
    END
```

func calling func2

```
PRESERVE8
    THUMB
    AREA |.text|, CODE, READONLY
    EXPORT func2

func2
    MOVS r2,#08
    BX LR
    END
```

USING BICS and ORRS to Set the priority register with priority levels.

- __main

  ldr r0,=0x20000000
- ldr r1,=0xffffffff
- movs r2, #0xff
- lsls r2,r2,#16
- bics r1,r1,r2
- movs r2, #0xc0
- lsls r2,r2,#16
- orrs r1,r1,r2
- str r1,[r0]
- stop b stop
- end

The 4 priority register levels are

1.00

2.40

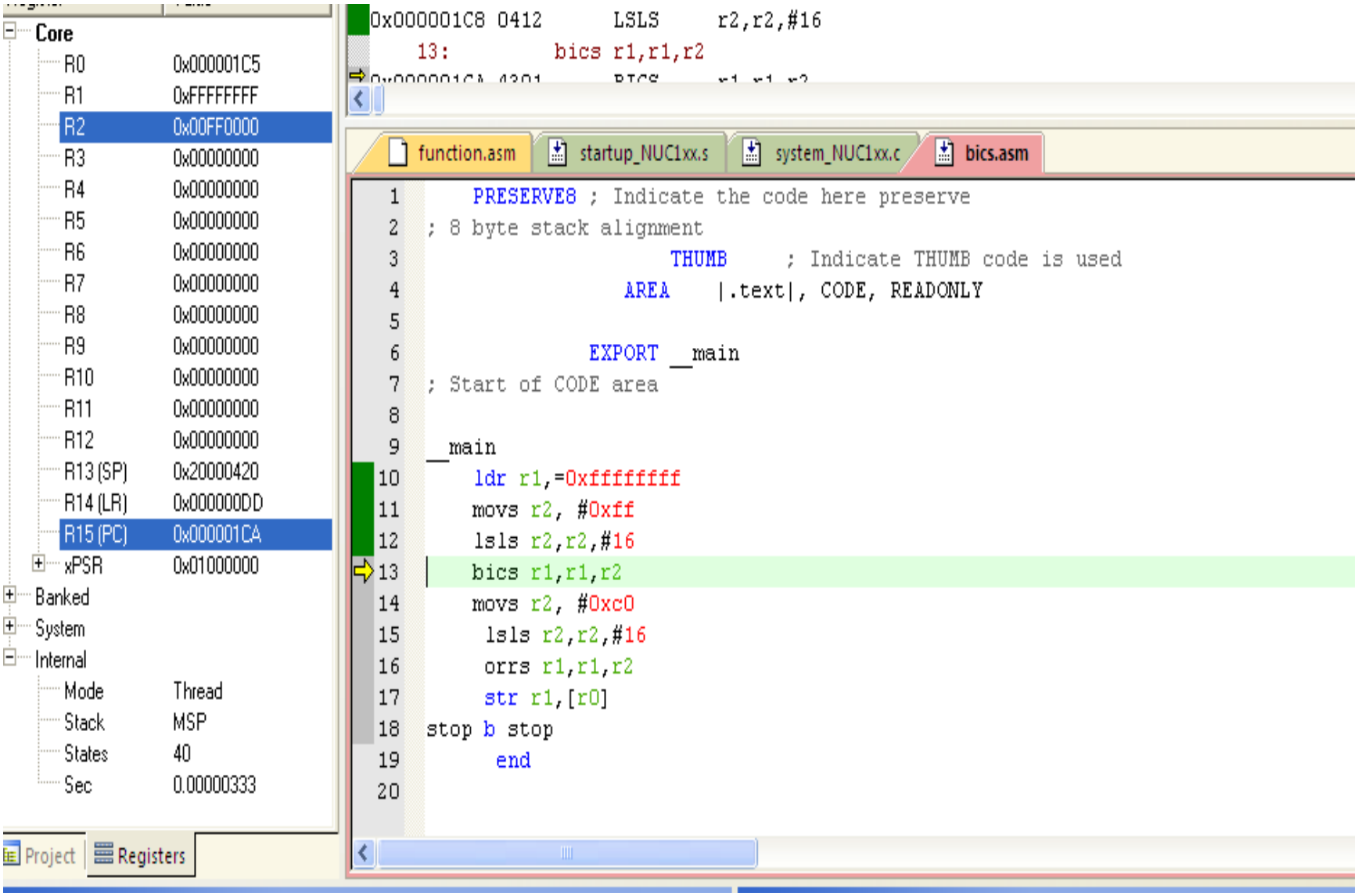
3.80

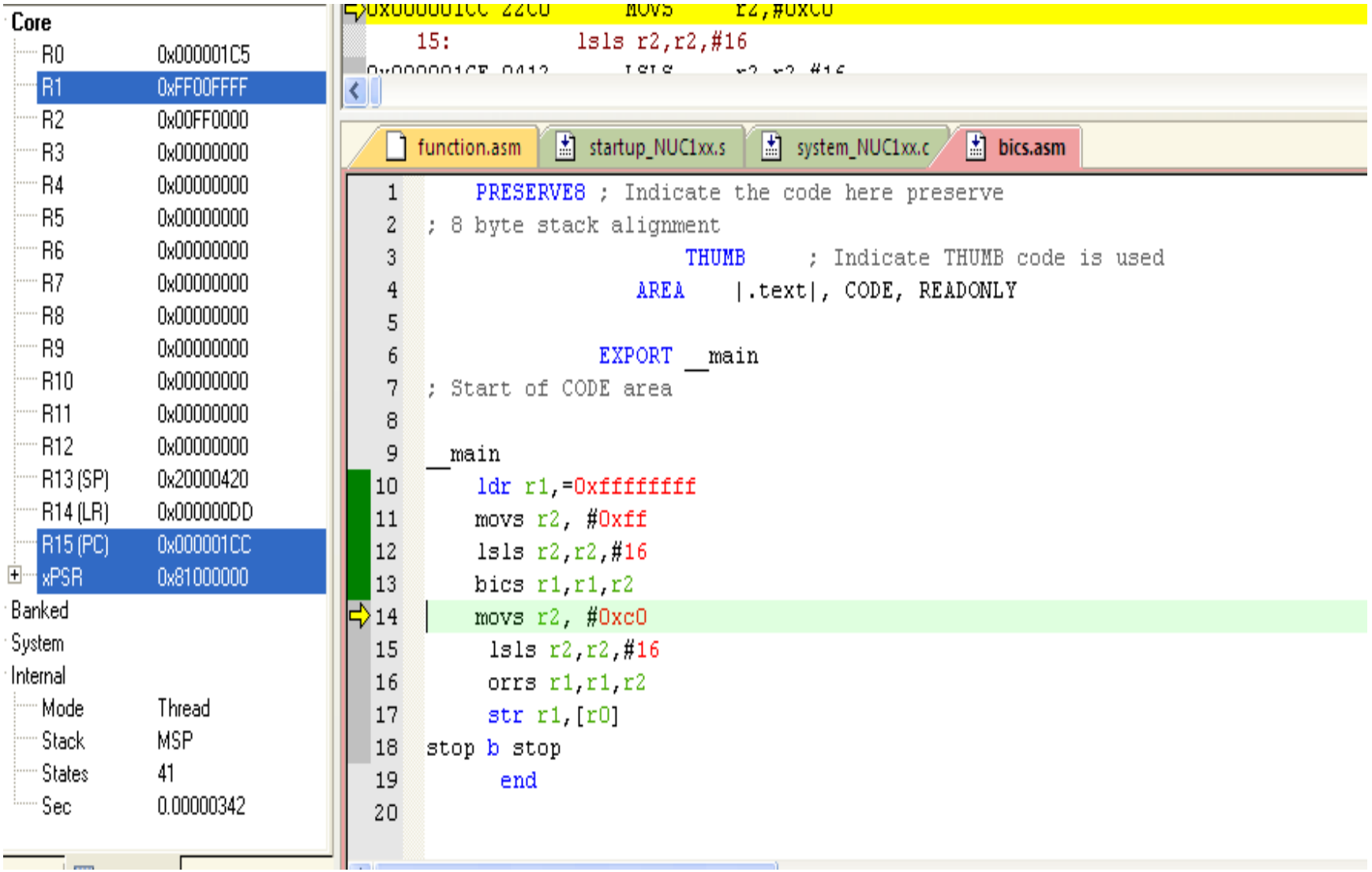
4.C0

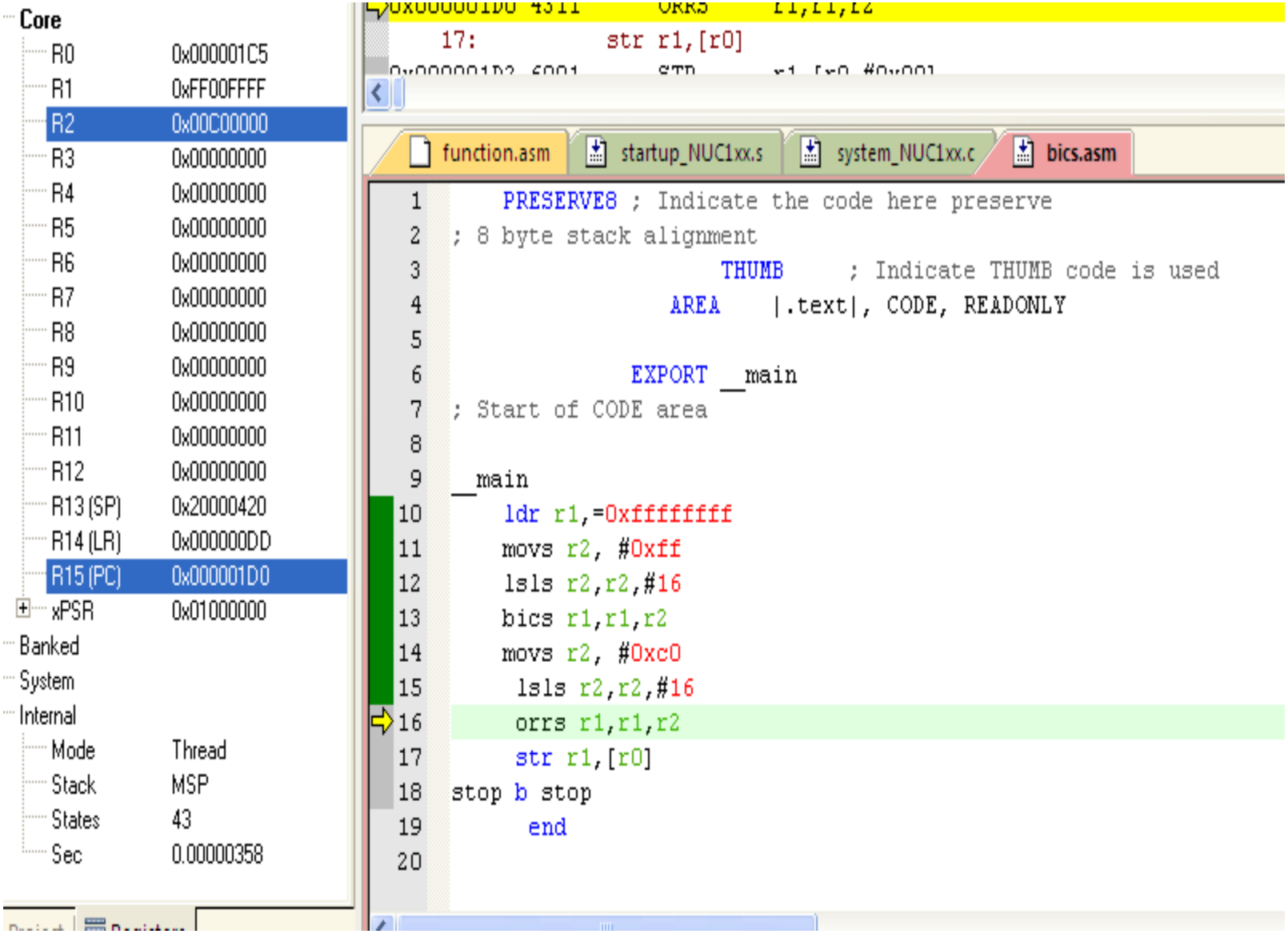
When we are require to set interrupt priority level with 00 having higher priority and CO having lowest priority into a register mapping to a particular interrupt, we use this code.

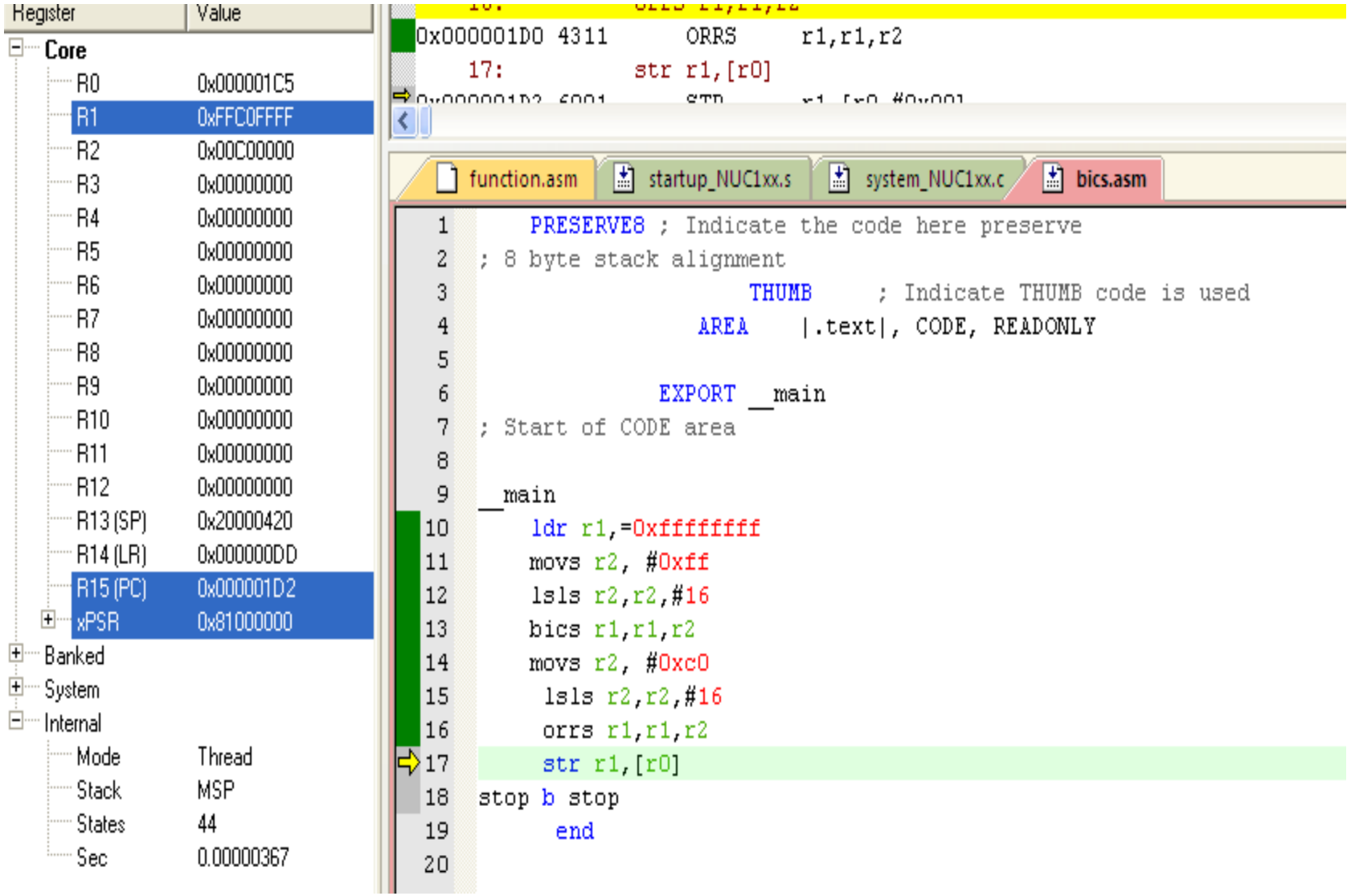
BIC (Logical Bitwise Clear)

Instruction	BIC
Function	Logical Bitwise Clear
Syntax (UAL)	BICS <Rd>, <Rd>, <Rm>
Syntax (Thumb)	BIC <Rd>, <Rm>
Note	Rd = AND(Rd, NOT(Rm)), APSR.N, and APSR.Z update. Rd and Rm are low registers.


Core
R0 0x000001C5
R1 0xFFFFFFFF
R2 0x00FF0000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CA
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Stack MSP
States 40
Sec 0.00000333
Project
Registers
0x000001C8 0412 LSLS r2,r2,#16
13: bics r1,r1,r2
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Core
R0 0x000001C5
R1 0xFF00FFFF
R2 0x00FF0000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CC
xPSR 0x81000000
Banked
System
Internal
Mode Thread
Stack MSP
States 41
Sec 0.00000342
0x000001CC 22C0 MOVS r2,#0xC0
15: lsls r2,r2,#16
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Core
R0 0x000001C5
R1 0xFF00FFFF
R2 0x00C00000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D0
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Stack MSP
States 43
Sec 0.00000358
17: str r1,[r0]
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Register
Value
Core
R0 0x000001C5
R1 0xFFC0FFFF
R2 0x00C00000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D2
xPSR 0x81000000
Banked
System
Internal
Mode Thread
Stack MSP
States 44
Sec 0.00000367
0x000001D0 4311 ORRS r1,r1,r2
17: str r1,[r0]
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20

SWITCH

- ;CASE BRANCHING PRESERVE8 ; Indicate the code here preserve ; 8 byte stack alignment THUMB ; Indicate THUMB code is used AREA |.text|, CODE, READONLY EXPORT main ; Start of CODE area

__main

LDR R0, =0

CMP R0, #3 ; Compare input to maximum valid choice

BHI default_case ; Branch to default case if higher than 3

MOVS R2, #4 ; Multiply branch table offset by 4

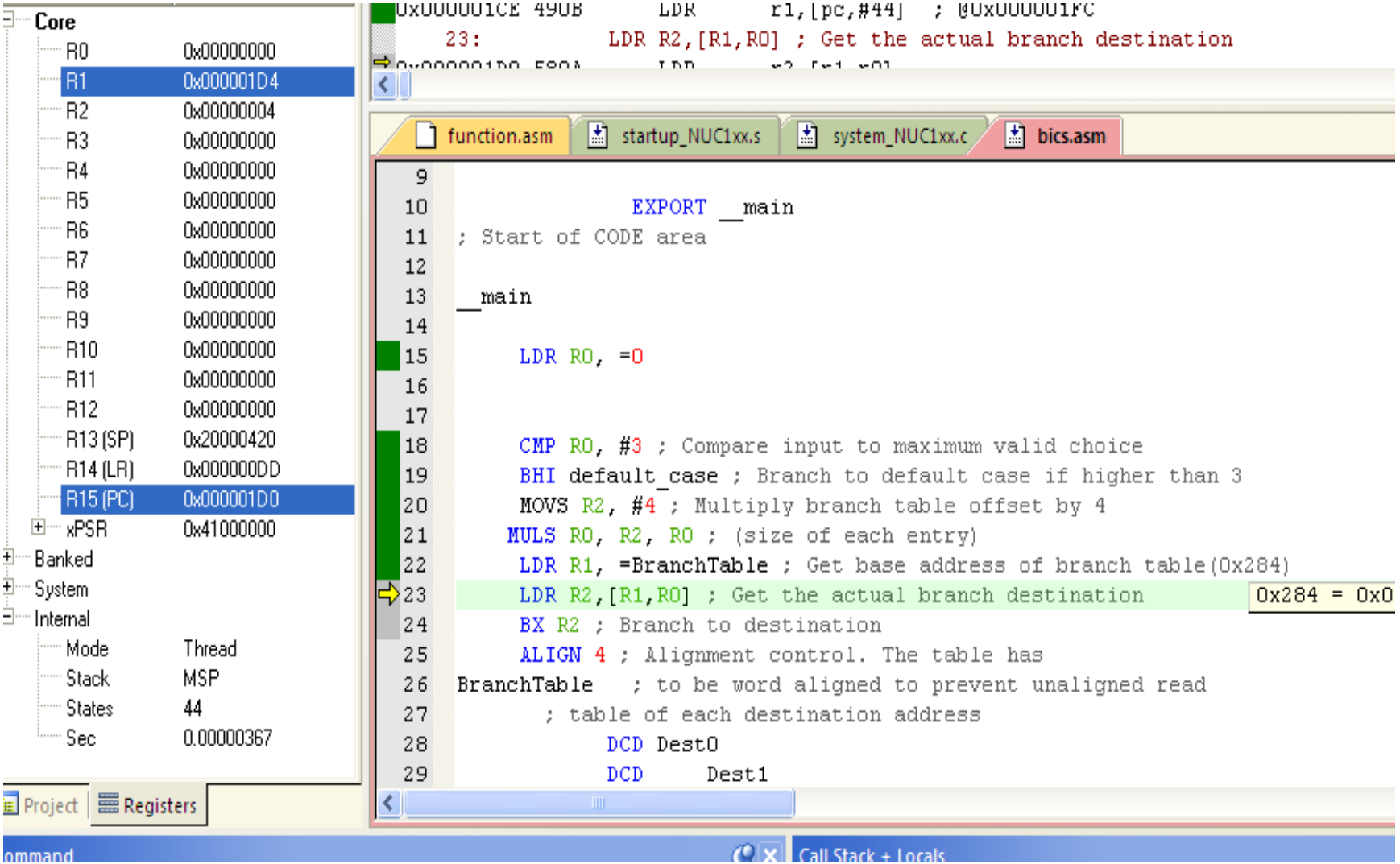
MULS R0, R2, R0 ; (size of each entry)

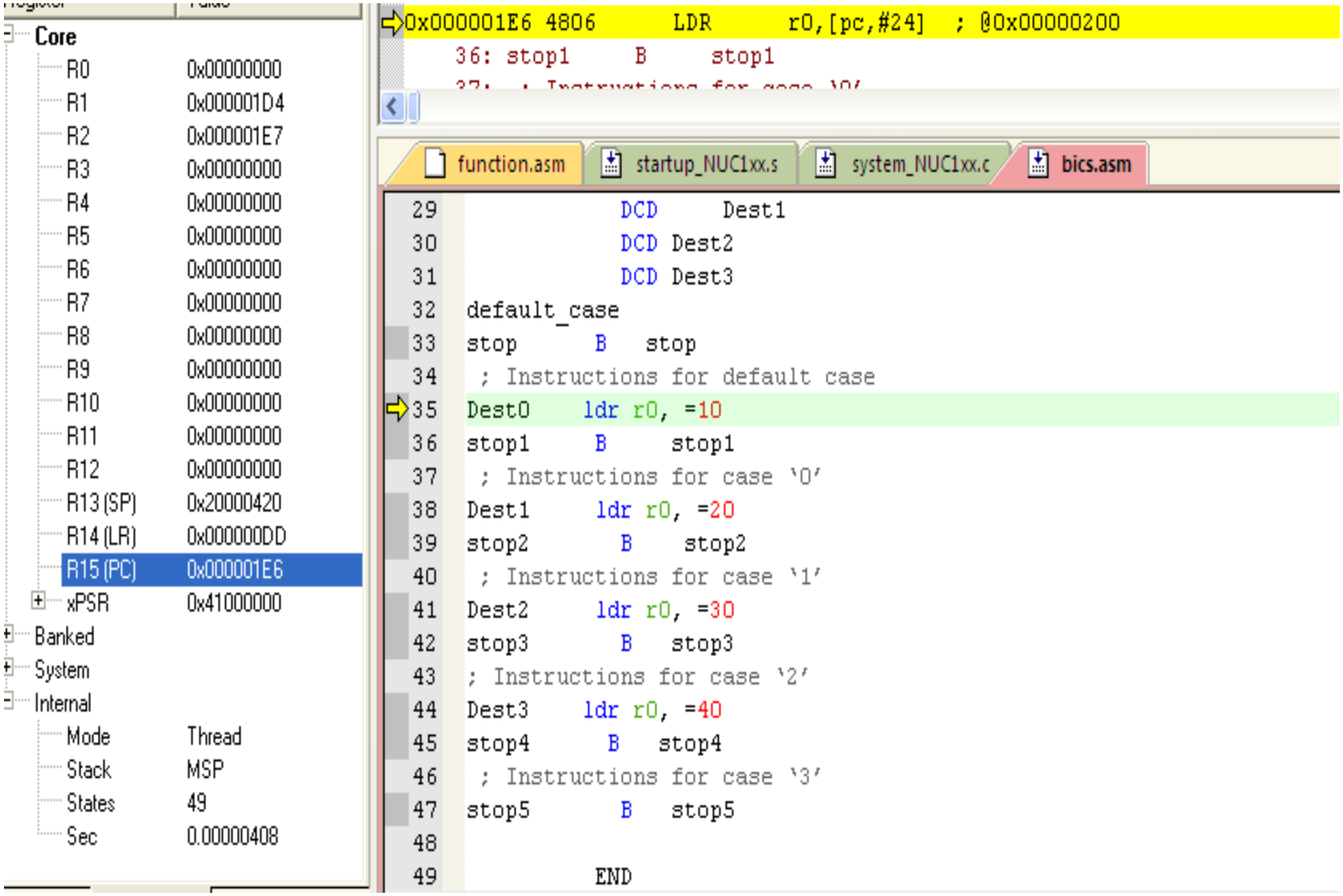
- LDR R1, =BranchTable ; Get base address of branch table(0x284)
- LDR R2,[R1,R0] ; Get the actual branch destination
- BX R2 ; Branch to destination
- ALIGN 4 ; Alignment control. The table has
- BranchTable ; to be word aligned to prevent unaligned read ;
-

Continued Switch

- table of each destination address
  - DCD Dest0
  - DCD Dest1
  - DCD Dest2
  - DCD Dest3

```
default_case
stop B stop ; Instructions for default case
Dest0 ldr r0, =10
 stop1 B stop1 ; Instructions for case ‘0’
Dest1 ldr r0, =20
 stop2 B stop2 ; Instructions for case ‘1’
Dest2 ldr r0, =30
stop3 B stop3 ; Instructions for case ‘2’
Dest3 ldr r0, =40
 stop4 B stop4 ; Instructions for case ‘3’
END
```


Core
R0 0x00000000
R1 0x000001D4
R2 0x00000004
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D0
xPSR 0x41000000
Banked
System
Internal
Mode Thread
Stack MSP
States 44
Sec 0.00000367
Project
Registers
0x00000ICE 490B LDR r1,[pc,#44] ; @0x00000IFC
23: LDR R2,[R1,R0] ; Get the actual branch destination
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
EXPORT __main
; Start of CODE area
__main
LDR R0, =0
CMP R0, #3 ; Compare input to maximum valid choice
BHI default_case ; Branch to default case if higher than 3
MOVS R2, #4 ; Multiply branch table offset by 4
MULS R0, R2, R0 ; (size of each entry)
LDR R1, =BranchTable ; Get base address of branch table(0x284)
LDR R2,[R1,R0] ; Get the actual branch destination
0x284 = 0x0
BX R2 ; Branch to destination
ALIGN 4 ; Alignment control. The table has
BranchTable ; to be word aligned to prevent unaligned read
; table of each destination address
DCD Dest0
DCD Dest1
Call Stack + Locals


Core
R0 0x00000000
R1 0x000001D4
R2 0x000001E7
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001E6
xPSR 0x41000000
Banked
System
Internal
Mode Thread
Stack MSP
States 49
Sec 0.00000408
0x000001E6 4806 LDR r0,[pc,#24] ; @0x00000200
36: stop1 B stop1
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
29 DCD Dest1
30 DCD Dest2
31 DCD Dest3
32 default_case
33 stop B stop
34 ; Instructions for default case
35 Dest0 ldr r0, =10
36 stop1 B stop1
37 ; Instructions for case '0'
38 Dest1 ldr r0, =20
39 stop2 B stop2
40 ; Instructions for case '1'
41 Dest2 ldr r0, =30
42 stop3 B stop3
43 ; Instructions for case '2'
44 Dest3 ldr r0, =40
45 stop4 B stop4
46 ; Instructions for case '3'
47 stop5 B stop5
48
49 END

RORS

Figure 5.6:
Rotate Right.

Instruction	ROR
Function	Rotate Right
Syntax (UAL)	RORS <Rd>, <Rd>, <Rm>
Syntax (Thumb)	ROR <Rd>, <Rm>
Note	Rd = Rd rotate right by Rm bits, last bit shifted out is copied to APSR.C, APSR.N and APSR.Z are also updated. Rd and Rm are low registers.

LSLS and LSRS

- For logical shift operations, the instructions are LSL (Figure 5.4) and LSR (Figure 5.5).

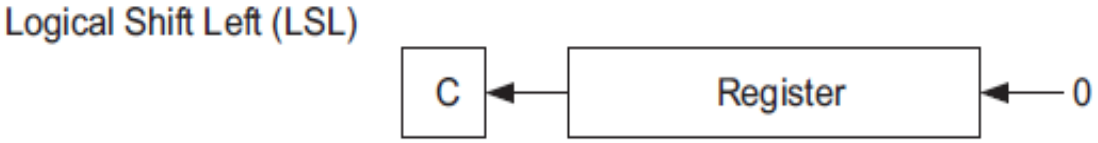
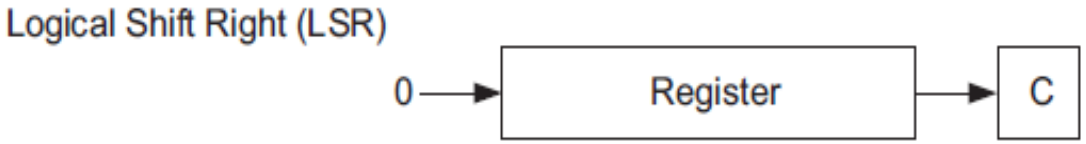


Figure 5.4:
Logical Shift Left.



Extraction of data value with W bits wide and P is the starting position

if we need to extract a bit field in a data value that is "W" bits wide, starting with bit positi "P" (LSB of the bit field), we can extract the bit field using the following instruction:

```
LSLS R0, R0,  #(32-W-P) ; Remove un-needed top bits
LSRS R0, R0,  #(32-W)   ; Align required bits to bit 0
```

For example, if we need to extract an 8-bit-width bit field from bit 4 to bit 11 (Figure 6.9), can use this instruction sequence:

```
LSLS R0, R0,  #(32-8-4) ; Remove un-needed top bits
LSRS R0, R0,  #(32-8)   ; Align required bits to bit 0
```

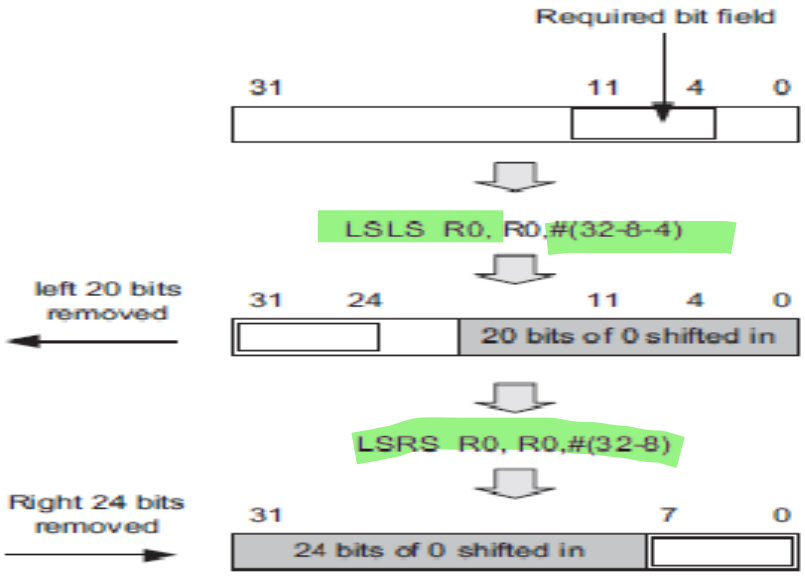


Figure 6.9:
Bit field extract operation.

Clearing the bits starting position P and width of the field being W

In a similar way, we can clear the bit field in a register by a few shift and rotate instructions (Figure 6.10):

```
RORS  R0, R0,  #4          ; Shift unneeded bit to bit 0
LSRS  R0, R0,  #8          ; Align required bits to bit 0
RORS  R0, R0,  #(32-8-4)   ; Store value to original position
```

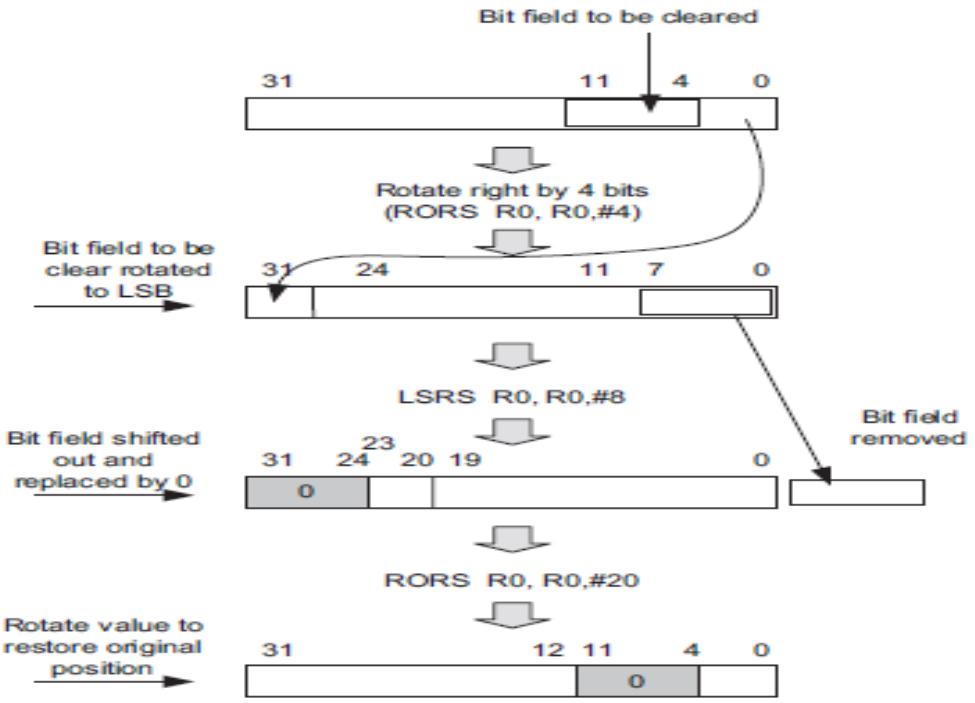
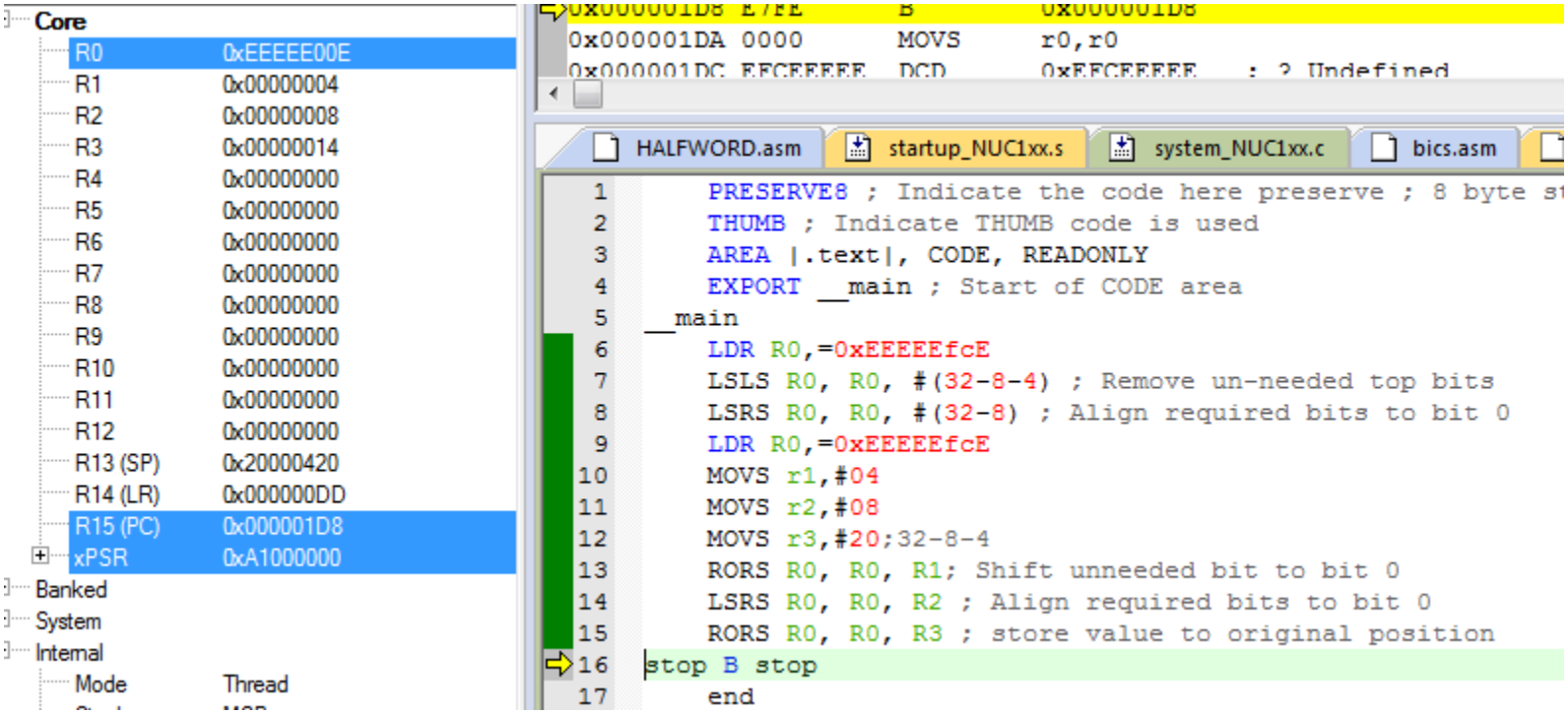


Figure 6.10:
Bit field clear operation.

Extracting and Clearing the bits starting position P and width of the field being W



ASR

Instruction	ASR
Syntax (Thumb)	ASR <Rd>, <Rm>, #immed5
Note	Rd = Rm >> immed5, last bit shifted out is copied to APSR.C, APSR.N and APSR.Z are also updated. Rd and Rm are low registers.

When ASR is used, the MSB of the result is unchanged, and the Carry flag is updated using the last bit shifted out (Figure 5.3).

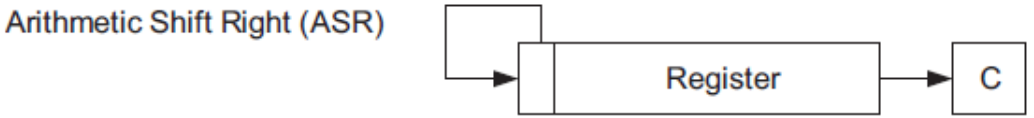
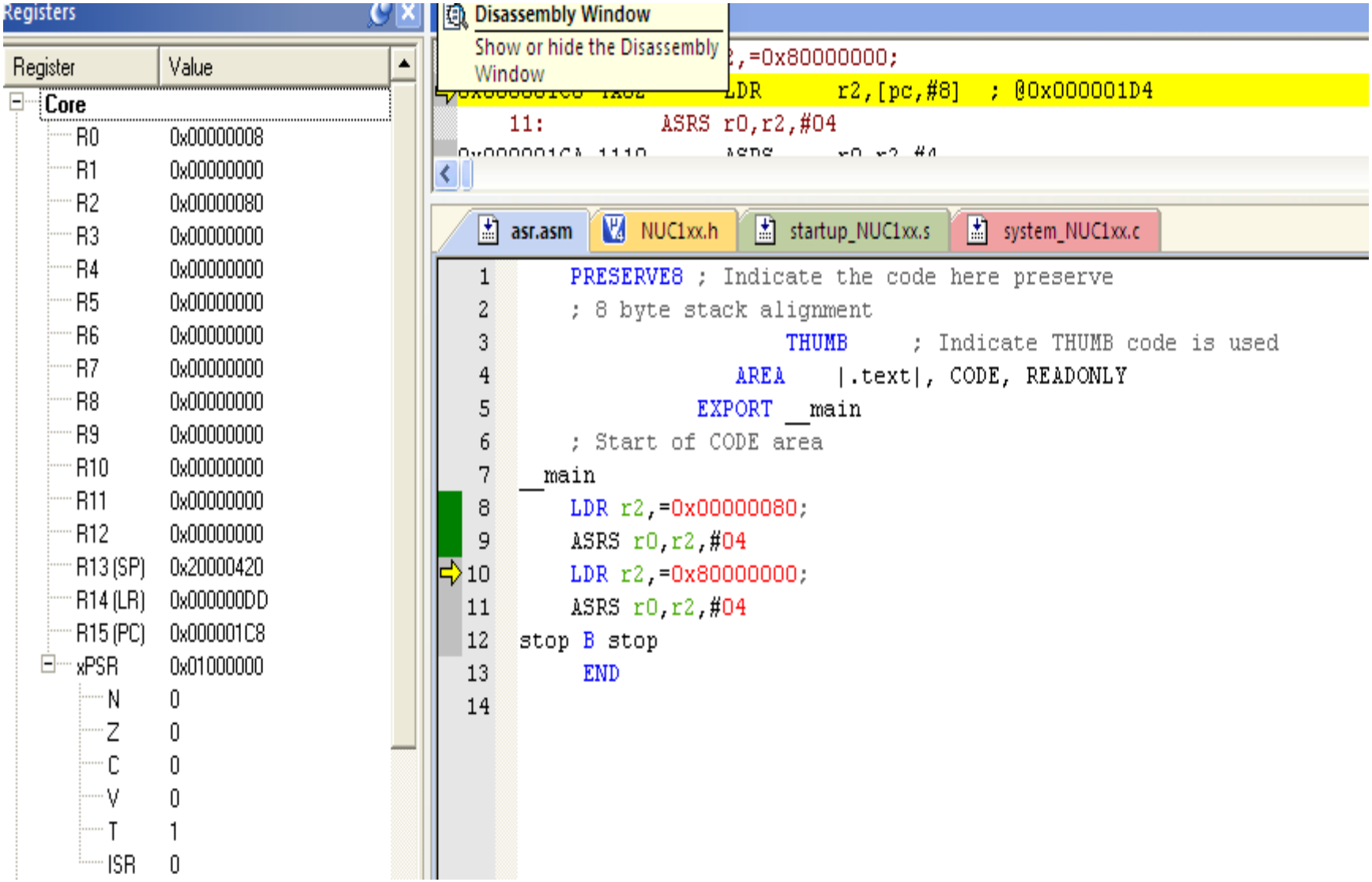
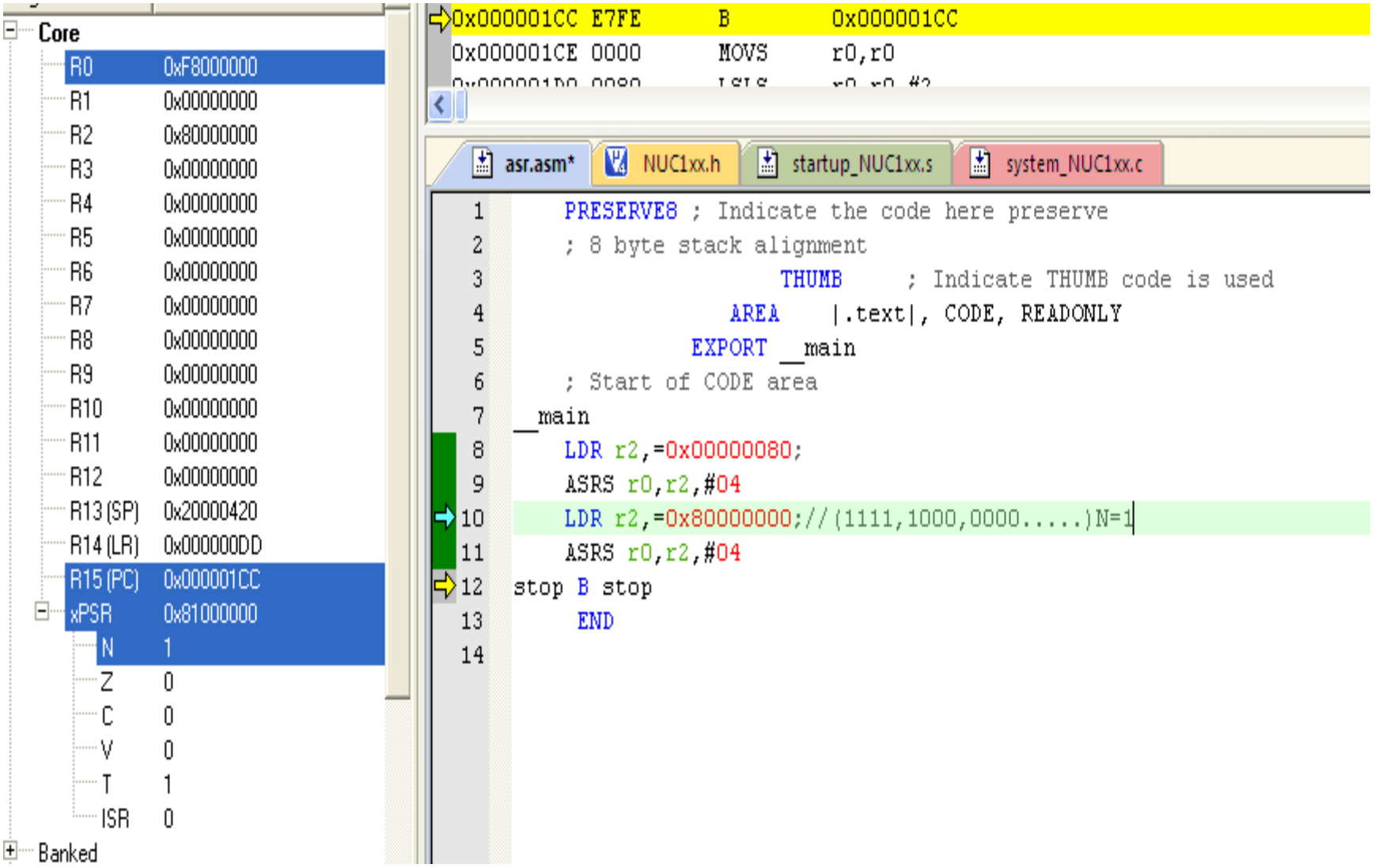


Figure 5.3:
Arithmetic Shift Right

ASR

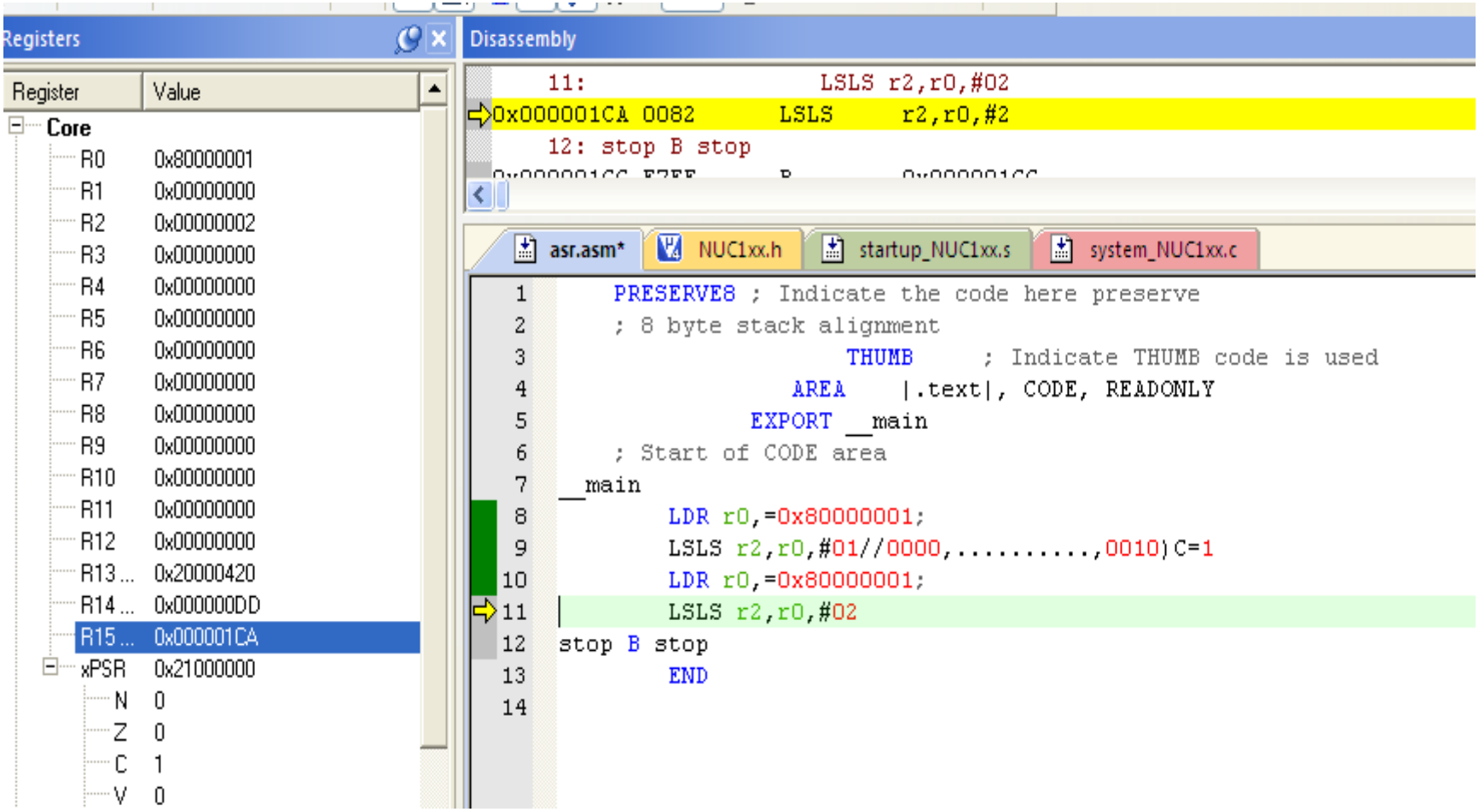
```
PRESERVE8 ; Indicate the code here preserve
        ; 8 byte stack alignment
                THUMB     ; Indicate THUMB code is used
              AREA    |.text|, CODE, READONLY
            EXPORT __main
        ; Start of CODE area
__main
        LDR r2,=0x00000080;
        ASRS r0,r2,#04
        LDR r2,=0x80000000;
        ASRS r0,r2,#04
stop B stop
     END
```

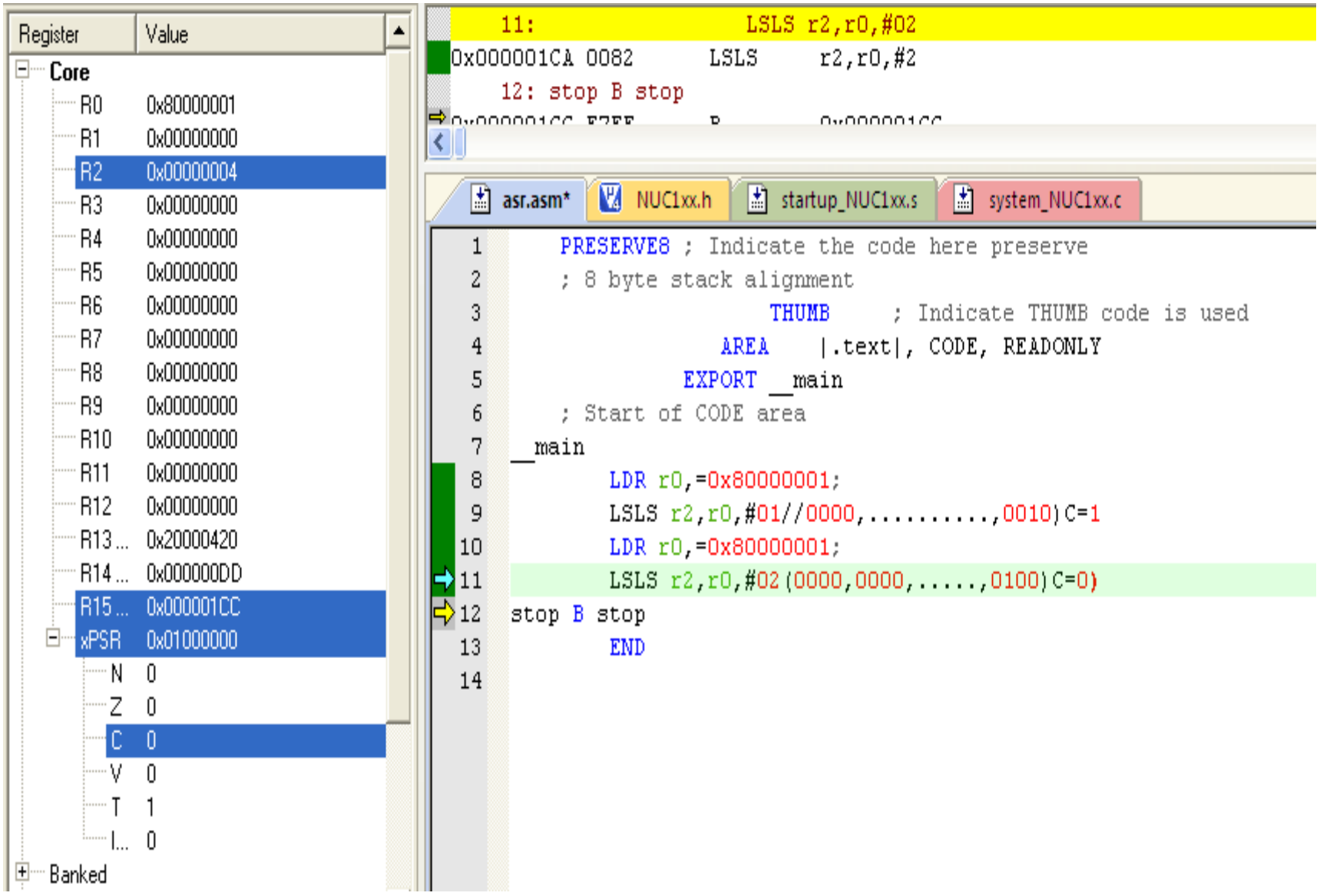

Registers
Register
Value
Core
R0 0x00000008
R1 0x00000000
R2 0x00000080
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001C8
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
ISR 0
Disassembly Window
Show or hide the Disassembly Window
,=0x80000000;
LDR r2,[pc,#8] ; @0x000001D4
11: ASRS r0,r2,#04
asr.asm
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r2,=0x00000080;
9 ASRS r0,r2,#04
10 LDR r2,=0x80000000;
11 ASRS r0,r2,#04
12 stop B stop
13 END
14


Core
R0 0xF8000000
R1 0x00000000
R2 0x80000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CC
xPSR 0x81000000
N 1
Z 0
C 0
V 0
T 1
ISR 0
Banked
0x000001CC E7FE B 0x000001CC
0x000001CE 0000 MOVS r0,r0
asr.asm*
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r2,=0x00000080;
9 ASRS r0,r2,#04
10 LDR r2,=0x80000000;//(1111,1000,0000.....)N=1
11 ASRS r0,r2,#04
12 stop B stop
13 END
14

LSLS

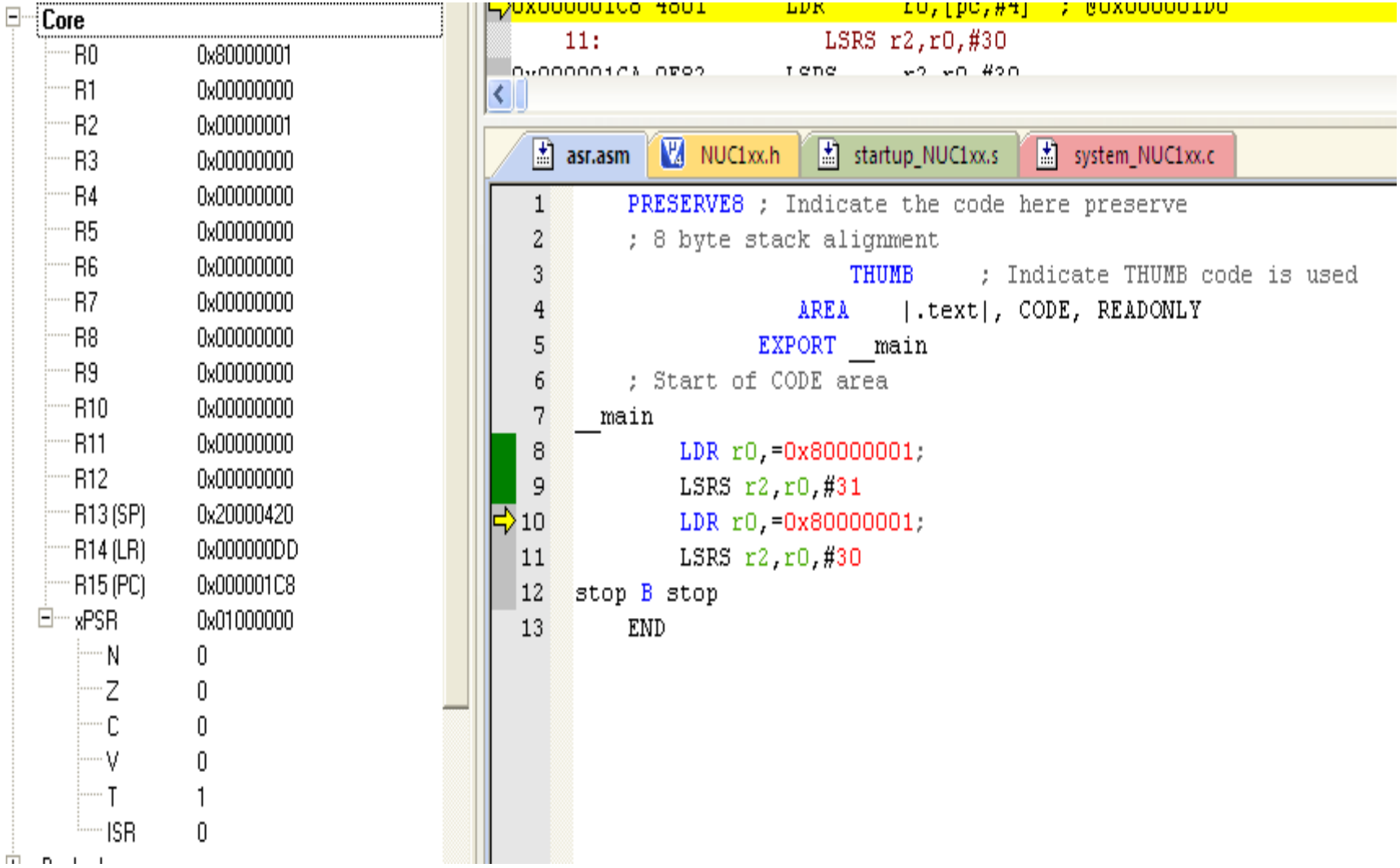
```
PRESERVE8 ; Indicate the code here preserve
        ; 8 byte stack alignment
                THUMB     ; Indicate THUMB code is used
              AREA    |.text|, CODE, READONLY
            EXPORT __main
        ; Start of CODE area
__main
                    LDR r0,=0x80000001;
                    LSLS r2,r0,#01
                    LDR r0,=0x80000001;
                    LSLS r2,r0,#02
stop B stop
                    END
```

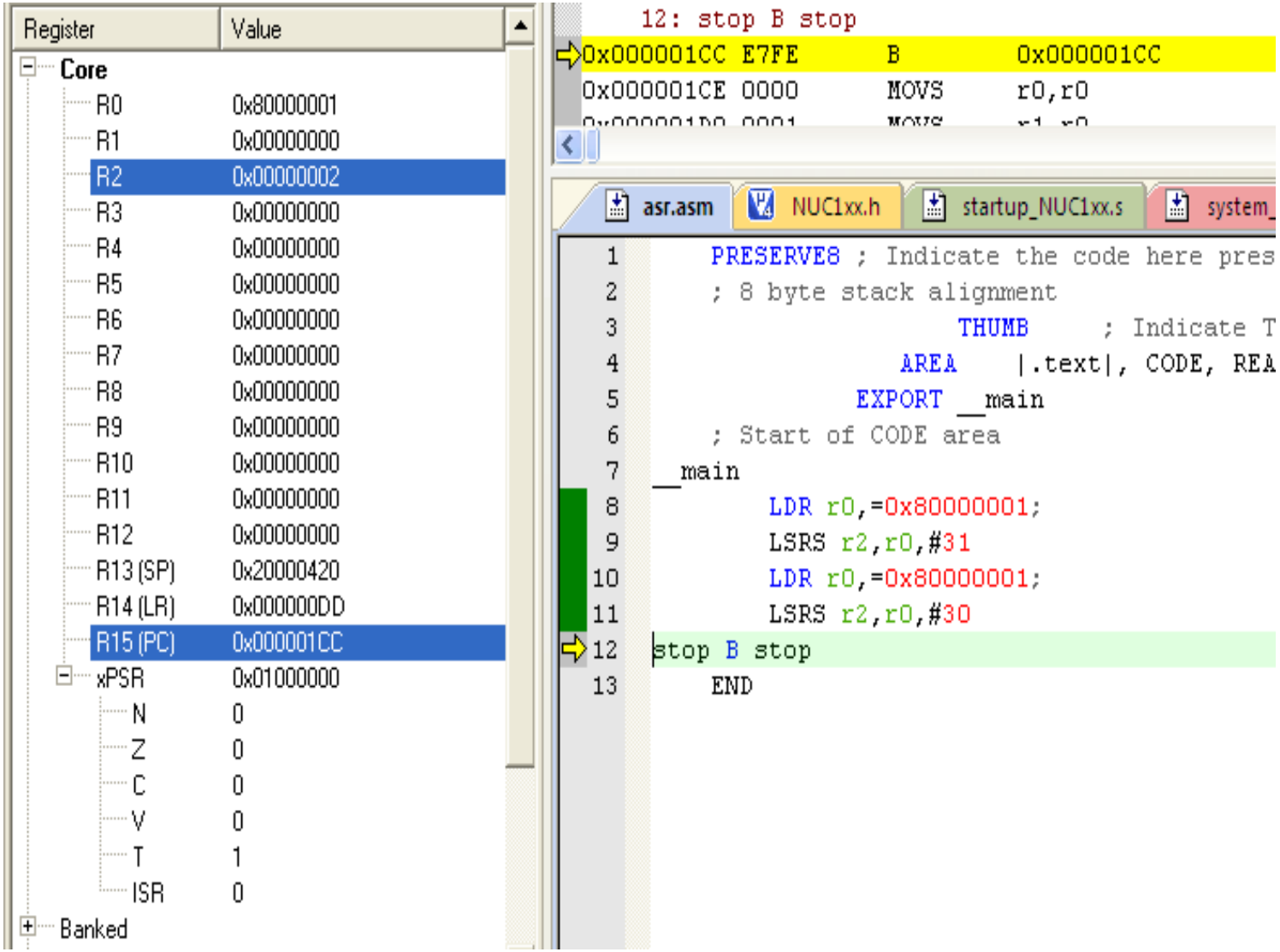

Registers
Register
Value
Core
R0 0x80000001
R1 0x00000000
R2 0x00000002
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 ... 0x20000420
R14 ... 0x000000DD
R15 ... 0x000001CA
xPSR 0x21000000
N 0
Z 0
C 1
V 0
Disassembly
11: LSLS r2,r0,#02
0x000001CA 0082 LSLS r2,r0,#2
12: stop B stop
asr.asm*
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 LSLS r2,r0,#01//0000,.....,0010)C=1
10 LDR r0,=0x80000001;
11 LSLS r2,r0,#02
12 stop B stop
13 END
14


Register
Value
Core
R0 0x80000001
R1 0x00000000
R2 0x00000004
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 ... 0x20000420
R14 ... 0x000000DD
R15 ... 0x000001CC
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
I... 0
Banked
11: LSLS r2,r0,#02
0x000001CA 0082 LSLS r2,r0,#2
12: stop B stop
asr.asm*
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 LSLS r2,r0,#01//0000,.....,0010)C=1
10 LDR r0,=0x80000001;
11 LSLS r2,r0,#02(0000,0000,.....,0100)C=0)
12 stop B stop
13 END
14

LSRS

```
PRESERVE8 ; Indicate the code here preserve
        ; 8 byte stack alignment
                THUMB     ; Indicate THUMB code is used
              AREA    |.text|, CODE, READONLY
            EXPORT __main
        ; Start of CODE area
__main
                    LDR r0,=0x80000001;
                    LSRS r2,r0,#31
                    LDR r0,=0x80000001;
                    LSRS r2,r0,#30
stop B stop
    END
```


Core
R0 0x80000001
R1 0x00000000
R2 0x00000001
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001C8
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
ISR 0
11: LSRS r2,r0,#30
asr.asm
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 LSRS r2,r0,#31
10 LDR r0,=0x80000001;
11 LSRS r2,r0,#30
12 stop B stop
13 END


Register
Value
Core
R0 0x80000001
R1 0x00000000
R2 0x00000002
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CC
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
ISR 0
Banked
12: stop B stop
0x000001CC E7FE B 0x000001CC
0x000001CE 0000 MOVS r0,r0
asr.asm
NUC1xx.h
startup_NUC1xx.s
system_
1 PRESERVE8 ; Indicate the code here pres
2 ; 8 byte stack alignment
3 THUMB ; Indicate T
4 AREA |.text|, CODE, REA
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 LSRS r2,r0,#31
10 LDR r0,=0x80000001;
11 LSRS r2,r0,#30
12 stop B stop
13 END

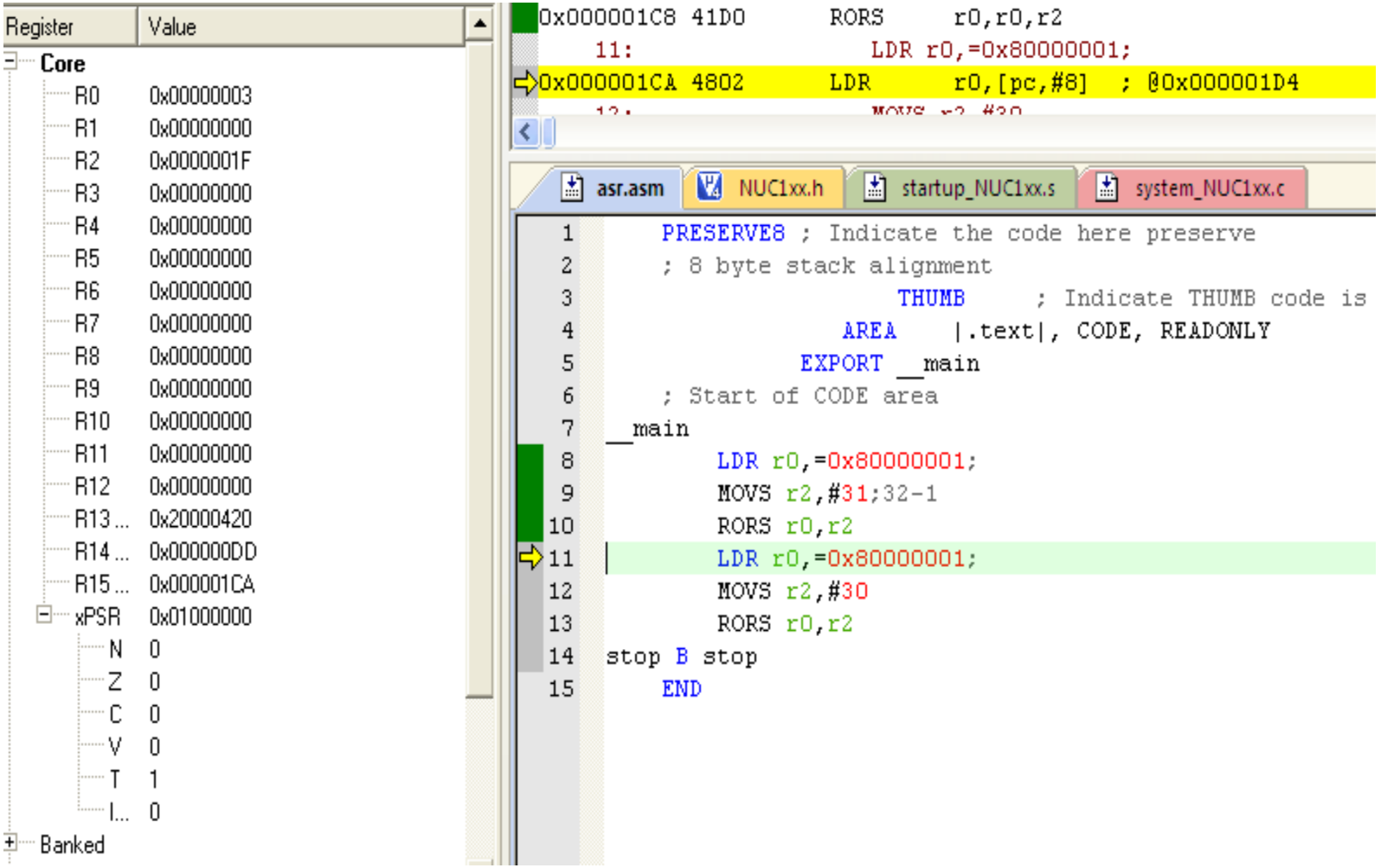
RORS

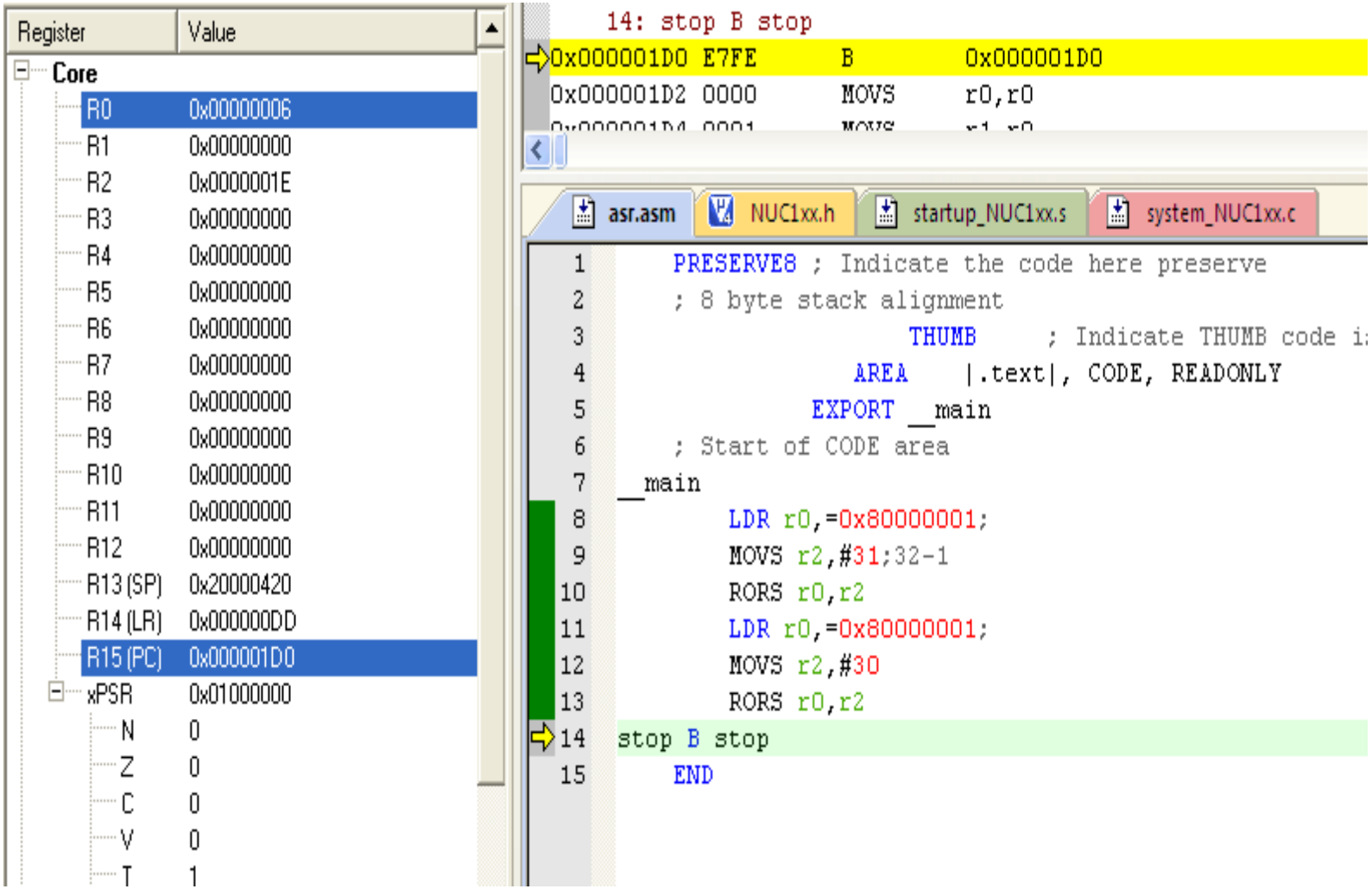
Figure 5.6:
Rotate Right.

Instruction	ROR
Function	Rotate Right
Syntax (UAL)	RORS <Rd>, <Rd>, <Rm>
Syntax (Thumb)	ROR <Rd>, <Rm>
Note	Rd = Rd rotate right by Rm bits, last bit shifted out is copied to APSR.C, APSR.N and APSR.Z are also updated. Rd and Rm are low registers.

RORS

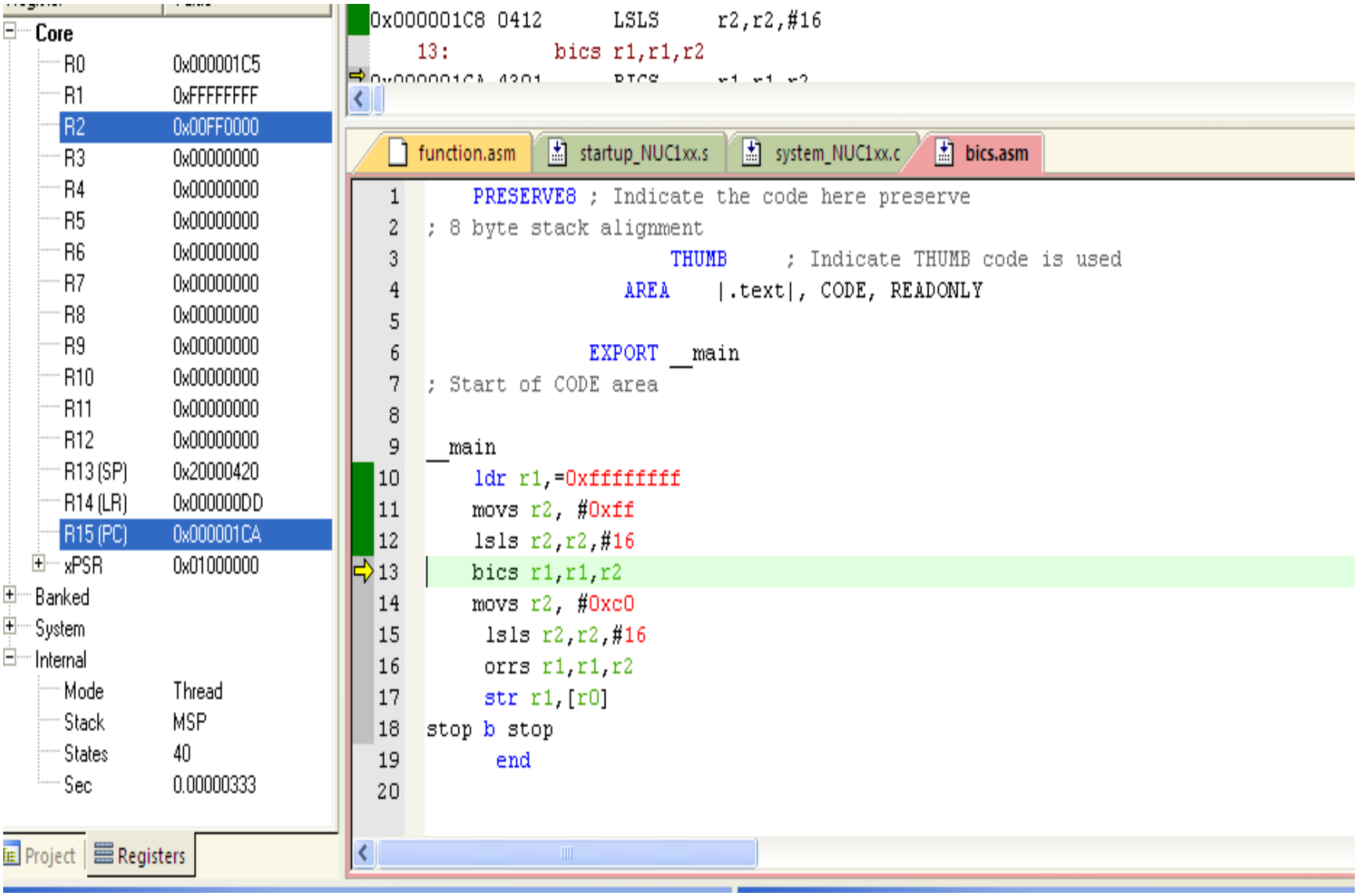
```
PRESERVE8 ; Indicate the code here preserve
        ; 8 byte stack alignment
                THUMB    ; Indicate THUMB code is used
            AREA    |.text|, CODE, READONLY
          EXPORT __main
        ; Start of CODE area
__main
                    LDR r0,=0x80000001;
                    MOVS r2,#31;32-1
                    RORS r0,r2
                    LDR r0,=0x80000001;
                    MOVS r2,#30
                    RORS r0,r2
stop B stop
   END
```

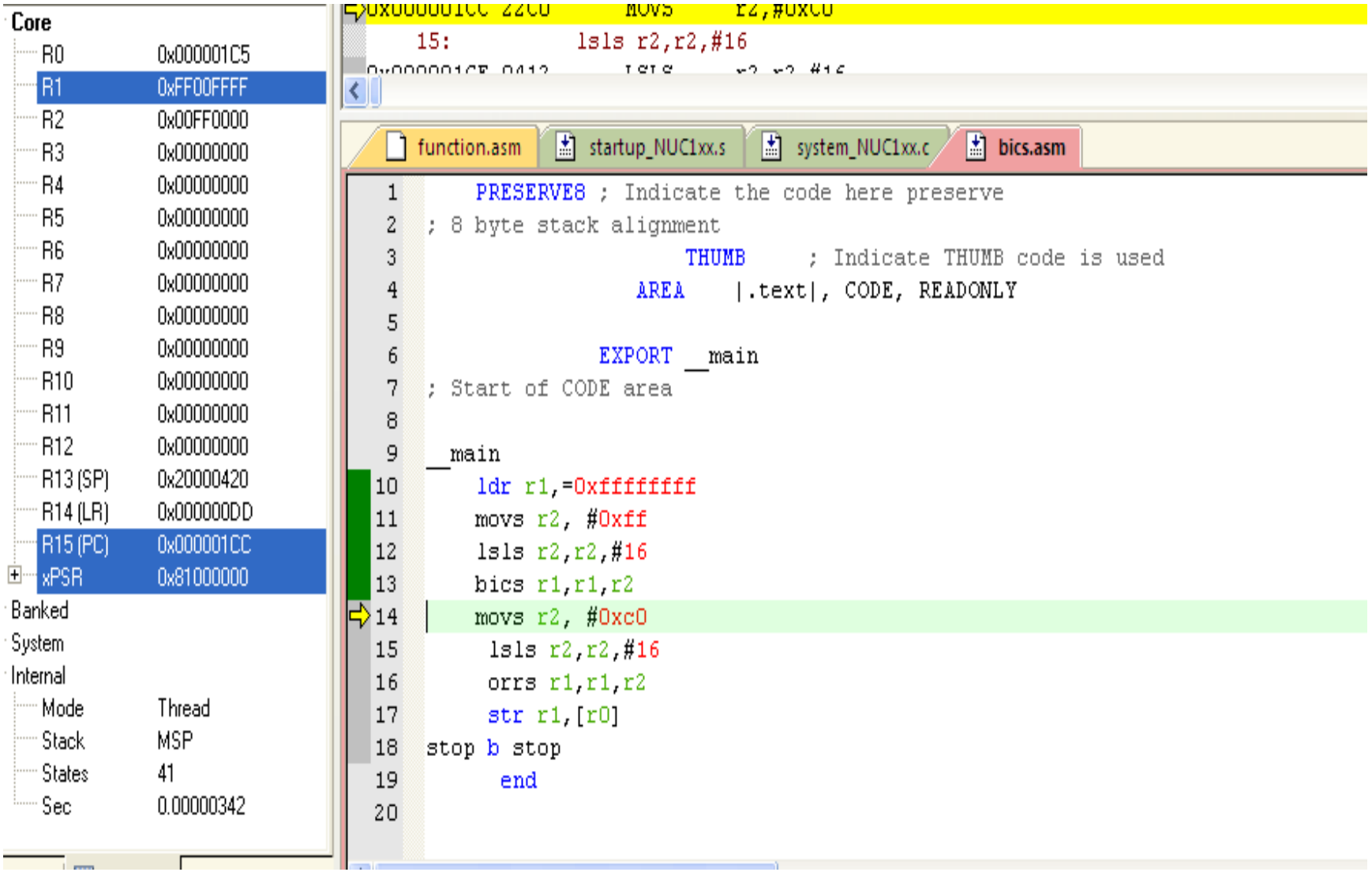

Register
Value
Core
R0 0x00000003
R1 0x00000000
R2 0x0000001F
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 ... 0x20000420
R14 ... 0x000000DD
R15 ... 0x000001CA
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
I... 0
Banked
0x000001C8 41D0 RORS r0,r0,r2
11: LDR r0,=0x80000001;
0x000001CA 4802 LDR r0,[pc,#8] ; @0x000001D4
asr.asm
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 MOVS r2,#31;32-1
10 RORS r0,r2
11 LDR r0,=0x80000001;
12 MOVS r2,#30
13 RORS r0,r2
14 stop B stop
15 END

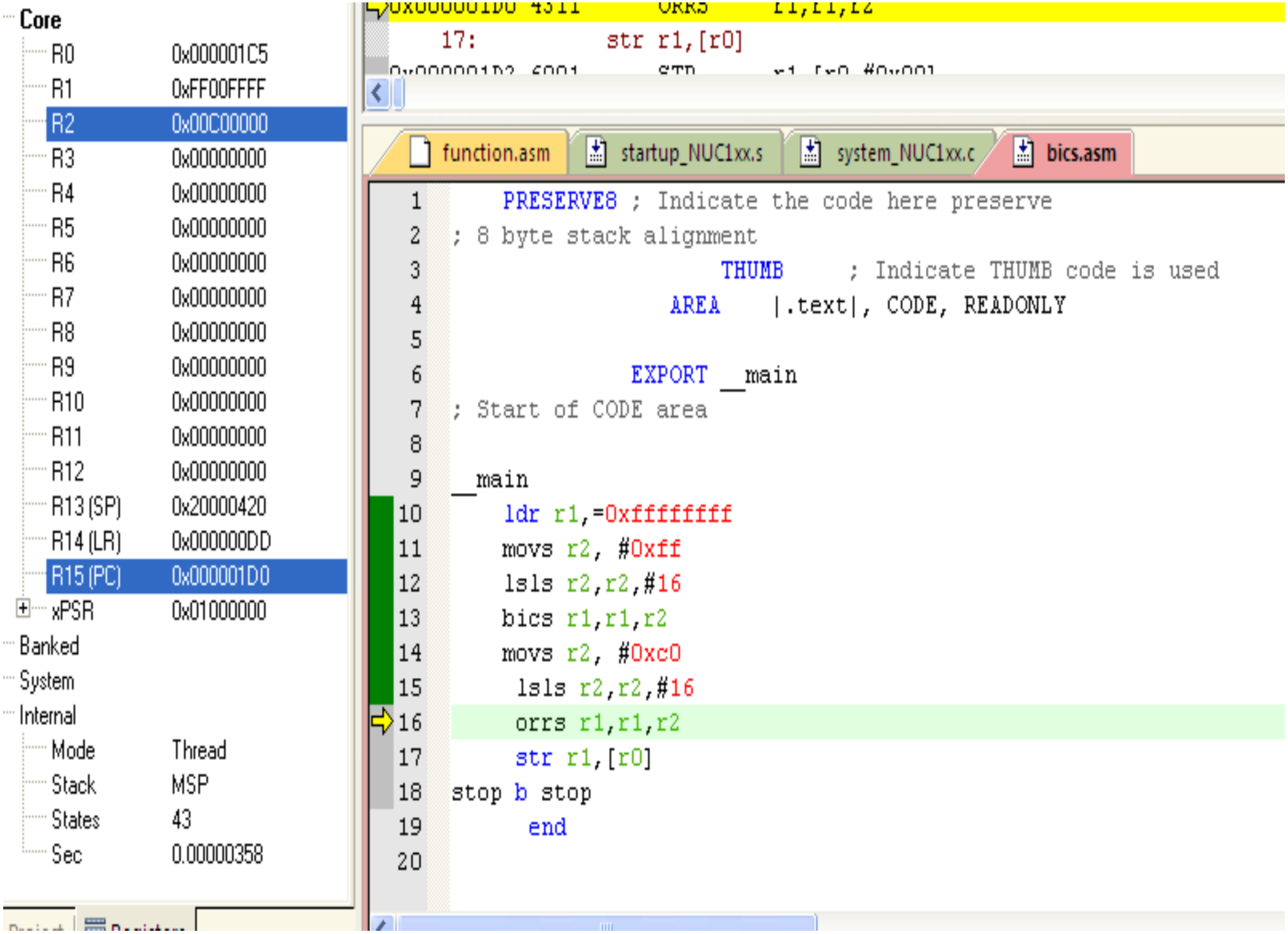

Register
Value
Core
R0 0x00000006
R1 0x00000000
R2 0x0000001E
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D0
xPSR 0x01000000
N 0
Z 0
C 0
V 0
T 1
14: stop B stop
0x000001D0 E7FE B 0x000001D0
0x000001D2 0000 MOVS r0,r0
asr.asm
NUC1xx.h
startup_NUC1xx.s
system_NUC1xx.c
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code i
4 AREA |.text|, CODE, READONLY
5 EXPORT __main
6 ; Start of CODE area
7 __main
8 LDR r0,=0x80000001;
9 MOVS r2,#31;32-1
10 RORS r0,r2
11 LDR r0,=0x80000001;
12 MOVS r2,#30
13 RORS r0,r2
14 stop B stop
15 END

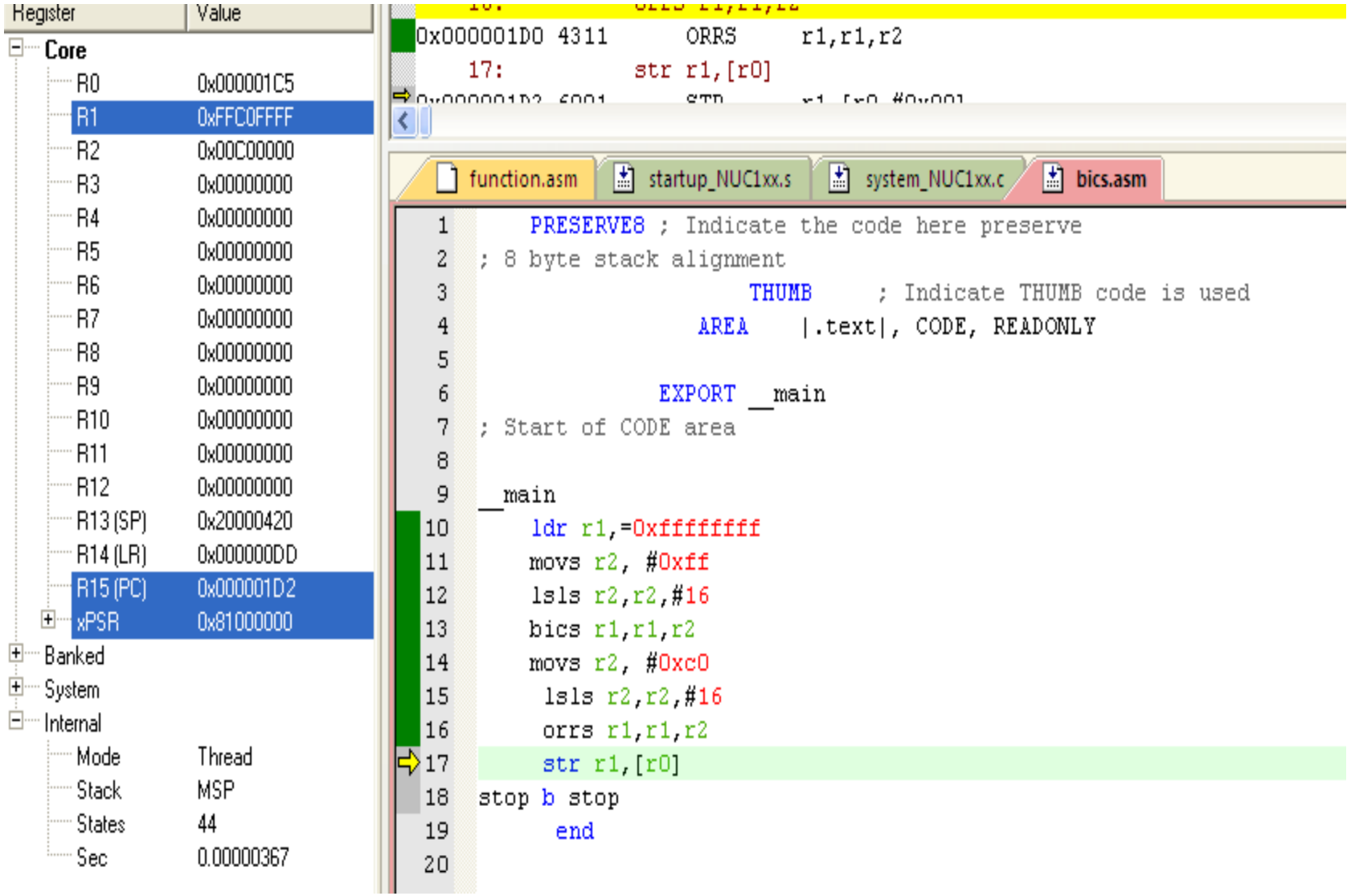
BIC (Logical Bitwise Clear)

Instruction	BIC
Function	Logical Bitwise Clear
Syntax (UAL)	BICS <Rd>, <Rd>, <Rm>
Syntax (Thumb)	BIC <Rd>, <Rm>
Note	Rd = AND(Rd, NOT(Rm)), APSR.N, and APSR.Z update. Rd and Rm are low registers.


Core
R0 0x000001C5
R1 0xFFFFFFFF
R2 0x00FF0000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CA
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Stack MSP
States 40
Sec 0.00000333
Project
Registers
0x000001C8 0412 LSLS r2,r2,#16
13: bics r1,r1,r2
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Core
R0 0x000001C5
R1 0xFF00FFFF
R2 0x00FF0000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001CC
xPSR 0x81000000
Banked
System
Internal
Mode Thread
Stack MSP
States 41
Sec 0.00000342
0x000001CC 22C0 MOVS r2,#0xC0
15: lsls r2,r2,#16
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Core
R0 0x000001C5
R1 0xFF00FFFF
R2 0x00C00000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D0
xPSR 0x01000000
Banked
System
Internal
Mode Thread
Stack MSP
States 43
Sec 0.00000358
17: str r1,[r0]
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20


Register
Value
Core
R0 0x000001C5
R1 0xFFC0FFFF
R2 0x00C00000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000420
R14 (LR) 0x000000DD
R15 (PC) 0x000001D2
xPSR 0x81000000
Banked
System
Internal
Mode Thread
Stack MSP
States 44
Sec 0.00000367
0x000001D0 4311 ORRS r1,r1,r2
17: str r1,[r0]
function.asm
startup_NUC1xx.s
system_NUC1xx.c
bics.asm
1 PRESERVE8 ; Indicate the code here preserve
2 ; 8 byte stack alignment
3 THUMB ; Indicate THUMB code is used
4 AREA |.text|, CODE, READONLY
5
6 EXPORT __main
7 ; Start of CODE area
8
9 __main
10 ldr r1,=0xffffffff
11 movs r2, #0xff
12 lsls r2,r2,#16
13 bics r1,r1,r2
14 movs r2, #0xc0
15 lsls r2,r2,#16
16 orrs r1,r1,r2
17 str r1,[r0]
18 stop b stop
19 end
20