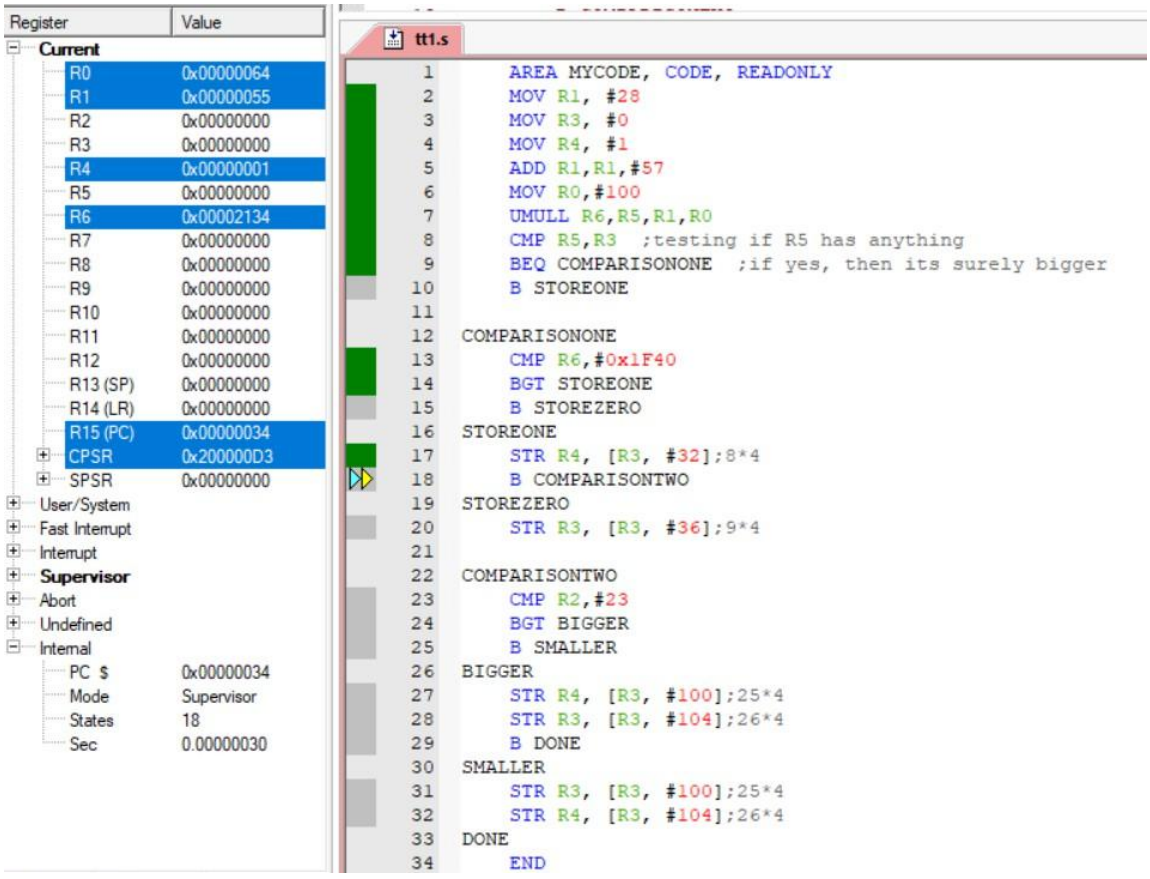
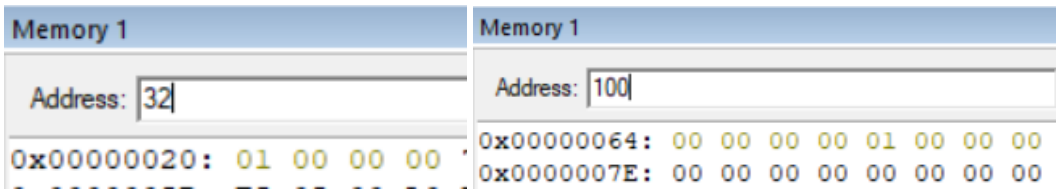
Problem 1

Take any number *a* in your mind and move it to R1 register. Add 57 with it. Then multiply (unsigned) with 100. If the result is (signed) greater than 0x1F40, store 1 to mem[8], otherwise store 0 to mem[9]. If you assume a>23, mem[25] should be 1 and mem[26] should be 0. Else mem[25] should be 0 and mem[26] should be 1.

```
AREA MYCODE, CODE, READONLY

      ; Write Your Code Here

LABEL B LABEL; Program gets stuck here

END; End of the program
```

Solution:

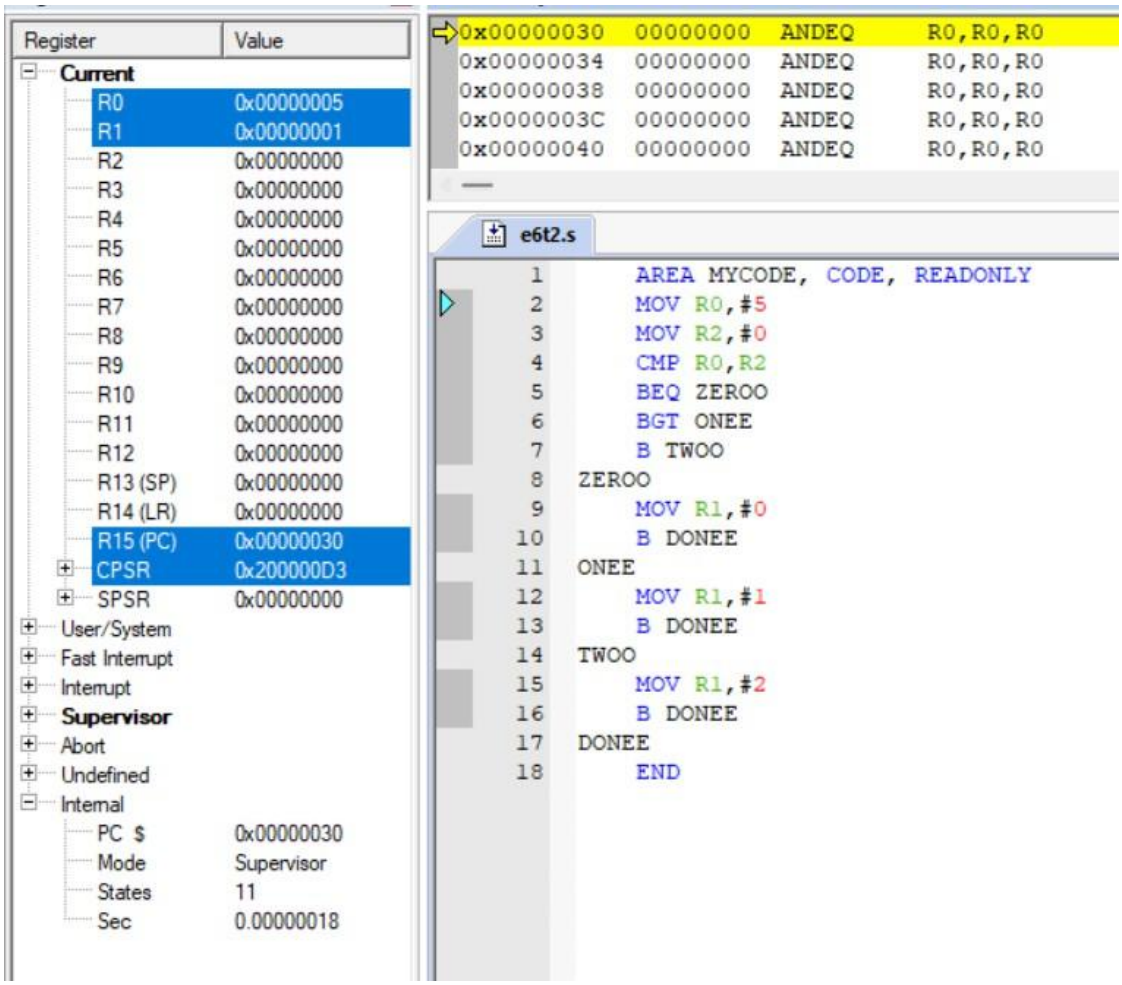


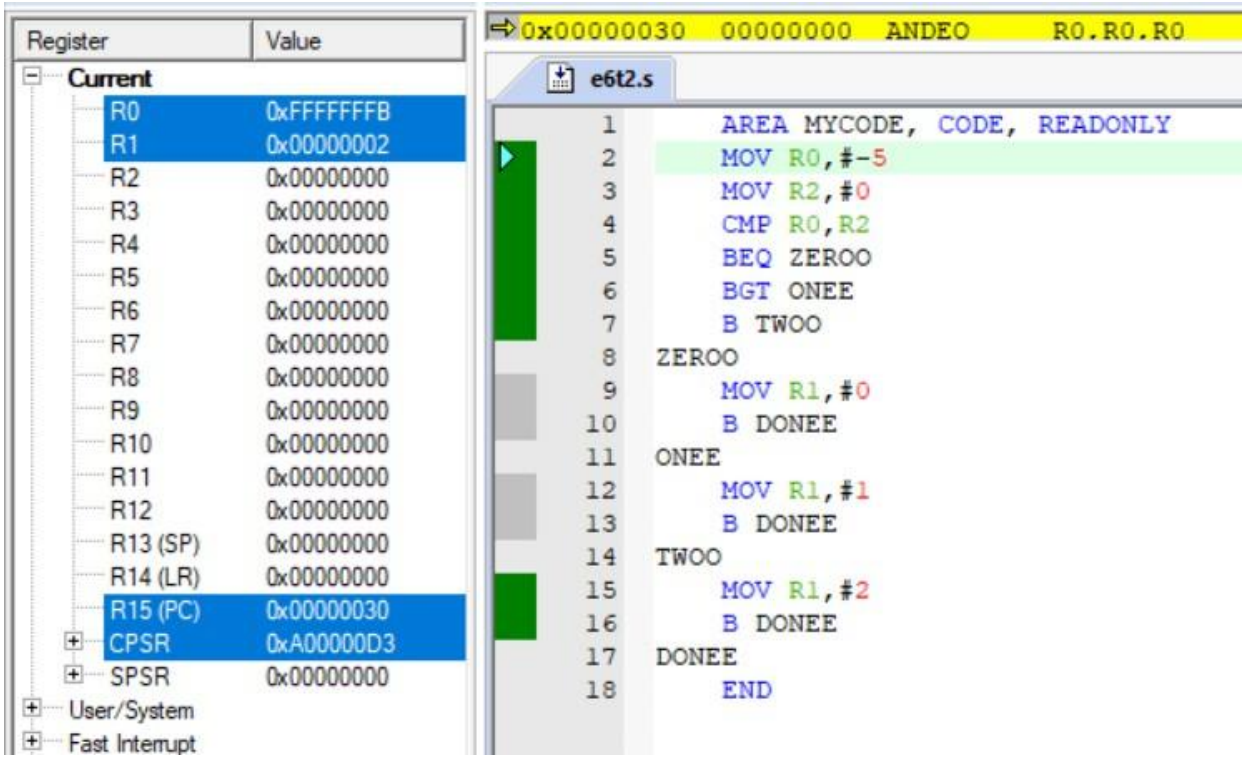


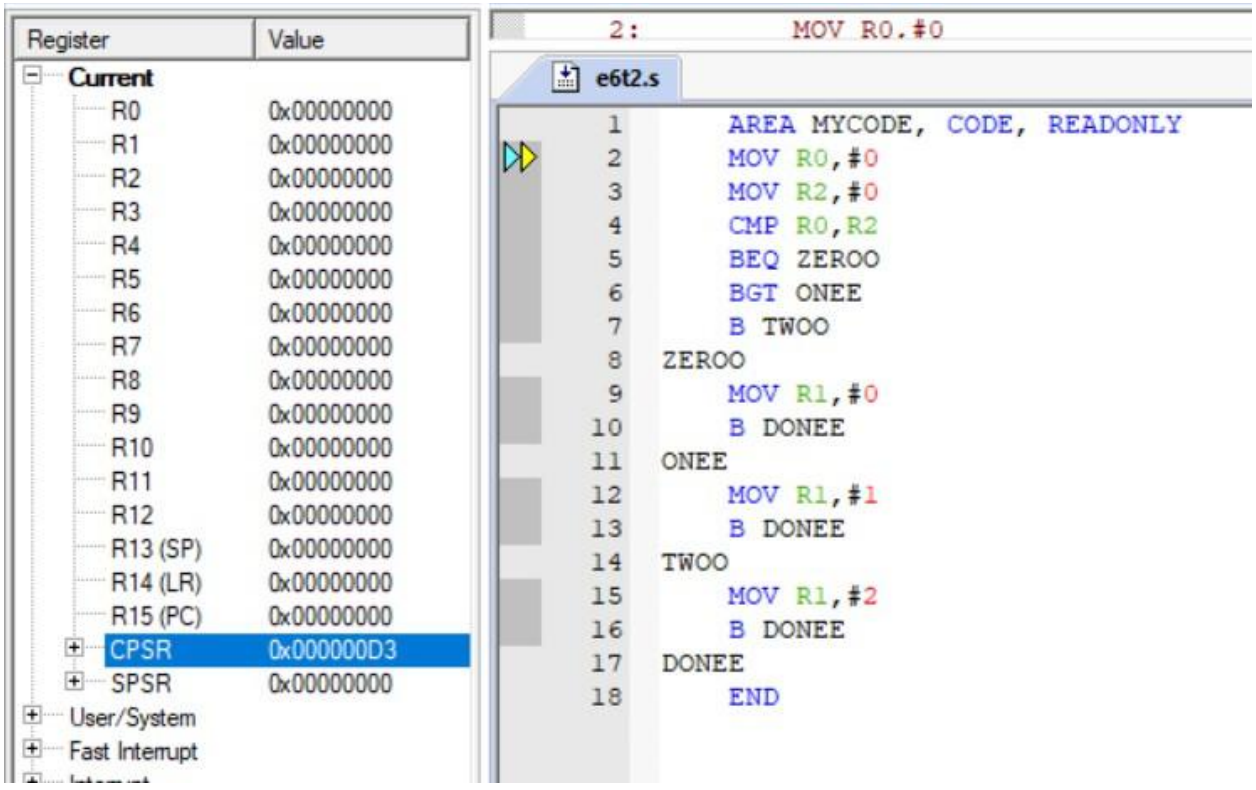
Problem 2

Write a code in Arm Assembly language that will store an integer value in register R0. If the value is positive, it will store 1 in R1, if negative then it will store 2 in R1, otherwise R1 will have 0 value.

Solution:



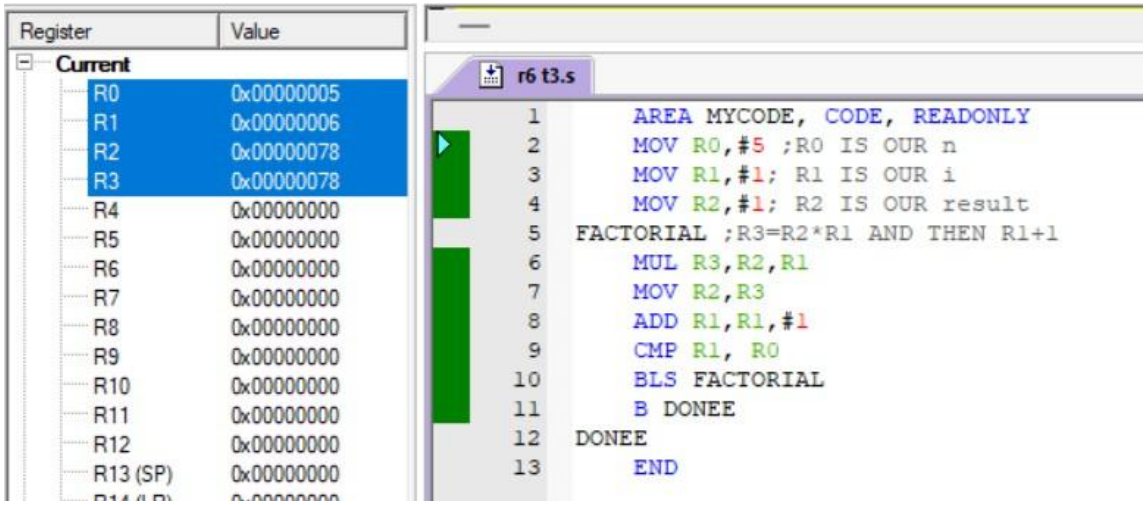

Register
Value
Current
R0 0xFFFFFFFB
R1 0x00000002
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000030
CPSR 0xA00000D3
SPSR 0x00000000
User/System
Fast Interrupt
0x00000030 00000000 ANDEO R0.R0.R0
e6t2.s
1 AREA MYCODE, CODE, READONLY
2 MOV R0,#-5
3 MOV R2,#0
4 CMP R0,R2
5 BEQ ZEROO
6 BGT ONEE
7 B TWOO
8 ZEROO
9 MOV R1,#0
10 B DONEE
11 ONEE
12 MOV R1,#1
13 B DONEE
14 TWOO
15 MOV R1,#2
16 B DONEE
17 DONEE
18 END


Register
Value
Current
R0 0x00000000
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x00000000
R15 (PC) 0x00000000
CPSR 0x000000D3
SPSR 0x00000000
User/System
Fast Interrupt
2: MOV R0.#0
e6t2.s
1 AREA MYCODE, CODE, READONLY
2 MOV R0,#0
3 MOV R2,#0
4 CMP R0,R2
5 BEQ ZEROO
6 BGT ONEE
7 B TWOO
8 ZEROO
9 MOV R1,#0
10 B DONEE
11 ONEE
12 MOV R1,#1
13 B DONEE
14 TWOO
15 MOV R1,#2
16 B DONEE
17 DONEE
18 END

Problem 3

Write an Assembly Language Code to compute the factorial of a given integer n.

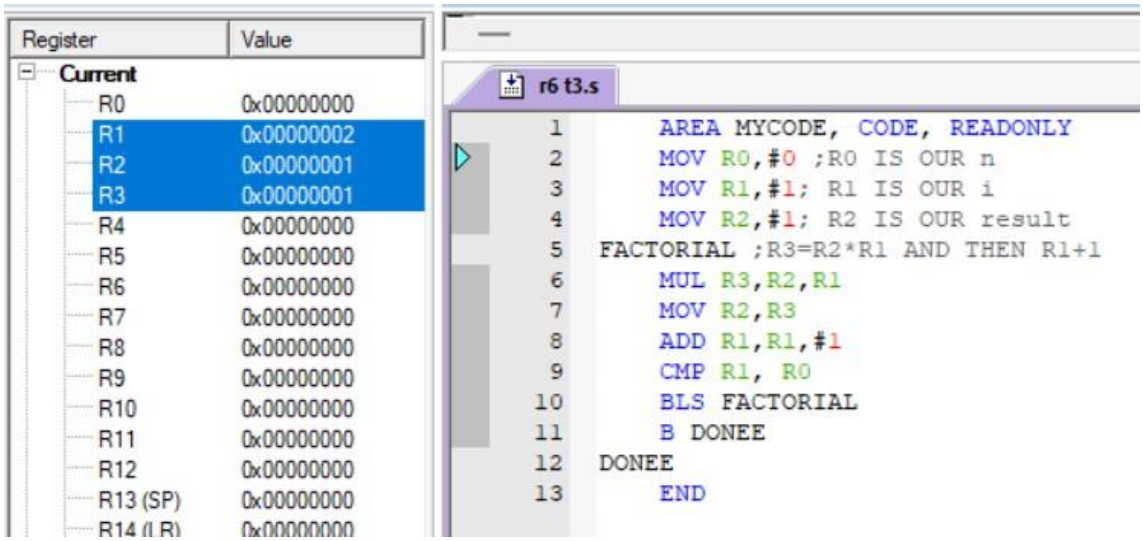
Solution



Register	Value
Current	
R0	0x00000005
R1	0x00000006
R2	0x00000078
R3	0x00000078
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000

r6 t3.s

```
1      AREA MYCODE, CODE, READONLY
2      MOV R0,#5 ;R0 IS OUR n
3      MOV R1,#1; R1 IS OUR i
4      MOV R2,#1; R2 IS OUR result
5  FACTORIAL ;R3=R2*R1 AND THEN R1+1
6      MUL R3,R2,R1
7      MOV R2,R3
8      ADD R1,R1,#1
9      CMP R1, R0
10     BLS FACTORIAL
11     B DONEE
12 DONEE
13     END
```



Register	Value
Current	
R0	0x00000000
R1	0x00000002
R2	0x00000001
R3	0x00000001
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000

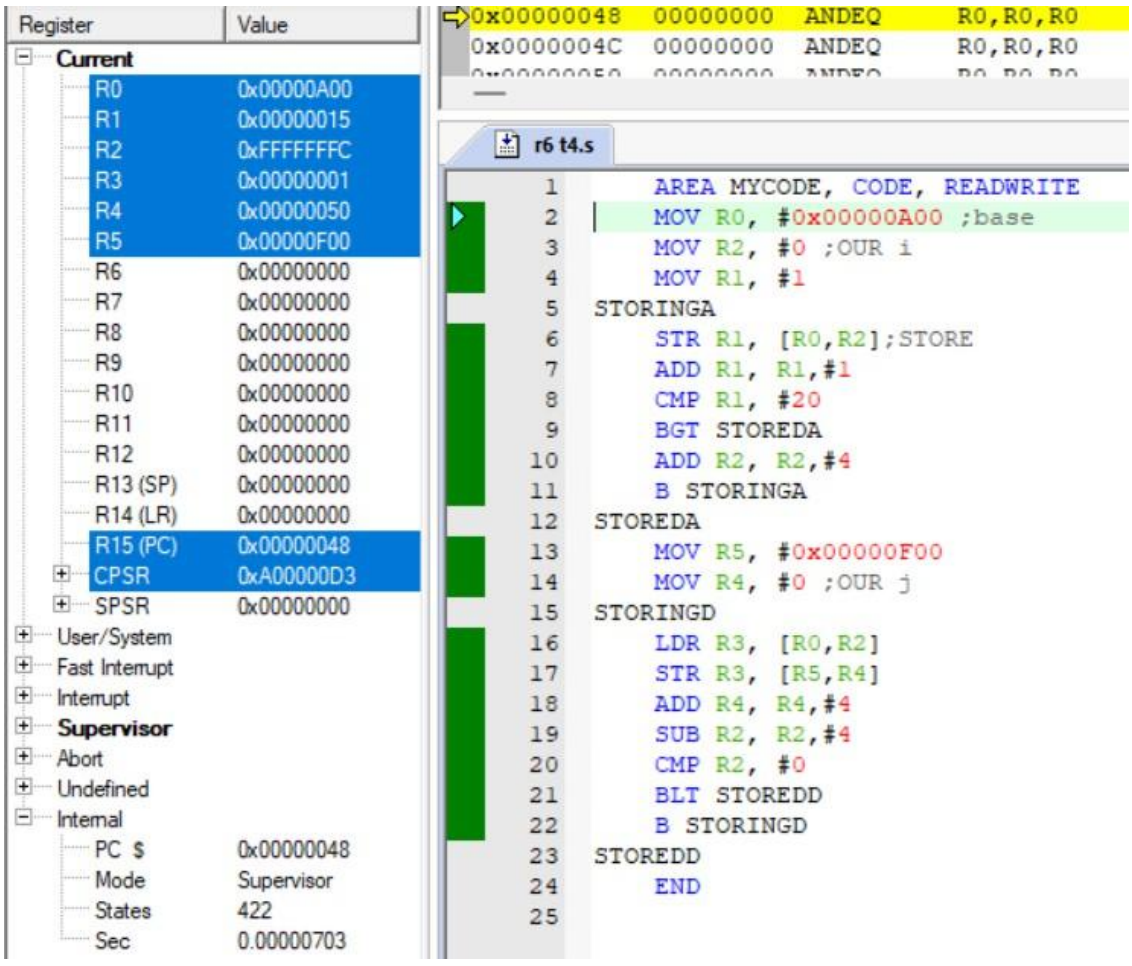
r6 t3.s

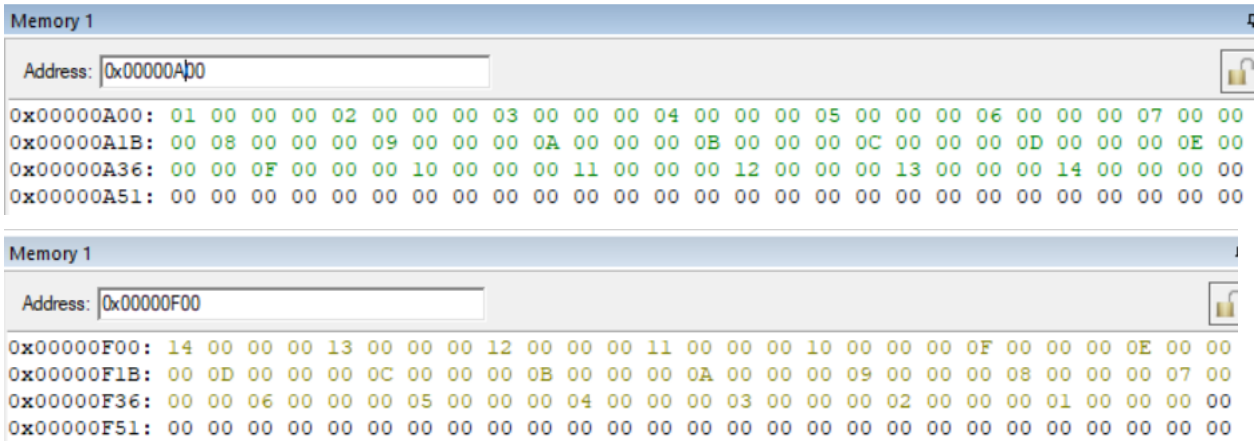
```
1      AREA MYCODE, CODE, READONLY
2      MOV R0,#0 ;R0 IS OUR n
3      MOV R1,#1; R1 IS OUR i
4      MOV R2,#1; R2 IS OUR result
5  FACTORIAL ;R3=R2*R1 AND THEN R1+1
6      MUL R3,R2,R1
7      MOV R2,R3
8      ADD R1,R1,#1
9      CMP R1, R0
10     BLS FACTORIAL
11     B DONEE
12 DONEE
13     END
```

Problem 4

Write an assembly program to store all natural numbers up to 20 in an array in ascending order. Then copy the numbers to another array so the numbers appear in descending order.

Solution

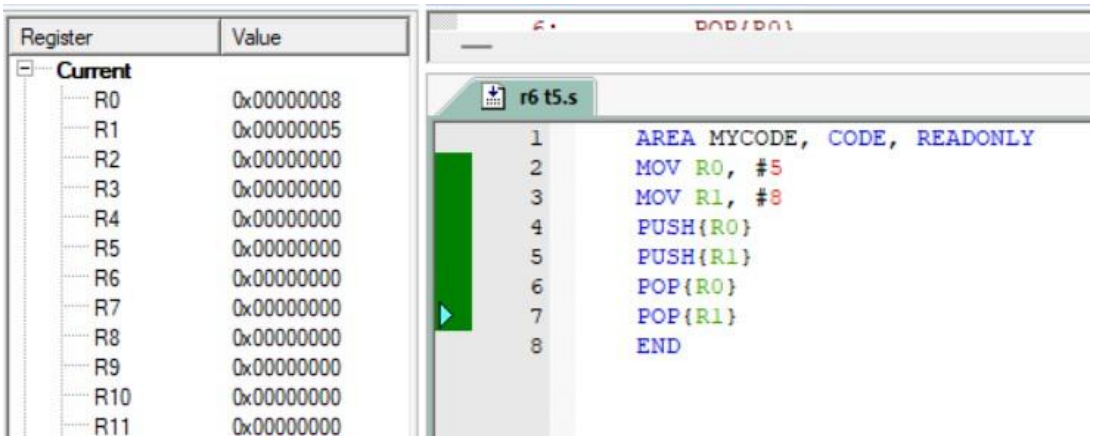




Problem 5

Write an assembly code to swap two register values using stack.

Solution:

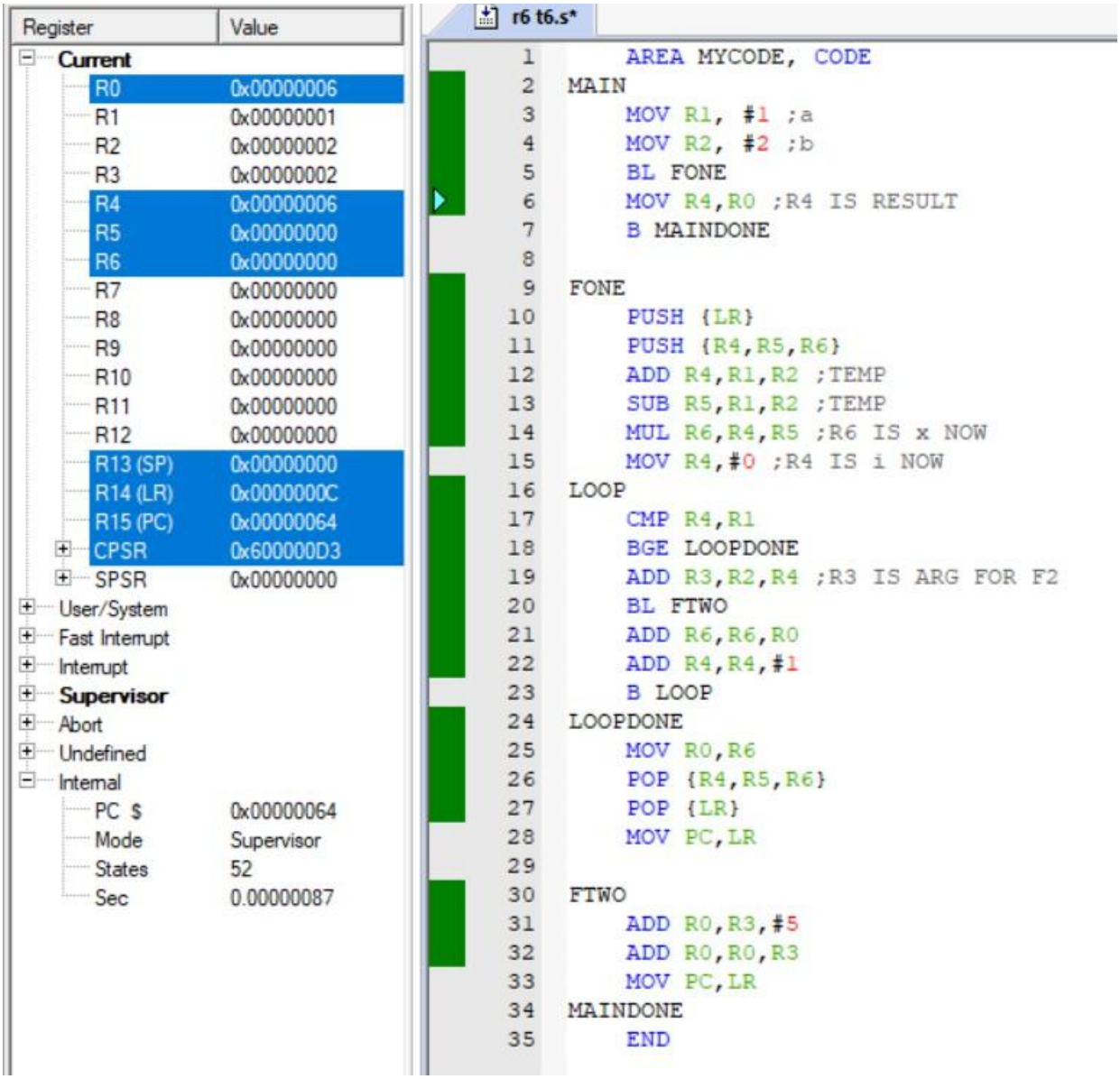


Problem 6

The following snippet shows an example of non-leaf function calling, where function *f1* calls another function *f2*. However, the idea is the similar, we just need to differentiate which is registers are preserved and non-preserved with respect to the caller *f1* and callee *f2*, and save the non-preserved registers on the stack before we call *f2* inside of *f1* and then restore those registers afterward. Implement the following C code snippet in ARM assembly language.

```
int main() {
        int y;
        y = f1(1,2);
}
int f1(int a, int b) {
        int i, x;
        x = (a + b)*(a − b);
        for (i=0; i<a; i++) x = x + f2(b+i);

        return x;
}
int f2(int p) {
        int r;
        r = p + 5;
        return r + p;
}
```

Solution:

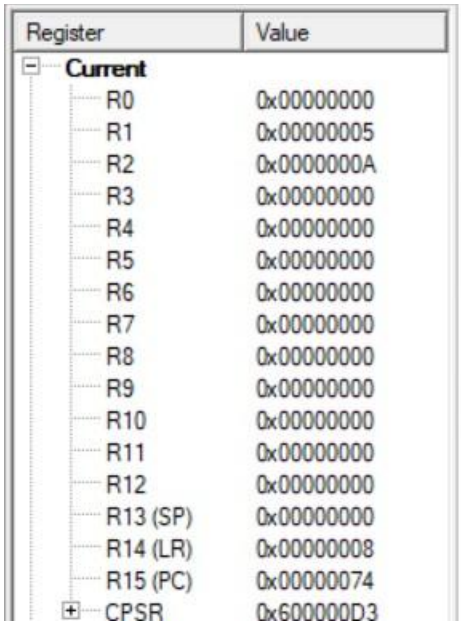

Register
Value
Current
R0 0x00000006
R1 0x00000001
R2 0x00000002
R3 0x00000002
R4 0x00000006
R5 0x00000000
R6 0x00000000
R7 0x00000000
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x00000000
R14 (LR) 0x0000000C
R15 (PC) 0x00000064
CPSR 0x600000D3
SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000064
Mode Supervisor
States 52
Sec 0.00000087
r6 t6.s*
1 AREA MYCODE, CODE
2 MAIN
3 MOV R1, #1 ;a
4 MOV R2, #2 ;b
5 BL FONE
6 MOV R4,R0 ;R4 IS RESULT
7 B MAINDONE
8
9 FONE
10 PUSH {LR}
11 PUSH {R4,R5,R6}
12 ADD R4,R1,R2 ;TEMP
13 SUB R5,R1,R2 ;TEMP
14 MUL R6,R4,R5 ;R6 IS x NOW
15 MOV R4,#0 ;R4 IS i NOW
16 LOOP
17 CMP R4,R1
18 BGE LOOPDONE
19 ADD R3,R2,R4 ;R3 IS ARG FOR F2
20 BL FTWO
21 ADD R6,R6,R0
22 ADD R4,R4,#1
23 B LOOP
24 LOOPDONE
25 MOV R0,R6
26 POP {R4,R5,R6}
27 POP {LR}
28 MOV PC,LR
29
30 FTWO
31 ADD R0,R3,#5
32 ADD R0,R0,R3
33 MOV PC,LR
34 MAINDONE
35 END

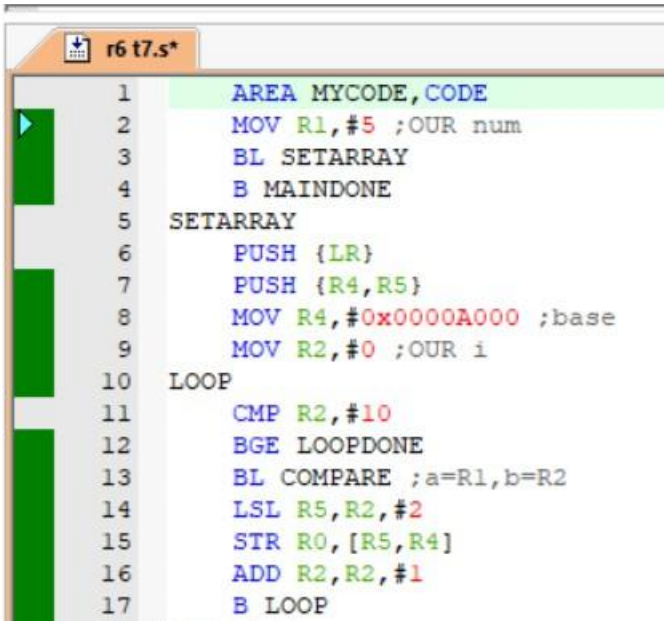
Problem 7

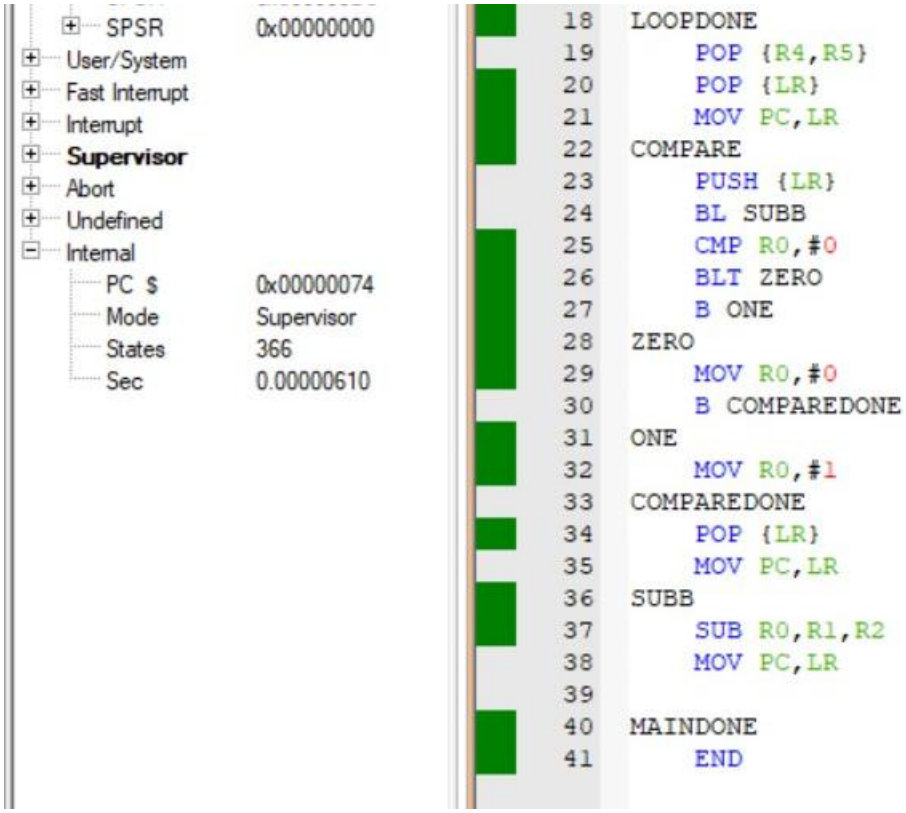
Implement the following C code snippet in ARM assembly language.

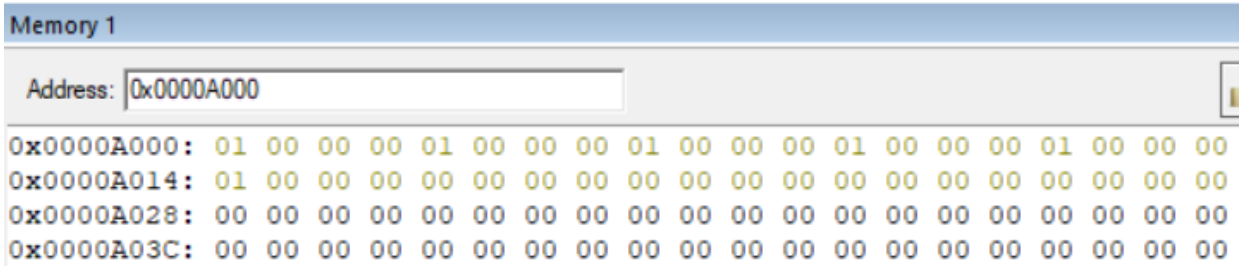
```
// C code
void setArray(int num) {
        int i;
        int array[10];
        for (i = 0; i < 10; i = i + 1)
        array[i] = compare(num, i);
}
int compare(int a, int b) {
        if (sub(a, b) >= 0)
           return 1;
        else
           return 0;
}
int sub(int a, int b) {
        return a − b;
}
```

Solution:






SPSR 0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
PC $ 0x00000074
Mode Supervisor
States 366
Sec 0.00000610
18 LOOPDONE
19 POP {R4,R5}
20 POP {LR}
21 MOV PC,LR
22 COMPARE
23 PUSH {LR}
24 BL SUBB
25 CMP R0,#0
26 BLT ZERO
27 B ONE
28 ZERO
29 MOV R0,#0
30 B COMPAREDONE
31 ONE
32 MOV R0,#1
33 COMPAREDONE
34 POP {LR}
35 MOV PC,LR
36 SUBB
37 SUB R0,R1,R2
38 MOV PC,LR
39
40 MAINDONE
41 END


Memory 1
Address: 0x0000A000
0x0000A000: 01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00 01 00 00 00
0x0000A014: 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000A028: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0000A03C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00