

CODE BLOG

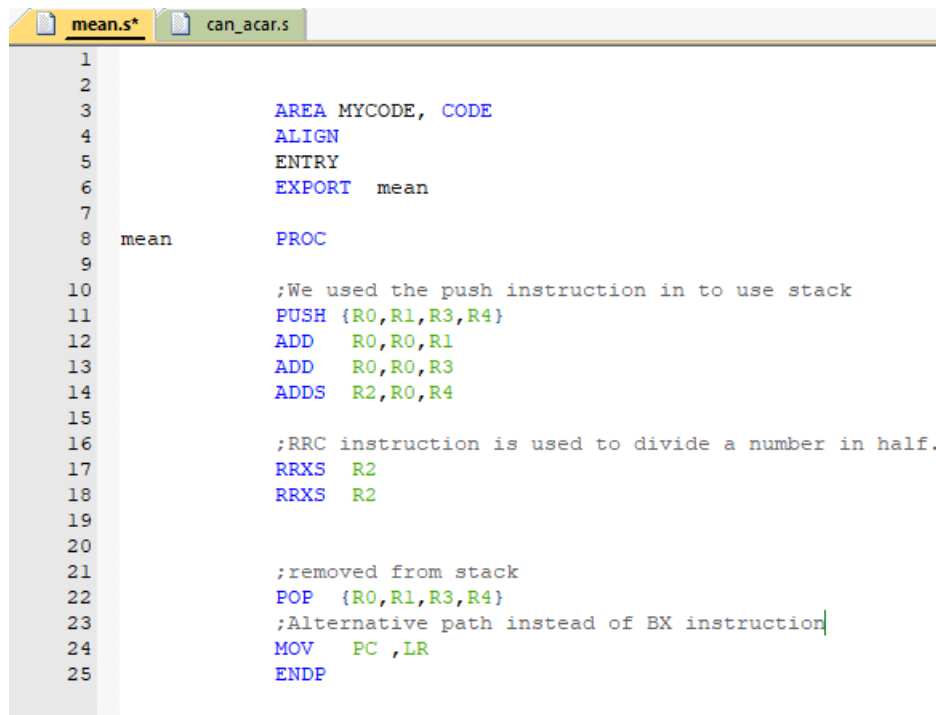
- Main

```
1 ;4 number values given in the question
2 num1 EQU 0x40000004
3 num2 EQU 0x40000002
4 num3 EQU 0x3FFFFFFE
5 num4 EQU 0x3FFFFFFC
6
7 ;address initial values
8 adress1 EQU 0X20000040
9 adress2 EQU 0X20000060
10
11
12 AREA MYCODE, CODE
13 ALIGN
14 ENTRY
15 EXPORT __main
16 EXTERN mean
17 __main
18 ;the given has been saved to the register
19 LDR R0, = num1
20 LDR R1, = num2
21 LDR R3, = num3
22 LDR R4, = num4
23 LDR R5, = adress1
24 LDR R7, = adress2
```

```
25
26 ;r6 is offset value for the save process
27 ;values recorded in registers R0, R2, R3, R4
28 ;numbers saved starting from address 0X20000040
29 MOV R6, #0
30 STR R0, [R5, R6]
31 ADD R6, R6, #4
32 STR R1, [R5, R6]
33 ADD R6, R6, #4
34 STR R3, [R5, R6]
35 ADD R6, R6, #4
36 STR R4, [R5, R6]
37
38 ;subroutine
39 BL mean
40
41 ;We calculated the R2 value in subroutine
42 ;Given the question; I subtract the average value from the given numbers
43 SUB R0, R0, R2
44 SUB R1, R1, R2
45 SUB R3, R3, R2
46 SUB R4, R4, R2
47
```

```
48 ;r6 is offset value for the save process
49 MOV R6, #0
50
51 ;The last step is the value of the subtraction operations, starting from the address 0X20000060.
52 STR R0, [R7, R6]
53 ADD R6, R6, #4
54 STR R1, [R7, R6]
55 ADD R6, R6, #4
56 STR R3, [R7, R6]
57 ADD R6, R6, #4
58 STR R4, [R7, R6]
59
60
61
62 loop B loop
63 END
```

Subroutines

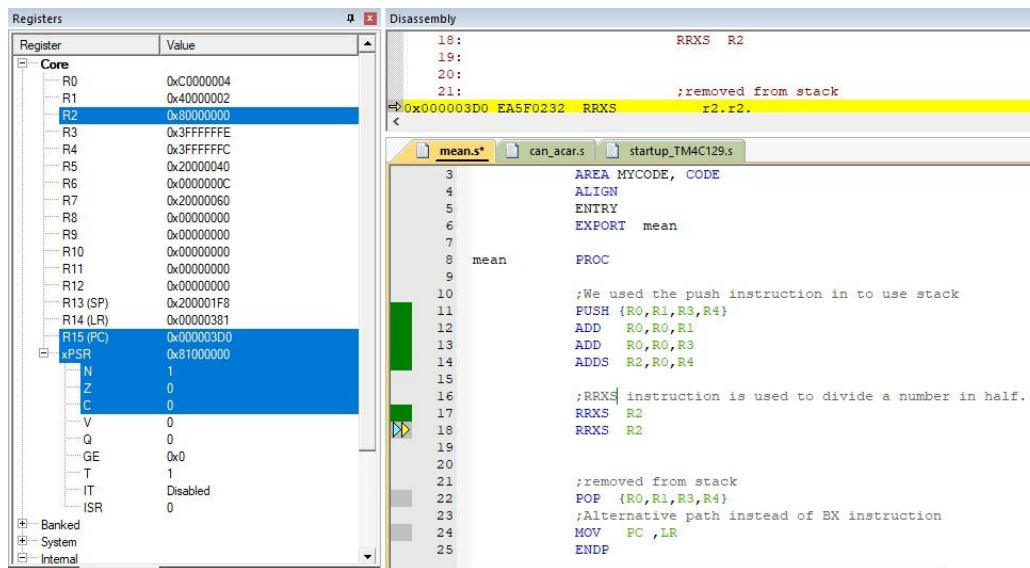


```
1
2
3      AREA MYCODE, CODE
4      ALIGN
5      ENTRY
6      EXPORT mean
7
8 mean  PROC
9
10     ;We used the push instruction in to use stack
11     PUSH {R0,R1,R3,R4}
12     ADD  R0,R0,R1
13     ADD  R0,R0,R3
14     ADDS R2,R0,R4
15
16     ;RRC instruction is used to divide a number in half.
17     RRXS R2
18     RRXS R2
19
20
21     ;removed from stack
22     POP  {R0,R1,R3,R4}
23     ;Alternative path instead of BX instruction
24     MOV  PC,LR
25     ENDP
```

ALGORITHM and CODE

- First, I defined the values given in the question. And then these values are recorded in registers. **EQU** instruction was used for defining operations, and **LDR** instruction was used to register to the Register.
 - The values given afterwards were recorded in the addresses with **STR** instruction. For example, R1 value is written to the address in R5, starting from the R6 offset.
 - The most important part; BL instruction is used when entering subroutine. Here the LR of the next step is kept.
 - On the Mean blog, numbers were collected and averaged. Adding 4 numbers, the number became 33 bits and 1 bit went to hand. Here, when making division, the **RRXS** instruction was used and the number was divided including the hand bit.
 - When exiting subroutine, instead of BX; Actually, when entering the subroutine, we said that the BL instruction holds the LR of the next step. Program Counter ; It is the register that performs operation in programs. If the PC value is LR, the subroutine output is provided.(**MOV PC, LR**)
 - Finally, the average value in the R2 register is subtracted from each element and stored starting at address 0x00000060.
-
- **PUSH** and **POP** commands provide input and output to stacks.
 - **ADDS** instruction also adds by checking the hand bit.

RRXS insturaction



- The visual shows the status of the registers while running the RRXS instruction.

KODLAR

MAIN

;4 number values given in the question

num1	EQU 0x40000004
num2	EQU 0x40000002
num3	EQU 0x3FFFFFFE
num4	EQU 0x3FFFFFFC

```
;address initial values
```

address1	EQU	0X20000040
address2	EQU	0X20000060

```
AREA MYCODE, CODE
ALIGN
ENTRY
EXPORT __main
EXTERN mean
```

__main__

;the given has been saved to the register

```
LDR    R0, = num1
LDR    R1, = num2
LDR    R3, = num3
LDR    R4, = num4
LDR    R5, = adress1
LDR    R7, = adress2
```

;r6 is offset value for the save process

;values recorded in registers R0, R2, R3, R4

```
;numbers saved starting from address 0X20000040
```

```
MOV    R6,#0
STR    R0,[R5,R6]
ADD    R6,R6,#4
STR    R1,[R5,R6]
ADD    R6,R6,#4
STR    R3,[R5,R6]
ADD    R6,R6,#4
STR    R4,[R5,R6]
```

```

;subroutine
BL          mean

;We calculated the R2 value in subroutine
;Given the question; I subtract the average value from the given numbers
SUB         R0,R0,R2
SUB         R1,R1,R2
SUB         R3,R3,R2
SUB         R4,R4,R2

;r6 is offset value for the save process
MOV         R6,#0

;The last step is the value of the subtraction operations, starting from the address 0X20000060.
STR         R0,[R7,R6]
ADD         R6,R6,#4
STR         R1,[R7,R6]
ADD         R6,R6,#4
STR         R3,[R7,R6]
ADD         R6,R6,#4
STR         R4,[R7,R6]

loop        B          loop
            END

```

Subroutines

```

AREA MYCODE, CODE
ALIGN
ENTRY
EXPORT mean

mean        PROC

;We used the push instruction in to use stack
PUSH {R0,R1,R3,R4}
ADD        R0,R0,R1
ADD        R0,R0,R3
ADDS R2,R0,R4

;RRXS instruction is used to divide a number in half.
RRXS R2
RRXS R2

;removed from stack
POP {R0,R1,R3,R4}
;Alternative path instead of BX instruction
MOV PC,LR
ENDP

```