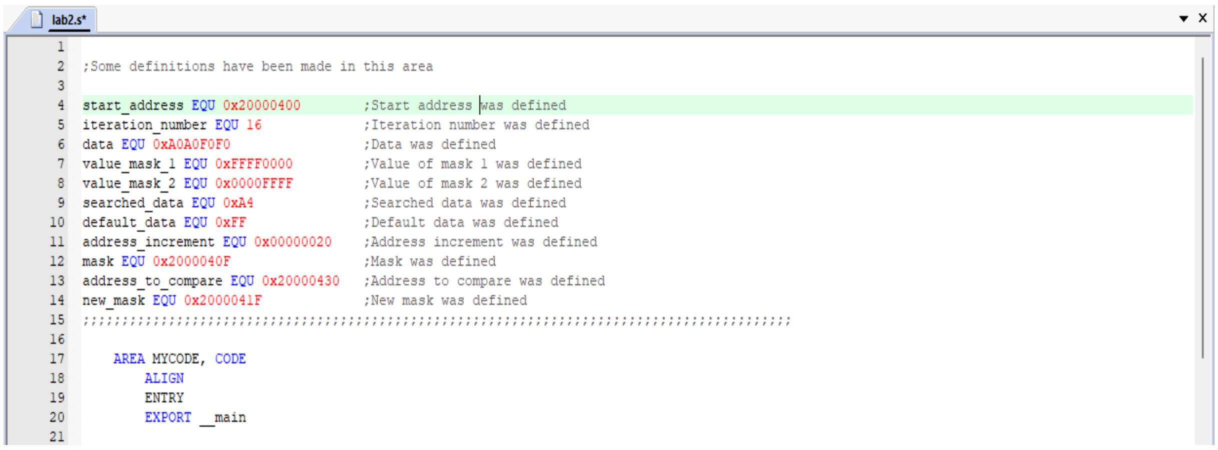
GAZI UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF
ELECTRICAL AND ELECTRONICS ENGINEERING

Experiment #2

Ahmet Emin Karakaya
191110046

First of all, we will understand the code by explaining the screenshots of the code piece by piece.

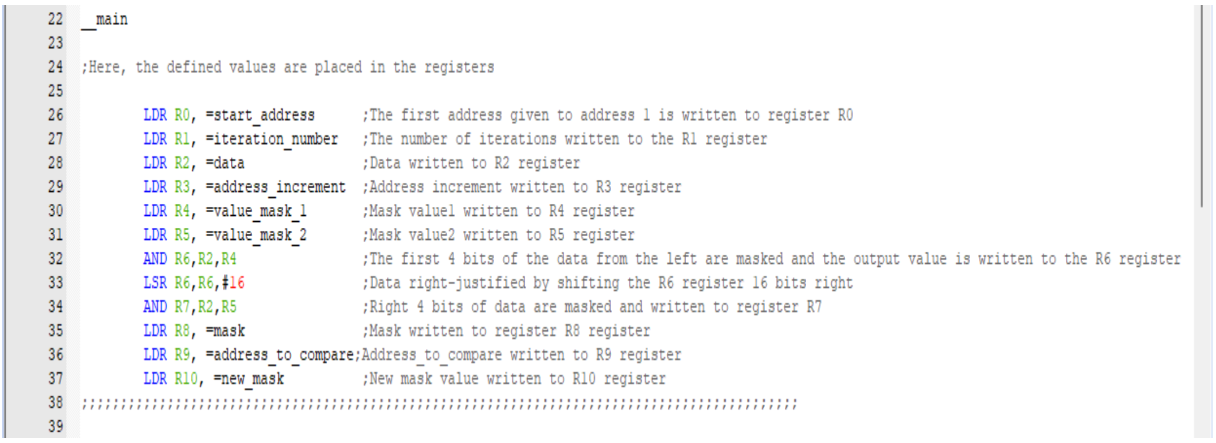
In Figure 1, I made some definitions with the EQU directive. First of all, I defined the address from which address to start writing data. For the loop, I defined the number of iterations for how much it should spin. I have defined the data to be written. I defined 2 masks to divide the data in half. I defined the value to be searched after the data is written to the address. If the searched data is not available at the address, we have defined a default value to indicate that it is not available. I implemented an algorithm to write the values 0xA0Ax and 0xF0Fx in the same loop. I have defined other definitions for this. Their purpose will be explained in the places of use.



```

;Some definitions have been made in this area

start_address EQU 0x20000400          ;Start address was defined
iteration_number EQU 16               ;Iteration number was defined
data EQU 0xA0A0F0F0                   ;Data was defined
value_mask_1 EQU 0xFFFF0000           ;Value of mask 1 was defined
value_mask_2 EQU 0x0000FFFF           ;Value of mask 2 was defined
searched_data EQU 0xA4                ;Searched data was defined
default_data EQU 0xFF                 ;Default data was defined
address_increment EQU 0x00000020      ;Address increment was defined
mask EQU 0x2000040F                   ;Mask was defined
address_to_compare EQU 0x20000430     ;Address to compare was defined
new_mask EQU 0x2000041F               ;New mask was defined
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    AREA MYCODE, CODE
        ALIGN
        ENTRY
        EXPORT __main

```

Figure 1

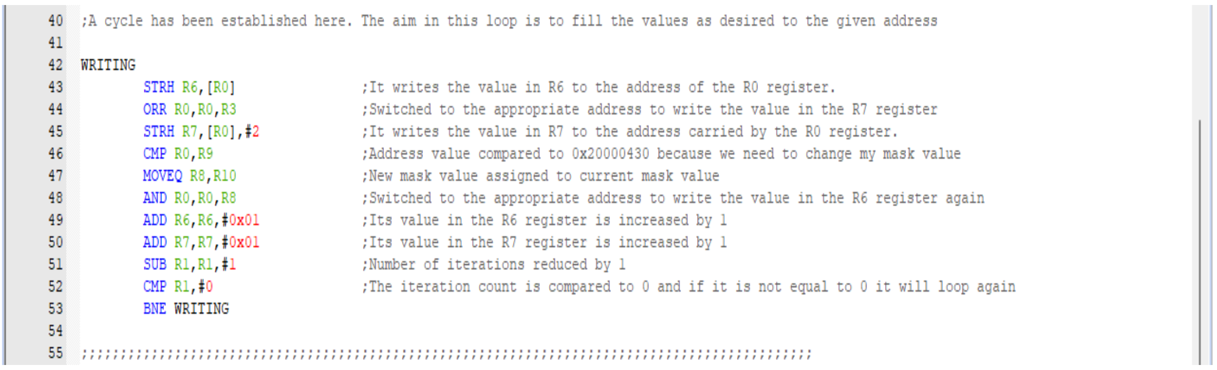
In Figure 2, I placed the things I defined into the relevant registers. There is one thing I should mention here. In the other stage, I need to complete the data in 16 loops. For this I masked the data with value_mask_1(0xFFFF0000) and kept the result in register R6. The result was 0xA0A00000, but I had to right justify meaningful values. For this reason, I wrote the value 0x0000A0A0 to the R6 register by shifting 16 bits. To get the other part of the data, I masked the data with value_mask_2(0x0000FFFF) and wrote the result to register r7.



```
__main

;Here, the defined values are placed in the registers

        LDR R0, =start_address      ;The first address given to address 1 is written to register R0
        LDR R1, =iteration_number   ;The number of iterations written to the R1 register
        LDR R2, =data               ;Data written to R2 register
        LDR R3, =address_increment  ;Address increment written to R3 register
        LDR R4, =value_mask_1       ;Mask value1 written to R4 register
        LDR R5, =value_mask_2       ;Mask value2 written to R5 register
        AND R6,R2,R4                ;The first 4 bits of the data from the left are masked and the output value is written to the R6 register
        LSR R6,R6,#16               ;Data right-justified by shifting the R6 register 16 bits right
        AND R7,R2,R5                ;Right 4 bits of data are masked and written to register R7
        LDR R8, =mask               ;Mask written to register R8 register
        LDR R9, =address_to_compare;Address_to_compare written to R9 register
        LDR R10, =new_mask          ;New mask value written to R10 register
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

```

Figure 2

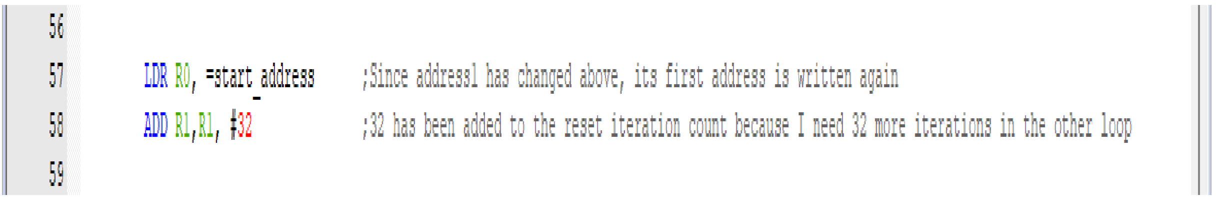
As seen in Figure 3, there is a loop. The purpose of this loop was to write the data as desired to the desired address. This has been done here as well. First of all, the value in the R6 register is written to start_address(0x20000400). Then the ORR operation was performed with the address R3(0x00000020). Thanks to this operation, the address became 0x20000420 and the value in the R7 register was written to that address and the address was increased by 2. The new value of the address is 0x20000422. After this process was completed, a comparison process was performed. The current address is compared with the register R9(0x20000430). The purpose of this was done because it was needed later. Because we need another mask when the address is 0x2000041x or 0x2000043x in the future. Let's forget about this process for now. The code will come to the AND directive because the CMP result is wrong. Here the last value of the address (0x20000422) is masked with R8(0x2000040F). After masking the address will be 0x20000402. After this process is finished, the value of R6 and R7 will increase by 1. The number of iterations will also be reduced by 1. If the iteration count is equal to 0, it will exit the loop. It will return inside the loop unless they are equal. This time the new R6 (0xA0A1) value will be written to our last address (0x20000402). Then with ORR the address will be 0x20000422 and the new R7(0xF0F1) will be written and increment the address by 2. This will continue until a value is written to address 0x20000428. However, after data is written to address 0x20000428, it will increase by 2 and become 0x20000430. After arriving at this address, another masking process has to be done. Because let's pretend we didn't change it. If we don't change it, we'll have ANDed 0x20000430 with R8(0x2000040F) and the new address becomes 0x20000400 and we start writing to the starting address. To avoid this, we compared the address value with 0x20000430 with CMP. If it is the same, now our new mask value will be 0x2000041F. In this way, if we mask with 0x20000430 and 0x2000041F, the result will be 0x20000410 and we will continue where we left off. In this way, I filled in the rest of the data.



```
;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]            ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3            ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2         ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9               ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10            ;New mask value assigned to current mask value
        AND R0,R0,R8            ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01         ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01         ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1            ;Number of iterations reduced by 1
        CMP R1,#0               ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Figure 3

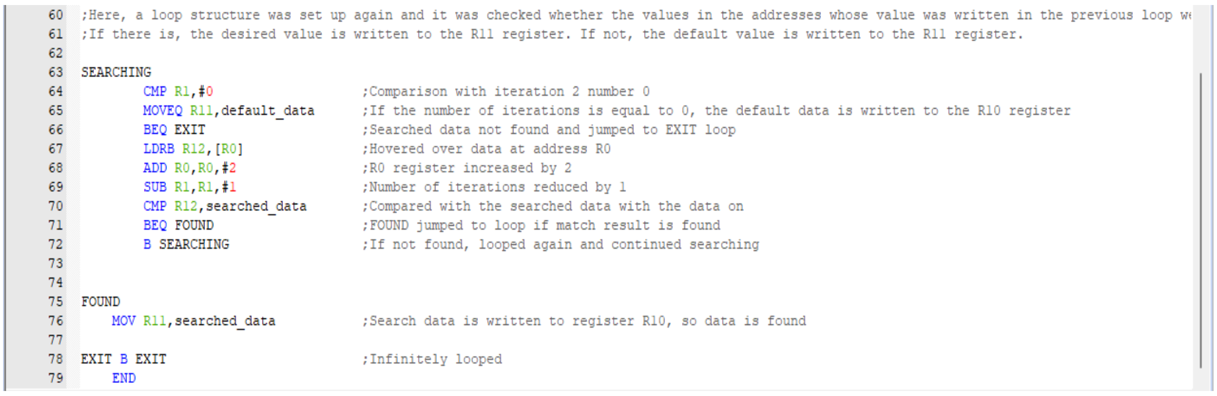
In Figure 4, we need to make 32 iterations of the address to the starting address for the other search loop. 32 iterations may be too much, but if the searched data is at the end, then 32 will only be enough. We've done that here.



```

        LDR R0, =start_address  ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32          ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

```

Figure 4

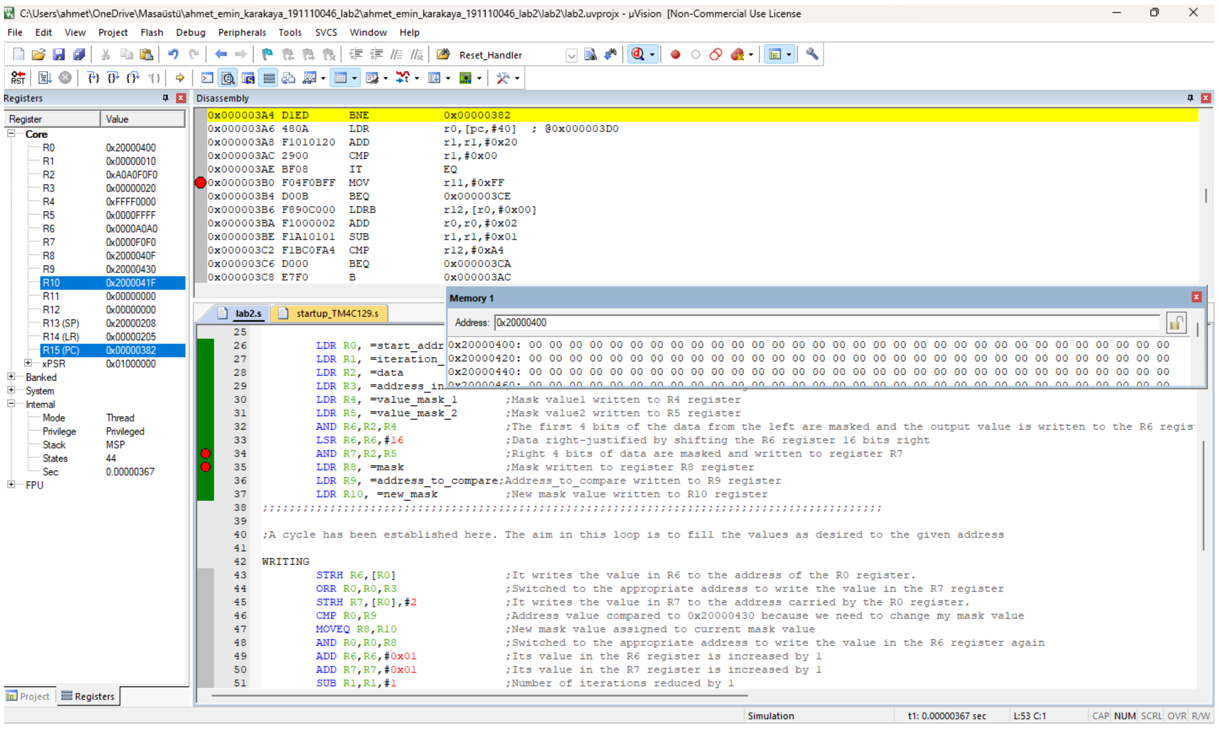
Figure 5 contains the rest of the code. Here I made a loop to find the sought value. In the SEARCHING loop, I checked whether we have the searched value in our addresses. First of all, I checked whether the number of iterations is 0 with the CMP command. The purpose of this is if after looking at all the addresses I have written, if the number sought is still not found, I set the default value to the R11 register and then told it to go to the EXIT command and put it into infinite balance there. Other than that, if my iteration count is not 0, I did the following. I pulled data from R0 to R12 and increased R0 by 2 because I did it to not control the A0 values in between. I reduced the number of iterations by 1. After that operation was done, I compared the searched value with the data I took from R12, and if they are the same, it will go to the FOUND command and write the sought value to the R11 register. In general, the working logic of the code was like this.



```
;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous loop w
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                   ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data      ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                    ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]               ;Hovered over data at address R0
        ADD R0,R0,#2                ;R0 register increased by 2
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R12,searched_data       ;Compared with the searched data with the data on
        BEQ FOUND                   ;FOUND jumped to loop if match result is found
        B SEARCHING                 ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data           ;Search data is written to register R10, so data is found

EXIT B EXIT                         ;Infinitely looped
    END
```

Figure 5

After that, I put screenshots showing the output of the code I described. In Figure6, it is observed that the relevant assignments to the registers are made.



Register	Value
Core	
R0	0x20000400
R1	0x00000010
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A0
R7	0x0000F0F0
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	44
Sec	0.00000367
FPU	

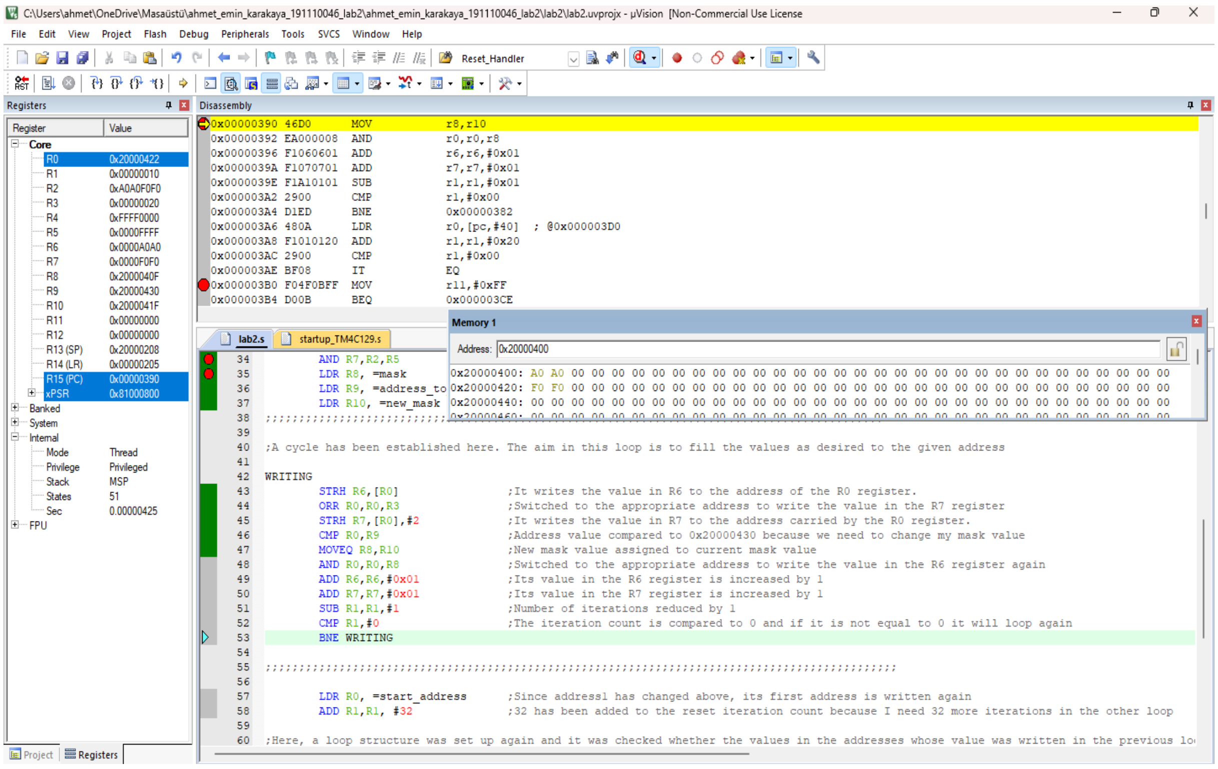
```
0x000003A4 D1ED      BNE          0x00000382
0x000003A6 480A      LDR          r0,[pc,#40]  ; @0x000003D0
0x000003A8 F1010120  ADD          r1,r1,#0x20
0x000003AC 2900      CMP          r1,#0x00
0x000003AE BF08      IT           EQ
0x000003B0 F04F0BFF  MOV          r11,#0xFF
0x000003B4 D00B      BEQ          0x000003CE
0x000003B6 F890C000  LDRB         r12,[r0,#0x00]
0x000003BA F1000002  ADD          r0,r0,#0x02
0x000003BE F1A10101  SUB          r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP          r12,#0xA4
0x000003C6 D000      BEQ          0x000003CA
0x000003C8 E7F0      B            0x000003AC
```

```
Address: 0x20000400
0x20000400: 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
0x20000420: 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
0x20000440: 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
```

```

        LDR R0, =start_addr
        LDR R1, =iteration_
        LDR R2, =data
        LDR R3, =address_in
        LDR R4, =value_mask_1       ;Mask value1 written to R4 register
        LDR R5, =value_mask_2       ;Mask value2 written to R5 register
        AND R6,R2,R4                ;The first 4 bits of the data from the left are masked and the output value is written to the R6 regis
        LSR R6,R6,#16               ;Data right-justified by shifting the R6 register 16 bits right
        AND R7,R2,R5                ;Right 4 bits of data are masked and written to register R7
        LDR R8, =mask               ;Mask written to register R8 register
        LDR R9, =address_to_compare;Address_to_compare written to R9 register
        LDR R10, =new_mask          ;New mask value written to R10 register
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
```

Figure 6

Data has been written to the address. It was shaped as described above in the address.


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x20000422
R1	0x00000010
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A0
R7	0x0000F0F0
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000390
xPSR	0x81000800
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	51
Sec	0.00000425
FPU	

```
0x00000390 46D0      MOV      r8,r10
0x00000392 EA000008  AND      r0,r0,r8
0x00000396 F1060601  ADD      r6,r6,#0x01
0x0000039A F1070701  ADD      r7,r7,#0x01
0x0000039E F1A10101  SUB      r1,r1,#0x01
0x000003A2 2900      CMP      r1,#0x00
0x000003A4 D1ED      BNE      0x00000382
0x000003A6 480A      LDR      r0,[pc,#40]  ; @0x000003D0
0x000003A8 F1010120  ADD      r1,r1,#0x20
0x000003AC 2900      CMP      r1,#0x00
0x000003AE BF08      IT       EQ
0x000003B0 F04F0BFF  MOV      r11,#0xFF
0x000003B4 D00B      BEQ      0x000003CE
```

Memory 1

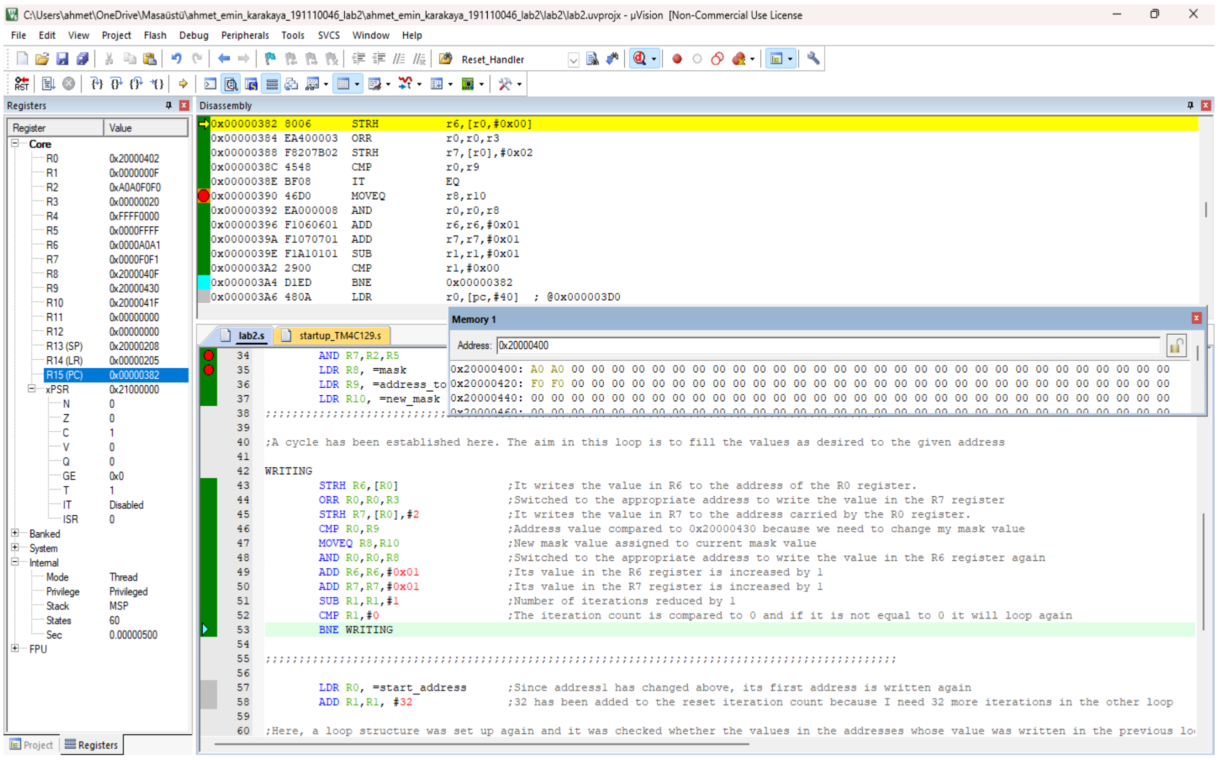
Address: 0x20000400

```
0x20000400: A0 A0 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
0x20000420: F0 F0 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
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]            ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3            ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2         ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9               ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10            ;New mask value assigned to current mask value
        AND R0,R0,R8            ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01         ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01         ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1            ;Number of iterations reduced by 1
        CMP R1,#0               ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address  ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32          ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 7

It's back in the loop. R6 and R7 increased by 1. Number of iterations(R1) decreased by 1. It came to the desired value at the address.(Figure 8)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x20000402
R1	0x0000000F
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A1
R7	0x0000F0F1
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x21000000
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	60
Sec	0.00000500
FPU	

```
0x00000382 8006      STRH     r6,[r0,#0x00]
0x00000384 EA400003  ORR      r0,r0,r3
0x00000388 F8207B02  STRH     r7,[r0],#0x02
0x0000038C 4548      CMP      r0,r9
0x0000038E BF08      IT       EQ
0x00000390 46D0      MOVEQ    r8,r10
0x00000392 EA000008  AND      r0,r0,r8
0x00000396 F1060601  ADD      r6,r6,#0x01
0x0000039A F1070701  ADD      r7,r7,#0x01
0x0000039E F1A10101  SUB      r1,r1,#0x01
0x000003A2 2900      CMP      r1,#0x00
0x000003A4 D1ED      BNE      0x00000382
0x000003A6 480A      LDR      r0,[pc,#40]  ; @0x000003D0
```

Memory 1

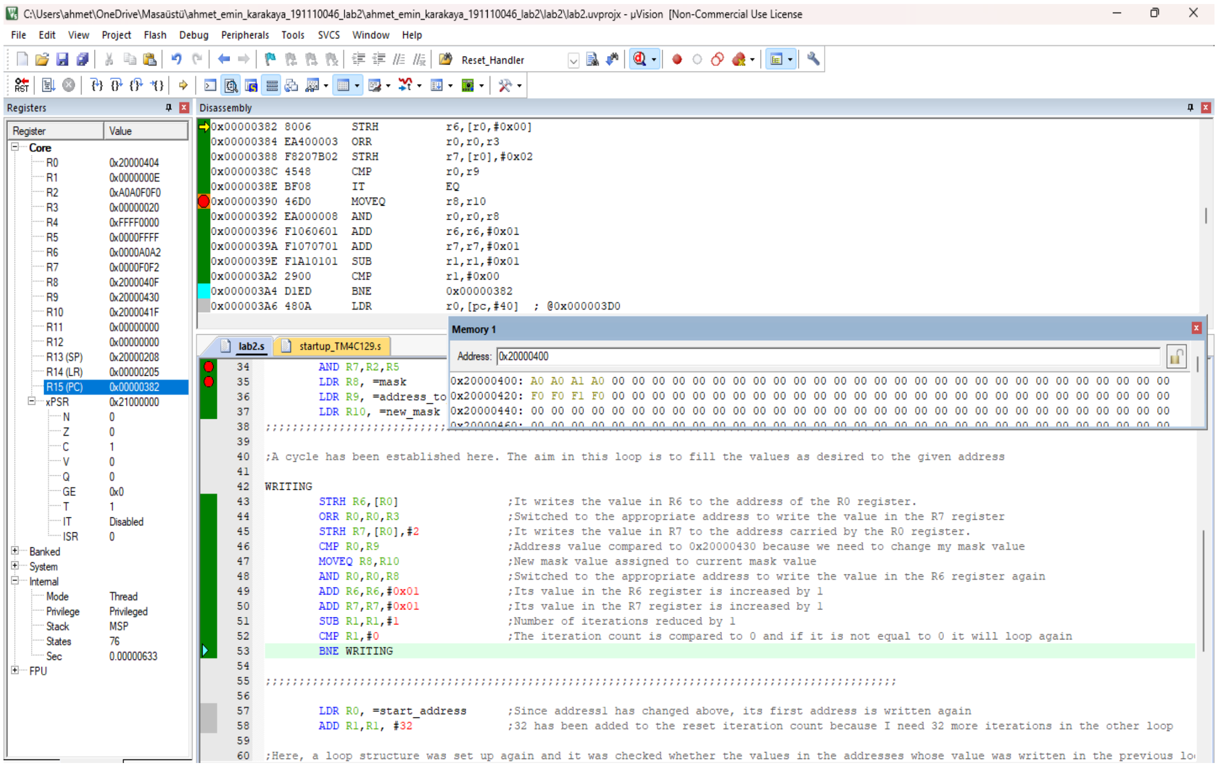
Address: 0x20000400

```
0x20000400: A0 A0 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
0x20000420: F0 F0 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
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]            ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3            ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2         ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9               ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10            ;New mask value assigned to current mask value
        AND R0,R0,R8            ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01         ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01         ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1            ;Number of iterations reduced by 1
        CMP R1,#0               ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address  ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32          ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 8

Next repeat (Figure 9)


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s

C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

```
Register      Value
Core
  R0          0x20000404
  R1          0x0000000E
  R2          0xA0A0F0F0
  R3          0x00000020
  R4          0xFFFF0000
  R5          0x0000FFFF
  R6          0x0000A0A2
  R7          0x0000F0F2
  R8          0x2000040F
  R9          0x20000430
  R10         0x2000041F
  R11         0x00000000
  R12         0x00000000
  R13 (SP)    0x20000208
  R14 (LR)    0x00000205
  R15 (PC)    0x00000382
  xPSR        0x21000000
    N         0
    Z         0
    C         1
    V         0
    Q         0
    GE        0x0
    T         1
    IT        Disabled
    ISR       0
Banked
System
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      76
  Sec         0.00000633
FPU
```

```
0x00000382 8006      STRH     r6,[r0,#0x00]
0x00000384 EA400003  ORR      r0,r0,r3
0x00000388 F8207B02  STRH     r7,[r0],#0x02
0x0000038C 4548      CMP      r0,r9
0x0000038E BF08      IT       EQ
0x00000390 46D0      MOVEQ    r8,r10
0x00000392 EA000008  AND      r0,r0,r8
0x00000396 F1060601  ADD      r6,r6,#0x01
0x0000039A F1070701  ADD      r7,r7,#0x01
0x0000039E F1A10101  SUB      r1,r1,#0x01
0x000003A2 2900      CMP      r1,#0x00
0x000003A4 D1ED      BNE      0x00000382
0x000003A6 480A      LDR      r0,[pc,#40]  ; @0x000003D0
```

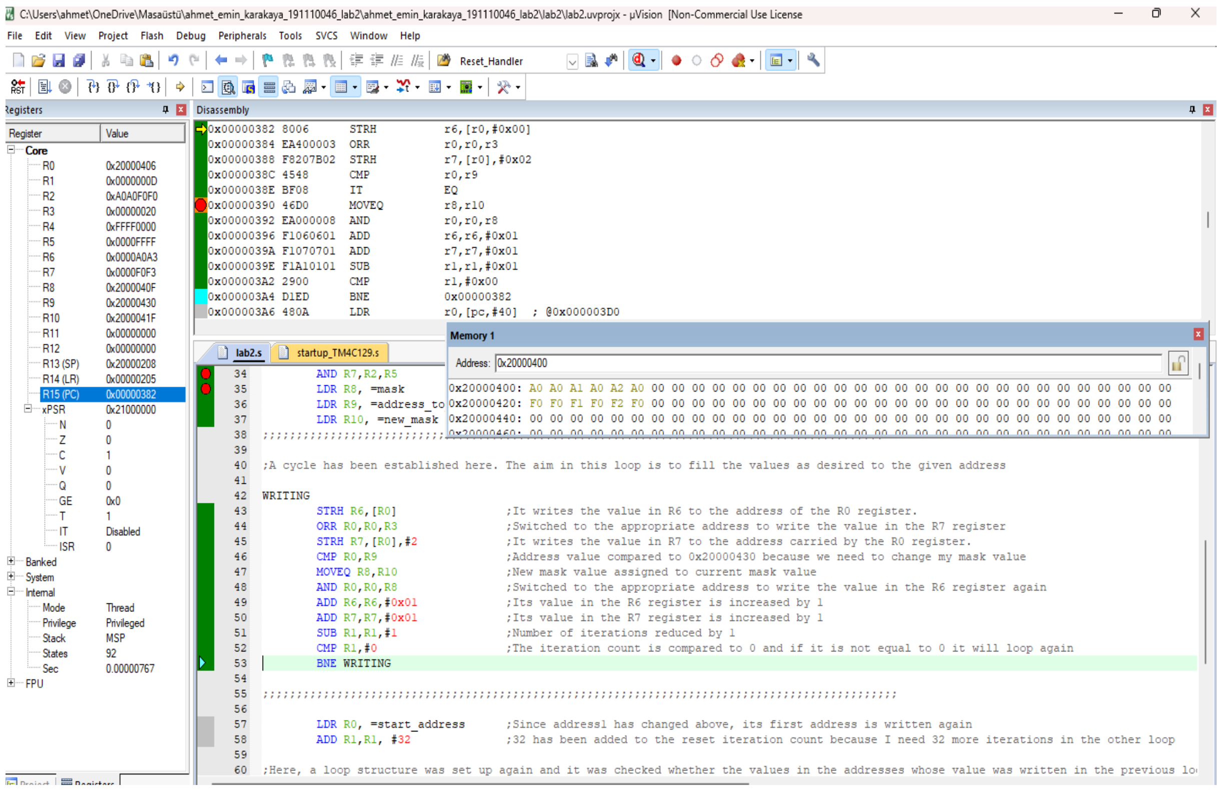
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 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
0x20000420: F0 F0 F1 F0 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
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]              ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3              ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2           ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                 ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10              ;New mask value assigned to current mask value
        AND R0,R0,R8              ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01           ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01           ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1              ;Number of iterations reduced by 1
        CMP R1,#0                 ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address    ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32            ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 9

Next repeat (Figure 10)


File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s

C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

```
Register      Value
Core
  R0          0x20000406
  R1          0x0000000D
  R2          0xA0A0F0F0
  R3          0x00000020
  R4          0xFFFF0000
  R5          0x0000FFFF
  R6          0x0000A0A3
  R7          0x0000F0F3
  R8          0x2000040F
  R9          0x20000430
  R10         0x2000041F
  R11         0x00000000
  R12         0x00000000
  R13 (SP)    0x20000208
  R14 (LR)    0x00000205
  R15 (PC)    0x00000382
  xPSR        0x21000000
    N         0
    Z         0
    C         1
    V         0
    Q         0
    GE        0x0
    T         1
    IT        Disabled
    ISR       0
Banked
System
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      92
  Sec         0.00000767
FPU
```

```
0x00000382 8006      STRH     r6,[r0,#0x00]
0x00000384 EA400003  ORR      r0,r0,r3
0x00000388 F8207B02  STRH     r7,[r0],#0x02
0x0000038C 4548      CMP      r0,r9
0x0000038E BF08      IT       EQ
0x00000390 46D0      MOVEQ    r8,r10
0x00000392 EA000008  AND      r0,r0,r8
0x00000396 F1060601  ADD      r6,r6,#0x01
0x0000039A F1070701  ADD      r7,r7,#0x01
0x0000039E F1A10101  SUB      r1,r1,#0x01
0x000003A2 2900      CMP      r1,#0x00
0x000003A4 D1ED      BNE      0x00000382
0x000003A6 480A      LDR      r0,[pc,#40]  ; @0x000003D0
```

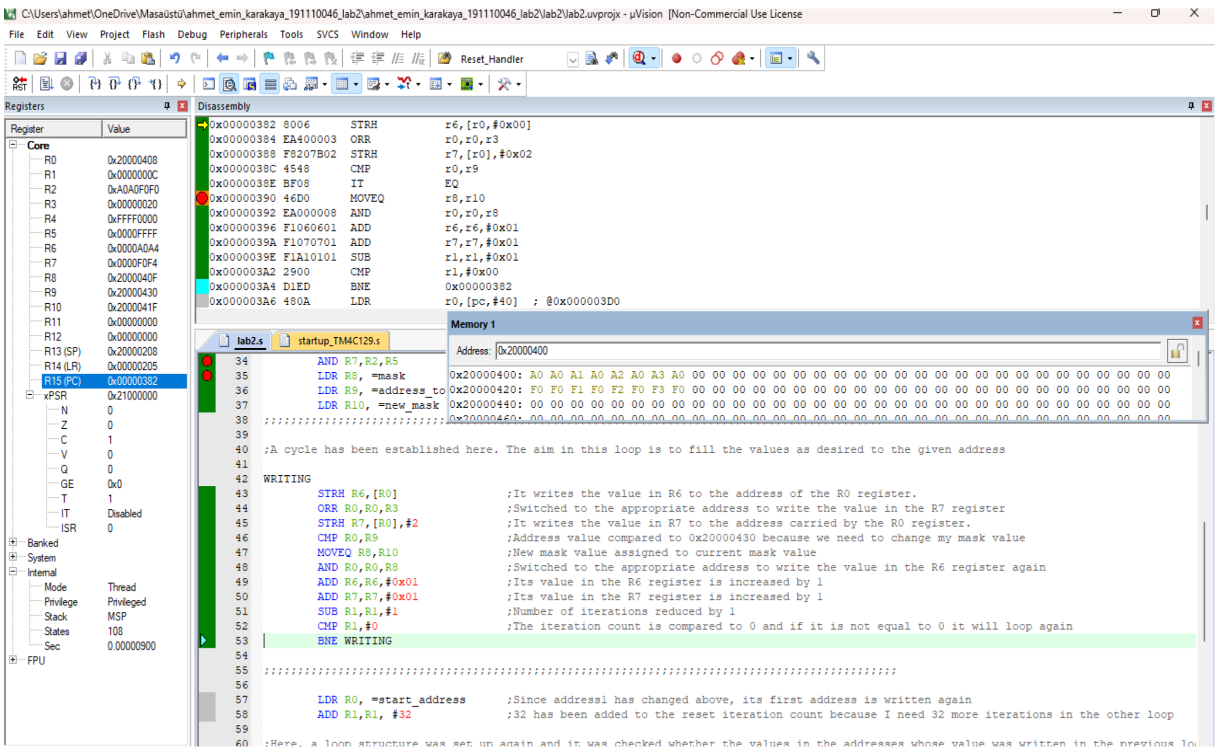
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 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
0x20000420: F0 F0 F1 F0 F2 F0 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
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]              ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3              ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2           ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                 ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10              ;New mask value assigned to current mask value
        AND R0,R0,R8              ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01           ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01           ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1              ;Number of iterations reduced by 1
        CMP R1,#0                 ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address    ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32            ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 10

Next repeat (Figure 11)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s

Register	Value
Core	
R0	0x20000408
R1	0x0000000C
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A4
R7	0x0000F0F4
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x21000000
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	108
Sec	0.00000900
FPU	

```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

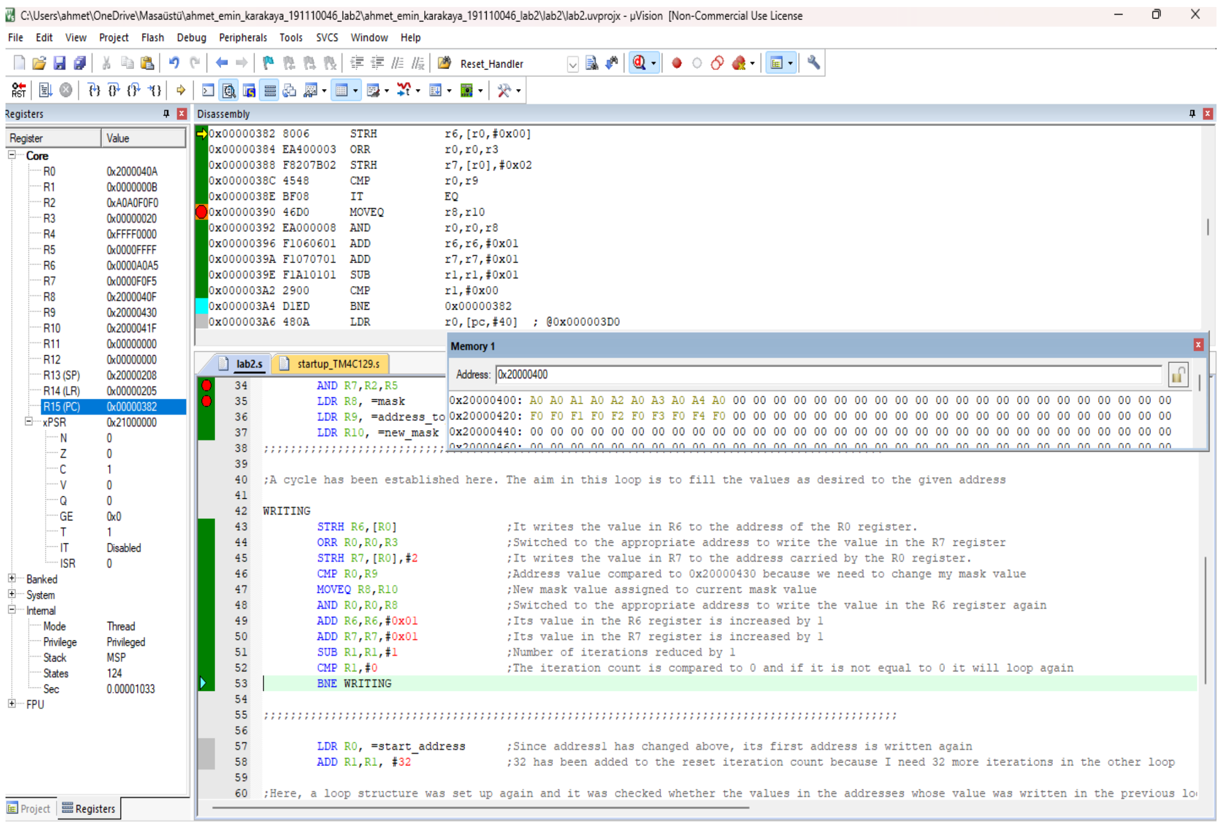
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R1,#0                   ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address      ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32              ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 11

Next repeat (Figure 12)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x2000040A
R1	0x0000000B
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A5
R7	0x0000F0F5
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x21000000
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	124
Sec	0.00001033
FPU	

```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

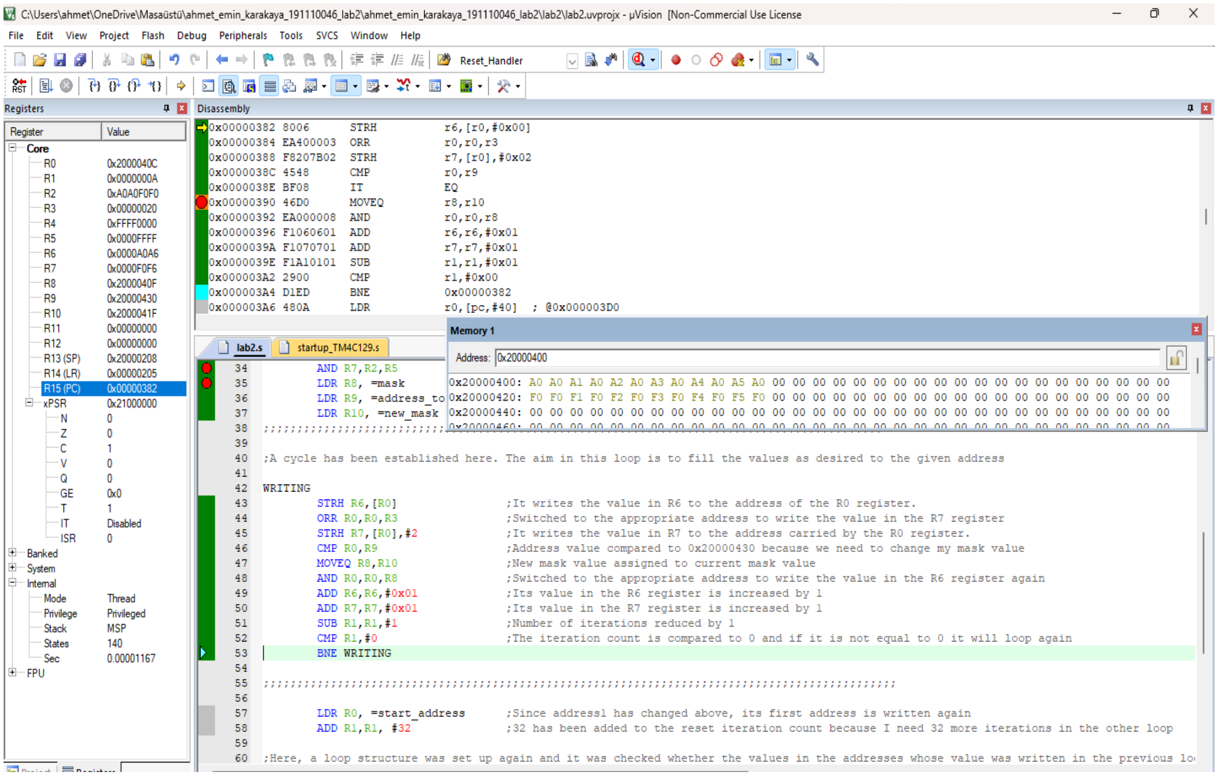
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R1,#0                   ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address      ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32              ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 12

Next repeat (Figure 13)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Register	Value
Core	
R0	0x2000040C
R1	0x0000000A
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A6
R7	0x0000F0F6
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x21000000
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	140
Sec	0.00001167
FPU	

```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

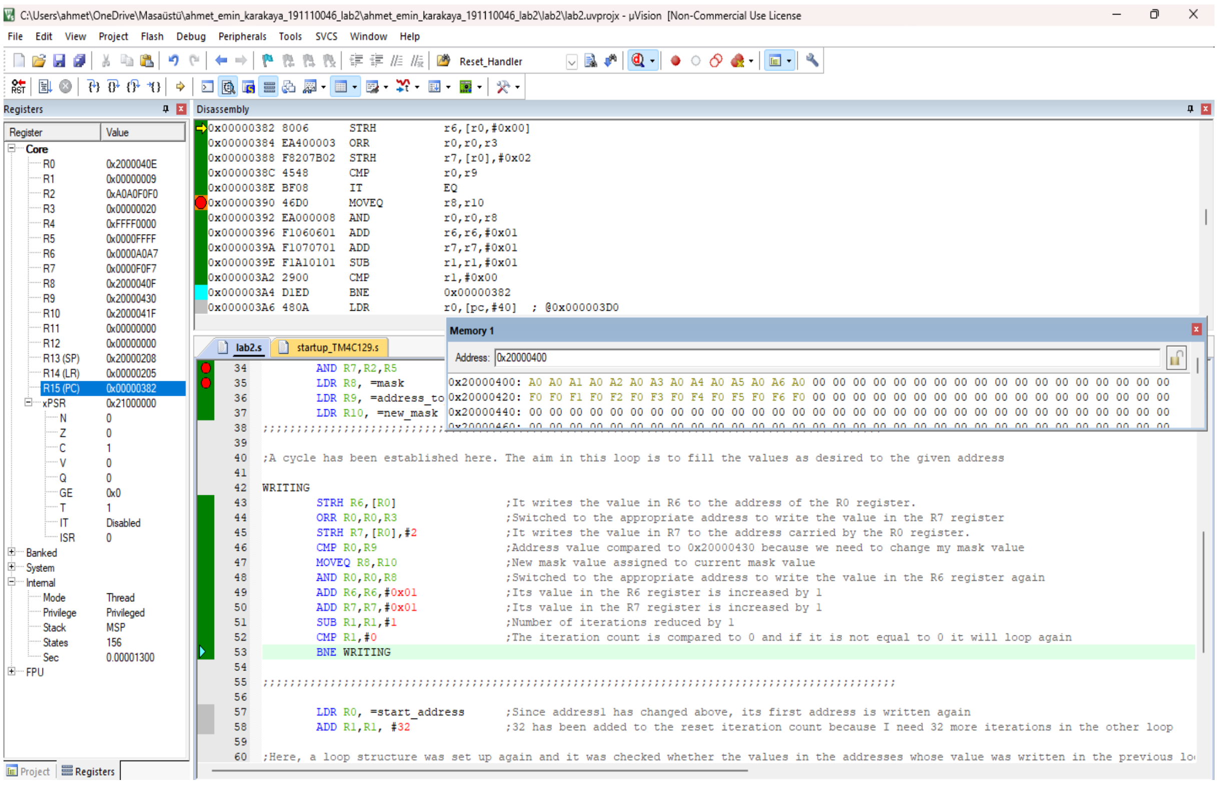
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R1,#0                   ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address      ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32              ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 13

Next repeat (Figure 14)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Register	Value
Core	
R0	0x2000040E
R1	0x00000009
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0A7
R7	0x0000F0F7
R8	0x2000040F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x00000382
xPSR	0x21000000
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	156
Sec	0.00001300
FPU	

```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

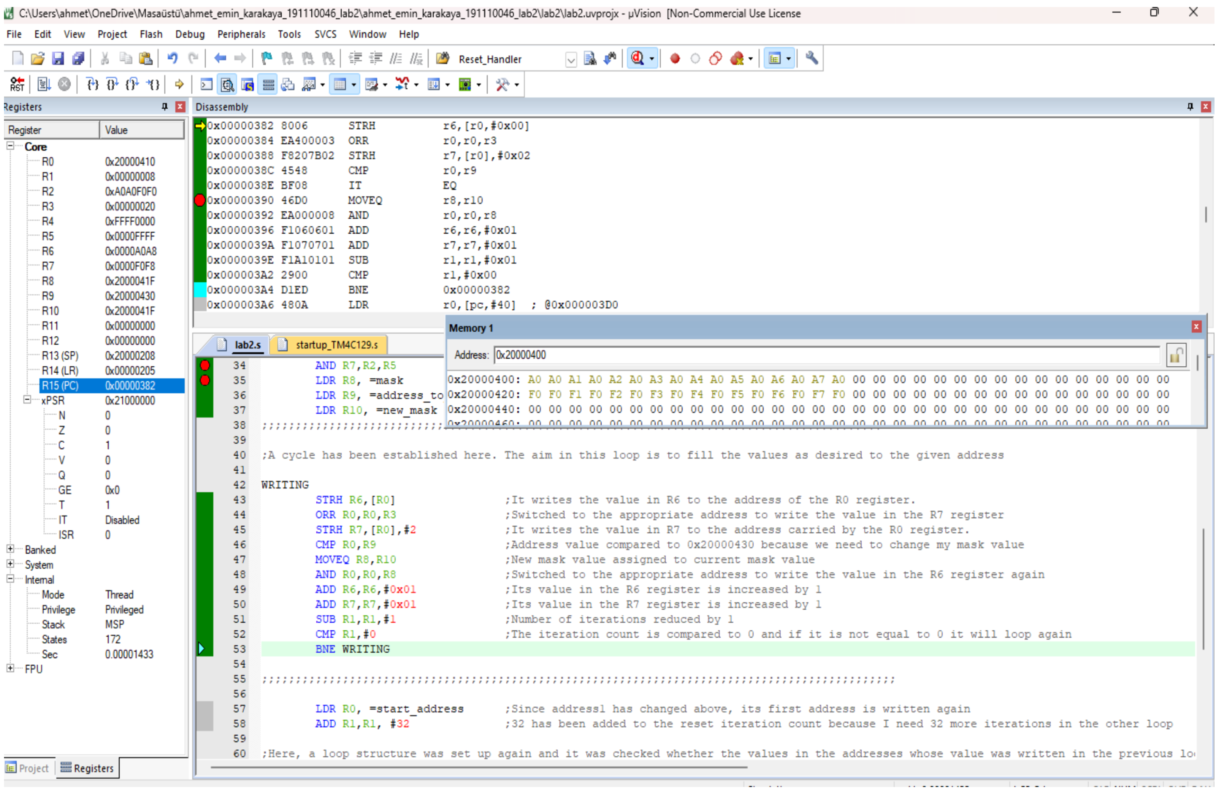
Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R1,#0                   ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address      ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32              ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 14

Here, the CMP event that I mentioned in the explanation took place. And the mask value is updated.(Figure 15)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s
Project
Registers

```
Register      Value
Core
  R0          0x20000410
  R1          0x00000008
  R2          0xA0A0F0F0
  R3          0x00000020
  R4          0xFFFF0000
  R5          0x0000FFFF
  R6          0x0000A0A8
  R7          0x0000F0F8
  R8          0x2000041F
  R9          0x20000430
  R10         0x2000041F
  R11         0x00000000
  R12         0x00000000
  R13 (SP)    0x20000208
  R14 (LR)    0x00000205
  R15 (PC)    0x00000382
  xPSR        0x21000000
    N         0
    Z         0
    C         1
    V         0
    Q         0
    GE        0x0
    T         1
    IT        Disabled
    ISR       0
Banked
System
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      172
  Sec         0.00001433
FPU
```

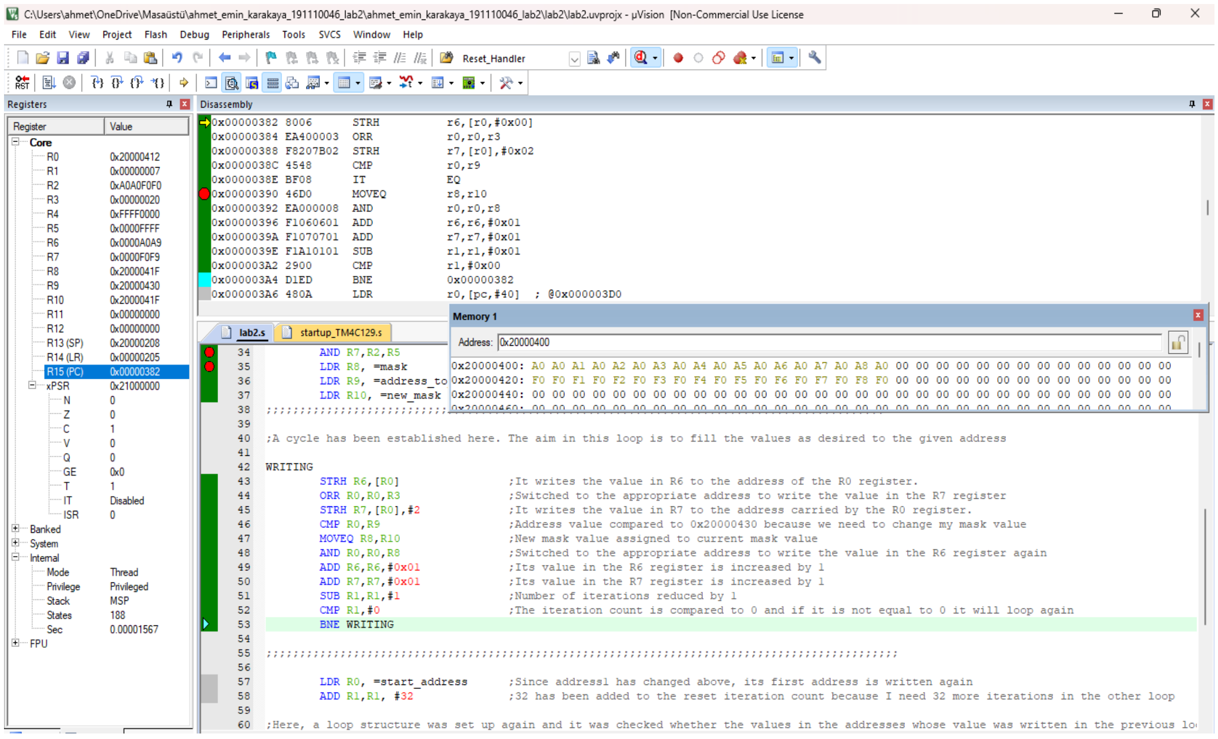
```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

```
Address: 0x20000400
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]               ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3               ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2            ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                  ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10               ;New mask value assigned to current mask value
        AND R0,R0,R8               ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01            ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01            ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1               ;Number of iterations reduced by 1
        CMP R1,#0                  ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address     ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32             ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 15

Next repeat (Figure 16)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s

```
Register      Value
Core
  R0          0x20000412
  R1          0x00000007
  R2          0xA0A0F0F0
  R3          0x00000020
  R4          0xFFFF0000
  R5          0x0000FFFF
  R6          0x0000A0A9
  R7          0x0000F0F9
  R8          0x2000041F
  R9          0x20000430
  R10         0x2000041F
  R11         0x00000000
  R12         0x00000000
  R13 (SP)    0x20000208
  R14 (LR)    0x00000205
  R15 (PC)    0x00000382
  xPSR        0x21000000
    N         0
    Z         0
    C         1
    V         0
    Q         0
    GE        0x0
    T         1
    IT        Disabled
    ISR       0
Banked
System
Internal
  Mode        Thread
  Privilege   Privileged
  Stack       MSP
  States      188
  Sec         0.00001567
FPU
```

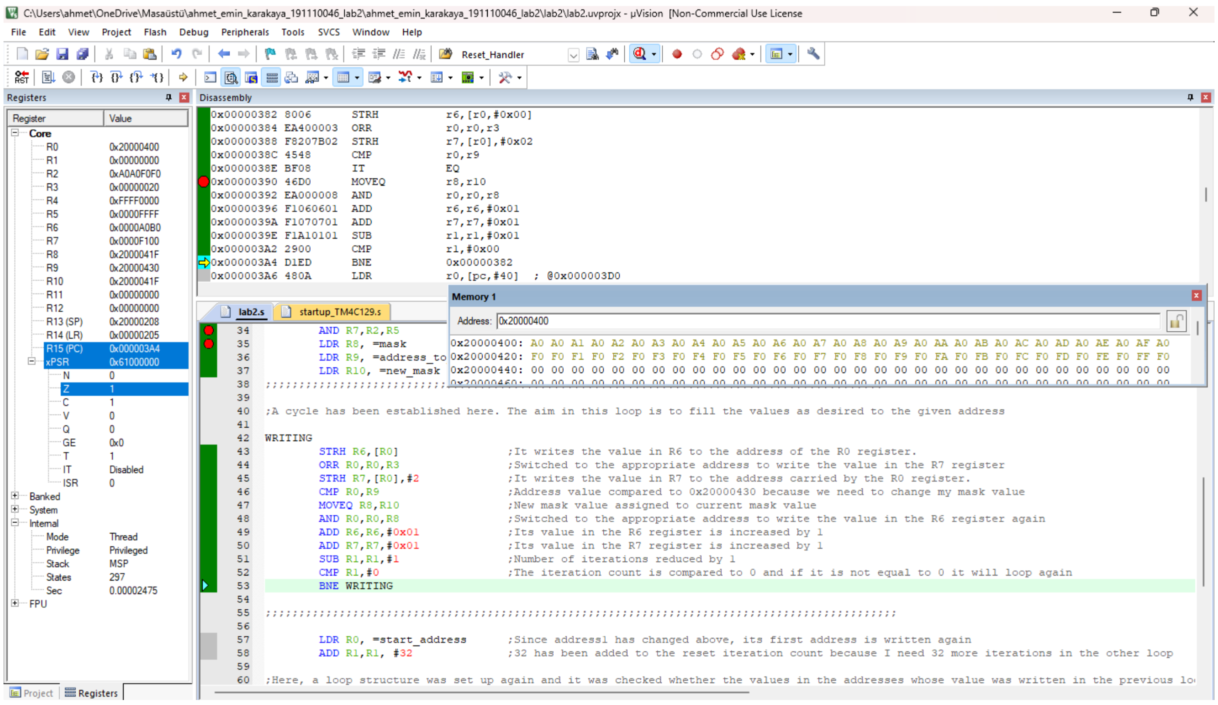
```
0x00000382 8006      STRH      r6,[r0,#0x00]
0x00000384 EA400003  ORR       r0,r0,r3
0x00000388 F8207B02  STRH      r7,[r0],#0x02
0x0000038C 4548      CMP       r0,r9
0x0000038E BF08      IT        EQ
0x00000390 46D0      MOVEQ     r8,r10
0x00000392 EA000008  AND       r0,r0,r8
0x00000396 F1060601  ADD       r6,r6,#0x01
0x0000039A F1070701  ADD       r7,r7,#0x01
0x0000039E F1A10101  SUB       r1,r1,#0x01
0x000003A2 2900      CMP       r1,#0x00
0x000003A4 D1ED      BNE       0x00000382
0x000003A6 480A      LDR       r0,[pc,#40]  ; @0x000003D0
```

```
Address: 0x20000400
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000440: 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
0x20000460: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]               ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3               ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2            ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                  ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10               ;New mask value assigned to current mask value
        AND R0,R0,R8               ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01            ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01            ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1               ;Number of iterations reduced by 1
        CMP R1,#0                  ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address     ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32             ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 16

After continuing in this way, I finally placed the data as desired.(Figure 17)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x20000400
R1	0x00000000
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003A4
xPSR	0x61000000
N	0
Z	1
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	297
Sec	0.00002475
FPU	

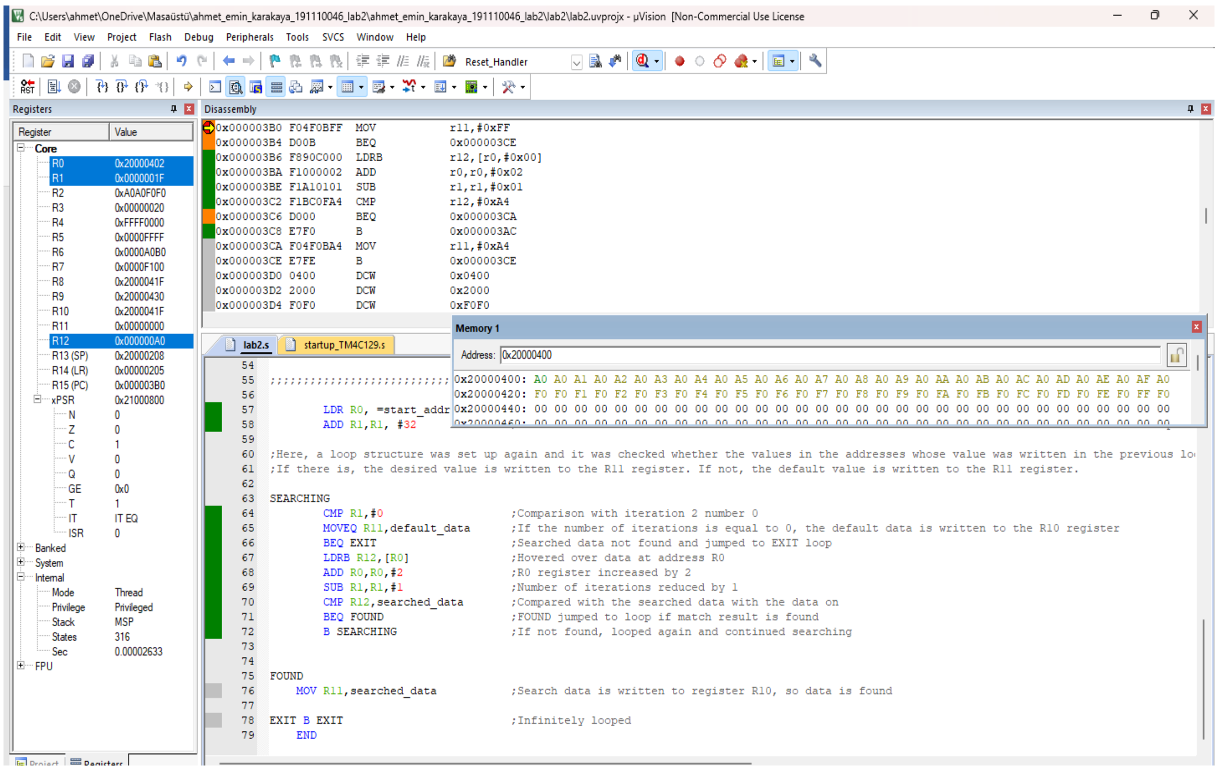
```
0x00000382 8006     STRH          r6,[r0,#0x00]
0x00000384 EA400003 ORR           r0,r0,r3
0x00000388 F8207B02 STRH          r7,[r0],#0x02
0x0000038C 4548     CMP           r0,r9
0x0000038E BF08     IT            EQ
0x00000390 46D0     MOVEQ         r8,r10
0x00000392 EA000008 AND           r0,r0,r8
0x00000396 F1060601 ADD           r6,r6,#0x01
0x0000039A F1070701 ADD           r7,r7,#0x01
0x0000039E F1A10101 SUB           r1,r1,#0x01
0x000003A2 2900     CMP           r1,#0x00
0x000003A4 D1ED     BNE           0x00000382
0x000003A6 480A     LDR           r0,[pc,#40]  ; @0x000003D0
```

```
Address: 0x20000400
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 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
```

```
        AND R7,R2,R5
        LDR R8, =mask
        LDR R9, =address_to
        LDR R10, =new_mask
;;;;;;;;;;;;;;;;;;;;;;;;;;;

;A cycle has been established here. The aim in this loop is to fill the values as desired to the given address

WRITING
        STRH R6,[R0]                ;It writes the value in R6 to the address of the R0 register.
        ORR R0,R0,R3                ;Switched to the appropriate address to write the value in the R7 register
        STRH R7,[R0],#2             ;It writes the value in R7 to the address carried by the R0 register.
        CMP R0,R9                   ;Address value compared to 0x20000430 because we need to change my mask value
        MOVEQ R8,R10                ;New mask value assigned to current mask value
        AND R0,R0,R8                ;Switched to the appropriate address to write the value in the R6 register again
        ADD R6,R6,#0x01             ;Its value in the R6 register is increased by 1
        ADD R7,R7,#0x01             ;Its value in the R7 register is increased by 1
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R1,#0                   ;The iteration count is compared to 0 and if it is not equal to 0 it will loop again
        BNE WRITING

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_address      ;Since address1 has changed above, its first address is written again
        ADD R1,R1, #32              ;32 has been added to the reset iteration count because I need 32 more iterations in the other loop

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
```

Figure 17

After these screenshots, screenshots will be shown showing that the searched value is found. Trying to find the search value one by one.(Figure 18)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x20000402
R1	0x0000001F
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x000000A0
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003B0
xPSR	0x21000800
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	IT EQ
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	316
Sec	0.00002633
FPU	

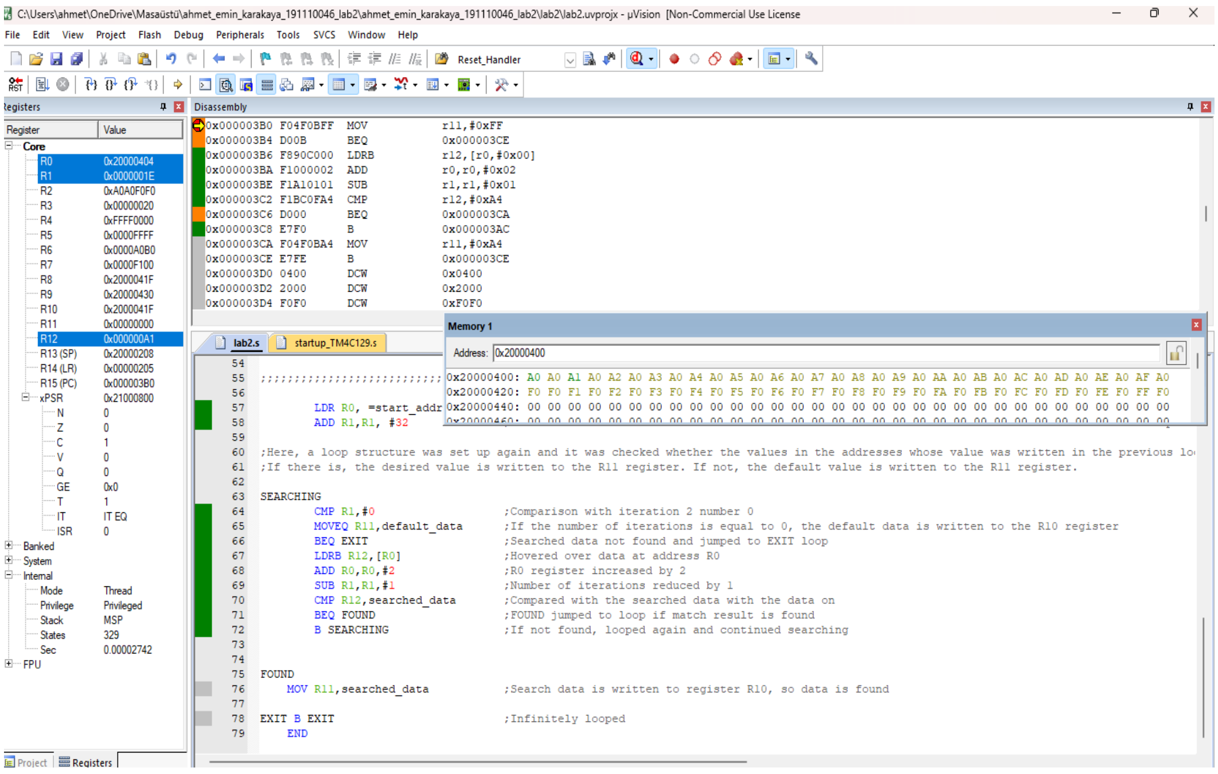
```
0x000003B0 F04F0BFF MOV           r11,#0xFF
0x000003B4 D00B     BEQ           0x000003CE
0x000003B6 F890C000 LDRB          r12,[r0,#0x00]
0x000003BA F1000002 ADD           r0,r0,#0x02
0x000003BE F1A10101 SUB           r1,r1,#0x01
0x000003C2 F1BC0FA4 CMP           r12,#0xA4
0x000003C6 D000     BEQ           0x000003CA
0x000003C8 E7F0     B             0x000003AC
0x000003CA F04F0BA4 MOV           r11,#0xA4
0x000003CE E7FE     B             0x000003CE
0x000003D0 0400     DCW           0x0400
0x000003D2 2000     DCW           0x2000
0x000003D4 F0F0     DCW           0xF0F0
```

```
Address: 0x20000400
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 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
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                   ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data      ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                    ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]               ;Hovered over data at address R0
        ADD R0,R0,#2                ;R0 register increased by 2
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R12,searched_data       ;Compared with the searched data with the data on
        BEQ FOUND                   ;FOUND jumped to loop if match result is found
        B SEARCHING                 ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data           ;Search data is written to register R10, so data is found

EXIT B EXIT                         ;Infinitely looped
    END
```

Figure 18

Compared with A1. (Figure 19)



C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Register	Value
Core	
R0	0x20000404
R1	0x0000001E
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x000000A1
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003B0
xPSR	0x21000800
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	IT EQ
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	329
Sec	0.00002742
FPU	

Disassembly

```
0x000003B0 F04F0BFF  MOV       r11,#0xFF
0x000003B4 D00B      BEQ       0x000003CE
0x000003B6 F890C000  LDRB      r12,[r0,#0x00]
0x000003BA F1000002  ADD       r0,r0,#0x02
0x000003BE F1A10101  SUB       r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP       r12,#0xA4
0x000003C6 D000      BEQ       0x000003CA
0x000003C8 E7F0      B         0x000003AC
0x000003CA F04F0BA4  MOV       r11,#0xA4
0x000003CE E7FE      B         0x000003CE
0x000003D0 0400      DCW       0x0400
0x000003D2 2000      DCW       0x2000
0x000003D4 F0F0      DCW       0xF0F0
```

Memory 1

Address: 0x20000400

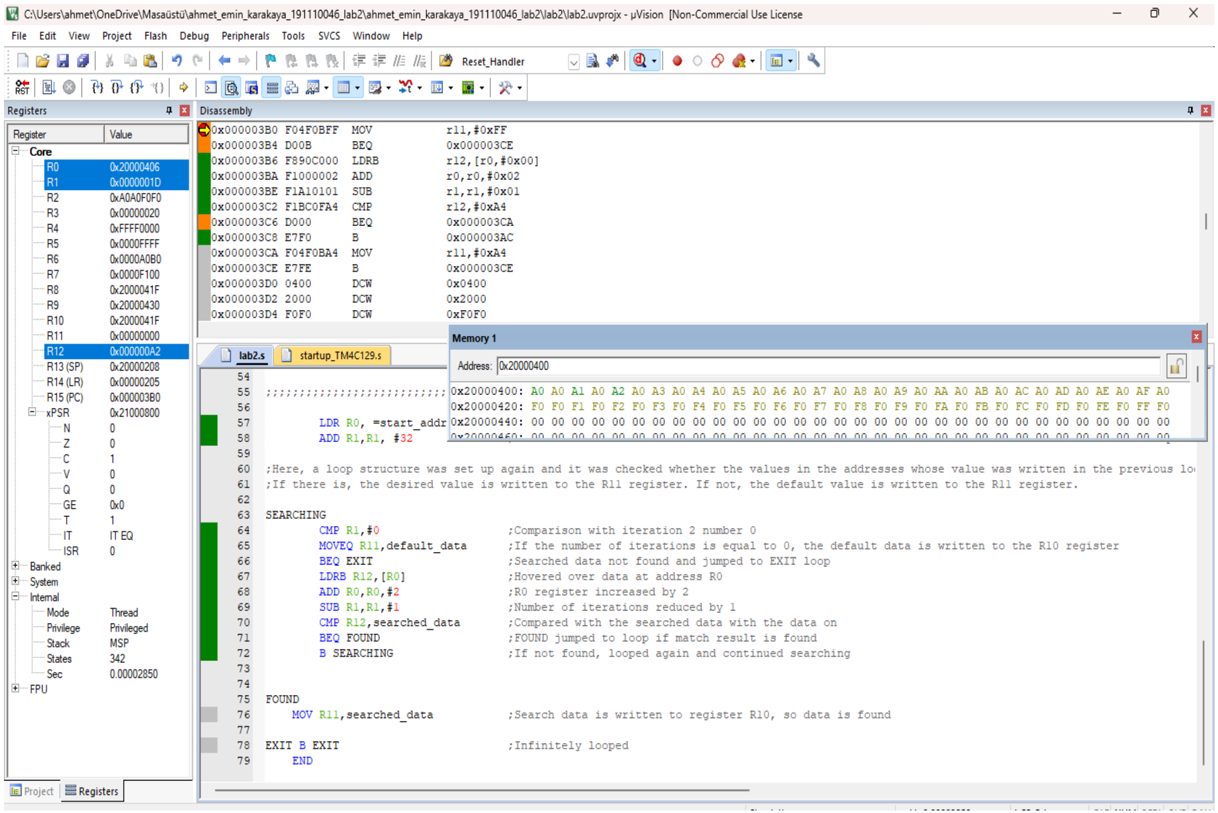
```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000460: 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
```

lab2.s | startup_TM4C129.s

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                   ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data      ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                    ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]               ;Hovered over data at address R0
        ADD R0,R0,#2                ;R0 register increased by 2
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R12,searched_data       ;Compared with the searched data with the data on
        BEQ FOUND                   ;FOUND jumped to loop if match result is found
        B SEARCHING                 ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data           ;Search data is written to register R10, so data is found

EXIT B EXIT                         ;Infinitely looped
    END
```

Figure 19

Compared with A2. (Figure 20)



C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Register	Value
Core	
R0	0x20000406
R1	0x0000001D
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x000000A2
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003B0
xPSR	0x21000800
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	IT EQ
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	342
Sec	0.00002850
FPU	

Disassembly

```
0x000003B0 F04F0BFF  MOV       r11,#0xFF
0x000003B4 D00B      BEQ       0x000003CE
0x000003B6 F890C000  LDRB      r12,[r0,#0x00]
0x000003BA F1000002  ADD       r0,r0,#0x02
0x000003BE F1A10101  SUB       r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP       r12,#0xA4
0x000003C6 D000      BEQ       0x000003CA
0x000003C8 E7F0      B         0x000003AC
0x000003CA F04F0BA4  MOV       r11,#0xA4
0x000003CE E7FE      B         0x000003CE
0x000003D0 0400      DCW       0x0400
0x000003D2 2000      DCW       0x2000
0x000003D4 F0F0      DCW       0xF0F0
```

Memory 1

Address: 0x20000400

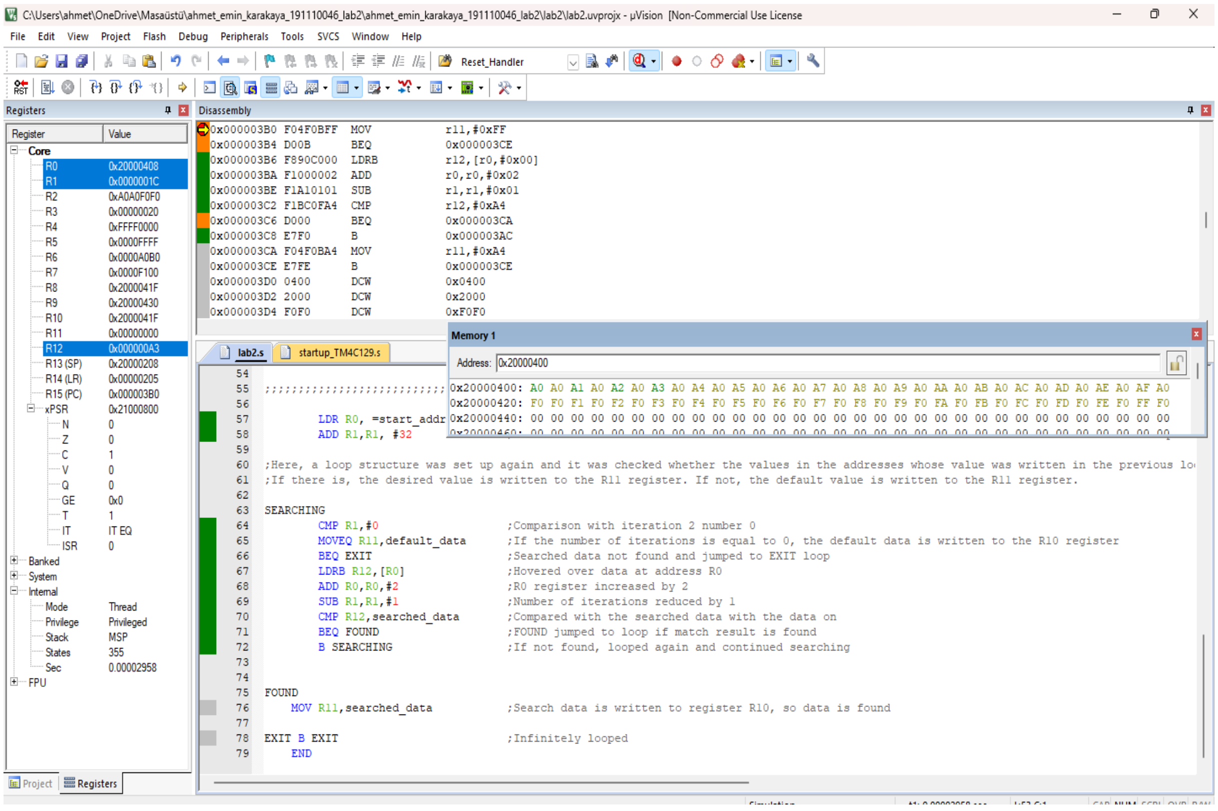
```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000460: 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
```

lab2.s | startup_TM4C129.s

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                   ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data      ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                    ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]               ;Hovered over data at address R0
        ADD R0,R0,#2                ;R0 register increased by 2
        SUB R1,R1,#1                ;Number of iterations reduced by 1
        CMP R12,searched_data       ;Compared with the searched data with the data on
        BEQ FOUND                   ;FOUND jumped to loop if match result is found
        B SEARCHING                 ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data           ;Search data is written to register R10, so data is found

EXIT B EXIT                         ;Infinitely looped
    END
```

Figure 20

Compared with A3. (Figure 21)



C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Registers

Register	Value
Core	
R0	0x20000408
R1	0x0000001C
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x000000A3
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003B0
xPSR	0x21000800
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	IT EQ
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	355
Sec	0.00002958
FPU	

Disassembly

```
0x000003B0 F04F0BFF  MOV          r11,#0xFF
0x000003B4 D00B      BEQ          0x000003CE
0x000003B6 F890C000  LDRB         r12,[r0,#0x00]
0x000003BA F1000002  ADD          r0,r0,#0x02
0x000003BE F1A10101  SUB          r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP          r12,#0xA4
0x000003C6 D000      BEQ          0x000003CA
0x000003C8 E7F0      B            0x000003AC
0x000003CA F04F0BA4  MOV          r11,#0xA4
0x000003CE E7FE      B            0x000003CE
0x000003D0 0400      DCW          0x0400
0x000003D2 2000      DCW          0x2000
0x000003D4 F0F0      DCW          0xF0F0
```

Memory 1

Address: 0x20000400

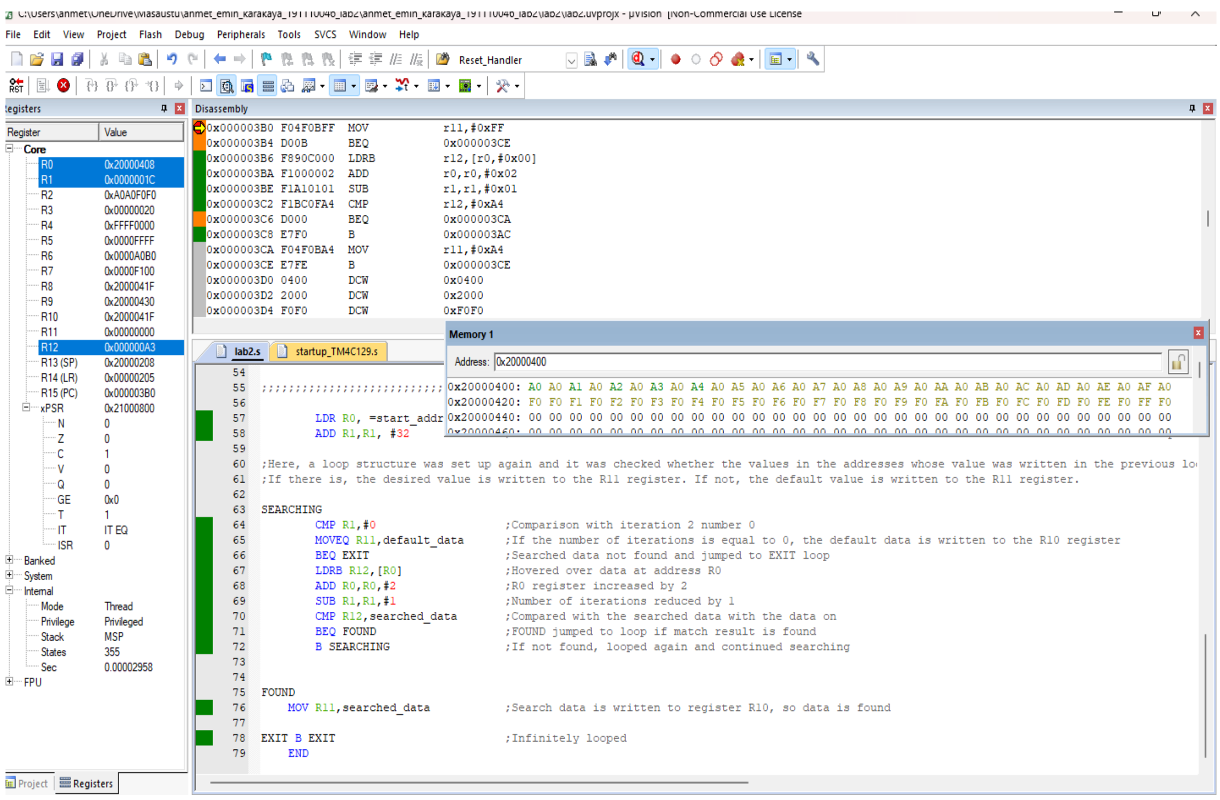
```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

lab2.s | startup_TM4C129.s

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                  ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data     ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                   ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]              ;Hovered over data at address R0
        ADD R0,R0,#2               ;R0 register increased by 2
        SUB R1,R1,#1               ;Number of iterations reduced by 1
        CMP R12,searched_data      ;Compared with the searched data with the data on
        BEQ FOUND                  ;FOUND jumped to loop if match result is found
        B SEARCHING                ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data          ;Search data is written to register R10, so data is found

EXIT B EXIT                        ;Infinitely looped
    END
```

Figure 21

Compared with A4. (Figure 22)



C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License

Registers

Register	Value
Core	
R0	0x20000408
R1	0x0000001C
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x00000000
R12	0x000000A3
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003B0
xPSR	0x21000800
N	0
Z	0
C	1
V	0
Q	0
GE	0x0
T	1
IT	IT EQ
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	355
Sec	0.00002958
FPU	

Disassembly

```
0x000003B0 F04F0BFF  MOV          r11,#0xFF
0x000003B4 D00B      BEQ          0x000003CE
0x000003B6 F890C000  LDRB         r12,[r0,#0x00]
0x000003BA F1000002  ADD          r0,r0,#0x02
0x000003BE F1A10101  SUB          r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP          r12,#0xA4
0x000003C6 D000      BEQ          0x000003CA
0x000003C8 E7F0      B            0x000003AC
0x000003CA F04F0BA4  MOV          r11,#0xA4
0x000003CE E7FE      B            0x000003CE
0x000003D0 0400      DCW          0x0400
0x000003D2 2000      DCW          0x2000
0x000003D4 F0F0      DCW          0xF0F0
```

Memory 1

Address: 0x20000400

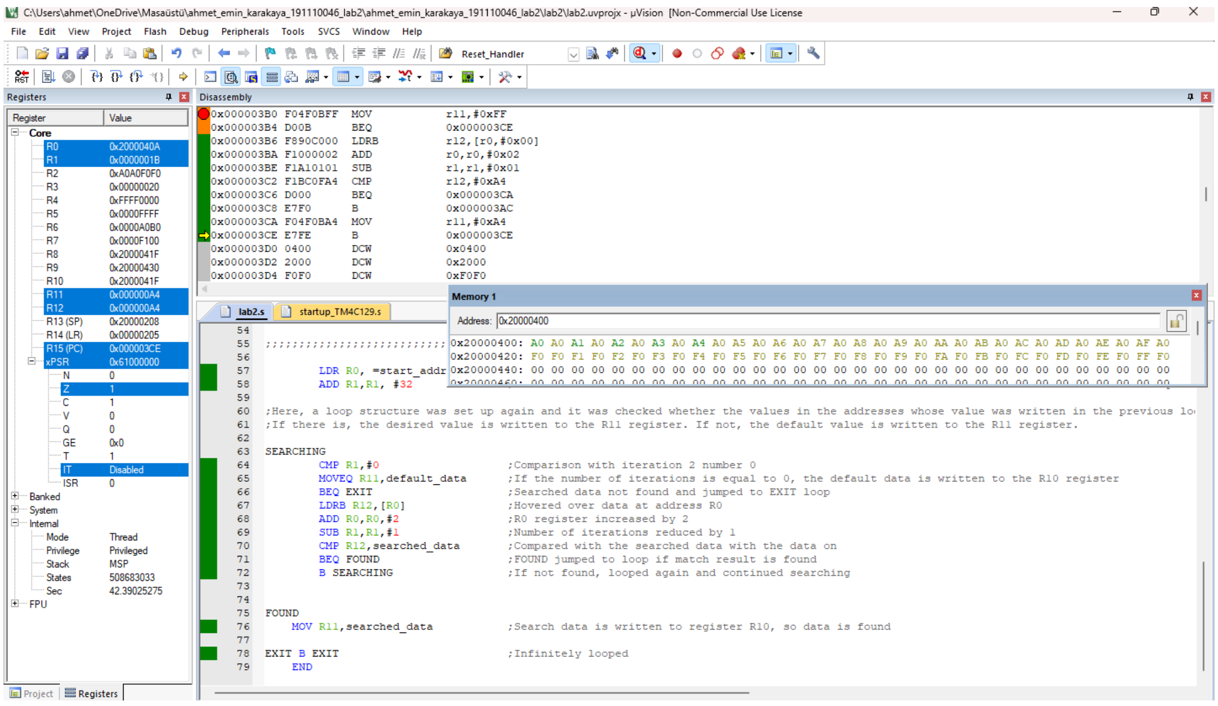
```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

lab2.s | startup_TM4C129.s

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                  ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data     ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                   ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]              ;Hovered over data at address R0
        ADD R0,R0,#2               ;R0 register increased by 2
        SUB R1,R1,#1               ;Number of iterations reduced by 1
        CMP R12,searched_data      ;Compared with the searched data with the data on
        BEQ FOUND                  ;FOUND jumped to loop if match result is found
        B SEARCHING                ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data          ;Search data is written to register R10, so data is found

EXIT B EXIT                        ;Infinitely looped
    END
```

Figure 22

The last one matched and wrote to register R11 to see if it found it. If he couldn't find it, he would have continued to search. (Figure 23)


C:\Users\ahmet\OneDrive\Masaüstü\ahmet_emin_karakaya_191110046_lab2\ahmet_emin_karakaya_191110046_lab2\lab2\lab2.uvprojx - µVision [Non-Commercial Use License
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Reset_Handler
Registers
Disassembly
Memory 1
lab2.s
startup_TM4C129.s
Project
Registers

Register	Value
Core	
R0	0x2000040A
R1	0x0000001B
R2	0xA0A0F0F0
R3	0x00000020
R4	0xFFFF0000
R5	0x0000FFFF
R6	0x0000A0B0
R7	0x0000F100
R8	0x2000041F
R9	0x20000430
R10	0x2000041F
R11	0x000000A4
R12	0x000000A4
R13 (SP)	0x20000208
R14 (LR)	0x00000205
R15 (PC)	0x000003CE
xPSR	0x61000000
N	0
Z	1
C	1
V	0
Q	0
GE	0x0
T	1
IT	Disabled
ISR	0
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	508683033
Sec	42.39025275
FPU	

```
0x000003B0 F04F0BFF  MOV        r11,#0xFF
0x000003B4 D00B      BEQ        0x000003CE
0x000003B6 F890C000  LDRB       r12,[r0,#0x00]
0x000003BA F1000002  ADD        r0,r0,#0x02
0x000003BE F1A10101  SUB        r1,r1,#0x01
0x000003C2 F1BC0FA4  CMP        r12,#0xA4
0x000003C6 D000      BEQ        0x000003CA
0x000003C8 E7F0      B          0x000003AC
0x000003CA F04F0BA4  MOV        r11,#0xA4
0x000003CE E7FE      B          0x000003CE
0x000003D0 0400      DCW        0x0400
0x000003D2 2000      DCW        0x2000
0x000003D4 F0F0      DCW        0xF0F0
```

Address: 0x20000400

```
0x20000400: A0 A0 A1 A0 A2 A0 A3 A0 A4 A0 A5 A0 A6 A0 A7 A0 A8 A0 A9 A0 AA A0 AB A0 AC A0 AD A0 AE A0 AF A0
0x20000420: F0 F0 F1 F0 F2 F0 F3 F0 F4 F0 F5 F0 F6 F0 F7 F0 F8 F0 F9 F0 FA F0 FB F0 FC F0 FD F0 FE F0 FF F0
0x20000440: 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
```

```

;;;;;;;;;;;;;;;;;;;;;;;;;;

        LDR R0, =start_addr
        ADD R1,R1, #32

;Here, a loop structure was set up again and it was checked whether the values in the addresses whose value was written in the previous lo
;If there is, the desired value is written to the R11 register. If not, the default value is written to the R11 register.

SEARCHING
        CMP R1,#0                       ;Comparison with iteration 2 number 0
        MOVEQ R11,default_data          ;If the number of iterations is equal to 0, the default data is written to the R10 register
        BEQ EXIT                        ;Searched data not found and jumped to EXIT loop
        LDRB R12,[R0]                   ;Hovered over data at address R0
        ADD R0,R0,#2                    ;R0 register increased by 2
        SUB R1,R1,#1                    ;Number of iterations reduced by 1
        CMP R12,searched_data           ;Compared with the searched data with the data on
        BEQ FOUND                       ;FOUND jumped to loop if match result is found
        B SEARCHING                     ;If not found, looped again and continued searching


FOUND
    MOV R11,searched_data               ;Search data is written to register R10, so data is found

EXIT B EXIT                             ;Infinitely looped
    END
```

Figure 23