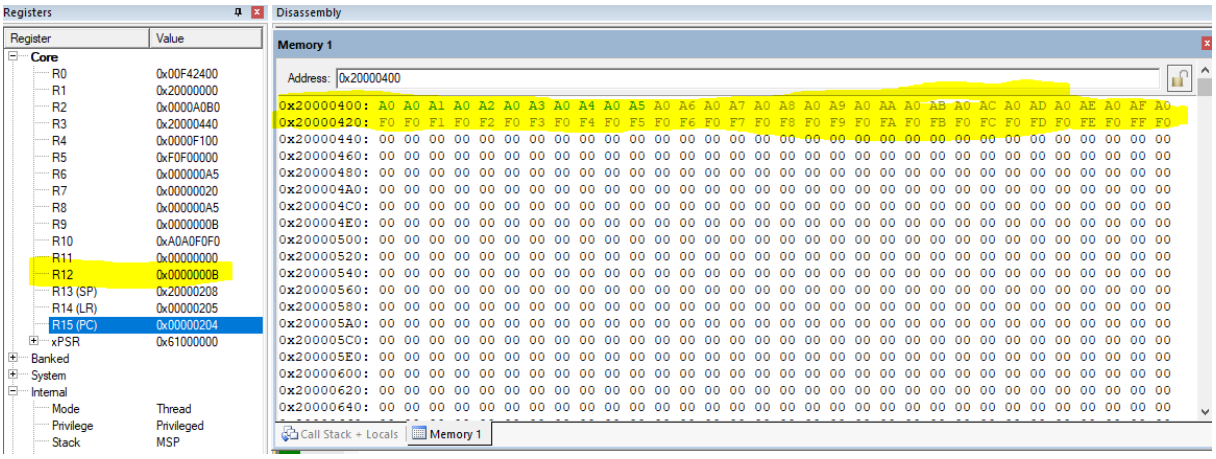
CODE BLOG

```
atama1          EQU     0XA0A0F0F0              ;odev icin verilen deger
iteration       EQU     16                      ;memory ye 16 cycle da kaydedebilcegimiz icin verilen deger
adress          EQU     0x20000400              ;soruda verilen ilk adress
aranan_eleman   EQU     0XA4                    ;aramak istedigim deger

            AREA MYCODE,CODE
                ALIGN
                ENTRY
                EXPORT __main
__main
                LDR         R0,=iteration       ;R0 registirana iterationa atadim
                LDR         R1,=adress
                LDR         R10,=atama1
                ADD         R7,R0,R0            ;r7 = r0 + r0
                ADD         R3,R7,R1
                MOV         R2,R10, LSR #16     ;0X0000A0A0
                MOV         R5,R10, LSL #16     ;0XF0F00000
                MOV         R4,R5,  LSR #16     ;0X0000F0F0


L1              STRH        R2,[R1]             ;STRH halfwors biçimde memory e kayit islemi yapar, adress R1 e R2 degerinin en anlamli 2 byte ni kaydetti
                STRH        R4,[R3]             ;R3 adresine r4 ün en anlamli 4 biti kaydedildi
                ADD         R1,R1,#2            ;memory de 400 ile baslayan adrese 16 bitlik degerleri yazdirdigimizdan dolayi adres 2 artirildi.
                ADD         R3,R3,#2            ;memory de 420 ile baslayan adrese 16 bitlik degerleri yazdirdigimizdan dolayi adres 2 artirildi.
                ADD         R2,R2,#1            ;soruda istenilen degeri birer birer arirarak hafizaya kaydetmemizdi, A0A0----- A1A0 olur
                ADD         R4,R4,#1            ;soruda istenilen degeri birer birer arirarak hafizaya kaydetmemizdi, F0F0----- F1A0 olur
                SUBS        R0,R0,#1            ;16 cycle icin donguyu her adimda 1 azaltmasi icin ve R0 degeri 0 oldugunda bayrak ile cikis saglamak icin SUBS kullanildi
                BNE         L1                  ;bayrak aktiflestiginde dongu den cikiyor

;LINEAR SEARCH
                LDR         R0,=iteration
                LDR         R1,=adress
                MOV         R9,#0               ;R9 a baslangic degeri atadik
                LDR         R8,=aranan_eleman


L2
                CMP         R8,R6               ;CMP komutu R8 ve R6 arasinda karsilastirma yapar
                BEQ         next                ;Eger R8,R6 esit olursa donguden next ile ciksin
                LDRB        R6 ,[R1,R9]         ;LDRB komutu R1 adresine R9 ofset i kadar gider ve o adresdeki degerini R6 ya atar
                ADD         R9, R9,#1
                B           L2
next
                MOV         R12,R9              ;R9 sayisi hangi kosulda saglanir ise biz R12 register ina atamamizi yapdi
                END
```

```
atama1          EQU     0XA0A0F0F0
iteration       EQU     16
adress          EQU     0x20000400
aranan_eleman   EQU     0XA4

            AREA MYCODE,CODE
                ALIGN
                ENTRY
                EXPORT __main
__main
                LDR         R0,=iteration
                LDR         R1,=adress
                LDR         R10,=atama1
                ADD         R7,R0,R0
                ADD         R3,R7,R1
                MOV         R2,R10, LSR #16
                MOV         R5,R10, LSL #16
                MOV         R4,R5,  LSR #16


L1              STRH        R2,[R1]
                STRH        R4,[R3]
                ADD         R1,R1,#2
                ADD         R3,R3,#2
                ADD         R2,R2,#1
                ADD         R4,R4,#1
                SUBS        R0,R0,#1
                BNE         L1
```

```
;LINEAR SEARCH
                LDR         R0,=iteration
                LDR         R1,=adress
                MOV         R9,#0
                LDR         R8,=aranan_eleman


L2
                CMP         R8,R6
                BEQ         next
                LDRB        R6 ,[R1,R9]
                ADD         R9, R9,#1
                B           L2
next
                MOV         R12,R9
                END
```

MEMORY AND REGISTER



Instructions

- **LSR** : Register x with logical shift left by n bits (0 = n = 31)
- **LSL** : Register x with logical shift right by n bits (1 = n = 32)
- **STRH** : Store unsigned Half Word
- **LDRB** : Load unsigned Byte

ALGORITHM and CODE

First of all, the address information given in the question, the number of cycles and the number of elements I are defined with the EQU instruction. The variables R0, R1, R10 are assigned to registers. Here, 2R0 is added to R3 to define the address that starts with 0x00000420. An important point; Using LSR AND LSL instruction, A0A0F0F0 was converted into 0000A0A0 and 0000F0F0 format.

LOOP 1; The divided values have been saved at 0x00000400 and 0x00000420. The saving was done with STRH, The value of R2 is assigned to the address owned by R1, and the value of R3 is assigned to the address owned by R4. Then A0A0 value is expected to be A1A0, and F0F0 value to be F1F0. For this, 2 is added to the addresses because A0A0 takes 2 bytes of space in memory. We wanted the new value to be added to address 0x00000402. In addition, we increased our value by 1. Before I forget, the reason why there is A1A0 in the memory blog is that it starts from low priority bits while saving to memory. To control the loop, the defined R0 is decremented one by one and if SUBS also changes the flag, the loop exits.

LINEAR SEARCH; Required values for the searched element and its loop are defined before entering freeze. While in loop, the values of R8 and R6 were compared with CMP instructions, If R8 and R6 are equal, use the BEQ instruction to exit the loop. If there is an equality, it is specified to exit the loop, but if there is no equality, we tried to find the value we are looking for by browsing the memory with the LDRB instructions. R6 value in LDRB; At the address R1, R9 is incrementally increased and thus the values starting from 400 to the last position in memory are scanned. R9 is offset value in LDRB.

Lastly I saved the offset value(R9's values) at R12' register.