جامعة بيرزيت

BIRZEIT UNIVERSITY

Electrical and Computer Engineering

Computer Organization and Microprocessors – ENCS2380

Assembly Assignment

Spring 2022

Name: Jehad Hamayel

ID: 1200348

Instructor: Dr. Abualseoud Hanani

Sec: 2

Date: 18/6/2022

My Code:

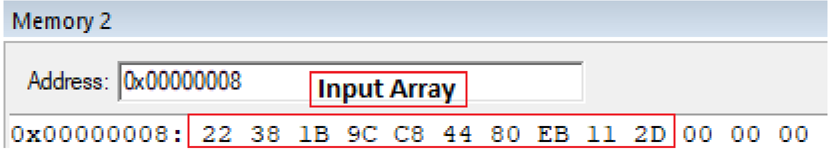
```
      AREA RESET, DATA, READONLY


        EXPORT __Vectors
__Vectors  DCD 0x20001000 ; stack pointer value when stack is empty
           DCD Reset_Handler; reset vector
                                align
;AREA DATA
Array DCB 34,56,27,156,200,68,128,235,17,45


                AREA MYRAM, DATA, READWRITE
;Assigning initial values to the variables
Sumation DCD 0
EVENSUM DCD 0
Array2 DCB 0,0,0,0,0,0,0,0,0,0
;AREA CODE
      area mycode, code, readonly
;Naming the registers with names to know what we want to store in them
SUM RN R10
EVENsum RN R11
count RN R12
count2 RN R9
returnValue RN R6
       ENTRY
       EXPORT Reset_Handler
Reset_Handler



                   BL GetSUM                                    ;Calling the procedure GetSUM
                                LDR R5,=Sumation       ;Take the address where we want to store the SUM
                                STR SUM,[R5]                             ;Data storage in memory
```

```
            BL GetEVENSUM               ;Calling the procedure GetEVENSUM
            LDR R5,=EVENSUM                     ;Take the address where we want to store the EVENsum
            STR EVENsum,[R5]            ;Data storage in memory


            MOV count,#10
            LDR R1,=Array               ;Marking the location of the first element in the Array To READ
            LDR R5,=Array2      ;Marking the location of the first element in the Second Array TO Store
AGAIN3 LDRB R2,[R1]             ;loop to pass the values
            BL POW                              ;Calling the procedure POW
            SUBS count,count,#1         ;Decrement for the counter
            ADD R1,R1,#1                ;Take the second item's location To READ
            STRB returnValue,[R5]   ;Store Data in Memory
            ADD R5,R5,#1                ;Take the second item's location To Store
            BNE AGAIN3                          ;Branch for loop


here B here
end
GetSUM                          ;Procedure For Get the Sumation
;Give the required values for the program in the registry
            MOV count,#10
            MOV SUM ,#0
            LDR R1, =Array    ;Marking the location of the first element in the Array
AGAIN   LDRB R2,[R1]    ;A loop to add the values in the Array and put the sum into the SUM register
            ADD SUM,SUM,R2
            SUBS count,count,#1   ;Decrement for the counter
            ADD R1,R1,#1          ;Take the second item's location To READ
            BNE AGAIN       ;Branch for loop
            BX LR
GetEVENSUM                  ;Procedure For Get the even numbers Sumation
;Give the required values for the program in the registry
            MOV EVENsum,#0
            MOV count,#10
            LDR R1, =Array              ;Marking the location of the first element in the Array
AGAIN2 LDRB R2,[R1]         ;A loop to add the EVEN values in the Array and put the sum into the SUM register
```

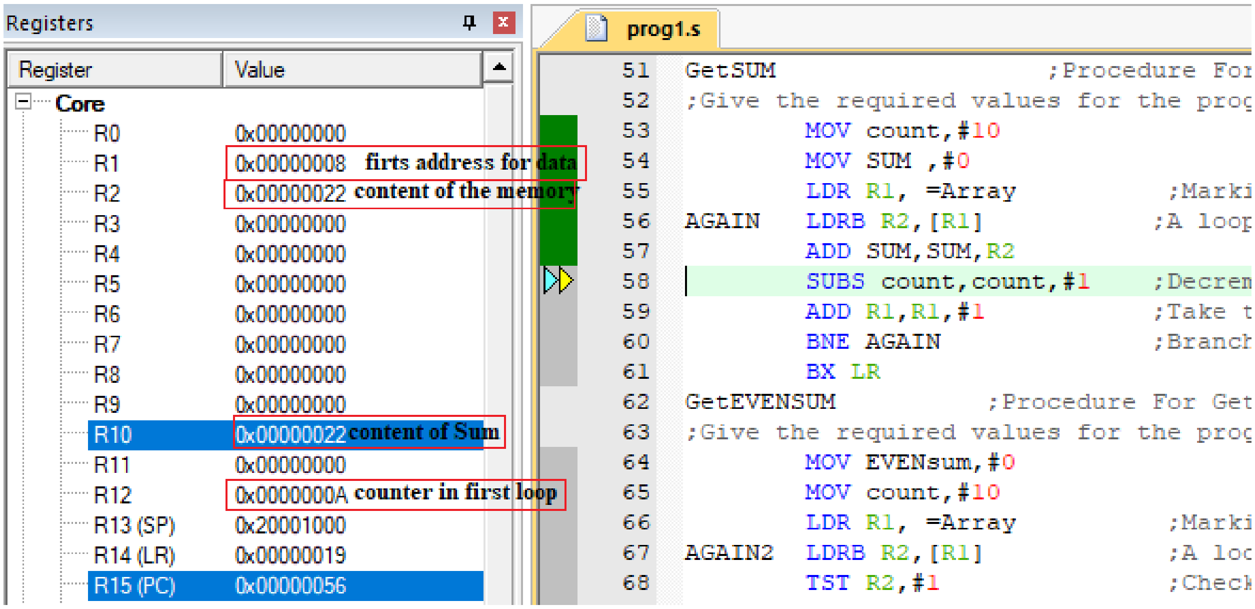
```
                    TST R2,#1                                  ;Check the first bit to see if the number is EVEN or Odd
                    BNE ODD                                              ;Branch if is in not EVEN
                    ADD EVENsum,EVENsum,R2          ;Add it to EVENsum if it is EVEN
ODD
                    SUBS count,count,#1                ;Decrement for the counter
                    ADD R1,R1,#1                ;Take the second item's location To READ
                    BNE AGAIN2                                 ;Branch for loop
                    BX LR
count3 RN R8
POW                                                ;Procedure For Find the largest power of 2 divisor that divides into a number
;Give the required values for the program in the registry
          MOV count2,#8
          mov R3,#1
          mov count3,#0
          CMP R2,#0        ;compare if it is ZERO
          MOV returnValue,#0
          BEQ leave ;Branch to leave
loop                          ;Loop for Find the largest power of 2 divisor that divides into a number
          TST R2,R3                  ;Check the specific bit to find which bit we arrive
          BNE out
          ADD count3,count3,#1
          SUBS count2,count2,#1      ;Decrement for the counter
          LSL R3 ,#1                 ;Shift left to check the second bit
          BNE loop                   ;Branch for loop
out
          MOV returnValue,#1
          MOV R7,#2
          CMP count3,#0
          BEQ leave
loop2   ;loop for get the Required Number
                    MUL returnValue,returnValue,R7      ; returnValue = returnValue * 2
                    SUBS count3,count3,#1      ;Decrement for the counter
                    BNE loop2                  ;Branch for loop
leave     BX LR
```

Output:

Input Array : 34,56,27,156,200,68,128,235,17,45

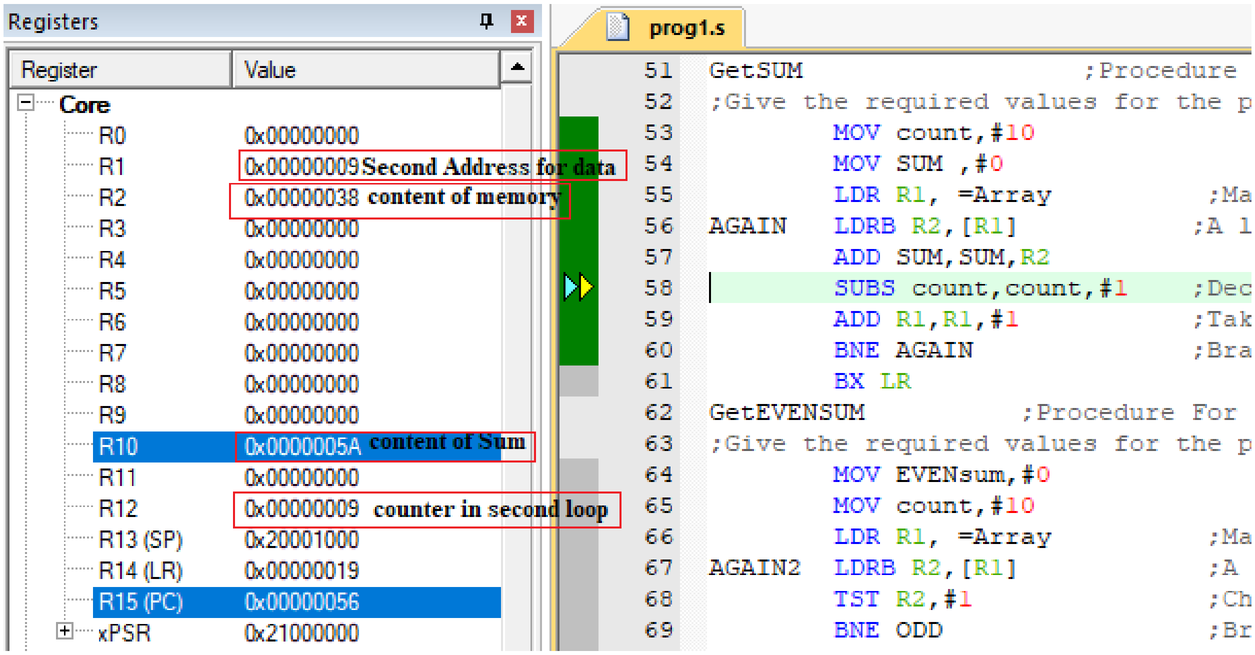


```
Memory 2
Address: 0x00000008        Input Array
0x00000008: 22 38 1B 9C C8 44 80 EB 11 2D 00 00 00
```



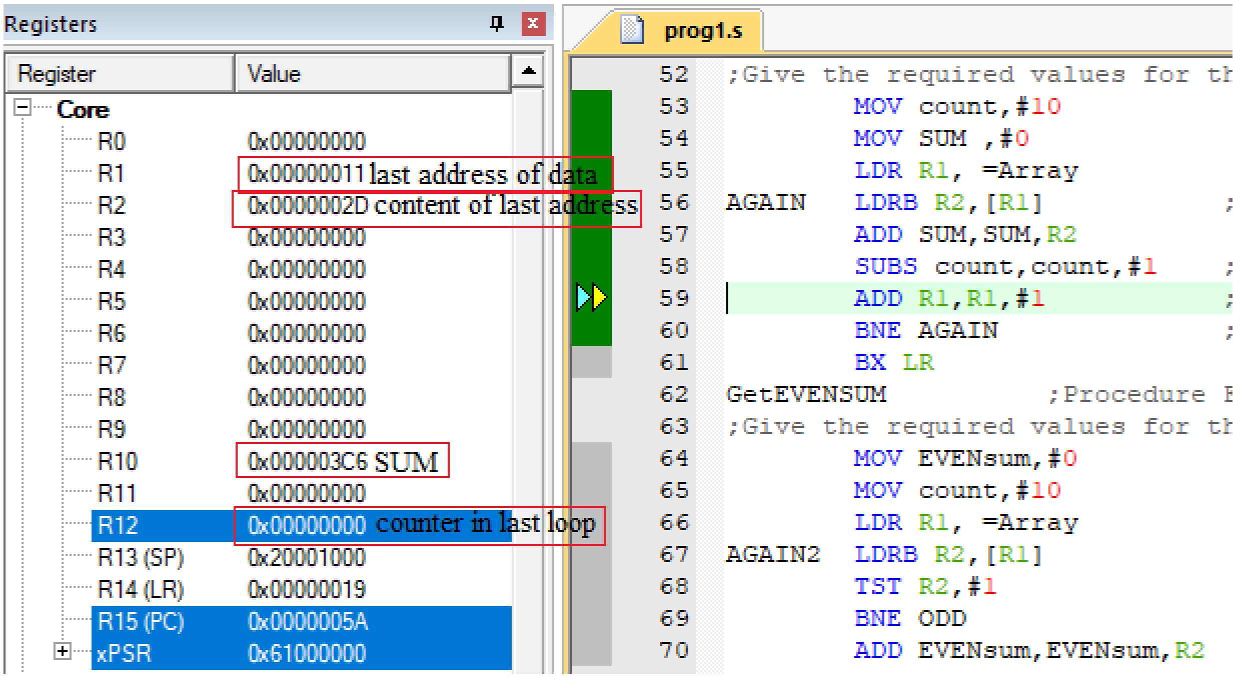
```
Registers
Register      Value
Core
  R0          0x00000000
  R1          0x00000008  firts address for data
  R2          0x00000022 content of the memory
  R3          0x00000000
  R4          0x00000000
  R5          0x00000000
  R6          0x00000000
  R7          0x00000000
  R8          0x00000000
  R9          0x00000000
  R10         0x00000022 content of Sum
  R11         0x00000000
  R12         0x0000000A counter in first loop
  R13 (SP)    0x20001000
  R14 (LR)    0x00000019
  R15 (PC)    0x00000056
```

```
prog1.s
GetSUM                                ;Procedure For
;Give the required values for the prog
        MOV   count,#10
        MOV   SUM ,#0
        LDR   R1, =Array              ;Marki
AGAIN   LDRB  R2,[R1]                 ;A loop
        ADD   SUM,SUM,R2
        SUBS  count,count,#1          ;Decrem
        ADD   R1,R1,#1                ;Take t
        BNE   AGAIN                   ;Branch
        BX    LR
GetEVENSUM                    ;Procedure For Get
;Give the required values for the prog
        MOV   EVENsum,#0
        MOV   count,#10
        LDR   R1, =Array              ;Marki
AGAIN2  LDRB  R2,[R1]                 ;A loo
        TST   R2,#1                   ;Check
```



```
Registers
Register      Value
Core
  R0          0x00000000
  R1          0x00000009Second Address for data
  R2          0x00000038 content of memory
  R3          0x00000000
  R4          0x00000000
  R5          0x00000000
  R6          0x00000000
  R7          0x00000000
  R8          0x00000000
  R9          0x00000000
  R10         0x0000005A  content of Sum
  R11         0x00000000
  R12         0x00000009  counter in second loop
  R13 (SP)    0x20001000
  R14 (LR)    0x00000019
  R15 (PC)    0x00000056
xPSR          0x21000000
```

```
prog1.s
GetSUM                                  ;Procedure
;Give the required values for the p
        MOV   count,#10
        MOV   SUM ,#0
        LDR   R1, =Array                ;Ma
AGAIN   LDRB  R2,[R1]                   ;A l
        ADD   SUM,SUM,R2
        SUBS  count,count,#1            ;Dec
        ADD   R1,R1,#1                  ;Tak
        BNE   AGAIN                     ;Bra
        BX    LR
GetEVENSUM                  ;Procedure For
;Give the required values for the p
        MOV   EVENsum,#0
        MOV   count,#10
        LDR   R1, =Array                ;Ma
AGAIN2  LDRB  R2,[R1]                   ;A
        TST   R2,#1                     ;Ch
        BNE   ODD                       ;Br
```

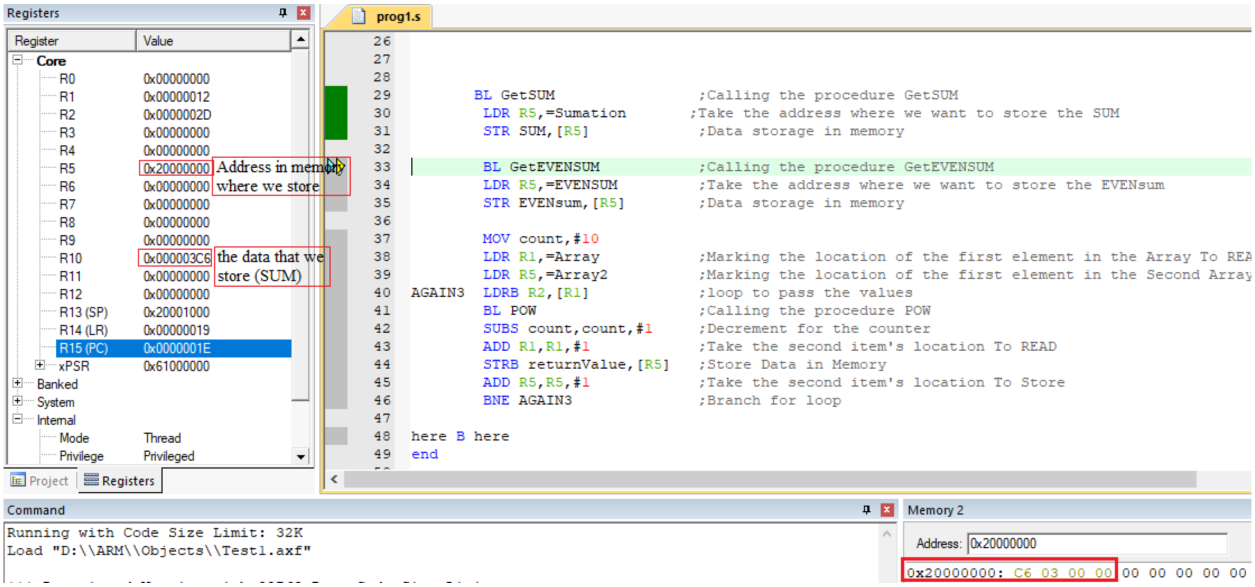


Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000011 last address of data
R2	0x0000002D content of last address
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6 SUM
R11	0x00000000
R12	0x00000000 counter in last loop
R13 (SP)	0x20001000
R14 (LR)	0x00000019
R15 (PC)	0x0000005A
xPSR	0x61000000

prog1.s

```
;Give the required values for th
        MOV count,#10
        MOV SUM ,#0
        LDR R1, =Array
AGAIN   LDRB R2,[R1]                       ;
        ADD SUM,SUM,R2
        SUBS count,count,#1                ;
        ADD R1,R1,#1                       ;
        BNE AGAIN                          ;
        BX LR
GetEVENSUM              ;Procedure F
;Give the required values for th
        MOV EVENsum,#0
        MOV count,#10
        LDR R1, =Array
AGAIN2  LDRB R2,[R1]
        TST R2,#1
        BNE ODD
        ADD EVENsum,EVENsum,R2
```



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000012
R2	0x0000002D
R3	0x00000000
R4	0x00000000
R5	0x20000000 Address in memory
R6	0x00000000 where we store
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6 the data that we
R11	0x00000000 store (SUM)
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0x00000019
R15 (PC)	0x0000001E
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged

prog1.s

```



        BL GetSUM                   ;Calling the procedure GetSUM
         LDR R5,=Sumation          ;Take the address where we want to store the SUM
         STR SUM,[R5]               ;Data storage in memory

         BL GetEVENSUM              ;Calling the procedure GetEVENSUM
         LDR R5,=EVENSUM            ;Take the address where we want to store the EVENsum
         STR EVENsum,[R5]           ;Data storage in memory

         MOV count,#10
         LDR R1,=Array              ;Marking the location of the first element in the Array To REA
         LDR R5,=Array2             ;Marking the location of the first element in the Second Array
AGAIN3   LDRB R2,[R1]               ;loop to pass the values
         BL POW                     ;Calling the procedure POW
         SUBS count,count,#1        ;Decrement for the counter
         ADD R1,R1,#1               ;Take the second item's location To READ
         STRB returnValue,[R5]      ;Store Data in Memory
         ADD R5,R5,#1               ;Take the second item's location To Store
         BNE AGAIN3                 ;Branch for loop

here B here
end
```

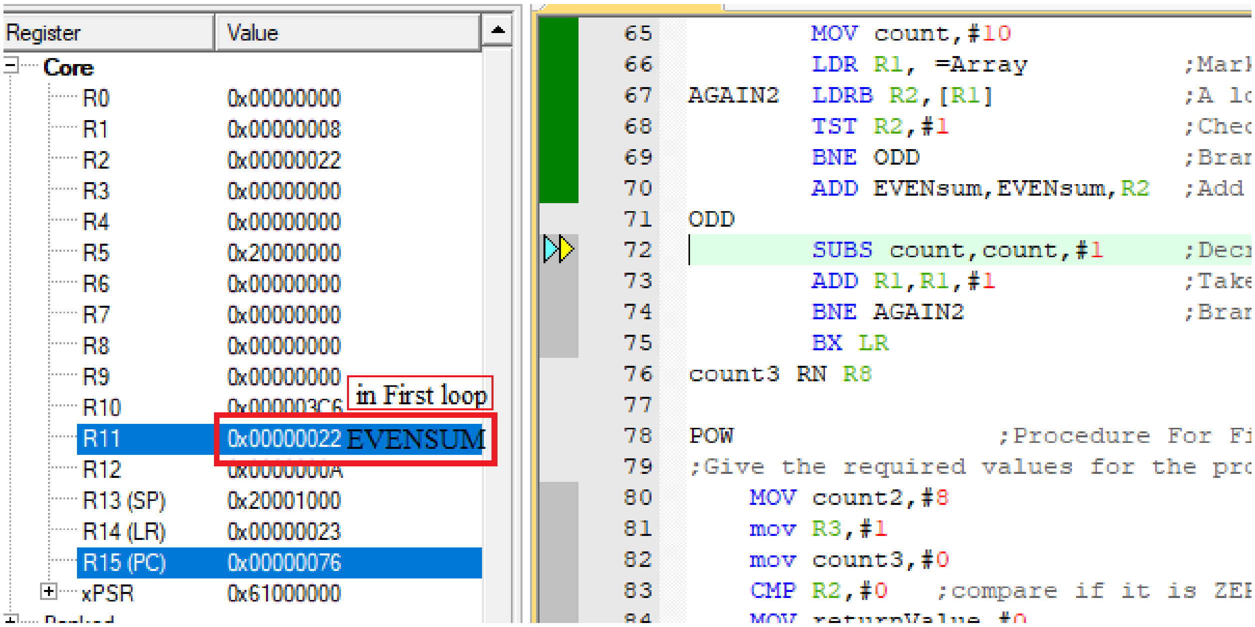
Command

```
Running with Code Size Limit: 32K
Load "D:\\ARM\\Objects\\Test1.axf"
```

Memory 2

Address: 0x20000000

```
0x20000000: C6 03 00 00 00 00 00 00 00
```

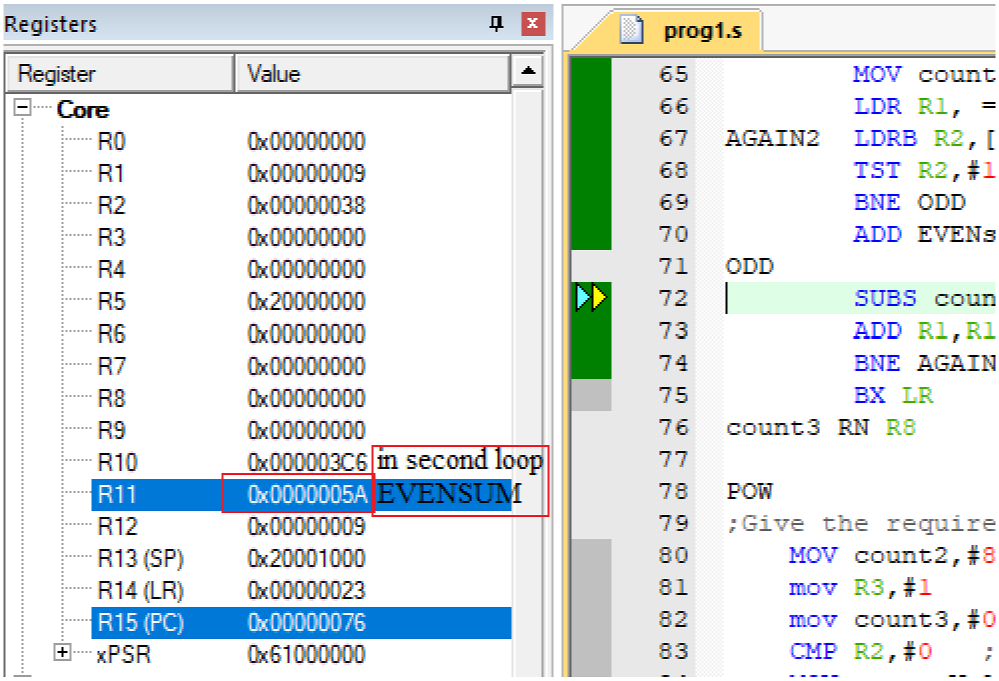


Register	Value
Core	
R0	0x00000000
R1	0x00000008
R2	0x00000022
R3	0x00000000
R4	0x00000000
R5	0x20000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6
R11	0x00000022
R12	0x0000000A
R13 (SP)	0x20001000
R14 (LR)	0x00000023
R15 (PC)	0x00000076
xPSR	0x61000000
Banked	

in First loop

EVENSUM

```
        MOV count,#10
        LDR R1, =Array              ;Mark
AGAIN2  LDRB R2,[R1]                ;A lo
        TST R2,#1                   ;Chec
        BNE ODD                     ;Bran
        ADD EVENsum,EVENsum,R2      ;Add
ODD
        SUBS count,count,#1         ;Decr
        ADD R1,R1,#1                ;Take
        BNE AGAIN2                  ;Bran
        BX LR
count3 RN R8

POW                     ;Procedure For Fi
;Give the required values for the pro
    MOV count2,#8
    mov R3,#1
    mov count3,#0
    CMP R2,#0     ;compare if it is ZEF
    MOV returnValue,#0
```



Registers

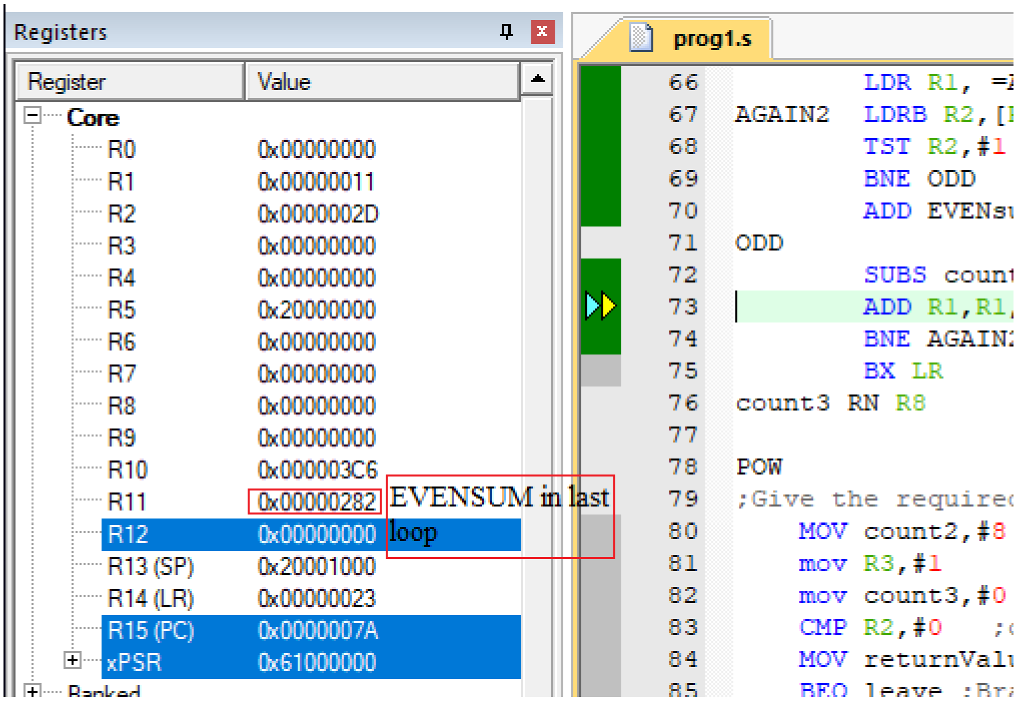
prog1.s

Register	Value
Core	
R0	0x00000000
R1	0x00000009
R2	0x00000038
R3	0x00000000
R4	0x00000000
R5	0x20000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6
R11	0x0000005A
R12	0x00000009
R13 (SP)	0x20001000
R14 (LR)	0x00000023
R15 (PC)	0x00000076
xPSR	0x61000000

in second loop

EVENSUM

```
        MOV count
        LDR R1, =
AGAIN2  LDRB R2,[
        TST R2,#1
        BNE ODD
        ADD EVENs
ODD
        SUBS coun
        ADD R1,R1
        BNE AGAIN
        BX LR
count3 RN R8

POW
;Give the require
    MOV count2,#8
    mov R3,#1
    mov count3,#0
    CMP R2,#0    ;
```



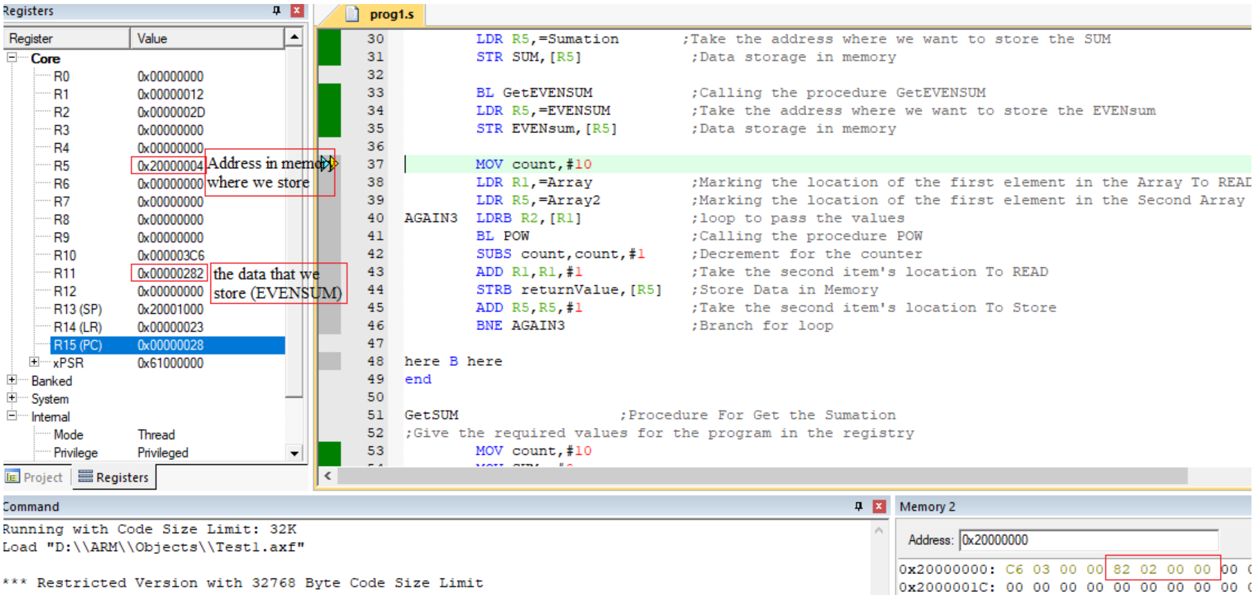
Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000011
R2	0x0000002D
R3	0x00000000
R4	0x00000000
R5	0x20000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6
R11	0x00000282
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0x00000023
R15 (PC)	0x0000007A
xPSR	0x61000000
Banked	

EVENSUM in last loop

prog1.s

```
        LDR R1, =
AGAIN2  LDRB R2,[
        TST R2,#1
        BNE ODD
        ADD EVENs
ODD
        SUBS count
        ADD R1,R1,
        BNE AGAIN
        BX LR
count3 RN R8

POW
;Give the require
    MOV count2,#8
    mov R3,#1
    mov count3,#0
    CMP R2,#0    ;
    MOV returnVal
    BEQ leave ;Br
```



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000012
R2	0x0000002D
R3	0x00000000
R4	0x00000000
R5	0x20000004
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x000003C6
R11	0x00000282
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0x00000023
R15 (PC)	0x00000028
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged

Address in memory where we store

the data that we store (EVENSUM)

prog1.s

```
        LDR R5,=Sumation        ;Take the address where we want to store the SUM
        STR SUM,[R5]             ;Data storage in memory

        BL GetEVENSUM            ;Calling the procedure GetEVENSUM
        LDR R5,=EVENSUM          ;Take the address where we want to store the EVENsum
        STR EVENsum,[R5]         ;Data storage in memory

        MOV count,#10
        LDR R1,=Array            ;Marking the location of the first element in the Array To READ
        LDR R5,=Array2           ;Marking the location of the first element in the Second Array
AGAIN3  LDRB R2,[R1]             ;loop to pass the values
        BL POW                   ;Calling the procedure POW
        SUBS count,count,#1      ;Decrement for the counter
        ADD R1,R1,#1             ;Take the second item's location To READ
        STRB returnValue,[R5]    ;Store Data in Memory
        ADD R5,R5,#1             ;Take the second item's location To Store
        BNE AGAIN3               ;Branch for loop

here B here
end

GetSUM                  ;Procedure For Get the Sumation
;Give the required values for the program in the registry
        MOV count,#10
```

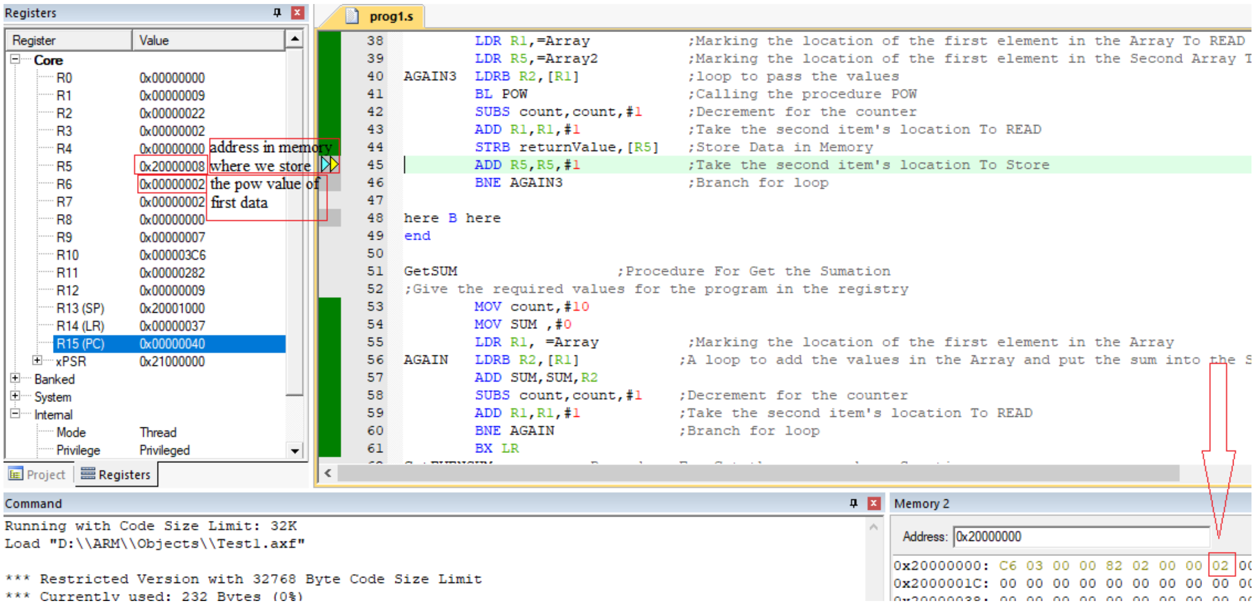
Command

```
Running with Code Size Limit: 32K
Load "D:\\ARM\\Objects\\Test1.axf"

*** Restricted Version with 32768 Byte Code Size Limit
```

Memory 2

Address: 0x20000000

```
0x20000000: C6 03 00 00 82 02 00 00 00
0x2000001C: 00 00 00 00 00 00 00 00 00
```



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000009
R2	0x00000022
R3	0x00000002
R4	0x00000000
R5	0x20000008
R6	0x00000002
R7	0x00000002
R8	0x00000000
R9	0x00000007
R10	0x000003C6
R11	0x00000282
R12	0x00000009
R13 (SP)	0x20001000
R14 (LR)	0x00000037
R15 (PC)	0x00000040
xPSR	0x21000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged

address in memory where we store

the pow value of first data

```
        LDR R1,=Array           ;Marking the location of the first element in the Array To READ
        LDR R5,=Array2          ;Marking the location of the first element in the Second Array T
AGAIN3  LDRB R2,[R1]            ;loop to pass the values
        BL POW                  ;Calling the procedure POW
        SUBS count,count,#1     ;Decrement for the counter
        ADD R1,R1,#1            ;Take the second item's location To READ
        STRB returnValue,[R5]   ;Store Data in Memory
        ADD R5,R5,#1            ;Take the second item's location To Store
        BNE AGAIN3              ;Branch for loop

here B here
end

GetSUM                  ;Procedure For Get the Sumation
;Give the required values for the program in the registry
        MOV count,#10
        MOV SUM ,#0
        LDR R1, =Array          ;Marking the location of the first element in the Array
AGAIN   LDRB R2,[R1]           ;A loop to add the values in the Array and put the sum into the S
        ADD SUM,SUM,R2
        SUBS count,count,#1    ;Decrement for the counter
        ADD R1,R1,#1           ;Take the second item's location To READ
        BNE AGAIN              ;Branch for loop
        BX LR
```

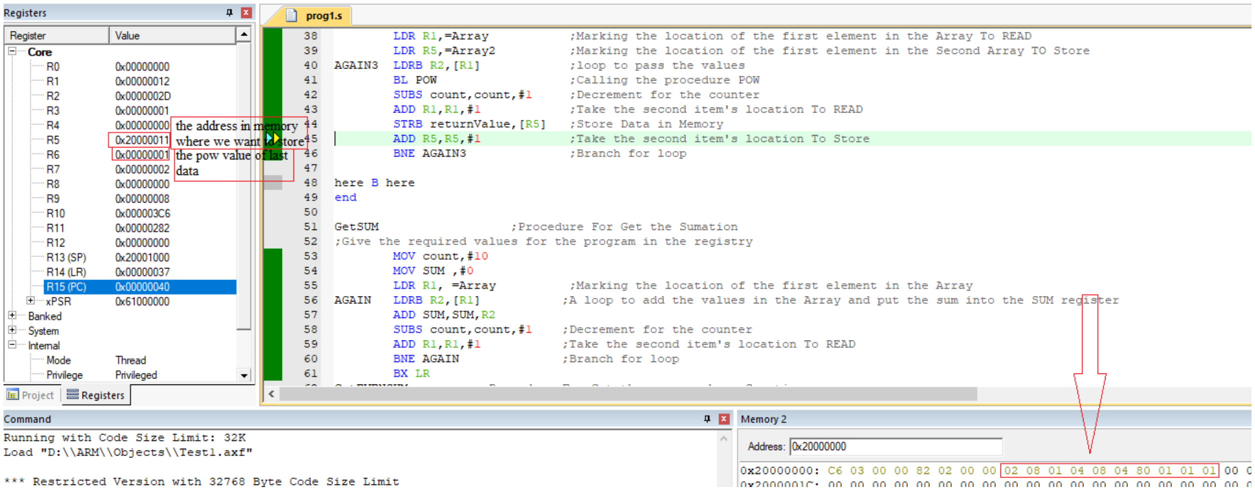
Command

```
Running with Code Size Limit: 32K
Load "D:\\ARM\\Objects\\Test1.axf"

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 232 Bytes (0%)
```

Memory 2

Address: 0x20000000

```
0x20000000: C6 03 00 00 82 02 00 00 02 00
0x2000001C: 00 00 00 00 00 00 00 00 00 00
```



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000012
R2	0x0000002D
R3	0x00000001
R4	0x00000000
R5	0x20000011
R6	0x00000001
R7	0x00000002
R8	0x00000000
R9	0x00000008
R10	0x000003C6
R11	0x00000282
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0x00000037
R15 (PC)	0x00000040
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged

the address in memory where we want to store

the pow value of last data

```
        LDR R1,=Array           ;Marking the location of the first element in the Array To READ
        LDR R5,=Array2          ;Marking the location of the first element in the Second Array TO Store
AGAIN3  LDRB R2,[R1]            ;loop to pass the values
        BL POW                  ;Calling the procedure POW
        SUBS count,count,#1     ;Decrement for the counter
        ADD R1,R1,#1            ;Take the second item's location To READ
        STRB returnValue,[R5]   ;Store Data in Memory
        ADD R5,R5,#1            ;Take the second item's location To Store
        BNE AGAIN3              ;Branch for loop

here B here
end

GetSUM                  ;Procedure For Get the Sumation
;Give the required values for the program in the registry
        MOV count,#10
        MOV SUM ,#0
        LDR R1, =Array          ;Marking the location of the first element in the Array
AGAIN   LDRB R2,[R1]           ;A loop to add the values in the Array and put the sum into the SUM register
        ADD SUM,SUM,R2
        SUBS count,count,#1    ;Decrement for the counter
        ADD R1,R1,#1           ;Take the second item's location To READ
        BNE AGAIN              ;Branch for loop
        BX LR
```

Command

```
Running with Code Size Limit: 32K
Load "D:\\ARM\\Objects\\Test1.axf"

*** Restricted Version with 32768 Byte Code Size Limit
```

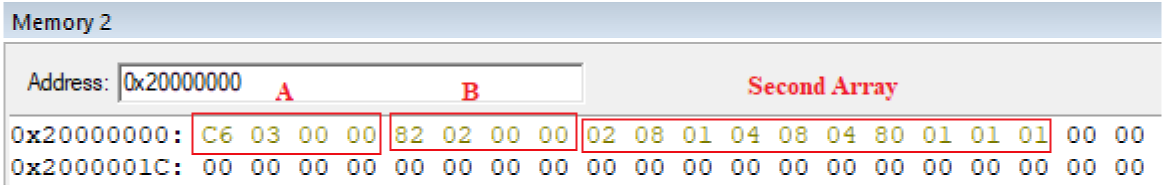
Memory 2

Address: 0x20000000

```
0x20000000: C6 03 00 00 82 02 00 00 02 08 01 04 08 04 80 01 01 01 00 0
0x2000001C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
```

Input Array : 34,56,27,156,200,68,128,235,17,45

Output:

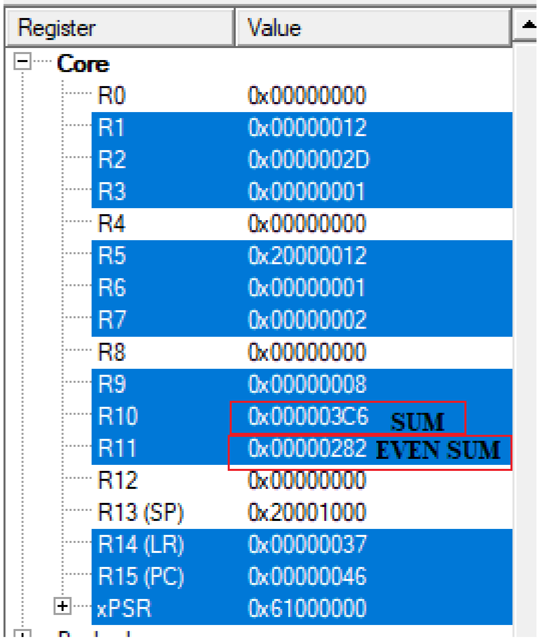


Memory 2

Address: 0x20000000

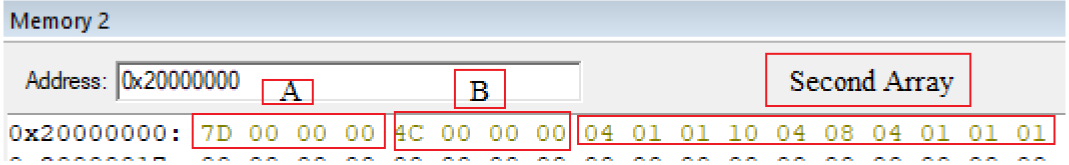
```
0x20000000: C6 03 00 00 82 02 00 00 02 08 01 04 08 04 80 01 01 01 00 00
0x2000001C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

1- A (SUM) = 0x3C6
2- B (EVENsum) =0x282
3- Second Array = (34) → 0x2,(56) → 0x8,(27) → 0x1,(156) → 0x4,(200) → 0x8,(68) → 0x4, (128) → 0x8,(235) → 0x1,(17) → 0x1,(45) → 0x1



Register	Value
Core	
R0	0x00000000
R1	0x00000012
R2	0x0000002D
R3	0x00000001
R4	0x00000000
R5	0x20000012
R6	0x00000001
R7	0x00000002
R8	0x00000000
R9	0x00000008
R10	0x000003C6
R11	0x00000282
R12	0x00000000
R13 (SP)	0x20001000
R14 (LR)	0x00000037
R15 (PC)	0x00000046
xPSR	0x61000000

Input Array : 4,5,7,16,20,8,28,25,7,5



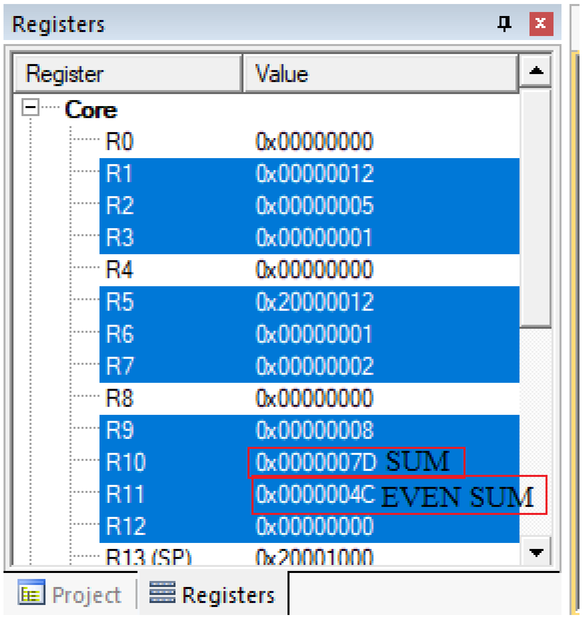
Memory 2

Address: 0x20000000

```
0x20000000: 7D 00 00 00 4C 00 00 00 04 01 01 10 04 08 04 01 01 01
```

1- A (SUM) = 0x7D
2- B (EVENsum) =0x4C
3- Second Array = (4) → 0x4,(5) → 0x1,(7) → 0x1,(16) → 0x10,(20) → 0x4,(8) → 0x8,

(28) → 0x4,(25) → 0x1,(7) → 0x1,(5) → 0x1



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000012
R2	0x00000005
R3	0x00000001
R4	0x00000000
R5	0x20000012
R6	0x00000001
R7	0x00000002
R8	0x00000000
R9	0x00000008
R10	0x0000007D SUM
R11	0x0000004C EVEN SUM
R12	0x00000000
R13 (SP)	0x20001000