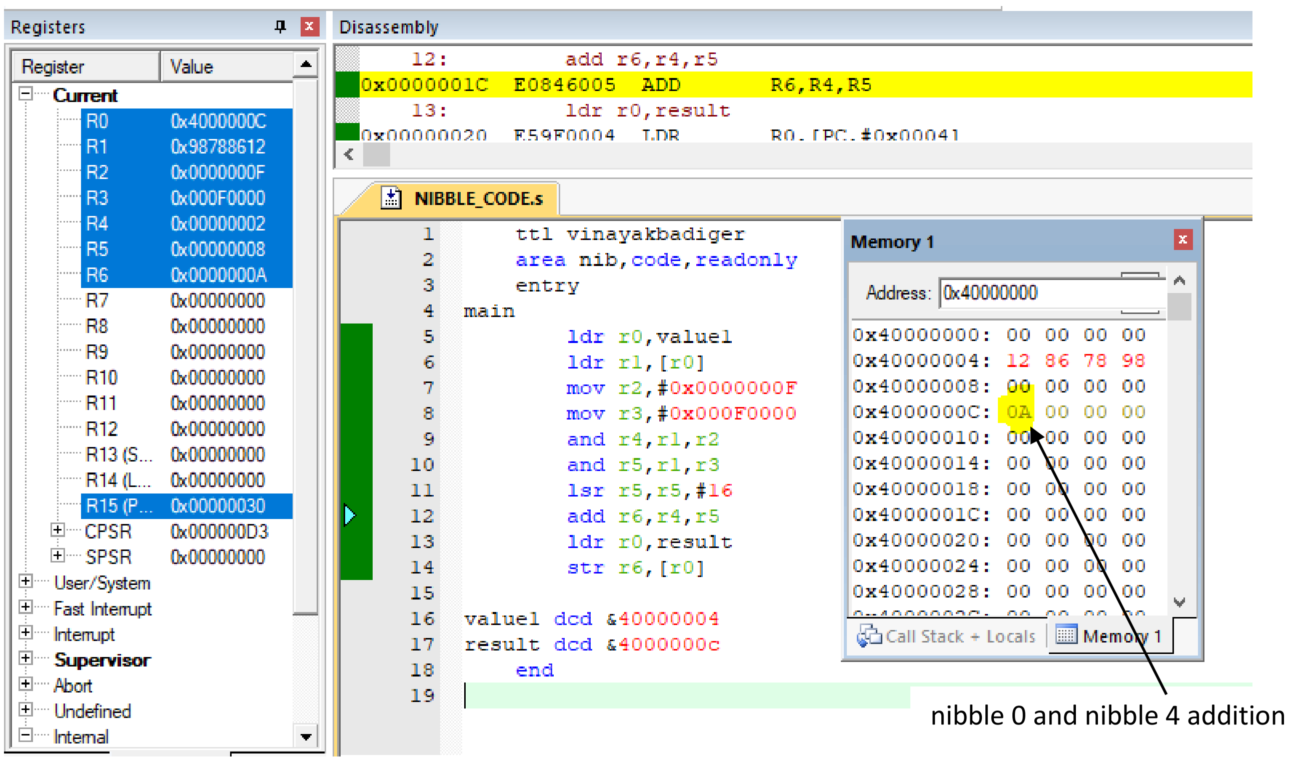
Name – Vinayak Badiger

1. **Assume a 32bit number in 40000004H add nibble 4 and nibble 0, and store the result in 4000000CH**

```
            area nib,code,readonly
   entry
main
                        ldr r0,value1
       ldr r1,[r0] ;address of r0 to r1
       mov r2,#0x0000000F ; move this value into r2
       mov r3,#0x000F0000 ; move this value into r3
       and r4,r1,r2   ; Masking of r1 bits with r2 value(anding) keeps
       and r5,r1,r3   ; Masking of r1 bits with r3 value(anding)
       lsr r5,r5,#16   ; logical right shift by 16bits as nibble 4 is on 17th bit
       add r6,r4,r5  ; adding of shifted bits
       ldr r0,result ; storing result address in r0
       str r6,[r0]  ; storing final nibble addition result in r6.

value1 dcd &40000004 ;define variable and assign address
result dcd &4000000c ; define variable and assign address
   end
```



```
Register        Value
Current
  R0            0x4000000C
  R1            0x98788612
  R2            0x0000000F
  R3            0x000F0000
  R4            0x00000002
  R5            0x00000008
  R6            0x0000000A
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (S...     0x00000000
  R14 (L...     0x00000000
  R15 (P...     0x00000030
  CPSR          0x000000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
```

```
    12:            add r6,r4,r5
0x0000001C  E0846005  ADD       R6,R4,R5
    13:            ldr r0,result
0x00000020  E59F0004  LDR       R0,[PC,#0x0004]
```

```
      ttl vinayakbadiger
      area nib,code,readonly
      entry
main
            ldr r0,value1
            ldr r1,[r0]
            mov r2,#0x0000000F
            mov r3,#0x000F0000
            and r4,r1,r2
            and r5,r1,r3
            lsr r5,r5,#16
            add r6,r4,r5
            ldr r0,result
            str r6,[r0]

value1 dcd &40000004
result dcd &4000000c
      end
```

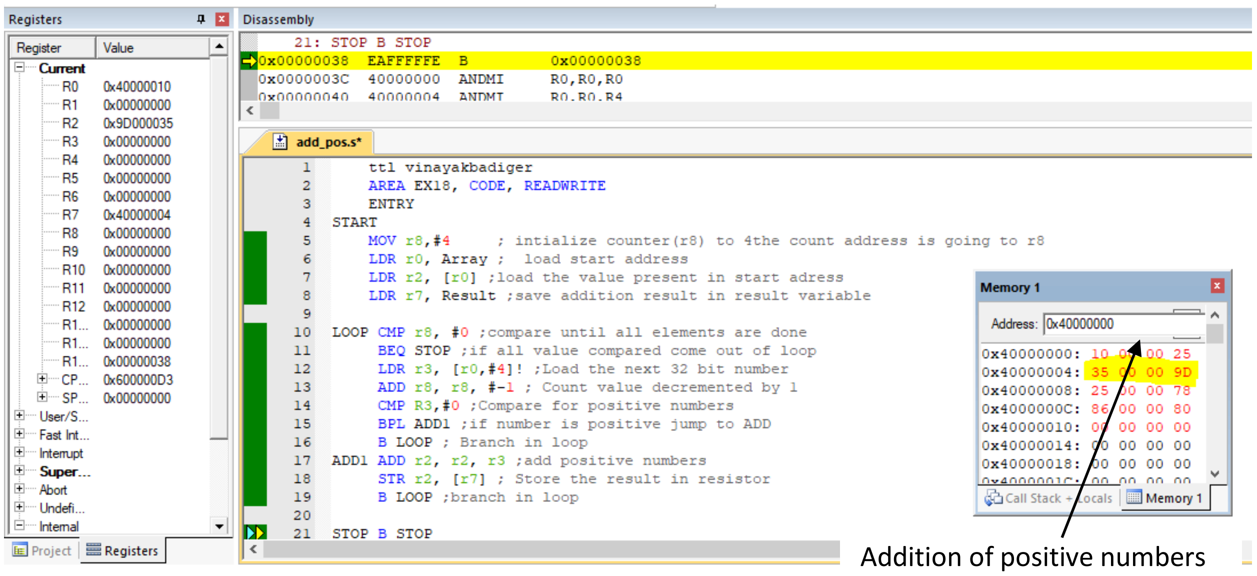
```
Address: 0x40000000
0x40000000:  00  00  00  00
0x40000004:  12  86  78  98
0x40000008:  00  00  00  00
0x4000000C:  0A  00  00  00
0x40000010:  00  00  00  00
0x40000014:  00  00  00  00
0x40000018:  00  00  00  00
0x4000001C:  00  00  00  00
0x40000020:  00  00  00  00
0x40000024:  00  00  00  00
0x40000028:  00  00  00  00
```

2. Consider array of numbers present from 40000004h. add only if numbers are positive 40000000h has the count of the array

```
                    ttl vinayakbadiger
          AREA EX18, CODE, READWRITE
          ENTRY
START
          MOV r8,#4          ; intialize counter(r8) to 4the count address is going to r8
          LDR r0, Array ;  load start address
          LDR r2, [r0] ;load the value present in start adress
          LDR r7, Result ;save addition result in result variable

LOOP CMP r8, #0 ;compare until all elements are done
           BEQ STOP ;if all value compared come out of loop
           LDR r3, [r0,#4]! ;Load the next 32 bit number
           ADD r8, r8, #-1 ; Count value decremented by 1
           CMP R3,#0 ;Compare for positive numbers
           BPL ADD1 ;if number is positive jump to ADD
           B LOOP ; Branch in loop
ADD1 ADD r2, r2, r3 ;add positive numbers
           STR r2, [r7] ; Store the result in resistor
           B LOOP ;branch in loop

STOP B STOP
Array  DCD &40000000
Result DCD &40000004
          END
```



Registers

Register	Value
Current	
R0	0x40000010
R1	0x00000000
R2	0x9D000035
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x40000004
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R1...	0x00000000
R1...	0x00000000
R1...	0x00000038
CP...	0x600000D3
SP...	0x00000000
User/S...	
Fast Int...	
Interrupt	
Super...	
Abort	
Undefi...	
Internal	

Disassembly

```
    21: STOP B STOP
0x00000038  EAFFFFFE  B         0x00000038
0x0000003C  40000000  ANDMI     R0,R0,R0
0x00000040  40000004  ANDMI     R0,R0,R4
```

add_pos.s*

```
     ttl vinayakbadiger
     AREA EX18, CODE, READWRITE
     ENTRY
START
     MOV r8,#4      ; intialize counter(r8) to 4the count address is going to r8
     LDR r0, Array ;  load start address
     LDR r2, [r0] ;load the value present in start adress
     LDR r7, Result ;save addition result in result variable

LOOP CMP r8, #0 ;compare until all elements are done
      BEQ STOP ;if all value compared come out of loop
      LDR r3, [r0,#4]! ;Load the next 32 bit number
      ADD r8, r8, #-1 ; Count value decremented by 1
      CMP R3,#0 ;Compare for positive numbers
      BPL ADD1 ;if number is positive jump to ADD
      B LOOP ; Branch in loop
ADD1 ADD r2, r2, r3 ;add positive numbers
      STR r2, [r7] ; Store the result in resistor
      B LOOP ;branch in loop

STOP B STOP
```

Memory 1

Address: 0x40000000

```
0x40000000: 10 00 00 25
0x40000004: 35 00 00 9D
0x40000008: 25 00 00 78
0x4000000C: 86 00 00 80
0x40000010: 00 00 00 00
0x40000014: 00 00 00 00
0x40000018: 00 00 00 00
0x4000001C: 00 00 00 00
```

Addition of positive numbers saved in 04 address