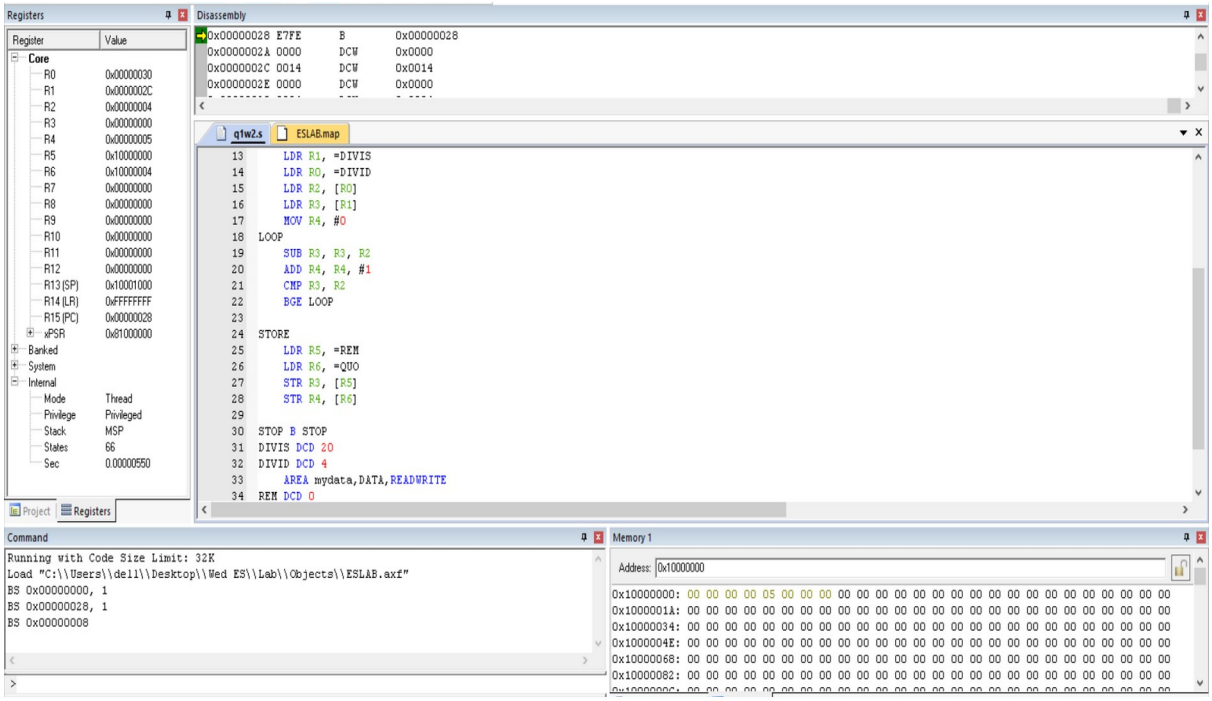
1.Write an assembly language program to implement division by repetitive subtraction.

```
AREA RESET,DATA,READONLY
EXPORT __Vectors

__Vectors
DCD 0X10001000
DCD Reset_Handler
AREA mycode,CODE,READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R1, =DIVIS
LDR R0, =DIVID
LDR R2, [R0]
LDR R3, [R1]
MOV R4, #0
LOOP
SUB R3, R3, R2
ADD R4, R4, #1
CMP R3, R2
BGE LOOP

STORE
LDR R5, =REM
LDR R6, =QUO
STR R3, [R5]
STR R4, [R6]

STOP B STOP
DIVIS DCD 20
DIVID DCD 4
```

```
AREA mydata,DATA,READWRITE
REM DCD 0
QUO DCD 0
END
```

OUTPUT :

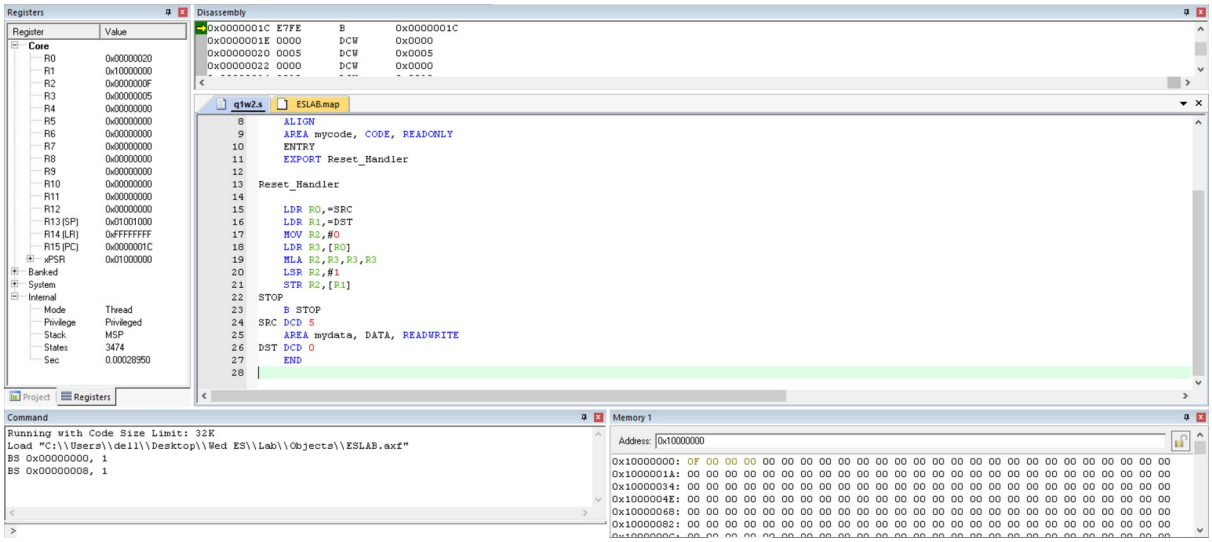


2.Find the sum of ‘n’ natural numbers using MLA instruction.

```
AREA RESET, DATA, READONLY
EXPORT __Vectors

__Vectors
DCD 0x1001000
DCD Reset_Handler
ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0,=SRC
LDR R1,=DST
MOV R2,#0
LDR R3,[R0]
```

```
MLA R2,R3,R3,R3
LSR R2,#1
STR R2,[R1]
STOP
B STOP
SRC DCD 5 ;n
AREA mydata, DATA, READWRITE
DST DCD 0
END
```

OUTPUT :

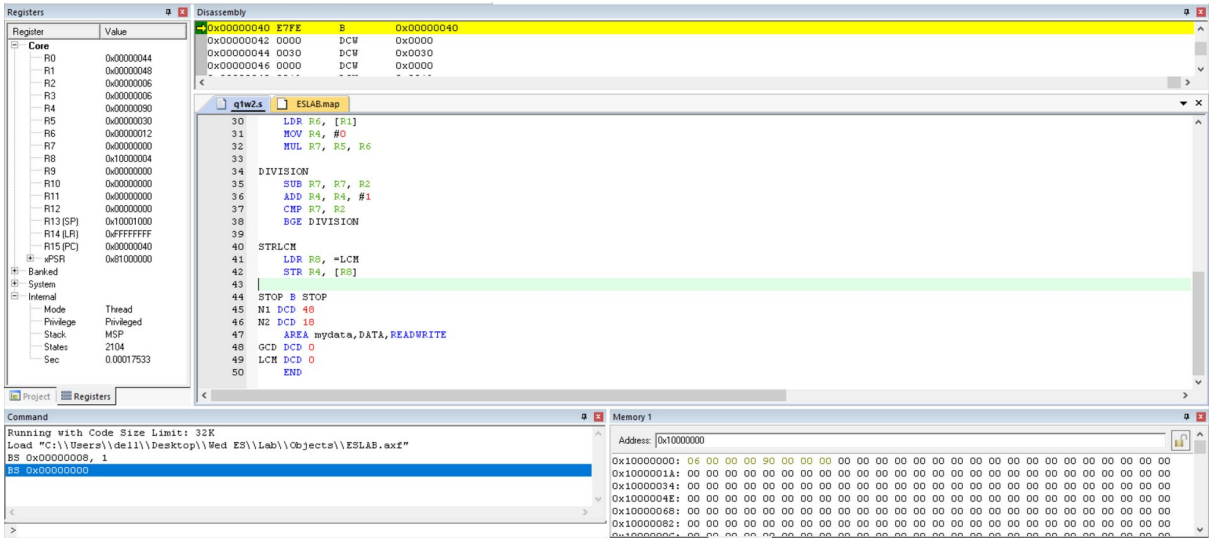


3.Write an assembly language program to find GCD and LCM of two 8 bit numbers.

```
AREA RESET, DATA, READONLY
EXPORT __Vectors

__Vectors
DCD 0X10001000
DCD Reset_Handler
AREA mycode,CODE,READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
```

```
LDR R0, =N1
LDR R1, =N2
LDR R2, [R0]
LDR R3, [R1]
GCDT
CMP R2, R3
SUBLT R3, R3, R2
SUBGT R2, R2, R3
BNE GCDT

STRGCD
LDR R4, =GCD
STR R2, [R4]
LCMT
LDR R5, [R0]
LDR R6, [R1]
MOV R4, #0
MUL R7, R5, R6

DIVISION
SUB R7, R7, R2
ADD R4, R4, #1
CMP R7, R2
BGE DIVISION

STRLCM
LDR R8, =LCM
STR R4, [R8]

STOP B STOP
N1 DCD 48
N2 DCD 18
AREA mydata,DATA,READWRITE
GCD DCD 0
LCM DCD 0
END
```

OUTPUT :

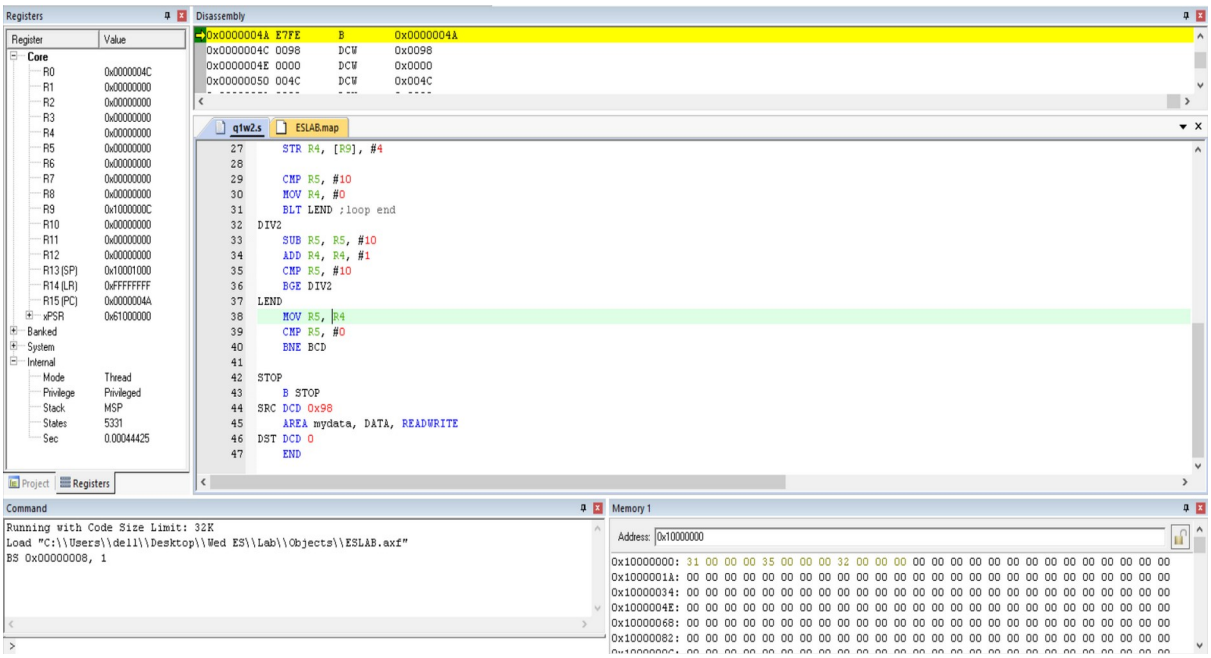


4. Write an ARM assembly language program to convert 2-digit hexadecimal number into ascii format.

```
AREA RESET,DATA,READONLY
EXPORT __Vectors
__Vectors
DCD 0X10001000
DCD Reset_Handler
ALIGN
AREA mycode,CODE,READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0, =SRC
LDR R1, [R0]
LDR R9, =DST
MOV R5, #100
BCD
CMP R1, R5
BLT DIV2
MOV R4, #0
DIV1
SUB R1, R1, R5
```

```
ADD R4, R4, #1
CMP R1, R5
BGE DIV1
ADD R4, R4, #48
STR R4, [R9], #4
CMP R5, #10
MOV R4, #0
BLT LEND ;loop end
DIV2
SUB R5, R5, #10
ADD R4, R4, #1
CMP R5, #10
BGE DIV2
LEND
MOV R5, R4
CMP R5, #0
BNE BCD
STOP
B STOP
SRC DCD 0x98
AREA mydata, DATA, READWRITE
DST DCD 0
END
```

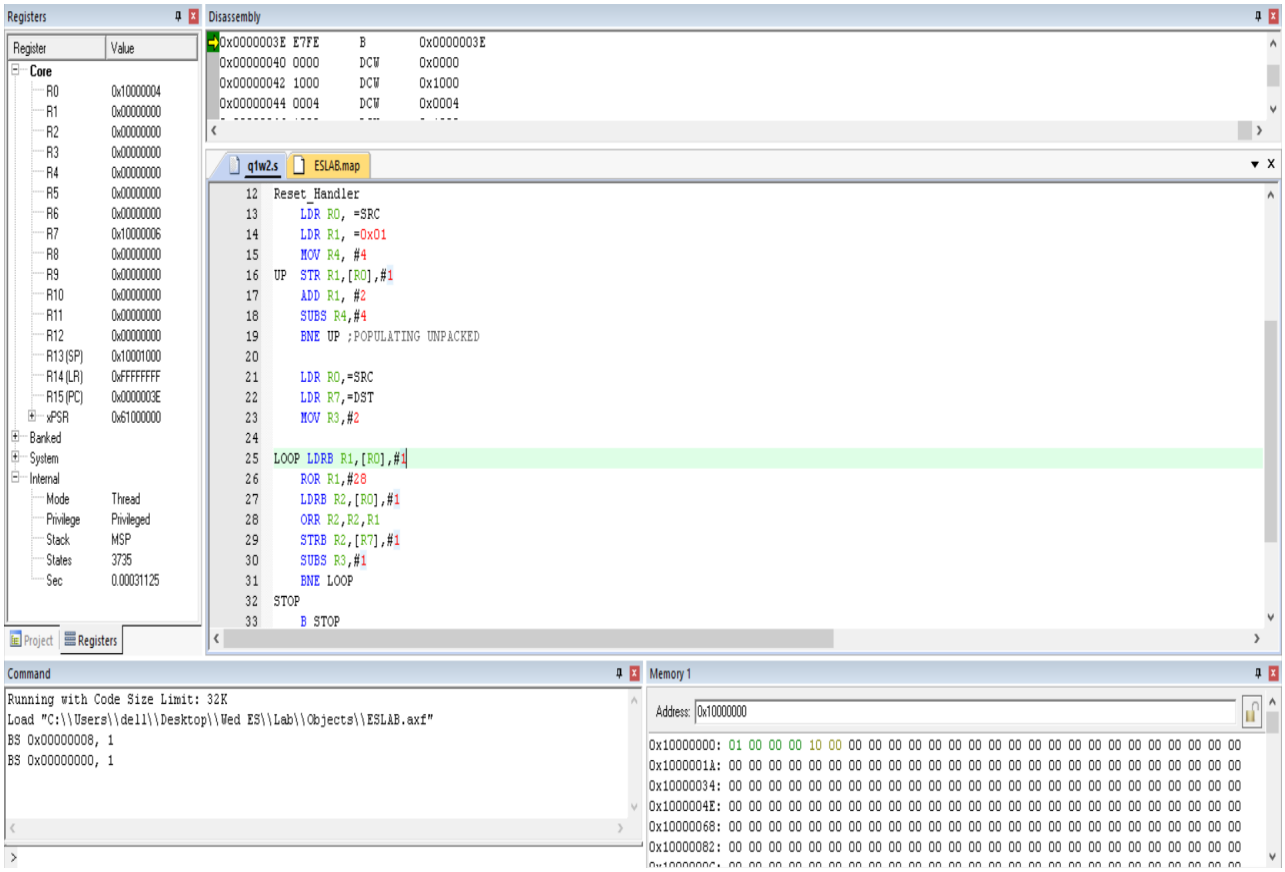
OUTPUT :



5. Write an ARM assembly language program to convert a 32-bit BCD number in the unpacked form into packed form.

```
AREA RESET,DATA,READONLY
EXPORT __Vectors
__Vectors
DCD 0X10001000
DCD Reset_Handler
ALIGN
AREA mycode,CODE,READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0, =SRC
LDR R1, =0x01
MOV R4, #4
UP STR R1,[R0],#1
ADD R1, #2
SUBS R4,#4
BNE UP ;POPULATING UNPACKED
LDR R0,=SRC
LDR R7,=DST
MOV R3,#2
LOOP LDRB R1,[R0],#1
ROR R1,#28
LDRB R2,[R0],#1
ORR R2,R2,R1
STRB R2,[R7],#1
SUBS R3,#1
BNE LOOP
STOP
B STOP
AREA mydata, DATA, READWRITE
SRC DCD 0
DST DCD 0
END
```

OUTPUT :


Registers
Register
Value
Core
R0 0x10000004
R1 0x00000000
R2 0x00000000
R3 0x00000000
R4 0x00000000
R5 0x00000000
R6 0x00000000
R7 0x10000006
R8 0x00000000
R9 0x00000000
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x10001000
R14 (LR) 0xFFFFFFFF
R15 (PC) 0x0000003E
xPSR 0x61000000
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 3735
Sec 0.00031125
Project
Registers
Disassembly
0x0000003E E7FE B 0x0000003E
0x00000040 0000 DCW 0x0000
0x00000042 1000 DCW 0x1000
0x00000044 0004 DCW 0x0004
q1w2.s
ESLAB.map
12 Reset_Handler
13 LDR R0, =SRC
14 LDR R1, =0x01
15 MOV R4, #4
16 UP STR R1,[R0],#1
17 ADD R1, #2
18 SUBS R4,#4
19 BNE UP ;POPULATING UNPACKED
20
21 LDR R0,=SRC
22 LDR R7,=DST
23 MOV R3,#2
24
25 LOOP LDRB R1,[R0],#1
26 ROR R1,#28
27 LDRB R2,[R0],#1
28 ORR R2,R2,R1
29 STRB R2,[R7],#1
30 SUBS R3,#1
31 BNE LOOP
32 STOP
33 B STOP
Command
Running with Code Size Limit: 32K
Load "C:\\Users\\dell\\Desktop\\Wed ES\\Lab\\Objects\\ESLAB.axf"
BS 0x00000008, 1
BS 0x00000000, 1
Memory 1
Address: 0x10000000
0x10000000: 01 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000001A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000034: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1000004E: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000068: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10000082: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00