LAB 4 :

Solved Exercise:

Write an assembly language program to unpack a 32-bit BCD number into eight 32-bit ASCII numbers.

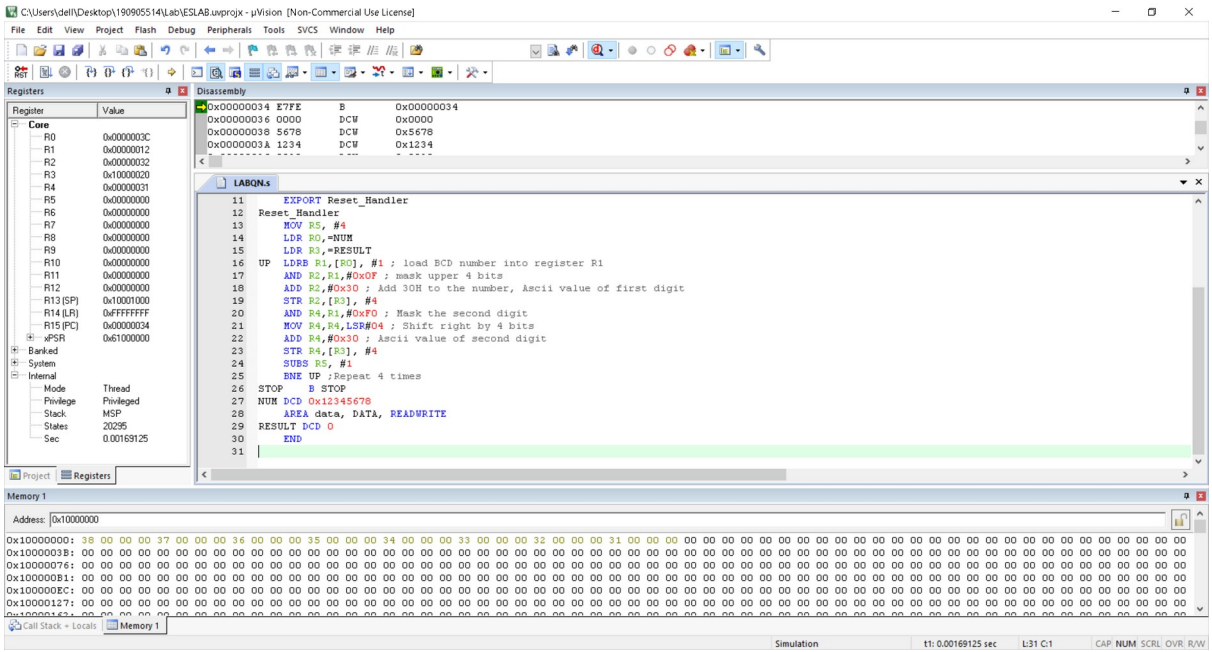
```
AREA RESET, DATA, READONLY
EXPORT __Vectors

__Vectors

DCD 0x10001000 ; stack pointer value when stack is empty
DCD Reset_Handler ; reset vector
ALIGN
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
MOV R5, #4
LDR R0,=NUM
LDR R3,=RESULT
UP LDRB R1,[R0], #1 ; load BCD number into register R1
AND R2,R1,#0x0F ; mask upper 4 bits
ADD R2,#0x30 ; Add 30H to the number, Ascii value of
first digit
STR R2,[R3], #4
AND R4,R1,#0xF0 ; Mask the second digit
MOV R4,R4,LSR#04 ; Shift right by 4 bits
ADD R4,#0x30 ; Ascii value of second digit
STR R4,[R3], #4
SUBS R5, #1
```

```
BNE UP ;Repeat 4 times
STOP B STOP
NUM DCD 0x12345678
AREA data, DATA, READWRITE
RESULT DCD 0
END
```

Output:



Lab Exercises:

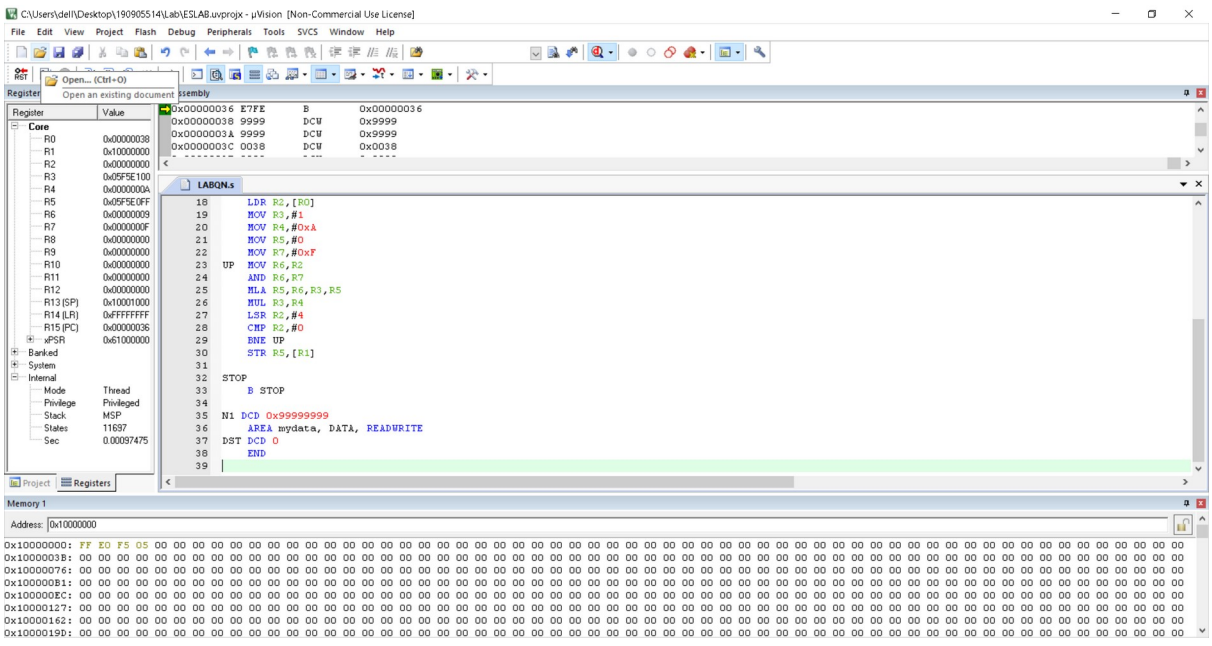
1. Convert a 32-bit packed BCD number into its equivalent hexadecimal number.

```
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,=N1
LDR R1,=DST
LDR R2,[R0]
MOV R3,#1
MOV R4,#0xA
MOV R5,#0
MOV R7,#0xF
UP MOV R6,R2
AND R6,R7
MLA R5,R6,R3,R5
MUL R3,R4
LSR R2,#4
CMP R2,#0
BNE UP
STR R5,[R1]

STOP
B STOP
N1 DCD 0x99999999
AREA mydata, DATA, READWRITE
DST DCD 0
END
```

Output:

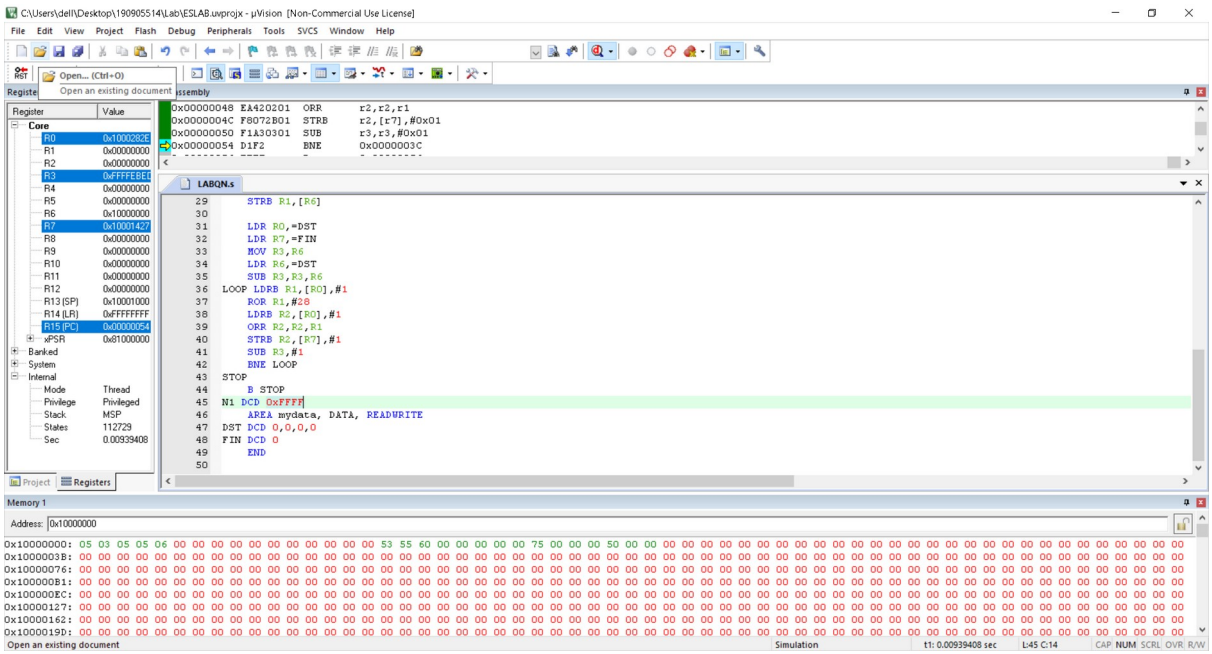


2. Convert a 16-bit hex number into its equivalent packed BCD.

```
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,=N1
MOV R2,#00
LDR R6,=DST
LDR R1,[R0]
UP CMP R1,#0xA
BCC STORE
SUB R1,#0xA
ADD R2,#0X1
B UP
STORE STRB R1,[R6],#1
MOV R1,R2
MOV R2,#0
CMP R1,#0xA
BCS UP
STRB R1,[R6]
LDR R0,=DST
LDR R7,=FIN
MOV R3,R6
LDR R6,=DST
SUB R3,R3,R6
LOOP LDRB R1,[R0],#1
```

```
ROR R1,#28
LDRB R2,[R0],#1
ORR R2,R2,R1
STRB R2,[R7],#1
SUB R3,#1
BNE LOOP
STOP
B STOP
N1 DCD 0xFFFF
AREA mydata, DATA, READWRITE
DST DCD 0,0,0,0
FIN DCD 0
END
```

Output:



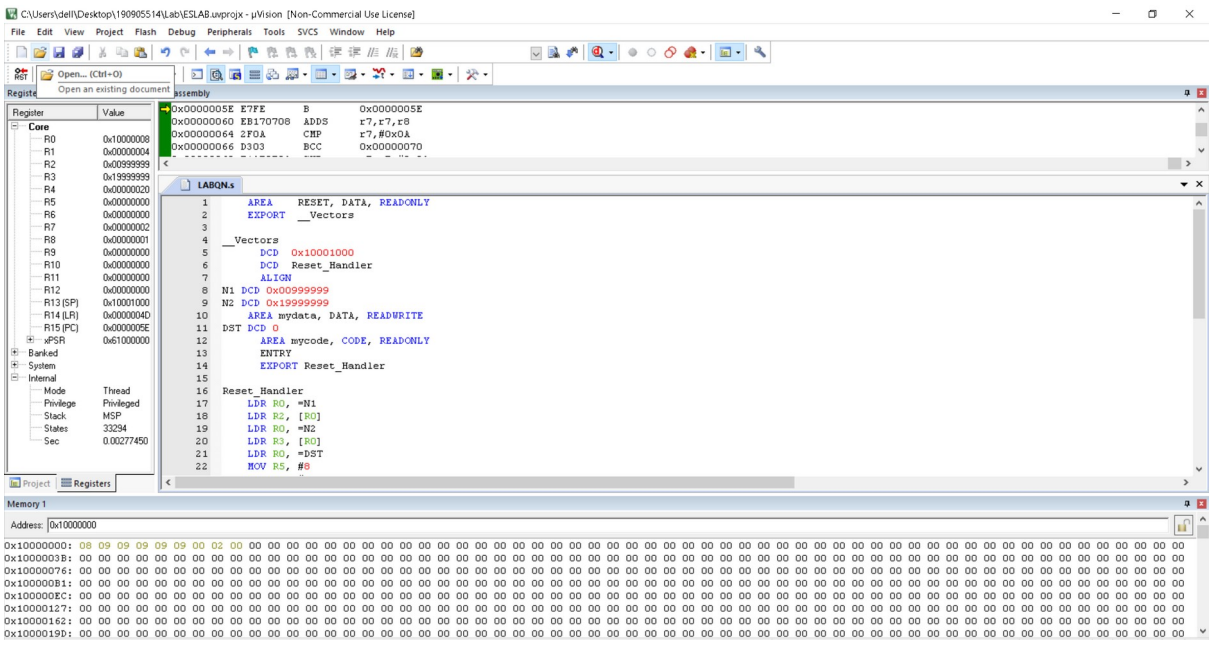
3. Add two 32-bit packed BCD numbers and store the result in packed BCD form.

```
AREA RESET, DATA, READONLY
EXPORT __Vectors
```

```
__Vectors
DCD 0x10001000
DCD Reset_Handler
ALIGN
N1 DCD 0x00999999
N2 DCD 0x19999999
AREA mydata, DATA, READWRITE
DST DCD 0
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0, =N1
LDR R2, [R0]
LDR R0, =N2
LDR R3, [R0]
LDR R0, =DST
MOV R5, #8
MOV R6, #0
MOV R9, #0xF
MOV R4, #0
UP MOV R1, #0
MOV R7, R2
MOV R8, R3
AND R7, R9
LSR R7, R4
ADD R7, R6
MOV R6, #0
AND R8, R9
LSR R8, R4
BL ADDN
ADD R1, #4
ADD R4, #4
LSL R9, R1
SUBS R5, #1
BNE UP
STRB R6, [R0]
```

```
STOP
B STOP
ADDN ADDS R7, R8
CMP R7, #0xA
BCC STORE
SUB R7, #0xA
ADD R6, #01
STORE STRB R7, [R0], #1
BX LR
END
```

Output:



4. Multiply two 16-bit packed BCD and store the result in packed BCD form.

```
AREA RESET, DATA, READONLY
EXPORT __Vectors

__Vectors
DCD 0x10001000
DCD Reset_Handler
ALIGN
```

```
N1 DCD 0x9999
N2 DCD 0x9999
AREA mydata, DATA, READWRITE
PRODUCT DCD 0,0
TEMP DCD 0
AREA mycode, CODE, READONLY
ENTRY
EXPORT Reset_Handler
Reset_Handler
LDR R0, =N1
LDR R2, [R0]
BL BCD_HEX
MOV R9, R5
LDR R0, =N2
LDR R2, [R0]
BL BCD_HEX
MOV R10, R5
LDR R0, =PRODUCT
MUL R9, R10
BL HEX_BCD
LDR R9, =TEMP
UP2 LDR R12, [R9], #1
LDR R11, [R9], #1
LSL R11, #4
ORR R12, R11
STRB R12, [R0], #1
SUBS R1, #1
BNE UP2
STOP
B STOP
BCD_HEX MOV R3, #1
MOV R4, #0xA
MOV R5, #0
MOV R7, #0xF
UP MOV R6, R2
AND R6, R7
MLA R5, R6, R3, R5
```

```
MUL R3, R4
LSR R2, #4
CMP R2, #0
BNE UP
BX LR
HEX_BCD
MOV R8, #0
LDR R1, =TEMP
UP1 CMP R9, #0xA
BCC STORE
SUB R9, #0xA
ADD R8, #01
B UP1
STORE
STRB R9, [R1], #1
MOV R9, R8
MOV R8, #0
CMP R9, #0xA
BCS UP1
STRB R9, [R1]
LDR R8, =TEMP
SUB R1, R8
BX LR
END
```

Output: