EC-210

MICROPROCESSORS LAB

LAB-6

UTKARSH MAHAJAN 201EC164

ARNAV RAJ 201EC109

Objective: To understand and use the multiplication instructions effectively.

Exercise:

6.2] Write an assembly program to perform multiplication c = a * b where

(a) a and b are both unsigned 32 bit numbers.

->

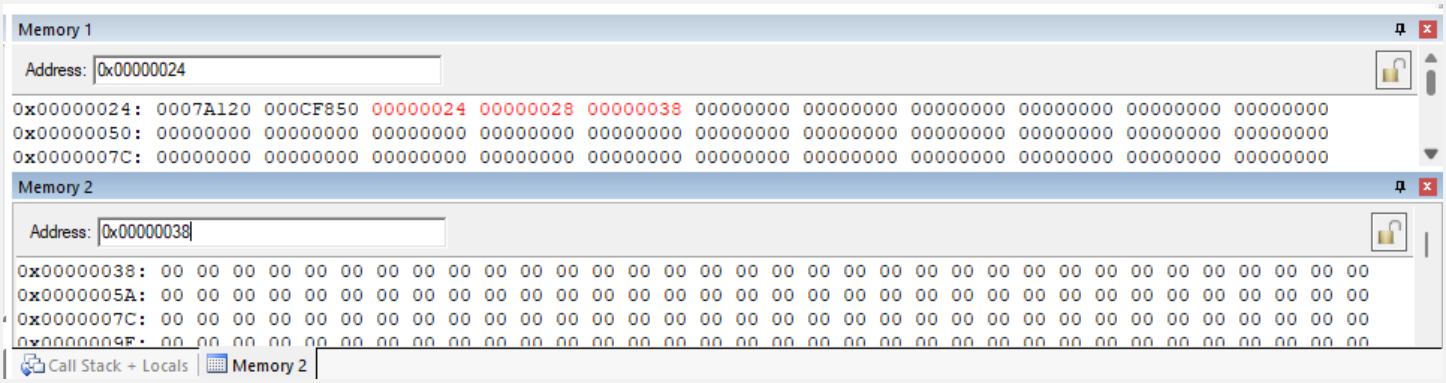
Source Code:

```
    AREA AllocSpace, DATA,NOINIT,READWRITE
    ;Space for storing result
    ;Utkarsh && Arnav Lab6
c SPACE 8;
;Actual Code
    AREA hmm, CODE, READWRITE
    EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R1, =a;
    LDR R2, =bi;
    ;address to the result
    LDR R3, =c ;
    LDR R4, [R1]; load a
    LDR R5, [R2]; load b
    UMULL R6, R7, R5, R4; unsigned mult
    STR R6, [R3]; storing low
    STR R7, [R3, #4]; storing high
stop BAL stop
; input_number
a   DCD 0x7A120;
bi  DCD 0xCF850;
    END
```

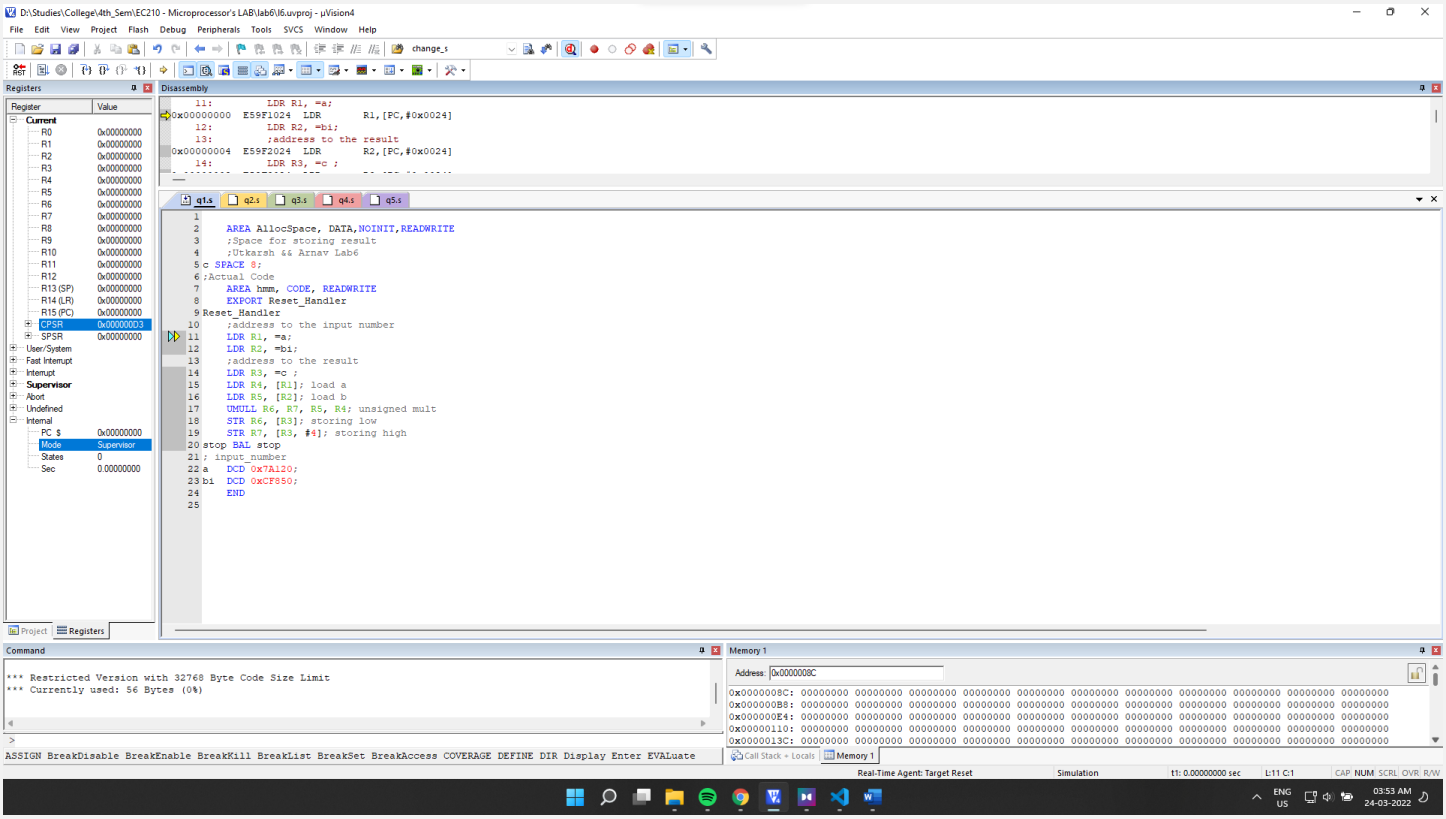
Debugging:

Initial Memory: (after getting the address through register)

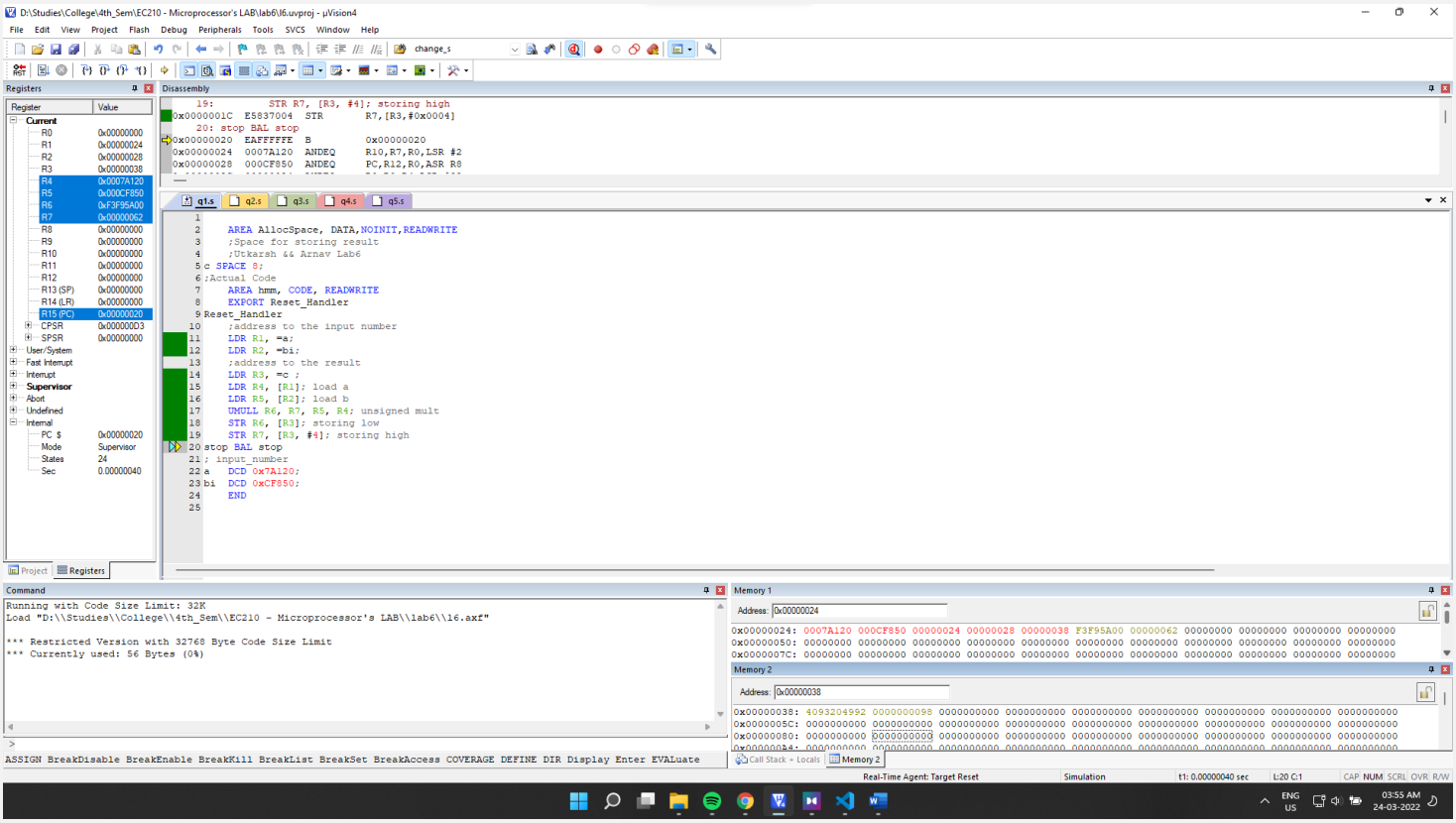
Memory 1 shows input and memory 2 will store output



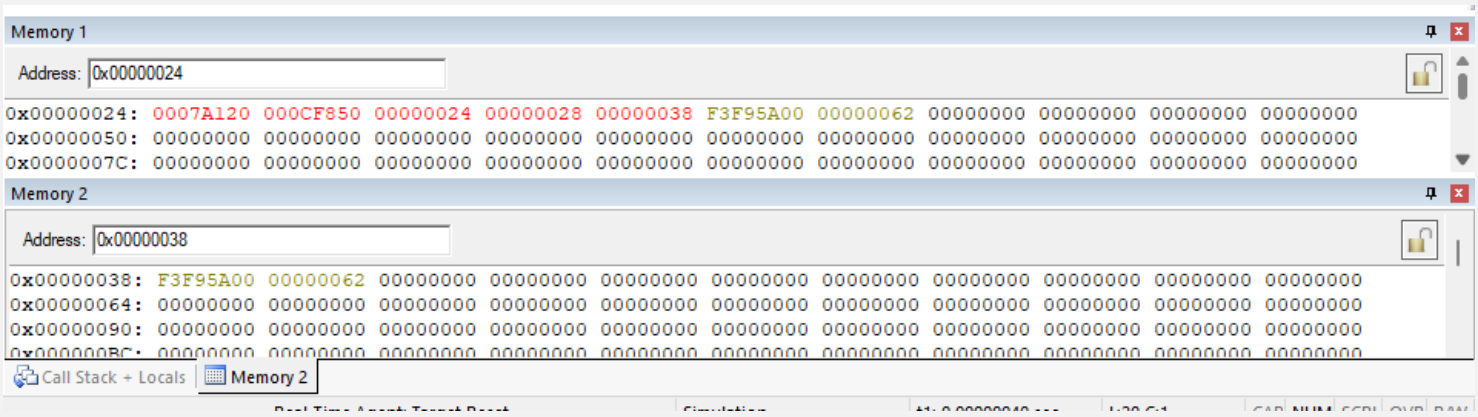
Setup:



Final Output:



Final Memory:



(b) a and b are both signed 32 bit numbers.

->

Source Code:

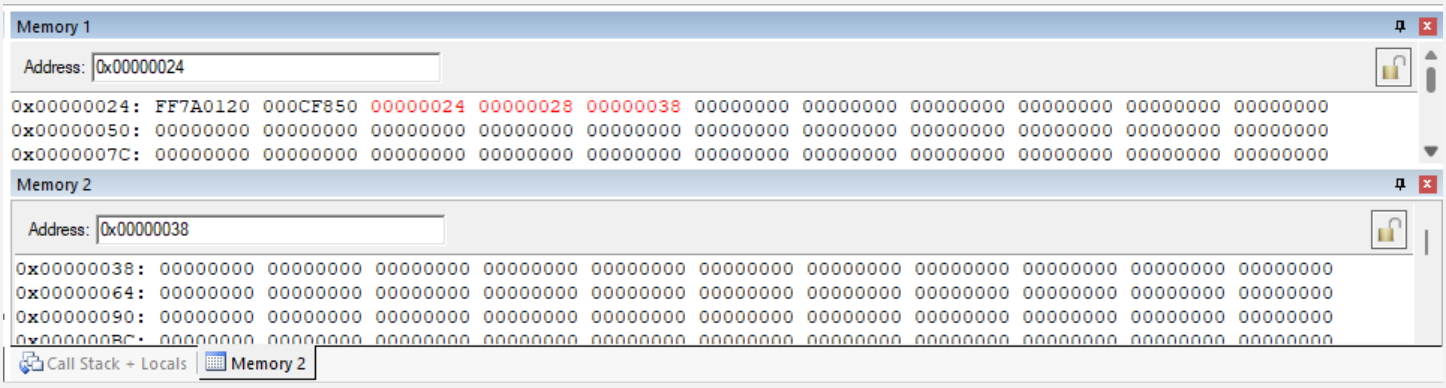
```
    AREA AllocSpace, DATA,NOINIT,READWRITE
    ;Space for storing result
    ;Utkarsh && Arnav Lab6
c SPACE 8;
;Actual Code
    AREA hmm, CODE, READWRITE
```

```
    EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R1, =a;
    LDR R2, =bi;
    ;address to the result
    LDR R3, =c ;
    LDR R4, [R1]; a
    LDR R5, [R2]; b
    SMULL R6, R7, R5, R4; multiply
    STR R6, [R3]; low
    STR R7, [R3, #4]; high
stop BAL stop
; input_number
a   DCD 0xFF7A0120;
bi  DCD 0xCF850;
    END
```

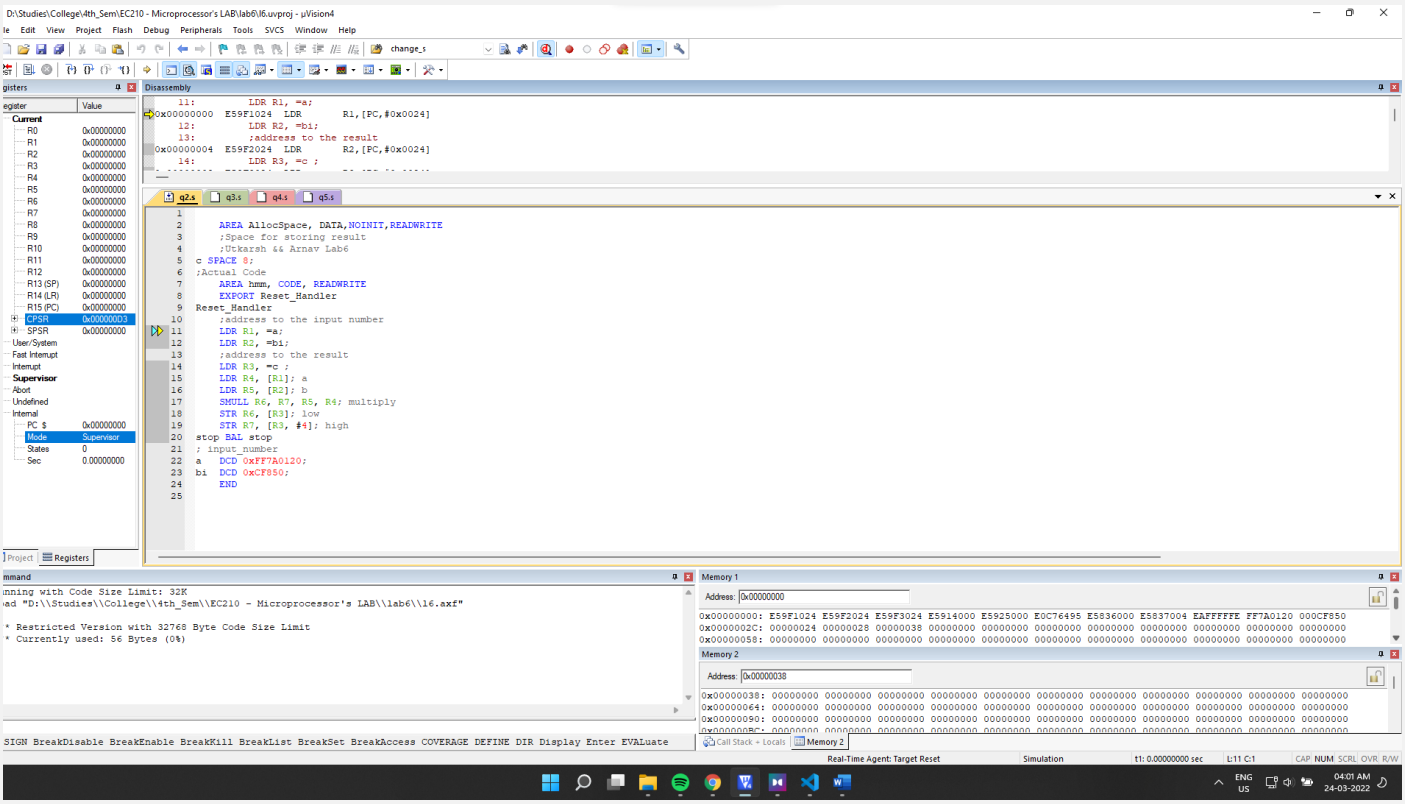
Debugging:

Initial Memory: (after getting the address through register)

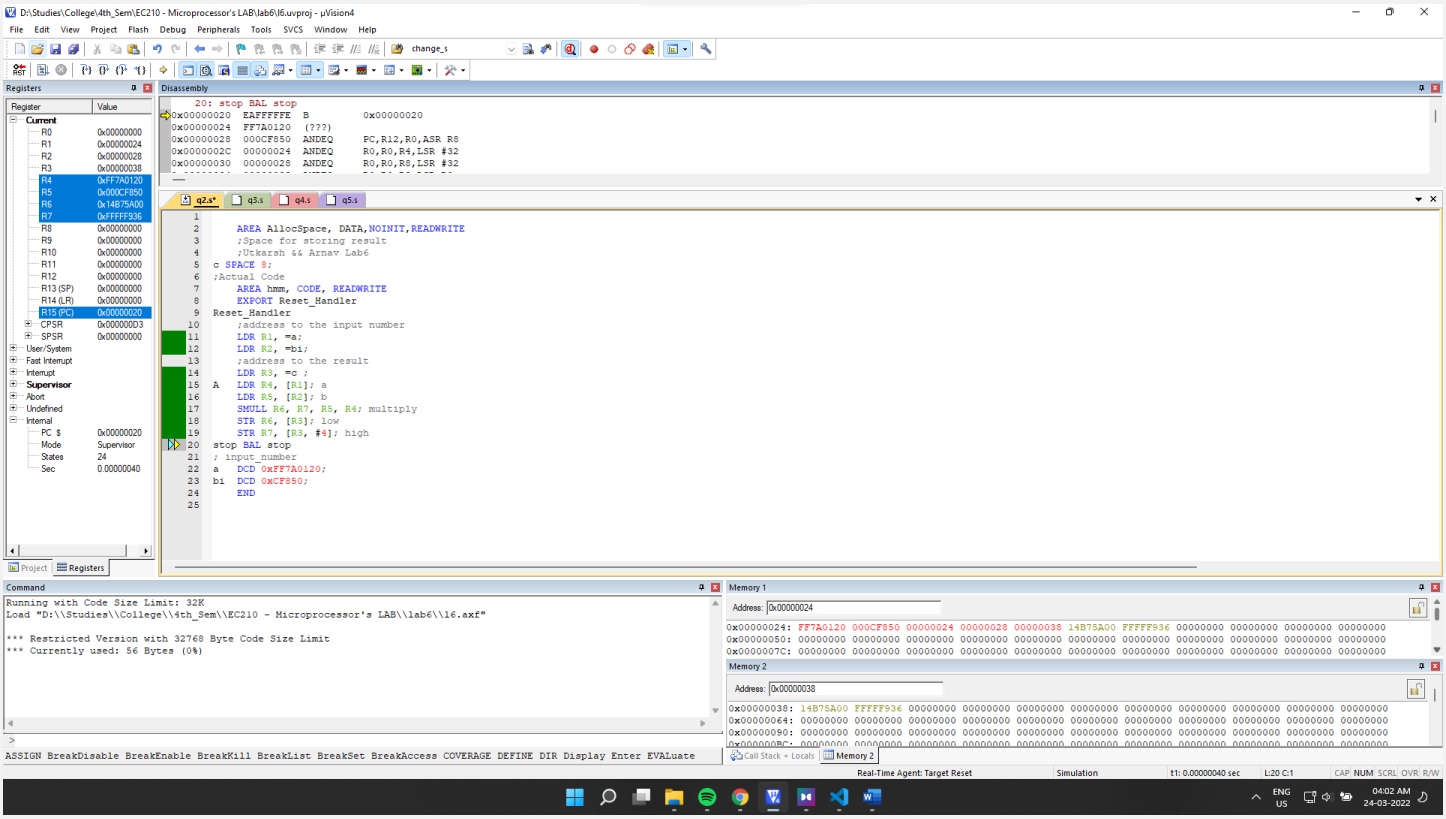
Memory 1 shows input number and memory 2 output



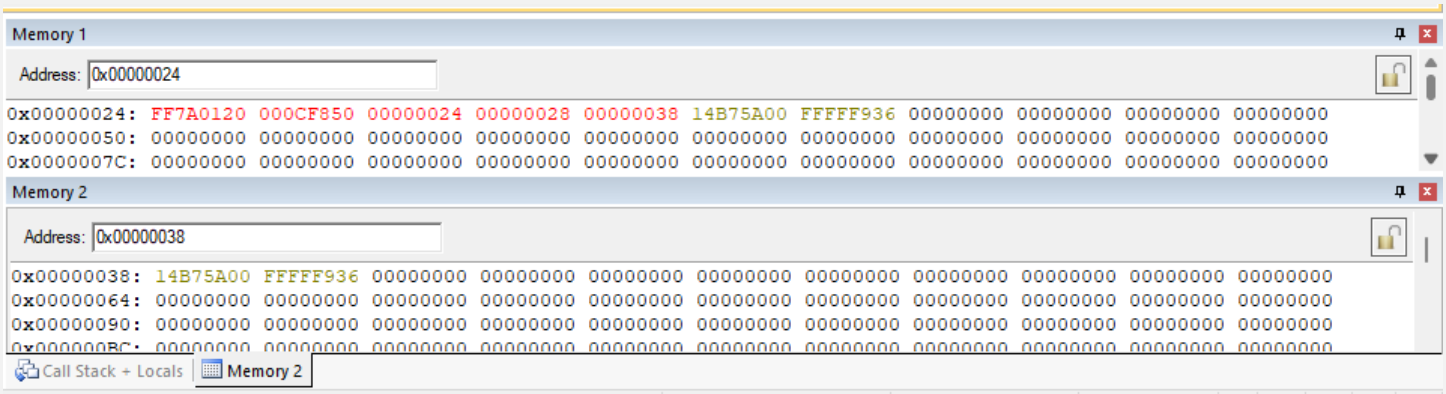
Setup:



Final Output:



Final Memory:



(c) a and b are both unsigned 64 bit numbers.

->Source Code:

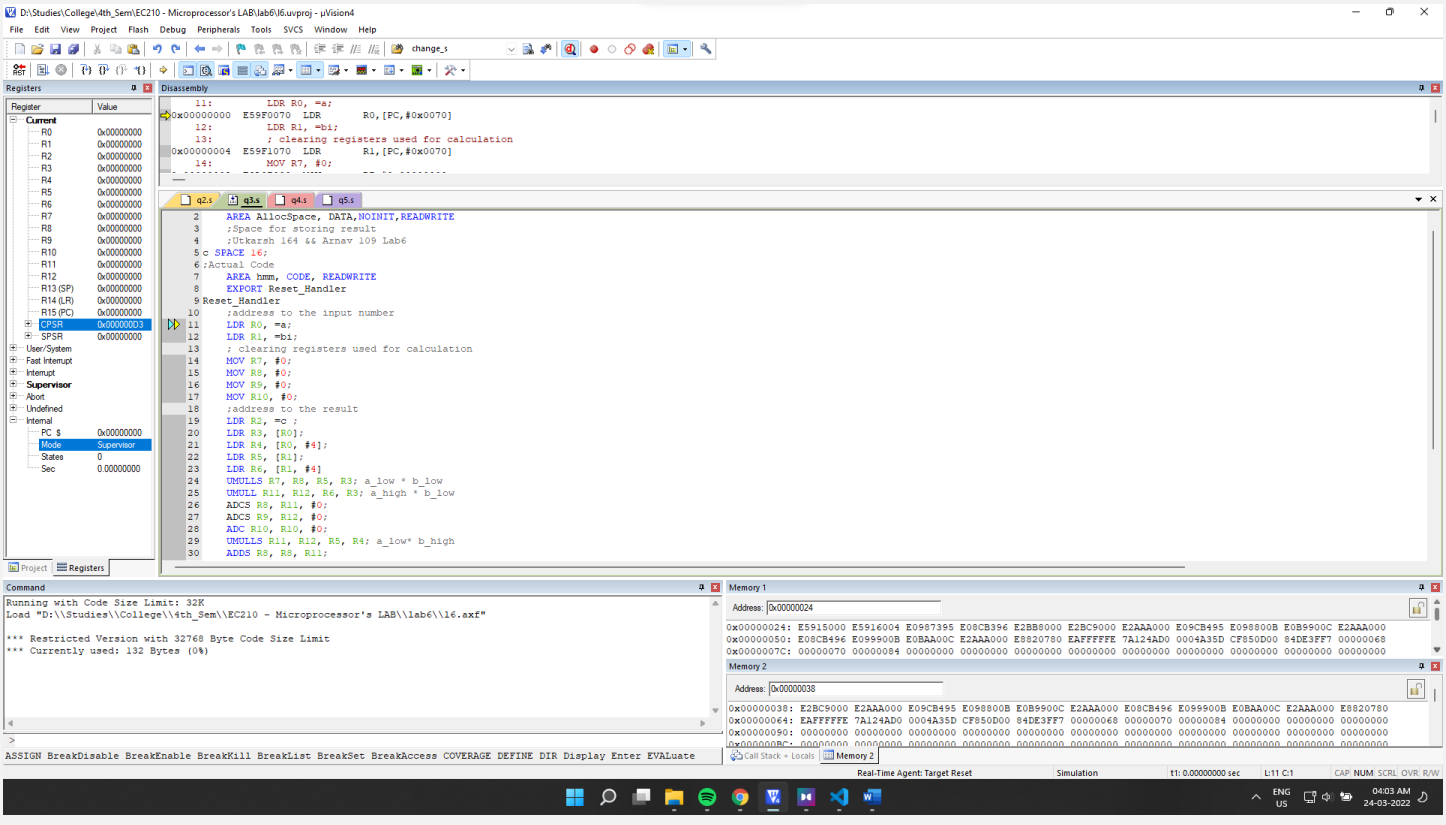
```
    AREA AllocSpace, DATA,NOINIT,READWRITE
    ;Space for storing result
    ;Utkarsh 164 && Arnav 109 Lab6
c SPACE 16;
;Actual Code
    AREA hmm, CODE, READWRITE
    EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R0, =a;
    LDR R1, =bi;
    ; clearing registers used for calculation
    MOV R7, #0;
    MOV R8, #0;
    MOV R9, #0;
    MOV R10, #0;
    ;address to the result
    LDR R2, =c ;
    LDR R3, [R0];
    LDR R4, [R0, #4];
    LDR R5, [R1];
    LDR R6, [R1, #4]
    UMULLS R7, R8, R5, R3; a_low * b_low
    UMULL R11, R12, R6, R3; a_high * b_low
    ADCS R8, R11, #0;
    ADCS R9, R12, #0;
    ADC R10, R10, #0;
    UMULLS R11, R12, R5, R4; a_low* b_high
    ADDS R8, R8, R11;
```

```
    ADCS R9, R9, R12;
    ADC R10, R10, #0;
    UMULL R11, R12, R6, R4; a_high * b_high
    ADDS R9, R9, R11;
    ADCS R10, R10, R12;
    ADC R10, R10, #0;
    STMIA R2, {R7-R10};

stop BAL stop
; input_number
a   DCD 0x7A124AD0, 0x4A35D;
bi  DCD 0xCF850D00, 0x84DE3FF7;
    END
```

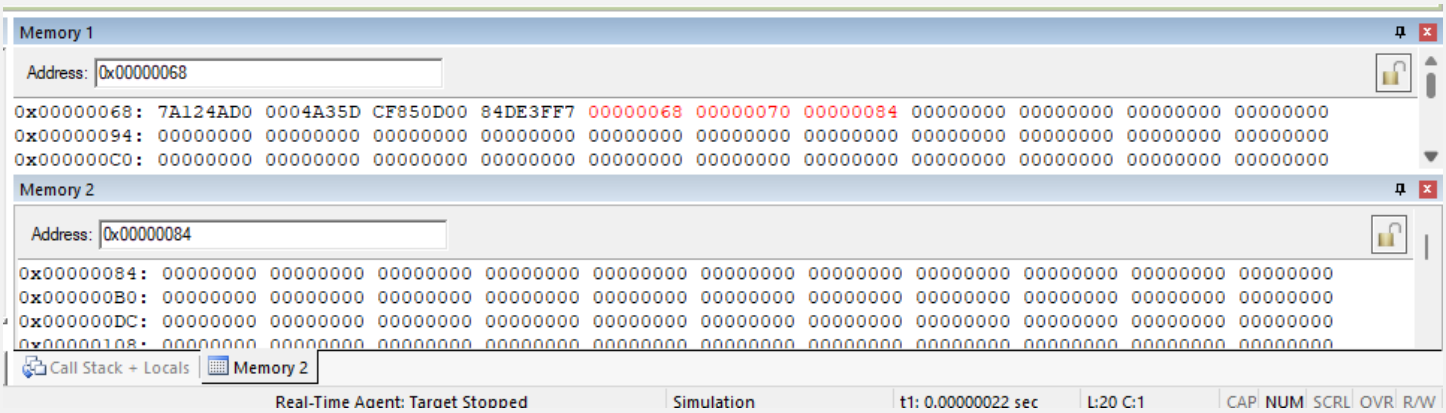
Debugging:

Setup:

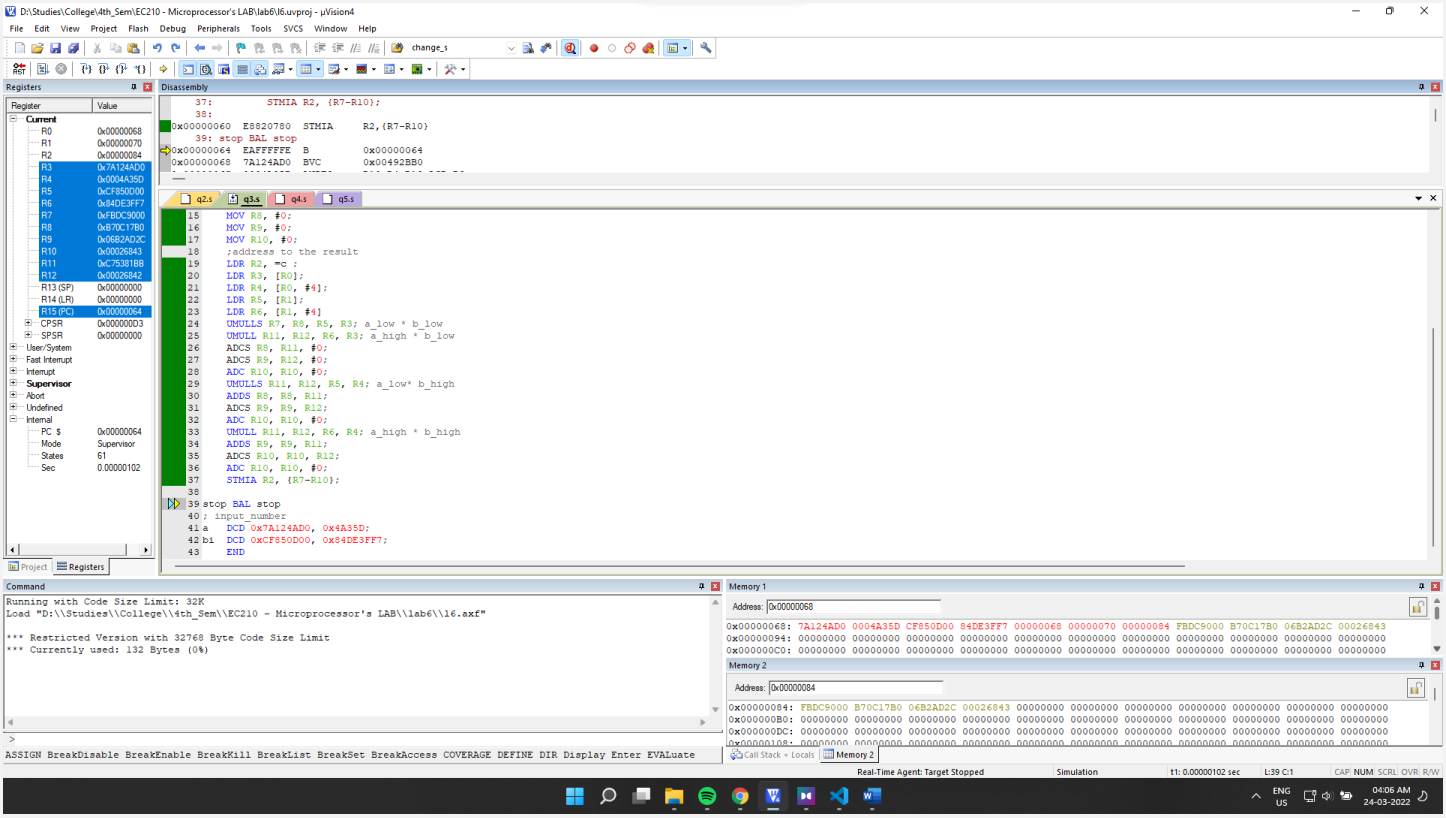


Initial Memory: (after getting the address through register)

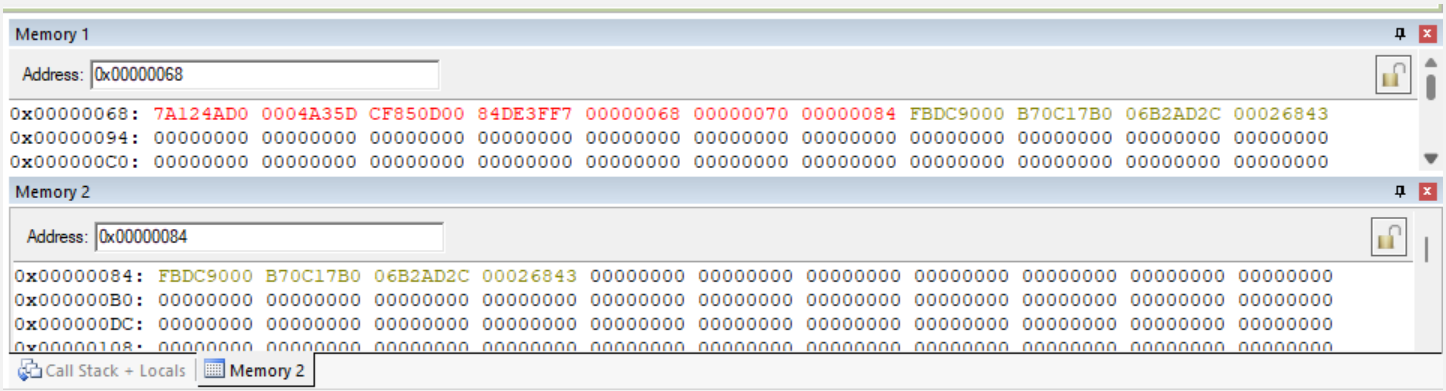
Memory 1 shows input number and memory 2 will hold output.



Final Output:



Final Memory Values:



(d) a and b are both signed 64 bit numbers.

->

Source Code:

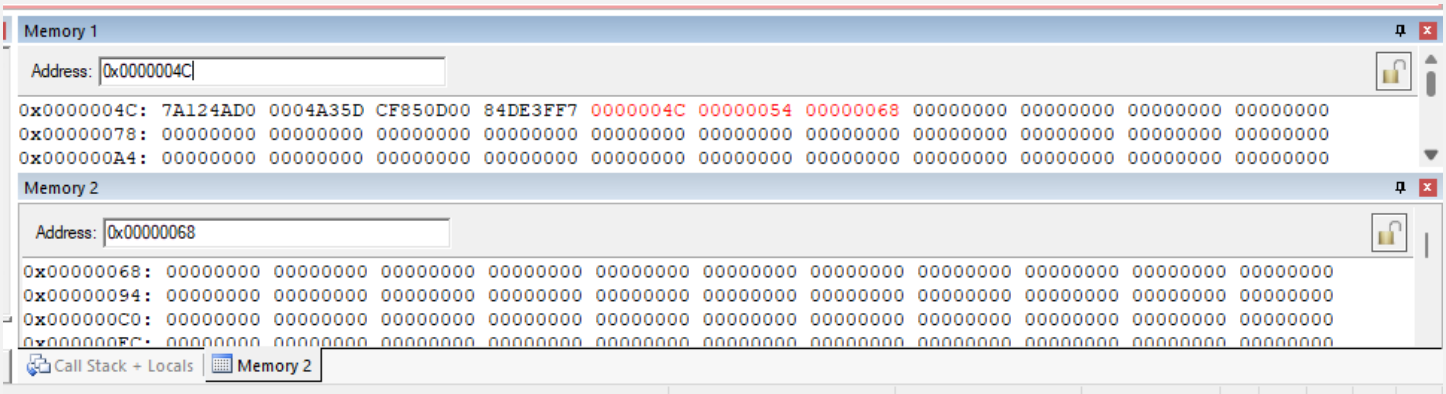
```
    AREA AllocSpace, DATA,NOINIT,READWRITE
    ;Space for storing result
    ;Utkarsh 164 && Arnav 109 Lab6
c SPACE 16;
;Actual Code
    AREA hmm, CODE, READWRITE
    EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R0, =a;
    LDR R1, =bi;
    ; clearing registers used for calculation
    MOV R7, #0;
    MOV R8, #0;
    MOV R9, #0;
    MOV R10, #0;
    ;address to the result
    LDR R2, =c ;
    LDR R3, [R0];
    LDR R4, [R0, #4];
    LDR R5, [R1];
    LDR R6, [R1, #4]
    UMULL R7, R8, R5, R3; unsigned a_low * b_low
    SMLALS R8, R9, R3, R6; signed a_low * b_high
    ADC R10, R10, #0; adding carry if any.
    SMLALS R8, R9, R4, R5; signed a_high * b_low
    ADC R10, R10, #0; adding carry if any.
    SMLAL R9, R10, R4, R6; signed a_high * b_high
    STMIA R2, {R7-R10};

stop BAL stop
; input_number
a   DCD 0x7A124AD0, 0x4A35D; 0x4A35D7A124AD0
bi  DCD 0xCF850D00, 0x84DE3FF7; 0x84DE3FF7CF850D00
    END
```

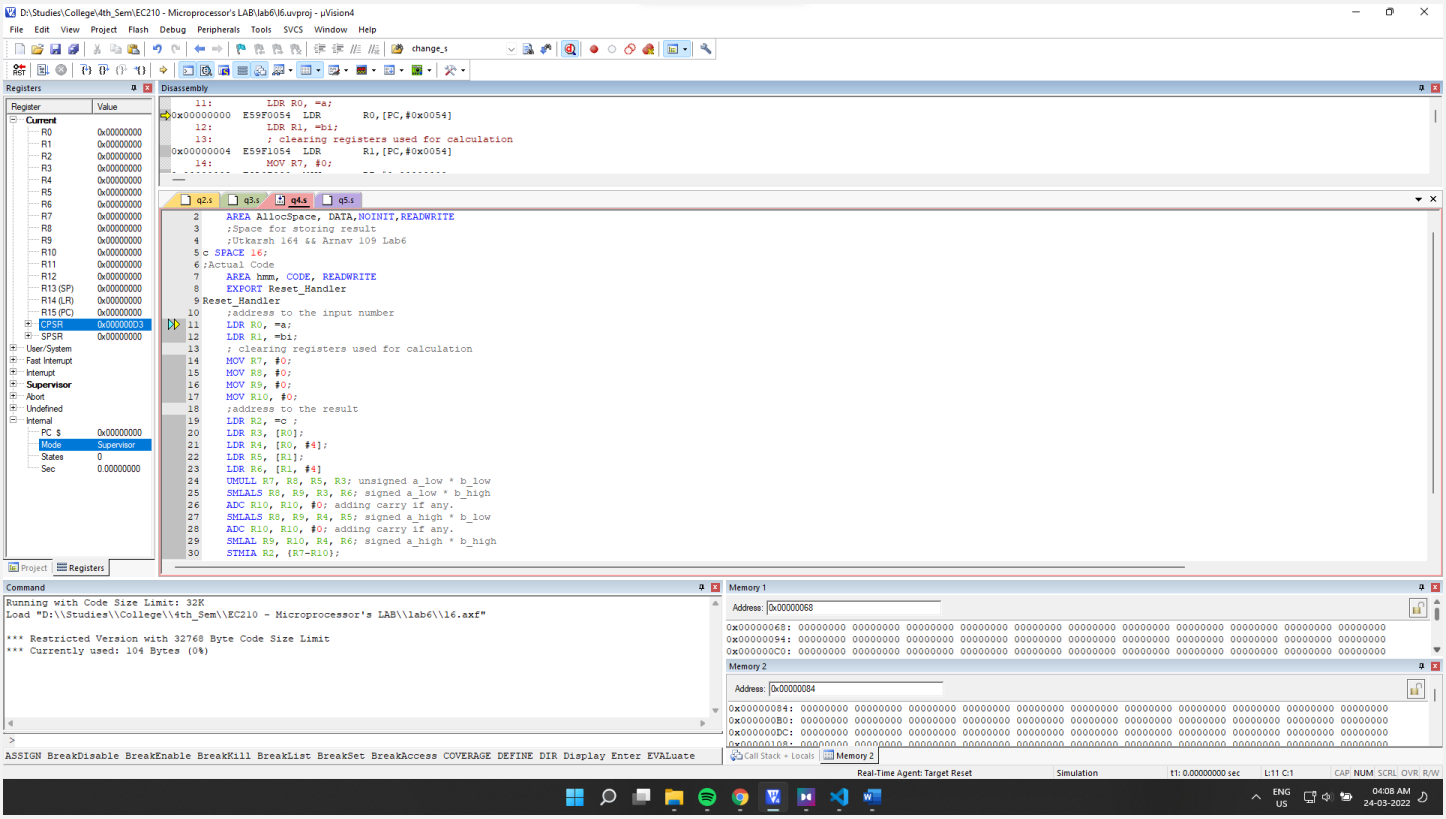
Debugging:

Initial Memory: (after getting the address through register)

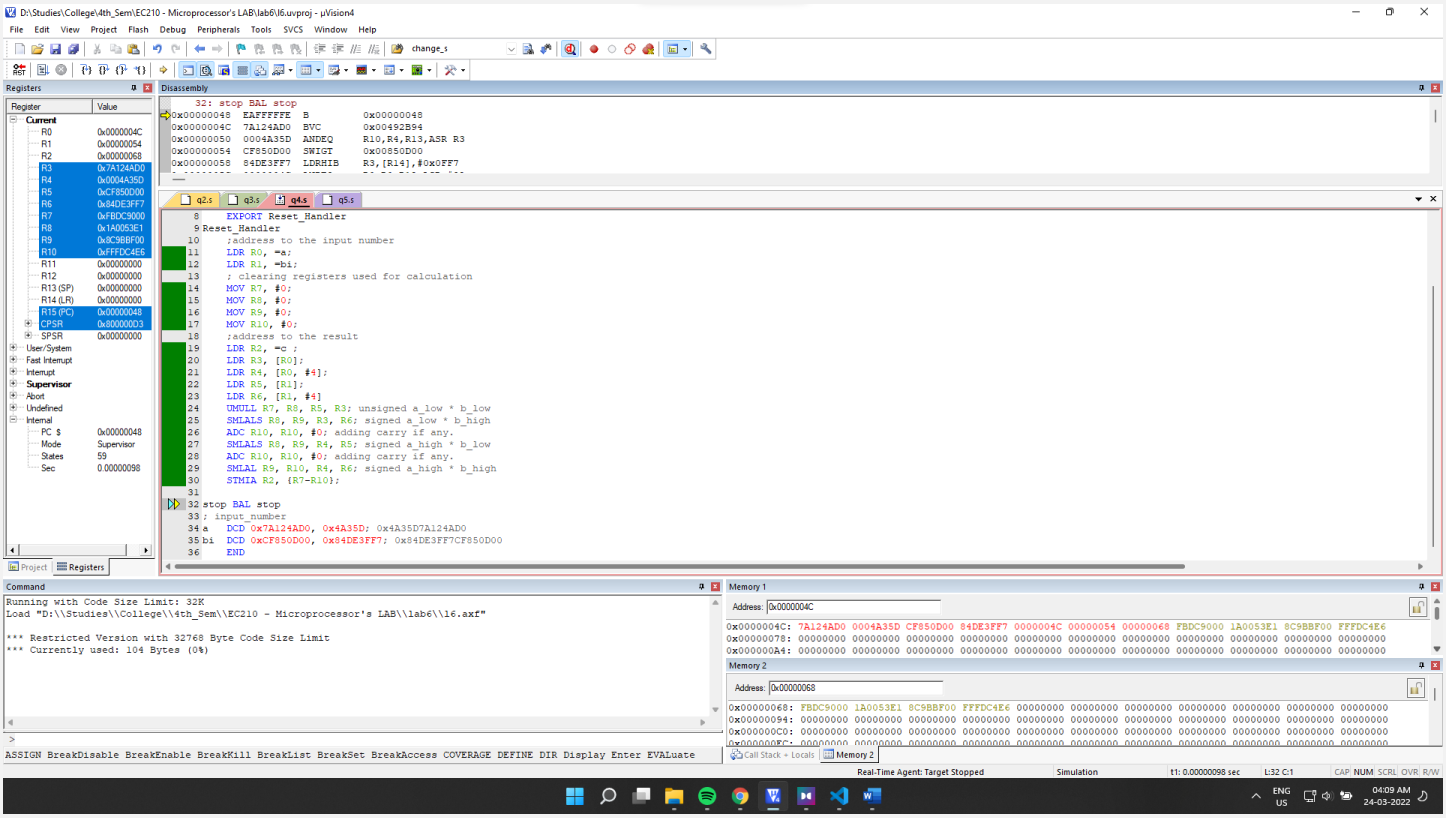
Memory 1 shows input, memory 2 will hold output.



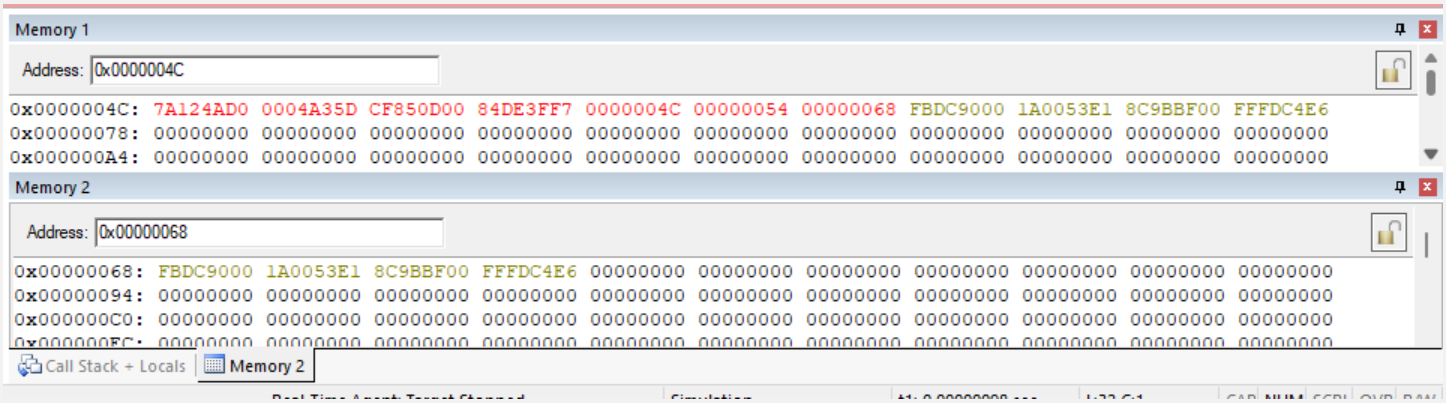
Setup:



Final Output:



Final Memory:



6.2] Assume that a signed long multiplication instruction is not available. Write a program that performs long multiplications, producing 64 bits of result. Use only the UMULL instruction and logical operations such as MVN to invert EOR and ORR

->

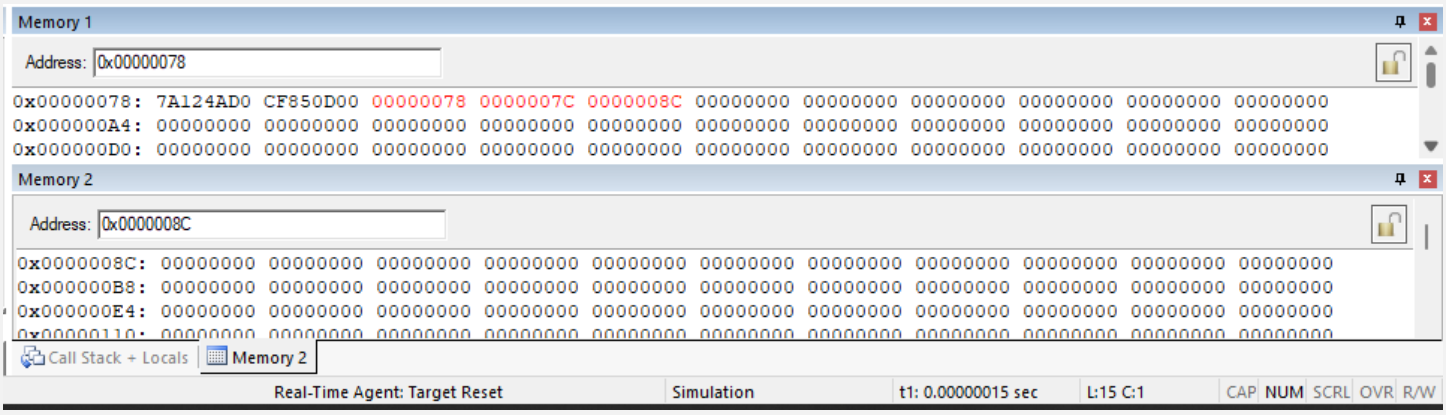
Source Code:

```
    AREA AllocSpace, DATA,NOINIT,READWRITE
    ;Space for storing result
    ;Utkarsh 164 && Arnav 109 Lab6
c SPACE 8;
;Actual Code
    AREA hmm, CODE, READWRITE
    EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R11, =a;
    LDR R12, =bi;
    ;address to the result
    LDR R0, =c ;
    LDR R1, [R11];
    LDR R2, [R12];
    LSR R3, R1, #31;
    LSR R4, R2, #31;
comp_1 CMP R3, #1;
    BEQ fn
comp_2 CMP R4, #1;
    BEQ sn
    B mult
fn MVN R5, R5;
    MVN R1, R1;
    ADD R1, R1, #1;
    B comp_2
sn MVN R5, R5;
    MVN R2, R2;
    ADD R2, R2, #1;
mult UMULL R6, R7, R1, R2;
    CMP R5, #0xFFFFFFFF;
    BEQ cs;
    B store;
cs MVN R6, R6;
    ADDS R6, R6, #1;
    MVN R7, R7;
    ADC R7, R7, #0;
store   STR R7, [R0];
    STR R2, [R0, #4];
stop BAL stop;
```

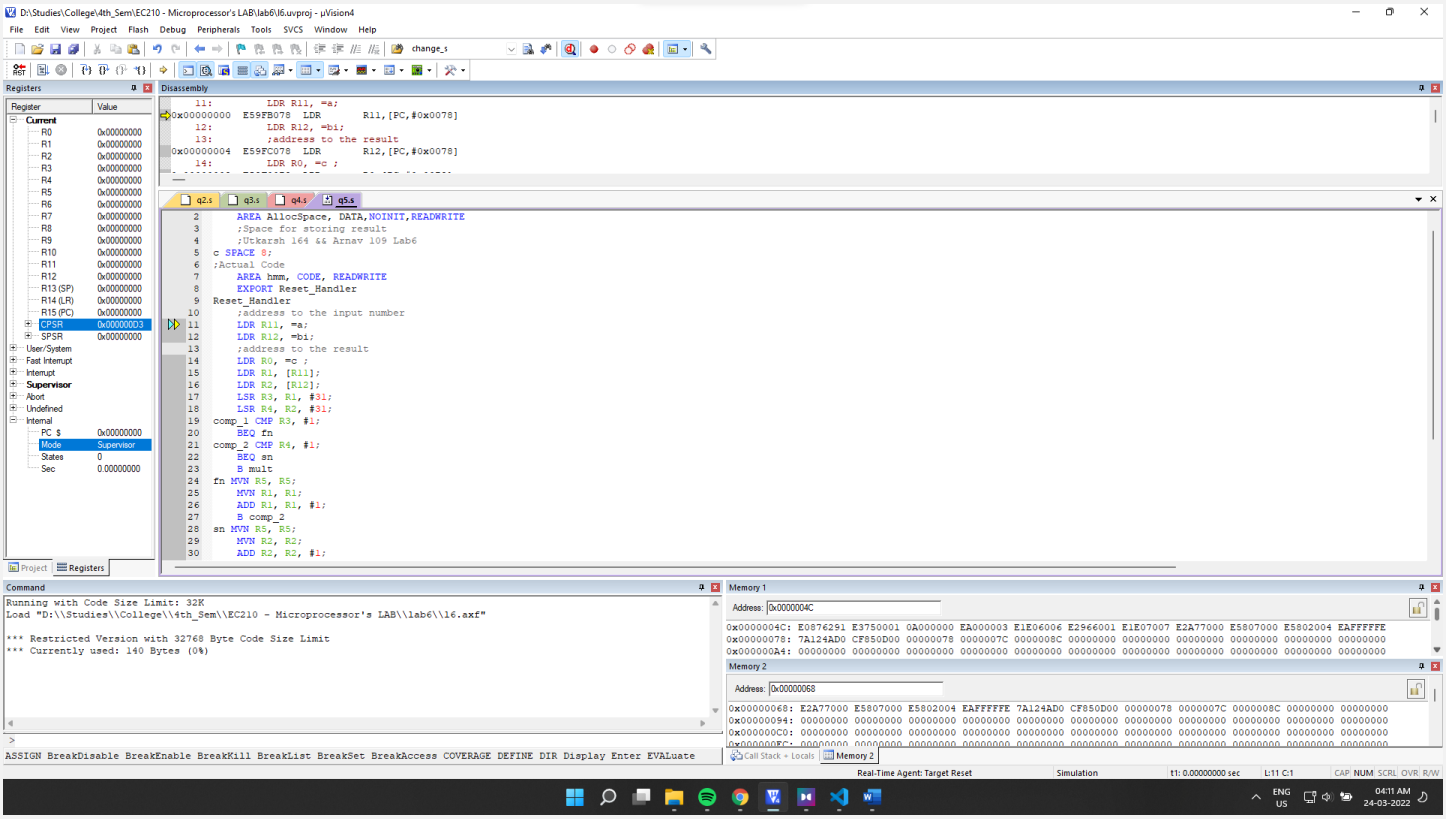
```
; input_number
a    DCD 0x7A124AD0; 0x7A124AD0
bi   DCD 0xCF850D00; 0xCF850D00
     END
```

Initial Memory: (after getting the address through register and loading the value into the reserved space)

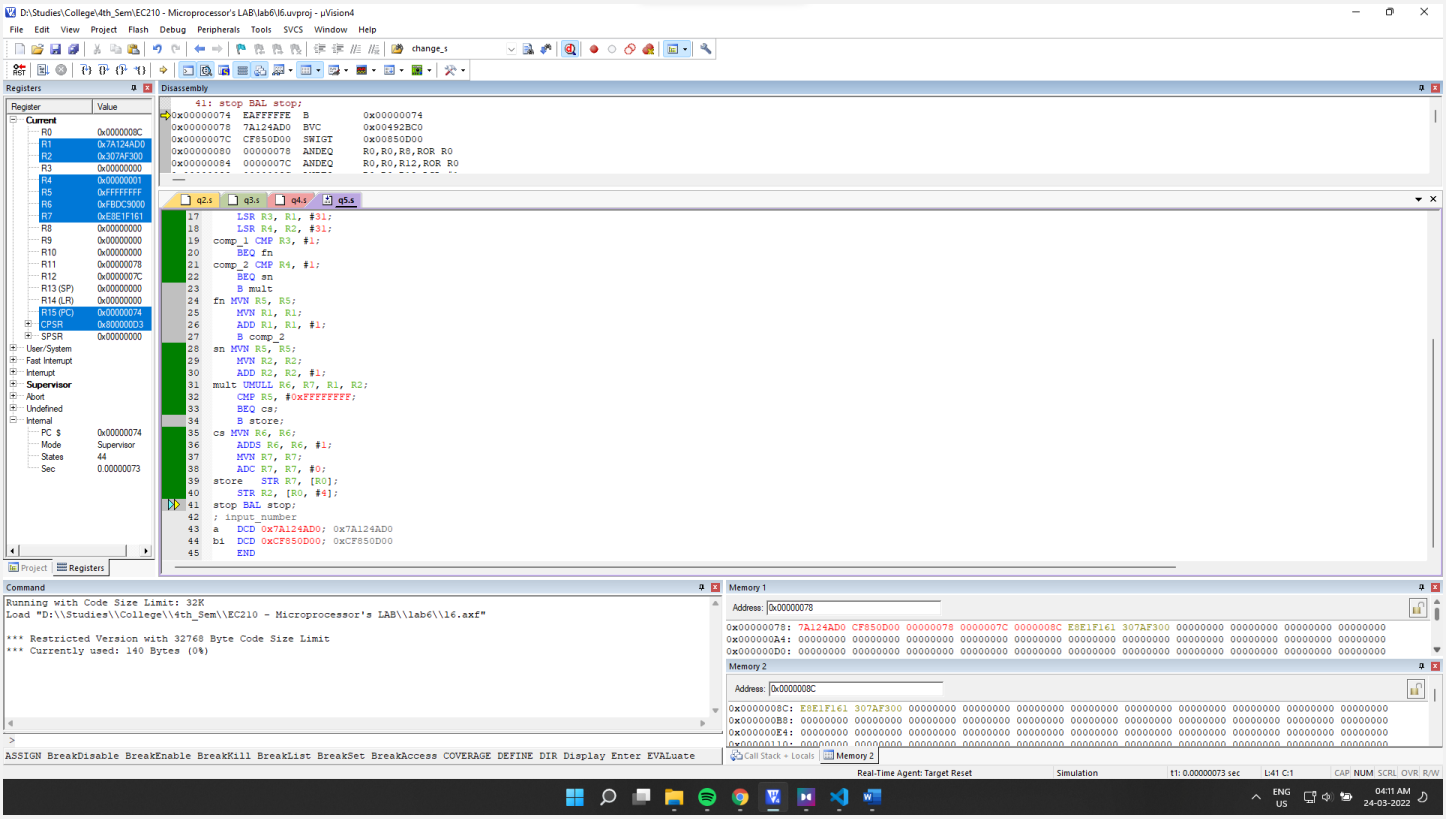
Memory 1 has the input no and memory 2 will hold result.



Setup:



Final Output:



Final Memory: