

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.

->

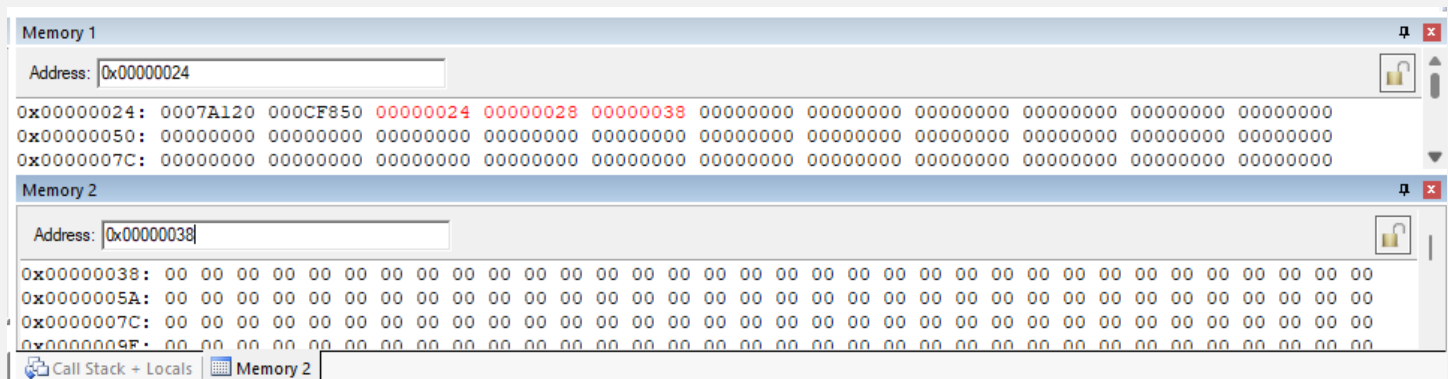
Source Code:

```
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
```

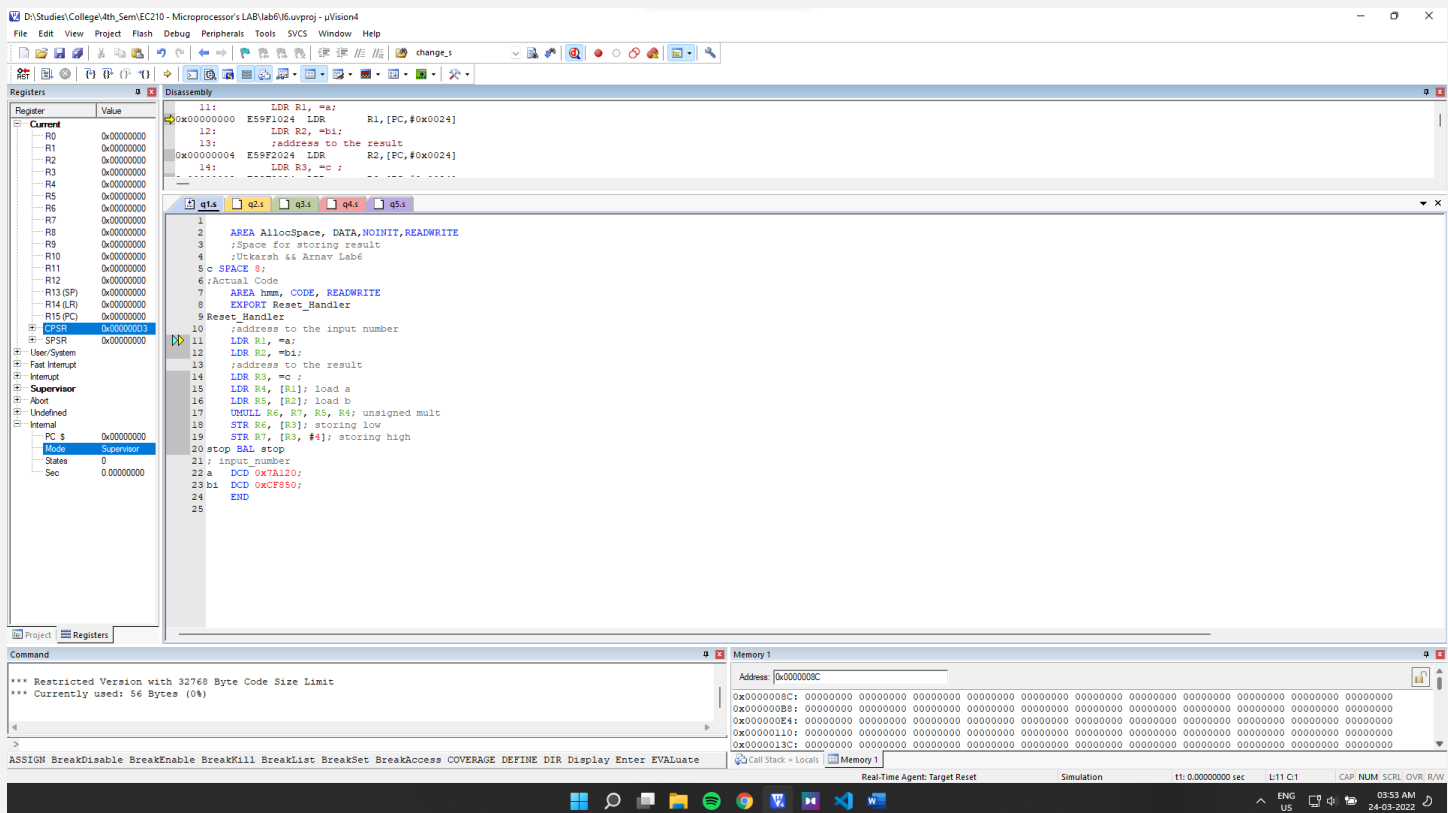
Debugging:

Initial Memory: (after getting the address through register)

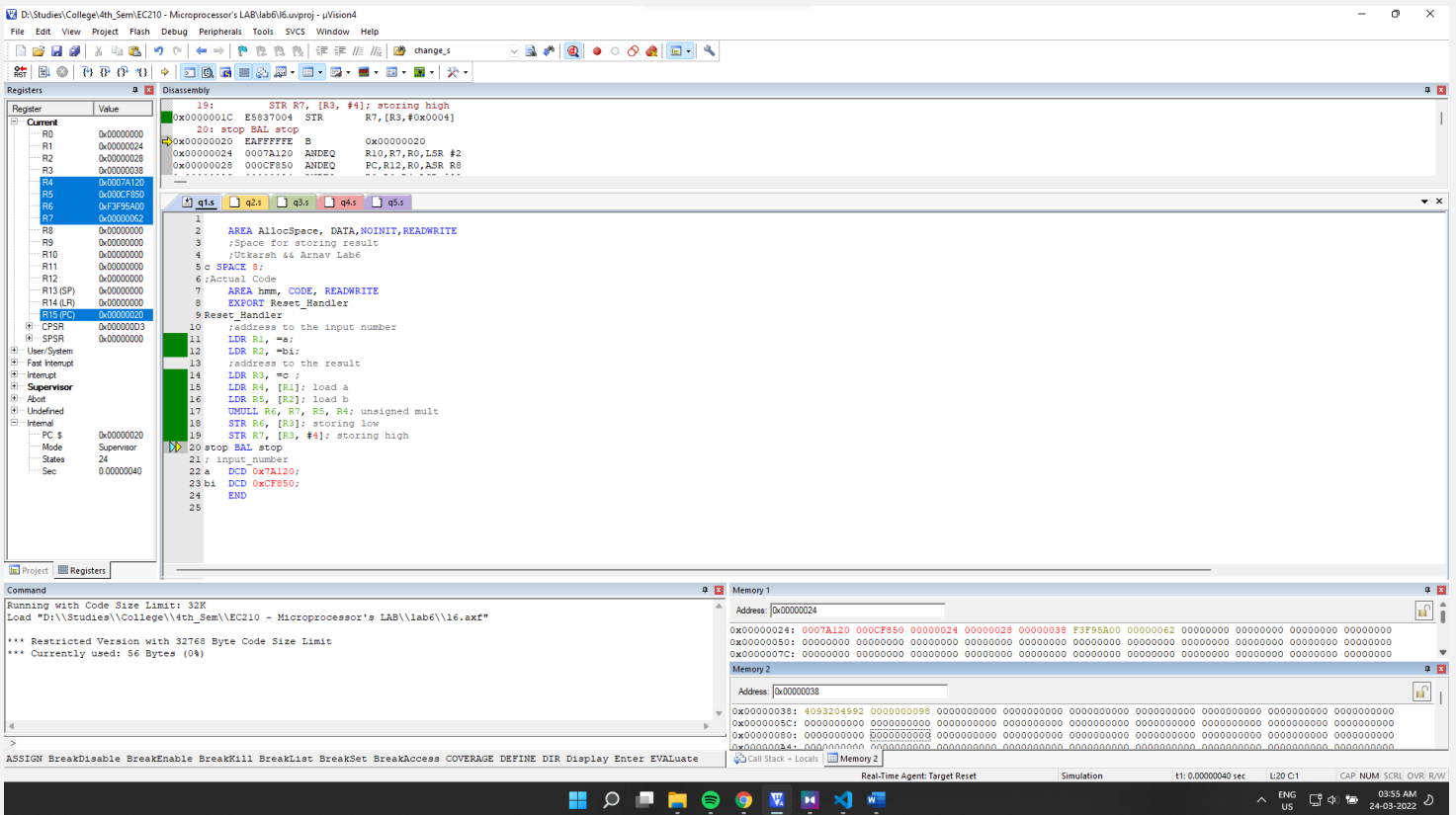
Memory 1 shows input and memory 2 will store output



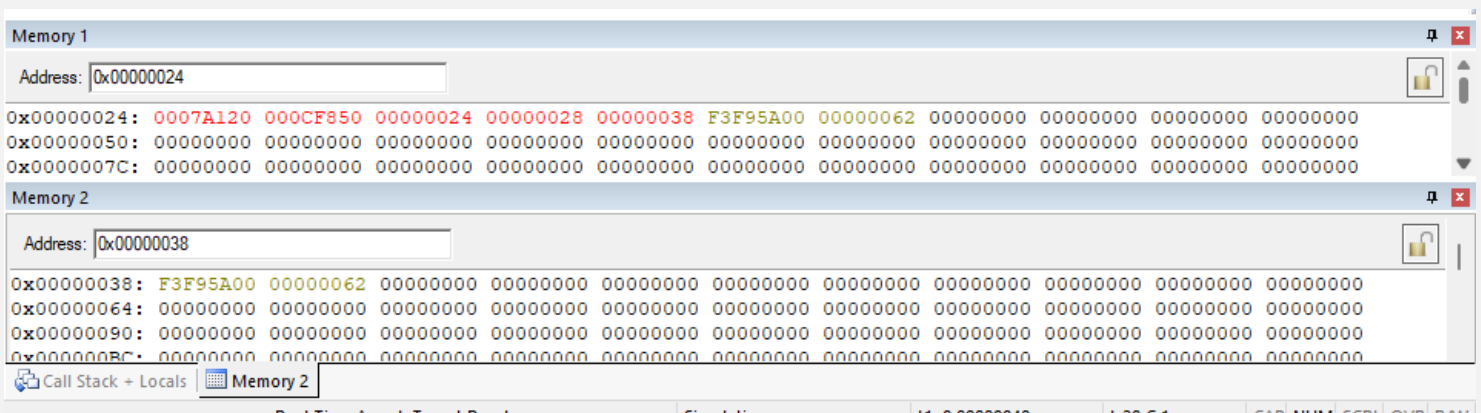
Setup:



Final Output:



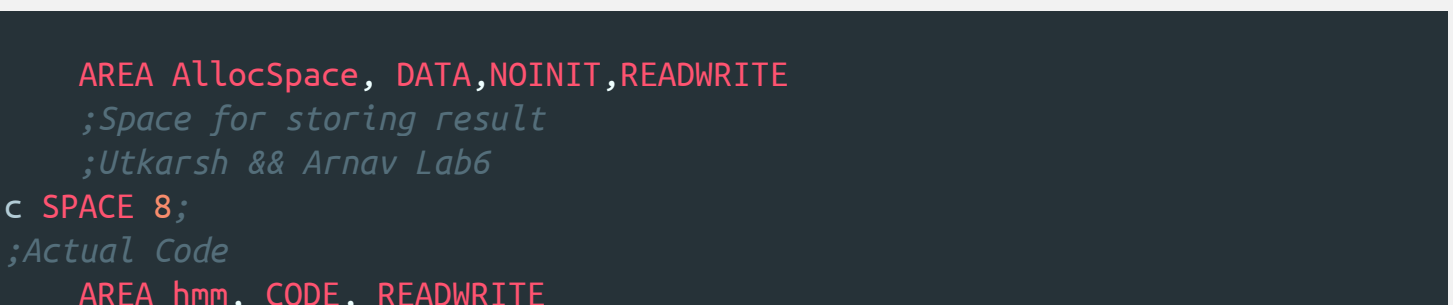
Final Memory:



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

->

Source Code:



```

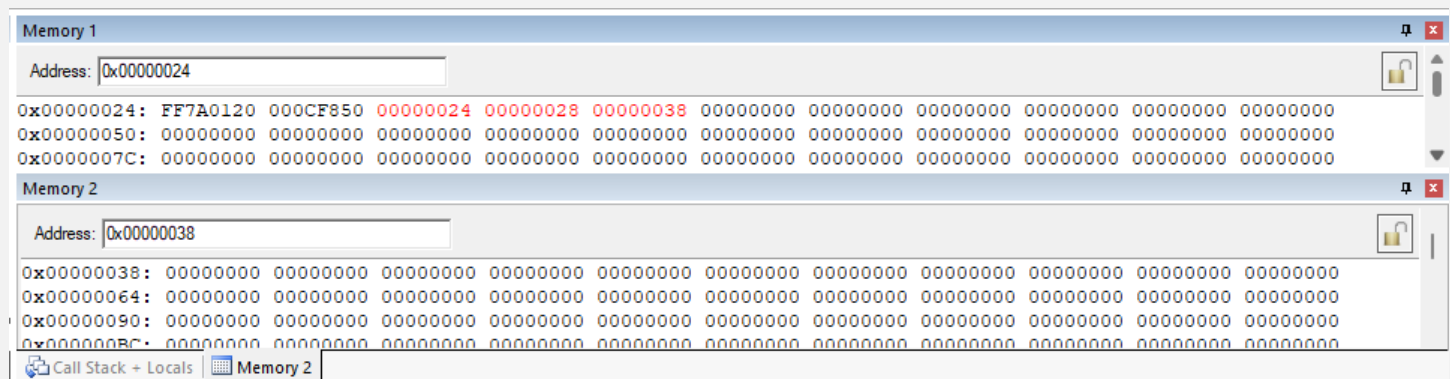
EXPORT Reset_Handler
Reset_Handler
    ;address to the input number
    LDR R1, =a;
    LDR R2, =bi;
    ;address to the result
    LDR R3, =c ;
    LDR R4, [R1]; a
    LDR R5, [R2]; b
    SMULL R6, R7, R5, R4; multiply
    STR R6, [R3]; low
    STR R7, [R3, #4]; high
stop BAL stop
; input_number
a 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

```

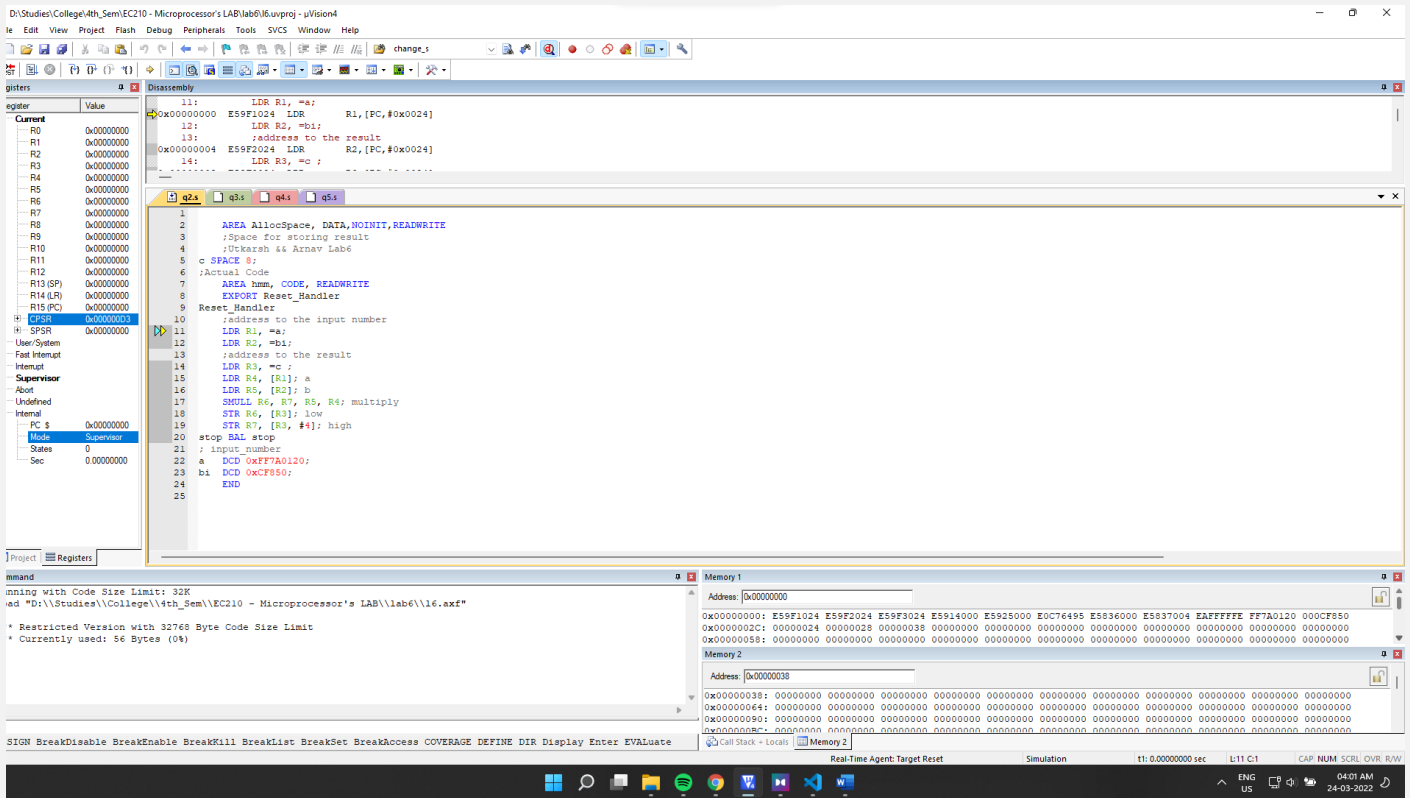
Debugging:

Initial Memory: (after getting the address through register)

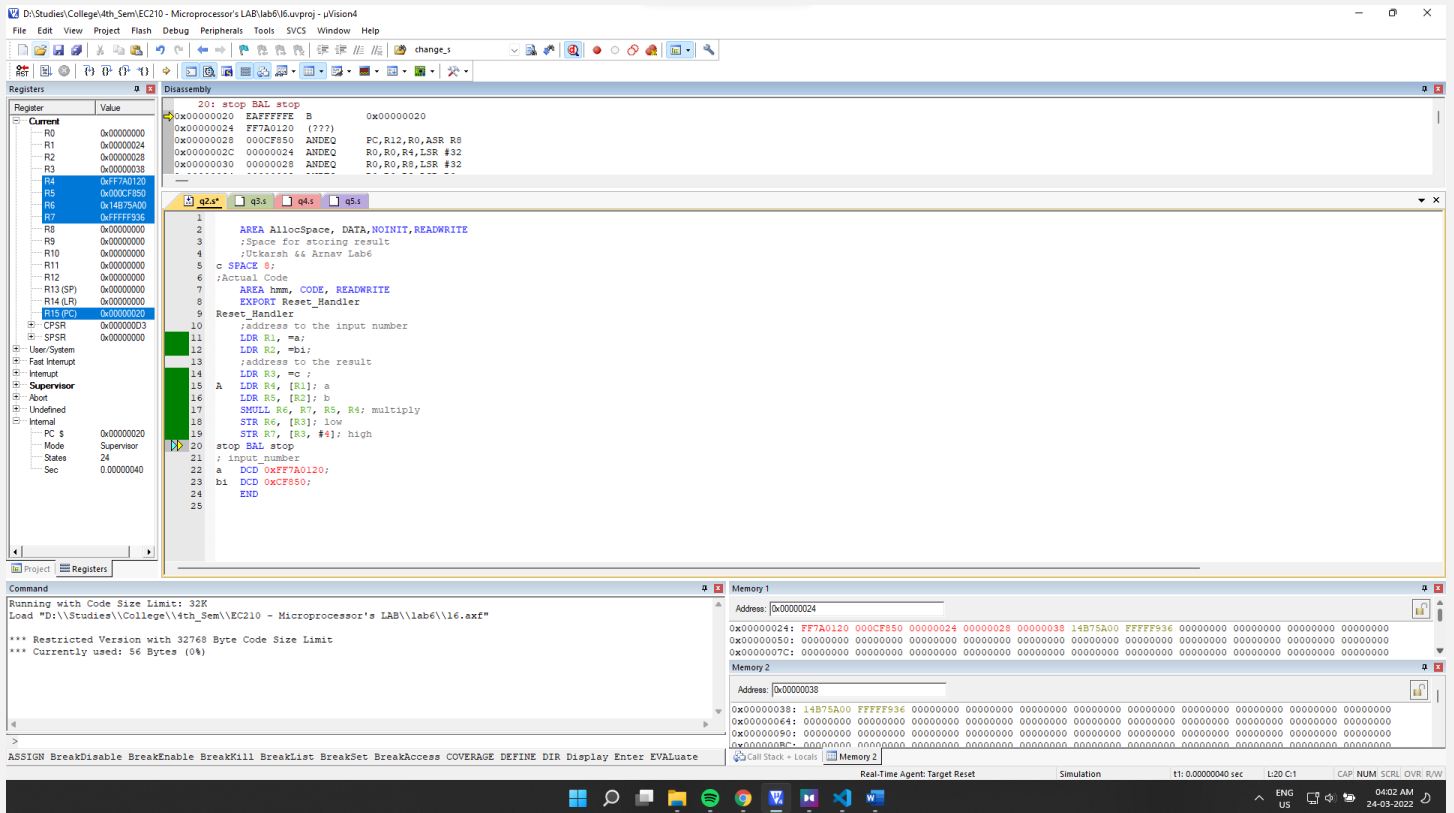
Memory 1 shows input number and memory 2 output



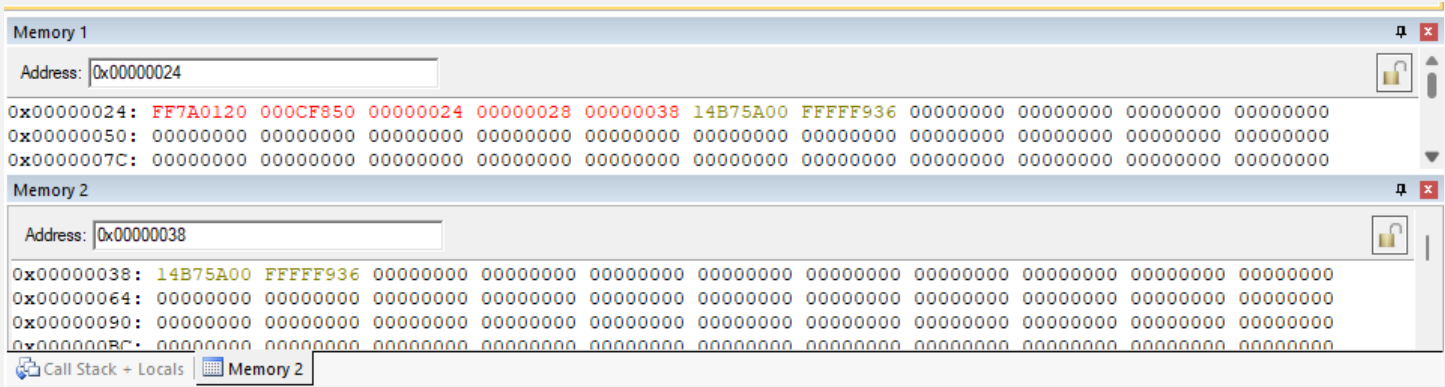
Setup:



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, =b;
; 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;

```

```

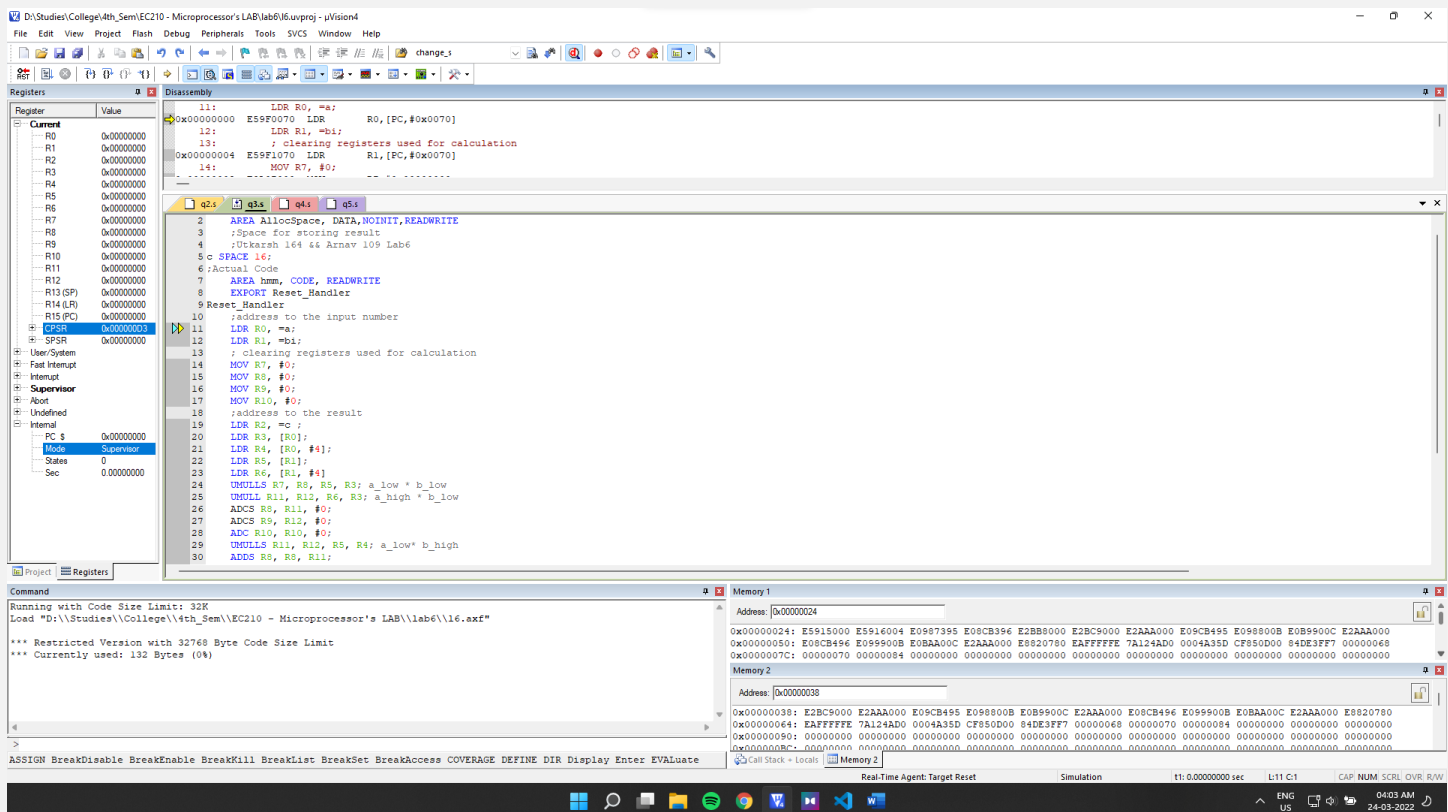
ADCS R9, R9, R12;
ADC R10, R10, #0;
UMULL R11, R12, R6, R4; a_high * b_high
ADDS R9, R9, R11;
ADCS R10, R10, R12;
ADC R10, R10, #0;
STMIA R2, {R7-R10};

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.

Memory 1

Address: 0x00000068

```

0x00000068: 7A124AD0 0004A35D CF850D00 84DE3FF7 00000068 00000070 00000084 00000000 00000000 00000000 00000000
0x00000094: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Memory 2

Address: 0x00000084

```

0x00000084: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000DC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000108: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Call Stack + Locals | **Memory 2**

Real-Time Agent: Target Stopped | Simulation | t1: 0.00000022 sec | L:20 C:1 | CAP NUM SCRL OVR R/W

Final Output:

D:\Studies\College\4th_Sem\EC210 - Microprocessor's LAB\lab6\I6.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x00000068
R1	0x00000070
R2	0x00000084
R3	0x7A124AD0
R4	0x0004A35D
R5	0xCF850D00
R6	0x84DE3FF7
R7	0xFBDC9000
R8	0xB70C17B0
R9	0x06B2AD2C
R10	0x00026843
R11	0xC75381B8
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000064
CPSR	0x00000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC	0x00000064
Mode	Supervisor
States	61
Sec	0.00000102

Disassembly

```

37: SIMIA R2, (R7-R10);
38:
39: stop BAL stop
40: input_number
41: DCD 0x7A124AD0, 0x4A35D;
42: b1 DCD 0xCF850D00, 0x84DE3FF7;
43: END

```

Command

Running with Code Size Limit: 32K

Load "D:\Studies\College\4th_Sem\EC210 - Microprocessor's LAB\lab6\I6.axf"

*** Restricted Version with 32768 Byte Code Size Limit

*** Currently used: 132 Bytes (0%)

ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate

Memory 1

Address: 0x00000068

```

0x00000068: 7A124AD0 0004A35D CF850D00 84DE3FF7 00000068 00000070 00000084 FBDC9000 B70C17B0 06B2AD2C 00026843
0x00000094: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Memory 2

Address: 0x00000084

```

0x00000084: FBDC9000 B70C17B0 06B2AD2C 00026843 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000DC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000108: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Call Stack + Locals | **Memory 2**

Real-Time Agent: Target Stopped | Simulation | t1: 0.00000102 sec | L:99 C:1 | CAP NUM SCRL OVR R/W

ENG US | 0408 AM | 24-03-2022

Final Memory Values:

Memory 1

Address: 0x00000068

```

0x00000068: 7A124AD0 0004A35D CF850D00 84DE3FF7 00000068 00000070 00000084 FBDC9000 B70C17B0 06B2AD2C 00026843
0x00000094: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000C0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Memory 2

Address: 0x00000084

```

0x00000084: FBDC9000 B70C17B0 06B2AD2C 00026843 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000B0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x000000DC: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0x00000108: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Call Stack + Locals | **Memory 2**

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

->

Source Code:

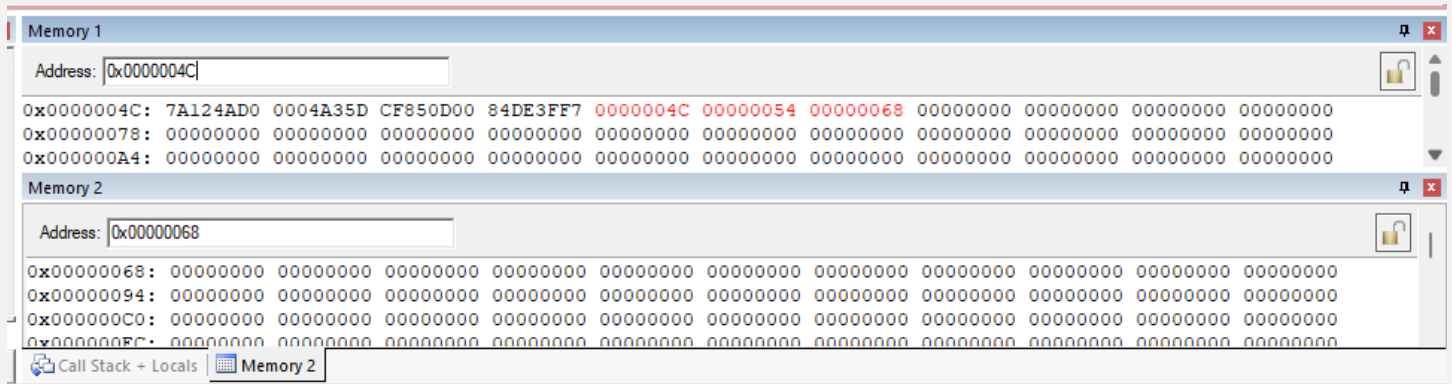
```
AREA AllocSpace, DATA, NOINIT, READWRITE
;Space for storing result
;Utkarsh 164 && Arnav 109 Lab6
c SPACE 16;
;Actual Code
AREA hmm, CODE, READWRITE
EXPORT Reset_Handler
Reset_Handler
;address to the input number
LDR R0, =a;
LDR R1, =bi;
; clearing registers used for calculation
MOV R7, #0;
MOV R8, #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
```

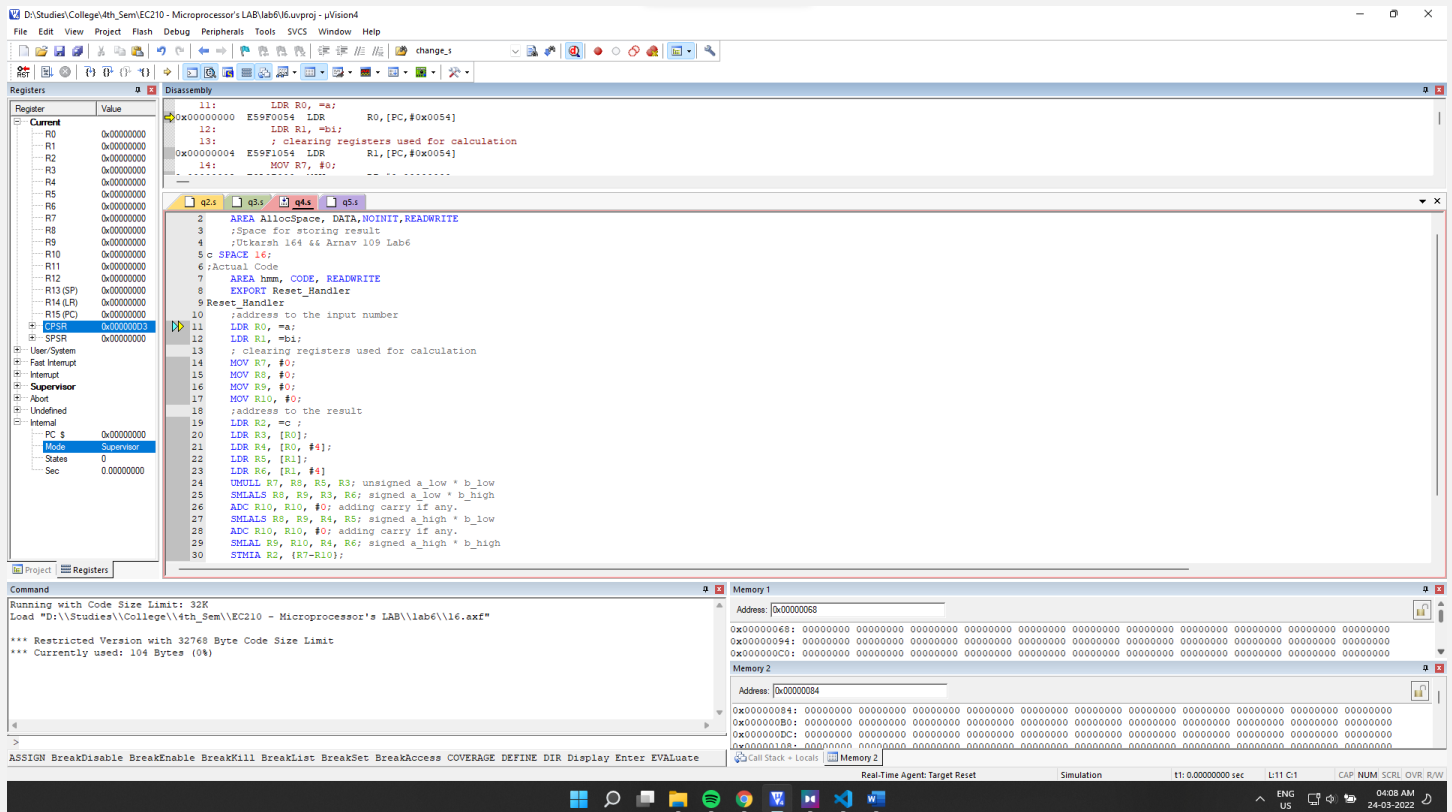
Debugging:

Initial Memory: (after getting the address through register)

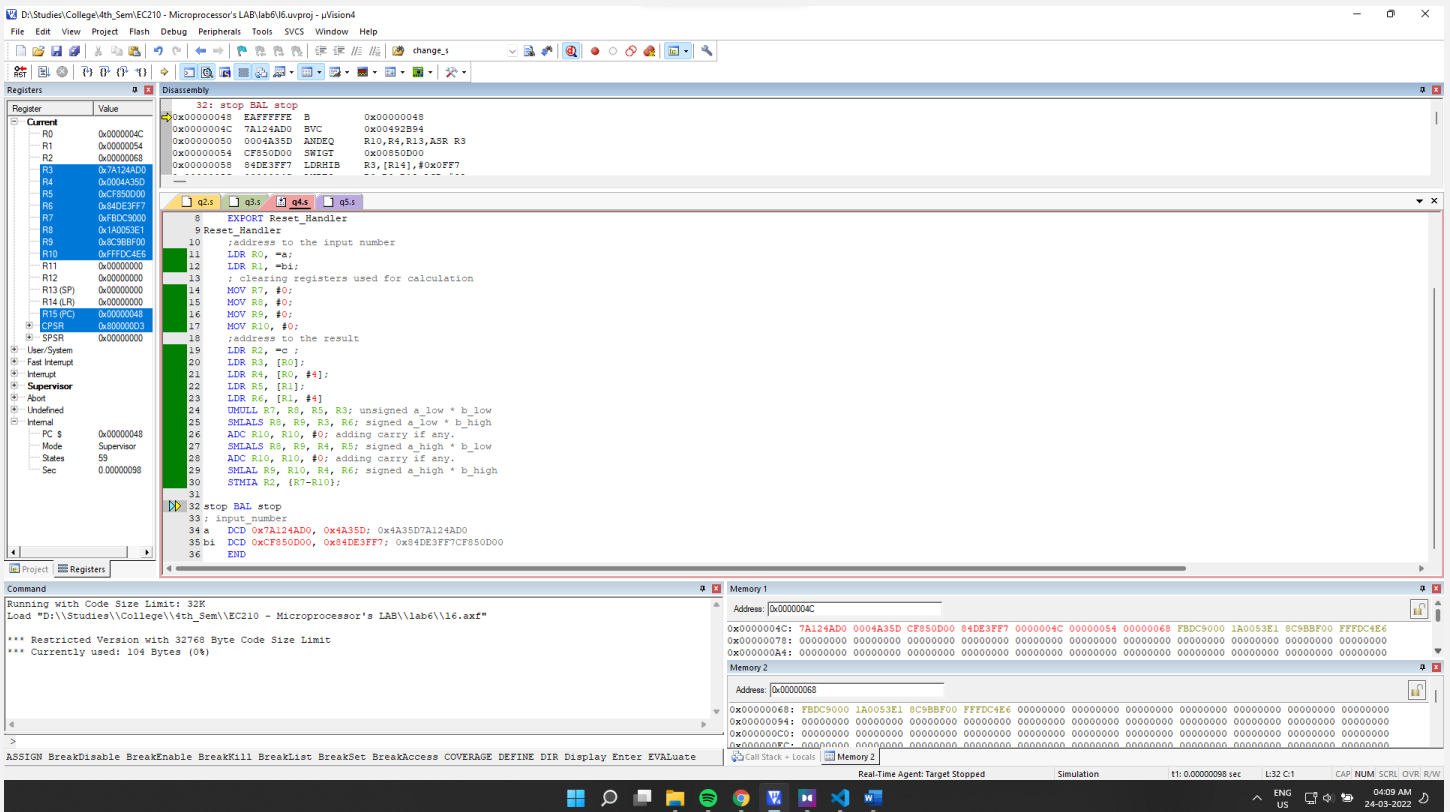
Memory 1 shows input, memory 2 will hold output.



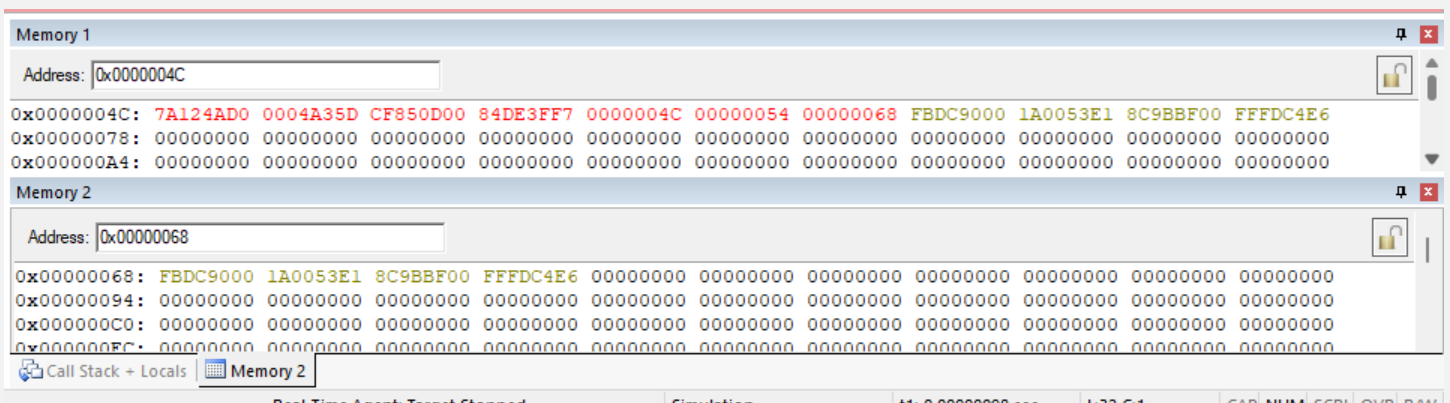
Setup:



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, =b;
    ;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.

The screenshot shows the Keil uVision IDE's memory window. It displays two memory locations: Memory 1 at address 0x00000078 and Memory 2 at address 0x0000008C. Memory 1 contains the value 7A124AD0, which is the input number. Memory 2 contains the value 00000000, which is the result. The status bar at the bottom indicates 'Real-Time Agent: Target Reset' and 'Simulation' mode.

Setup:

The screenshot shows the Keil uVision IDE's assembly view. The code defines the input number and result, and performs a multiplication operation. The code is as follows:

```

11: LDR R11, =a;
12: LDR R12, =b;
13: ;address to the result
14: LDR R0, =c;
15: ;
16: ;
17: ;
18: ;
19: ;
20: ;
21: ;
22: ;
23: ;
24: ;
25: ;
26: ;
27: ;
28: ;
29: ;
30: ;

```

The status bar at the bottom indicates 'Real-Time Agent: Target Reset' and 'Simulation' mode.

Final Output:

The screenshot displays the uVision4 IDE interface with the following components:

- Registers:** A list of registers (R0-R15, PC, CPSR, User/System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, Internal) with their current values. For example, R0 is 0x0000008C, R1 is 0x7A124AD0, and R2 is 0x307AF300.
- Disassembly:** A window showing the assembly code being executed. The code includes instructions like `stop BAL stop;`, `EAFFFFFE BVC`, `7A124AD0 BVC`, `CF850D00 SWIGT`, `00000078 ANDEQ`, `0000007C ANDEQ`, `LSR R3, R1, #31;`, `LSR R4, R2, #31;`, `comp_1 CMP R3, #1;`, `BEQ fn`, `comp_2 CMP R4, #1;`, `BEQ en`, `B mult`, `fn MVN R5, R5;`, `MVN R1, R1;`, `ADD R1, R1, #1;`, `B comp_2`, `en MVN R5, R5;`, `MVN R2, R2;`, `ADD R2, R2, #1;`, `mult UNDEF R6, R7, R1, R2;`, `CMP R5, #0xFFFFFFFF;`, `BEQ ca;`, `B store;`, `ca 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`, `DCD 0x7A124AD0; 0x7A124AD0`, `DCD 0xCF850D00; 0xCF850D00`, and `END`.
- Command:** A window showing the command line and the output of the program. The output includes the text: `Running with Code Size Limit: 32K`, `Load "D:\Studies\College\4th_Sem\EC210 - Microprocessor's LAB\lab6\l6.axf"`, `*** Restricted Version with 32768 Byte Code Size Limit`, and `*** Currently used: 140 Bytes (0%)`.
- Memory:** A window showing the memory contents. The memory is organized into two sections: **Memory 1** (Address: 0x00000078) and **Memory 2** (Address: 0x0000008C). The memory contents are displayed in hexadecimal and ASCII format.

Final Memory:

The screenshot displays the uVision4 IDE interface with the following components:

- Memory 1:** A window showing the memory contents starting at address 0x00000078. The memory is organized into two sections: **Memory 1** (Address: 0x00000078) and **Memory 2** (Address: 0x0000008C). The memory contents are displayed in hexadecimal and ASCII format.
- Memory 2:** A window showing the memory contents starting at address 0x0000008C. The memory is organized into two sections: **Memory 1** (Address: 0x00000078) and **Memory 2** (Address: 0x0000008C). The memory contents are displayed in hexadecimal and ASCII format.