

# **EC-210**

# **MICROPROCESSORS LAB**

## **LAB-3**



**UTKARSH MAHAJAN 201EC164**

**ARNAV RAJ 201EC109**

Objective: To demonstrate the use of load and Store instructions and their addressing.

**Exercise:**

**3.2]** Write an assembly program to add two numbers:  $C = a + b$  where

- (a) a and b are both unsigned 32 bit numbers
- (b) a and b are both signed 32 bit numbers
- (c) a and b are both unsigned 64 bit numbers
- (d) a and b are both signed 64 bit numbers

Load a and b from memory and store the results including carry into new location.

->

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

We will take values from the memory and store it in R2 and R3. Then store the Calculated result in R0 and R4. This result is later stored in memory positioned after the input value memory locations.

Source Code:

```
AREA qthree, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
      MOV R0, #0 ; R0 holds the result
Loop LDR R2, [R1], #4 ; Load the first number & advance ptr
      LDR R3, [R1], #4 ; Load the second number & advance ptr
      ADDS R0, R2, R3 ; Add the numbers
      ADC R4, #0;
      STR R0, [R1], #4 ; store the result in the next memory
      STRB R4, [R1], #4 ; store the carry result in the next memory
Stop B Stop
END
```

## Debugging:

## Initial Memory:

The screenshot shows the 'Memory 2' window with the following details:

- Address:** 0x40000000
- Content:** A memory dump starting at address 0x40000000, showing 16 bytes per row. The first few rows are:
  - 0x40000000: 4D 3F FF 47 5D 7C 2B EF 00 00 00 00 00 00 00 00
  - 0x40000022: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  - 0x40000044: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  - 0x40000066: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  - 0x40000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
- Bottom Navigation:** Call Stack + Locals, Memory 2
- Bottom Status:** Real-Time Agent: Target Reset, Simulation, t1: 0.000000000 sec, L4 C:1, CAP NUM SCRL OVR R/W

## Setup:

The screenshot shows the iVision4 development environment for Microprocessors. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help. The Registers window on the left lists CPU registers (R0-R15, SP, PC) with their current values. The Disassembly window in the center shows assembly code for a reset handler, including instructions like MOV R1, #0x40000000 and LDR R2, [R1], #4. The Memory window at the bottom displays a memory dump starting at address 0x40000000, showing binary data. The Command window at the bottom left provides build information and command-line tools.

## After Execution:

## Final Output:

## Final Register Values:

Register	Value
Current	
R0	0x372ABBA
R1	0x40000010
R2	0x47FF3F4D
R3	0xEF2B7C5D
R4	0x00000001
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
+ CPSR	0x200000D3
+ SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000020
Mode	Supervisor
States	20
Sec	0.00000033

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

We will take values from the memory and store it in R2 and R3. While calculating addition in signed numbers, we know that we need to consider N(Negative) as the additional bit if overflow V=0 otherwise C(carry) for the **additional bit**. We will handle this by appending **conditional execution code VC**. Then store the Calculated result in R0 and R4. This result is later **stored in memory** locations positioned after the input value memory locations.

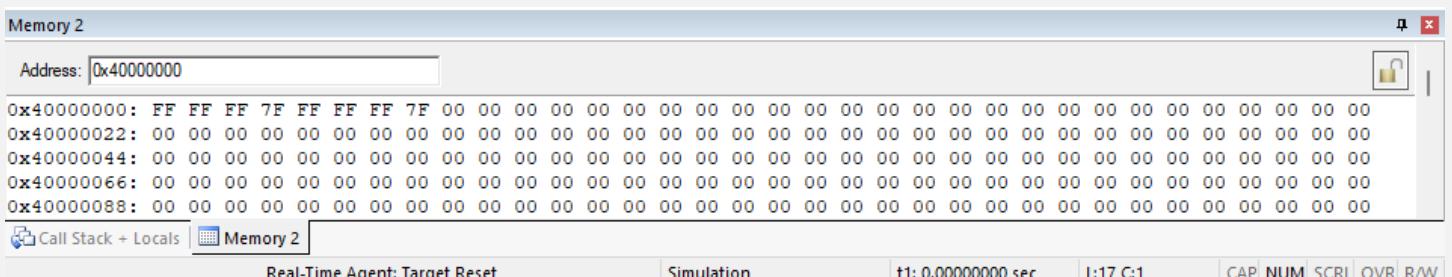
## Source code

```
AREA qthree, CODE, READONLY
EXPORT Reset_Handler

Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
        MOV R0, #0 ; R0 holds the result
Loop  LDR R2, [R1], #4 ; Load the first number -1 & advance ptr
      LDR R3, [R1], #4 ; Load the second number -2 & advance ptr
      ADDS R0, R2, R3 ; Add the numbers
      MRSVC R5, CPSR; using N as additional bit if V==0
      LSLSVC R5, #1; otherwise set C as the additional bit
      ADC R4, #0;The additional bit.
      STR R0, [R1], #4 ; store the result in the next memory
      STR R4, [R1], #1 ; store the carry result in the next memory
Stop B Stop
END
```

## Debugging:

## Initial Memory:



## Setup:

The screenshot shows a software interface for a Microprocessor LAB. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tool, SVCS, Window, Help. The left sidebar lists Registers (Registers, Current, R0-R15, CPSR, User-System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, Internal), Stack (PC \$, Mode, Supervisor, 0, States, 0, Sec, 0), and Command (Restricted Version with 32768 Byte Code Size Limit, Currently used: 44 Bytes (0\$)). The main window has tabs for Registers, Disassembly, and Memory. The Registers tab shows current register values. The Disassembly tab displays assembly code for a program, including a Reset Handler and a loop that adds two memory locations. The Memory tab shows memory starting at address 0x40000000 with various data patterns.

## After Execution:

The screenshot shows the µVision4 IDE interface with the following details:

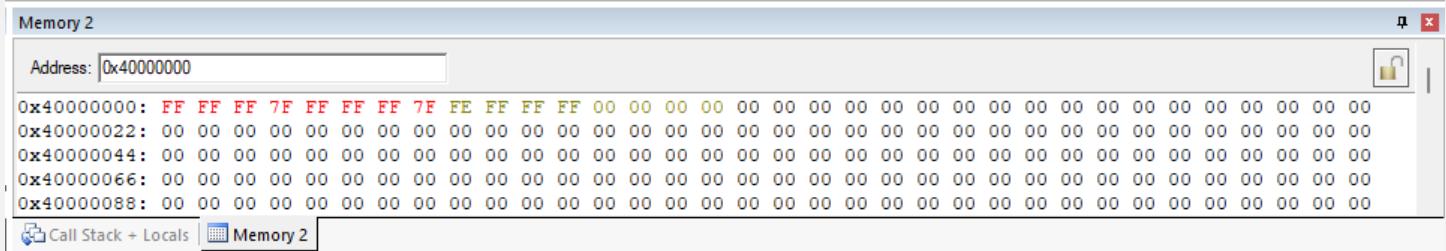
- Title Bar:** DA\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\uvproj - µVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tool, SVCS, Window, Help
- Toolbars:** Standard toolbar with icons for Open, Save, Run, Stop, Break, etc.
- Registers Window:** Shows the current register values:
  - Current: R0 = 0xFFFFFFF, R1 = 0x4000000D, R2 = 0xFFFFFFFF, R3 = 0x00000000, R4 = 0x00000000, R5 = 0x00000000, R6 = 0x00000000, R7 = 0x00000000, R8 = 0x00000000, R9 = 0x00000000, R10 = 0x00000000, R11 = 0x00000000, R12 = 0x00000000, R13 (SP) = 0x00000000, R14 (LR) = 0x00000000, R15 (PC) = 0x00000028
  - CPSR: CPSR = 0x00000003
  - Interrupts: User-System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, Internal
    - PC \$ Mode: Supervisor
    - States: 19
    - Sec: 0.00000032
- Disassembly Window:** Shows the assembly code for the signed32.s file:

```
0x00000000 E4810004 STR R0,[R1],#0x0004
14: 0x00000024 E4814001 STR R4,[R1],#0x0001
15: 0x00000028 EAFFFFE B 0x00000028
0x0000002C 00000000 ANDEQ R0,R0,R0

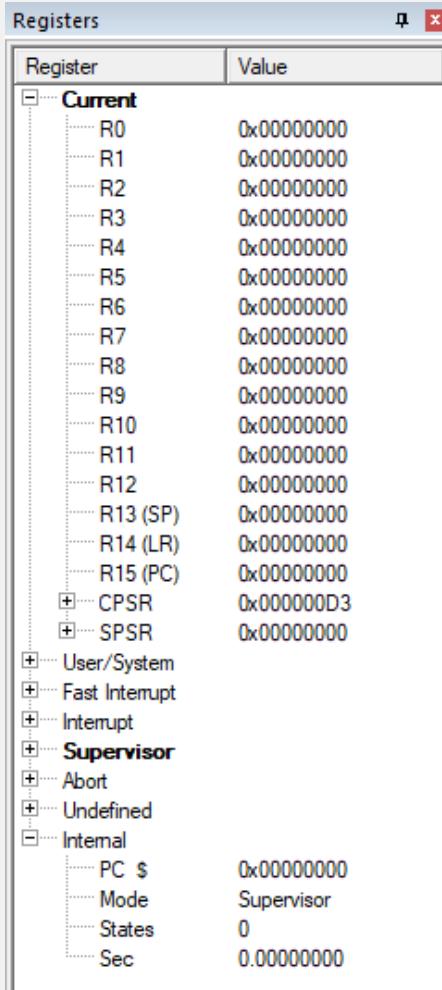
signed32.s*
1 AREA gthreetwo, CODE, READONLY
2 EXPORT Reset_Handler
3 Reset_Handler
4 SStart MOV R1, #0x40000000; Address of the memory where the
5 number is present
6 MOV R0, #0 : R0 holds the result
7 Loop LDR R2, [R1], #4 ; Load the first number -1 & advance ptr
8 LDR R3, [R1], #4 ; Load the second number -2 & advance ptr
9 ADDS R0, R2, R3 ; Add the numbers
10 MRSV R5, CPSR; using N as additional bit if V==0
11 LSLSVC R5, #1; otherwise set C as the additional bit
12 ADD R0, R5, #The additional bit.
13 STR R0, [R1], #4 ; store the result in the next memory
14 STR R4, [R1], #4 ; store the carry result in the next memory
15 Stop B Stop
16 END
17
```
- Memory Window:** Shows the memory dump starting at address 0x40000000. The memory is filled with zeros, except for the first four bytes which are FF FF FF FF.
- Command Window:** Displays compiler messages:

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 44 Bytes (0%)
d>
```
- Status Bar:** Real-Time Agent: Target Stopped, Simulation, t!: 0.00000032 sec, L: 15 C: 1, CAP NUM SCR OVR/RW, ENG US, 04:10 AM, 25-02-2022

## Final memory Output:



# Final Register Values:



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

We will take values from the memory and store it in R2, R3, R4 and R5. Then store the Calculated result in R6, R7 and R8. This result is later stored in memory positioned after the input value memory locations.

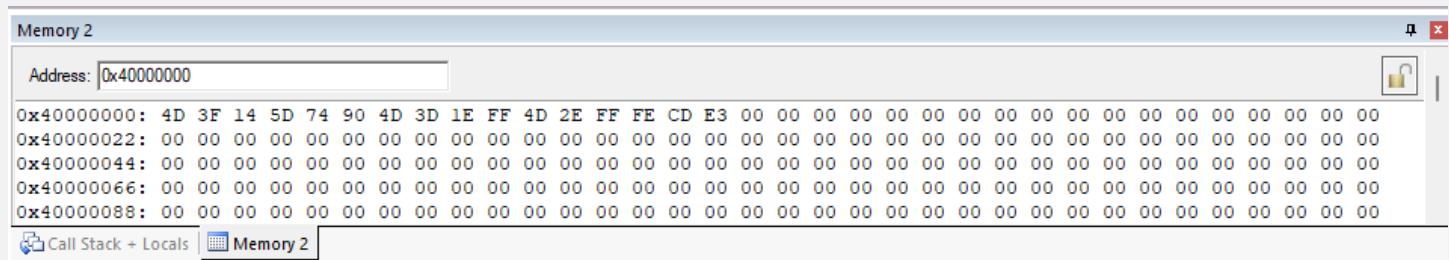
## Source code:

```
AREA qthreetwo, CODE, READONLY
EXPORT Reset_Handler

Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R6, #0 ; R4 and R5, R6 to hold the result
    MOV R7, #0 ;
    MOV R8, #0 ;
Loop LDR R2, [R1], #4 ; Load the first number's low & advance ptr
    LDR R3, [R1], #4 ; Load the first number's high & advance ptr
    LDR R4, [R1], #4 ; Load the second number's low & advance ptr
    LDR R5, [R1], #4 ; Load the second number's high & advance ptr
    ADDS R6, R4, R2 ; Add the numbers low
    ADCS R7, R5, R3 ; Adding high of both numbers with carry from low add.
    ADC R8, #0;The additional bit.
    STR R6, [R1], #4 ; store the Low result in the next memory
    STR R7, [R1], #4 ; store the High result in the next memory
    STR R8, [R1], #4 ; storing the carry in the next memory
    ; We have stored the result in 96bits(64 +32)
Stop B Stop
END
```

## Debugging:

## Initial Memory:



## Setup:

## After Execution:

## Final Output:

The screenshot shows the Keil uVision IDE interface with the following details:

- Title Bar:** D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - uVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Registers Window:** Shows CPU registers R0-R15, PC, and various flags. The PC register value is 0x00000034.
- Disassembly Window:** Displays assembly code for a 64-bit addition program. The code includes a Reset Handler, memory copy operations, and a loop for adding two 64-bit numbers. It ends with a Stop B instruction at address 0x0000003B.
- Memory Window:** Shows memory dump starting at address 0x40000000, displaying hex values for both low and high bytes of each word.
- Command Window:** Displays compiler messages and command history.
- Status Bar:** Real-Time Agent: Target Stopped, Simulation, 1t: 0.00000047 sec, L20 C1, CAP NUM SCRLL QVR IN, ENG US, 04:06 AM, 25-02-2022.

## Final Register Values:

Register	Value
Current	
R0	0x00000000
R1	0x4000001C
R2	0x5D143F4D
R3	0x3D4D9074
R4	0x2E4DFF1E
R5	0xE3CDFEFF
R6	0x8B623E6B
R7	0x211B8F73
R8	0x00000001
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000038
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000038
Mode	Supervisor
States	28
Sec	0.00000047

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

We will take values from the memory and store it in R2 and R3. While calculating signed numbers, we know that we need to consider N(Negative) as the additional bit if overflow V=0 otherwise C(carry) for the **additional** bit. We will handle this by appending **conditional execution** code VC. Then store the Calculated result in R6, R7 and R8. This result is later **stored in memory**

locations positioned after the input value memory locations. **65<sup>th</sup> bit will be sign**

## Source code

```

AREA qthreetwo, CODE, READONLY
EXPORT Reset_Handler

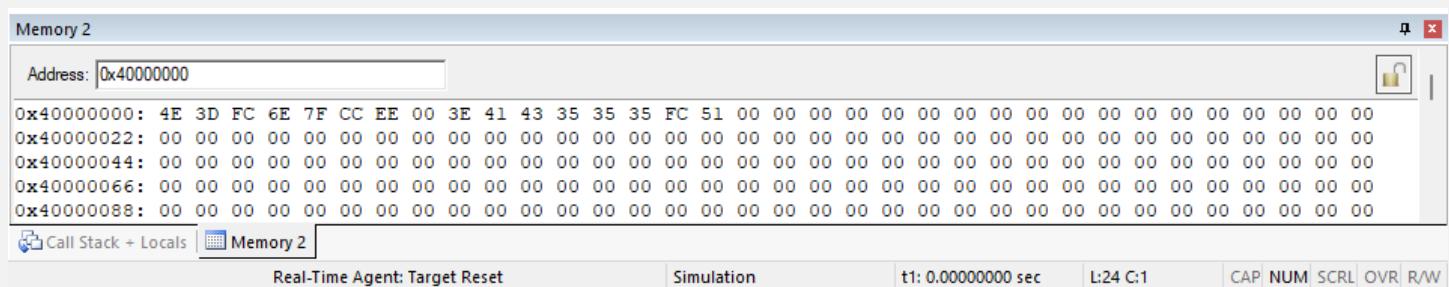
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R6, #0 ; R4 and R5, R6 to hold the result
    MOV R7, #0 ;
    MOV R8, #0 ;
Loop LDR R2, [R1], #4 ; Load the first number's low & advance ptr
    LDR R3, [R1], #4 ; Load the first number's high & advance ptr
    LDR R4, [R1], #4 ; Load the second number's low & advance ptr
    LDR R5, [R1], #4 ; Load the second number's high & advance ptr
    ADDS R6, R4, R2 ; Add the numbers low
    ADCS R7, R5, R3 ; Adding high of both numbers with carry from low add.
    MRSVC R9, CPSR; using N as additional bit if V==0
    LSLSVC R9, #1; otherwise set C as the additional bit
    ADC R8, #0;The additional bit.
    STR R6, [R1], #4 ; store the Low result in the next memory
    STR R7, [R1], #4 ; store the High result in the next memory
    STR R8, [R1], #4 ; storing the carry in the next memory
                           ; We have stored the 65 bit result in 96bits(64 +32).
; 65th bit will be sign.

Stop B Stop
END

```

## Debugging:

## Initial Memory:



## Setup:

The screenshot shows a software interface for a microcontroller project. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tool, SVCS, Window, Help. The left sidebar lists registers (R0-R15, CPSR, CPSR), stack information (PC \$, Mode, Supervisor, States, Sec), and interrupt levels (User System, Fast Interrupt, Interrupt, Supervisor, Abort, Undefined, Internal). The main window has tabs for Registers (selected) and Disassembly. The Registers tab shows current values for R0-R15 and CPSR. The Disassembly tab displays assembly code for a multiplication routine, including comments and assembly mnemonics. The bottom section includes a Command window with build logs, a Memory dump window showing memory starting at address 0x40000000, and a status bar with real-time agent information.

## After Execution:

## Final Output:

The screenshot shows the 'Memory 2' window with the following details:

- Address:** 0x40000000
- Content:** A memory dump starting at address 0x40000000. The first few bytes are highlighted in red: 4E 3D FC 6E 7F CC EE 00 3E 41 43 35 35 35 FC 51 8C 7E 3F A4 B4 01 EB 52. The rest of the memory is filled with zeros.
- Buttons:** Call Stack + Locals, Memory 2 (highlighted), and a lock icon.
- Status Bar:** Real-Time Agent: Target Reset, Simulation, t1: 0.00000055 sec, L2: C1, CAP NUM SCRL OVR R/W.

## Final Register Values:

Register	Value
Current	
R0	0x00000000
R1	0x4000001C
R2	0x6EFC3D4E
R3	0x00EECC7F
R4	0x3543413E
R5	0x51FC3535
R6	0xA43F7E8C
R7	0x52EB01B4
R8	0x00000000
R9	0x000001A6
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
+ CPSR	0x000000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
Internal	
PC \$	0x00000040
Mode	Supervisor
States	33
Sec	0.00000055

3.3] Write an assembly program to perform  $C = a - b$  where

- (a) a and b are both unsigned 32 bit numbers
- (b) a and b are both signed 32 bit numbers
- (c) a and b are both unsigned 64 bit numbers

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

->

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

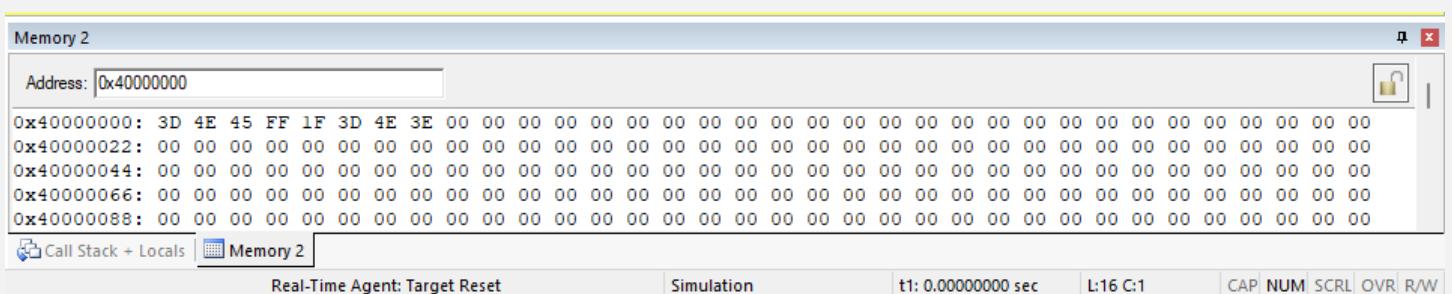
We will take values from the memory and store it in R2 and R3. Then store the Calculated result in R0 and R4. This result is later stored in memory positioned after the input value memory locations.

Source code:

```
AREA qthreeetwo, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R0, #0 ; R0 holds the result
Loop LDR R2, [R1], #4 ; Load the first number & advance ptr
    LDR R3, [R1], #4 ; Load the second number & advance ptr
    SUBS R0, R2, R3 ; Subtract the numbers
    SBC R4, #0;
    STR R0, [R1], #4 ; store the result in the next memory
    STRB R4, [R1], #4 ; store the carry result in the next memory
Stop B Stop
END
```

Debugging:

Initial Memory:



# Setup:

The screenshot shows the uVision IDE interface with the following details:

- Registers** window: Shows the current register values. The PC is at \$0x00000004.
- Disassembly** window:

```
4: Start MOV R1, #0x40000000; Address of the memory where the
5:                                     ; number is present
→ 0x00000000 E3A01101 MOV      R1,#0x40000000
6:                                     ; R0 holds the result
0x00000004 E3A01100 MOV      R0,#0x00000000
7: Loop LDR R2, [R1], #4 ; Load the first number & advance ptr
   ...
1 AREA qihretwo, CODE, READONLY
2 EXPORT Reset_Handler
3 Reset_Handler
4 Start MOV R1, #0x40000000; Address of the memory where the
5:                                     ; number is present
6:                                     ; R0 holds the result
7: Loop LDR R2, [R1], #4 ; Load the first number & advance ptr
8: LDR R3, [R1], #4 ; Load the second number & advance ptr
9: SUBS R0, R2, R3 ; Subtract the numbers
10: SBC R4, #0;
11: STR R0, [R1], #4 ; store the result in the next memory
12: STRB R4, [R1], #4 ; store the carry result in the next memory
13 Stop B Stop
14 END
15
16
```
- Memory** window: Shows memory starting at address 0x40000000 filled with zeros.
- Command** window: Displays assembly code for the subtraction logic.
- System Bar**: Shows the date (25-02-2022), time (04:32 AM), and user (ENG US).

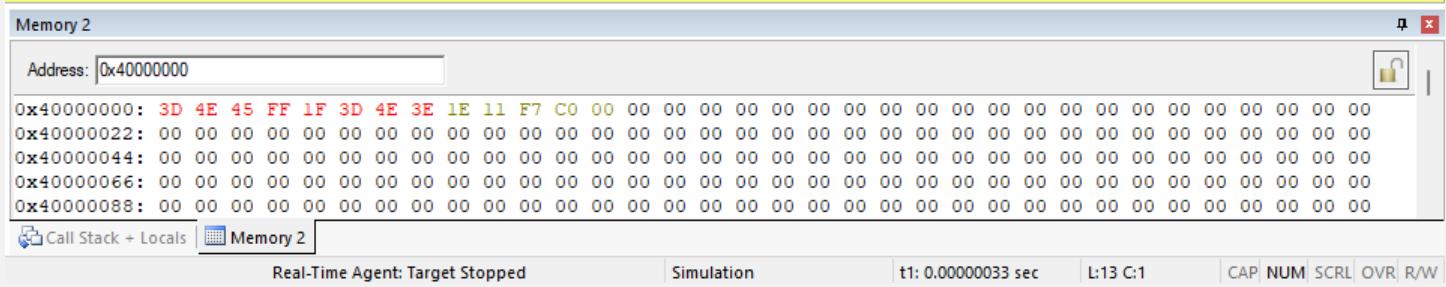
## After Execution:

### Final Output:

The screenshot shows the uVision IDE interface after execution, with the following details:

- Registers** window: Shows the final register values. The PC is at \$0x00000004.
- Disassembly** window:

```
0x00000000 E4010004 STR     R0,[R1],#0x0004
1:                                     ; store the carry result in the next memory
2: Stop B Stop
3: 
4: Start MOV R1, #0x40000000; Address of the memory where the
5:                                     ; number is present
6:                                     ; R0 holds the result
7: Loop LDR R2, [R1], #4 ; Load the first number & advance ptr
8: LDR R3, [R1], #4 ; Load the second number & advance ptr
9: SUBS R0, R2, R3 ; Subtract the numbers
10: SBC R4, #0;
11: STR R0, [R1], #4 ; store the result in the next memory
12: STRB R4, [R1], #4 ; store the carry result in the next memory
13 Stop B Stop
14 END
15
16
```
- Memory** window: Shows memory starting at address 0x40000000 filled with zeros.
- Command** window: Displays assembly code for the subtraction logic.
- System Bar**: Shows the date (25-02-2022), time (04:32 AM), and user (ENG US).



## Final Register Values:

Register	Value
R0	0xC0F7111E
R1	0x40000010
R2	0xFF454E3D
R3	0x3E4E3D1F
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000020
+ CPSR	0xA00000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Intemupt	
+ Supervisor	
+ Abort	
+ Undefined	
+ Internal	
PC \$	0x00000020
Mode	Supervisor
States	20
Sec	0.00000033

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

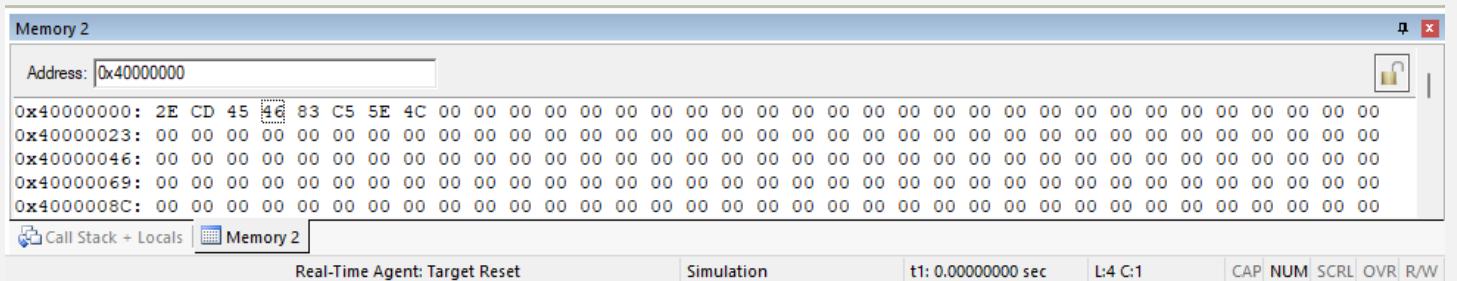
We will take values from the memory and store it in R2 and R3. While calculating subtraction in signed numbers, we know that we need to consider N(Negative) as the additional bit if overflow V=0 otherwise C(carry) for the additional bit. We will handle this by appending **conditional execution** code VC. Then store the Calculated result in R0 and R4. This result is later **stored in memory** locations positioned after the input value memory locations.

Source code:

```
AREA qthreeetwo, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R0, #0 ; R0 holds the result
Loop LDR R2, [R1], #4 ; Load the first number -1 & advance ptr
    LDR R3, [R1], #4 ; Load the second number -2 & advance ptr
    SUBS R0, R2, R3 ; Add the numbers
    MRSVC R5, CPSR; using N as additional bit if V==0
    LSLSVC R5, #1; otherwise set C as the additional bit
    ADC R4, #0;The additional bit.
    STR R0, [R1], #4 ; store the result in the next memory
    STR R4, [R1], #1 ; store the carry result in the next memory
Stop B Stop
END
```

## Debugging:

Initial Memory:



# Setup:

The screenshot shows the uVision IDE interface with the following details:

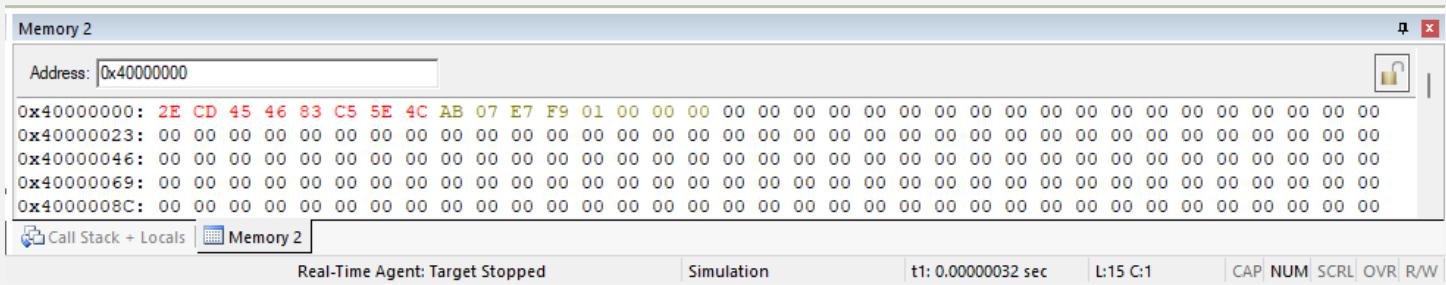
- Registers** window: Shows CPU registers R0-R15, CPSR, and System registers. PC is at \$0x00000000.
- Disassembly** window: Displays assembly code for a 32-bit addition program. The code includes instructions for moving numbers to memory, loading them back, and performing additions. It ends with a stop instruction.
- Memory** window: Shows memory starting at address 0x40000000 with all zeros.
- Command** window: Shows build logs indicating a restricted version with a 32768 byte code size limit and current usage of 44 bytes.
- System Bar**: Shows the date (25-02-2022), time (04:42 AM), and connection status (ENG US).

## After Execution:

### Final Output:

The screenshot shows the uVision IDE interface after execution, with the following details:

- Registers** window: Same state as before, with PC at \$0x00000000.
- Disassembly** window: Same assembly code as the setup screen.
- Memory** window: Shows memory starting at address 0x40000000 with the result of the addition (0x00000000).
- Command** window: Shows build logs indicating a restricted version with a 32768 byte code size limit and current usage of 44 bytes.
- System Bar**: Shows the date (25-02-2022), time (04:42 AM), and connection status (ENG US).



## Final Register Values:

Register	Value
R0	0xF9E707AB
R1	0x4000000D
R2	0x4645CD2E
R3	0x4C5EC583
R4	0x00000001
R5	0x000001A6
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000028
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
Internal	
PC \$	0x00000028
Mode	Supervisor
States	19
Sec	0.00000032

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

We will take values from the memory and store it in R2 and R3. While calculating subtraction of signed numbers, we know that we need to consider N(Negative) as the additional bit if overflow V=0 otherwise C(carry) for the additional bit. We will handle this by appending **conditional execution** code VC. Then store the Calculated result in R6, R7 and R8. This result is later

**stored in memory** locations positioned after the input value memory locations. **65<sup>th</sup> bit will be sign**

## Source code:

```

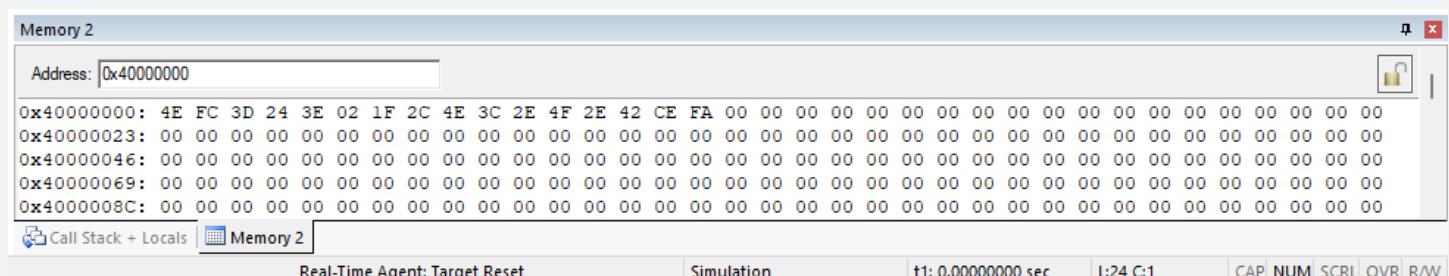
AREA qthree, CODE, READONLY
EXPORT Reset_Handler

Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R6, #0 ; R6 and R7, R8 to hold the result
    MOV R7, #0 ;
    MOV R8, #0 ;
Loop LDR R2, [R1], #4 ; Load the first number's low & advance ptr
    LDR R3, [R1], #4 ; Load the first number's high & advance ptr
    LDR R4, [R1], #4 ; Load the second number's low & advance ptr
    LDR R5, [R1], #4 ; Load the second number's high & advance ptr
    SUBS R6, R4, R2 ; Add the numbers low
    SBCS R7, R5, R3 ; Adding high of both numbers with carry from low add.
    MRSVC R9, CPSR; using N as additional bit if V==0
    LSLSVC R9, #1; otherwise set C as the additional bit
    ADC R8, #0;
    STR R6, [R1], #4 ; store the Low result in the next memory
    STR R7, [R1], #4 ; store the High result in the next memory
    STR R8, [R1], #4 ; storing the carry in the next memory
    ; We have stored the 65 bits result in memory of 96bits(64 +32)
Stop B Stop
END

```

## Debugging:

## Initial Memory:



## Setup:

The screenshot shows a debugger interface with several windows:

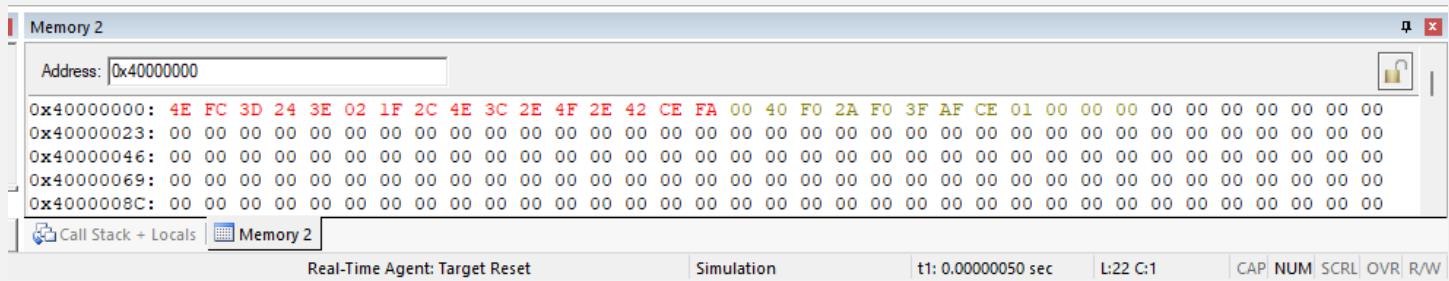
- Registers**: Shows CPU registers (R0-R15, PC, CPSR, SPSR) with their current values.
- Disassembly**: Displays assembly code for the signed64.s file. The highlighted instruction is `MOV R1, #0x40000000`. The assembly code includes comments explaining the purpose of each instruction, such as loading numbers from memory and performing a 64-bit addition.
- signed64.s**: The source code for the assembly file, showing the assembly mnemonics and comments.
- Memory**: A dump of memory starting at address 0x40000000, showing the result of the addition.
- Command**: A terminal window displaying build logs and command-line options.

## After Execution:

## Final Output:

The screenshot shows a debugger interface with the following details:

- Title Bar:** D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3\uvproj - μVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripheral, Tools, SVCS, Window, Help
- Registers Tab:** Shows CPU registers (R0-R7, R9-R12, R13-PC, CPSR) and their current values.
- Disassembly Tab:** Displays assembly code for the signed8x8.s file. The code performs a 16-bit multiplication of two 8-bit numbers (R0 and R1) and stores the result (64 bits) in memory at address R2. The assembly code includes comments explaining the operations like loading numbers, performing the multiplication, and storing the result.
- Memory Tab:** Shows a dump of memory starting at address 0x40000000. The dump area is labeled "Memory 2".
- Command Line:** Shows compiler messages indicating a restricted version and usage statistics.
- Bottom Status Bar:** Real-Time Agent: Target Reset, Simulation, t1: 0.0000050 sec, L122 C1, CAP NUM SCR1 DVI R/W, and a timestamp: 04:53 AM 25-02-2022.



## Final Register Values:

Register	Value
Current	
R0	0x00000000
R1	0x4000001C
R2	0x243DFC4E
R3	0x2C1F023E
R4	0x4F2E3C4E
R5	0xFACE422E
R6	0x2AF04000
R7	0xCEAF3FF0
R8	0x00000001
R9	0x400001A6
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000040
Mode	Supervisor
States	30
Sec	0.00000050

**3.4] If a and b are two 32 bit numbers stored in consecutive memory location, swap the register content using EOR instruction and store back the results.**

The EOR does bitwise XOR, this can be used to swap two variables. The XOR of two numbers x and y returns a number that has all the bits as 1 wherever bits of x and y differ.

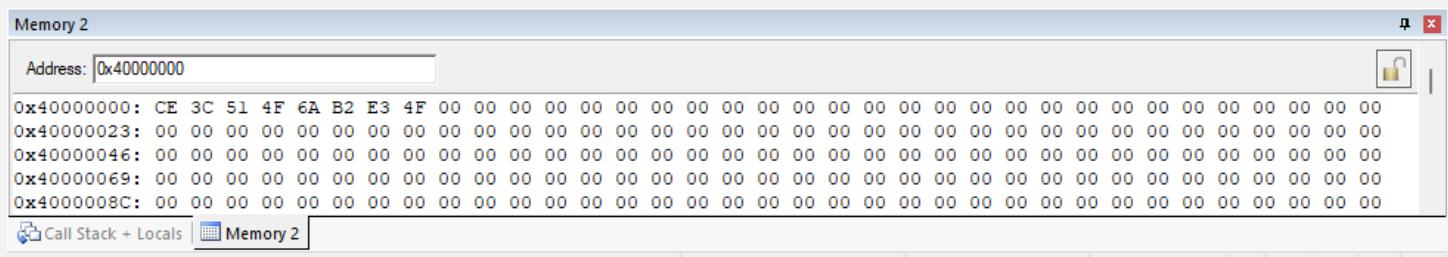
## Source code:

```
AREA hmm, CODE, READONLY
EXPORT Reset_Handler

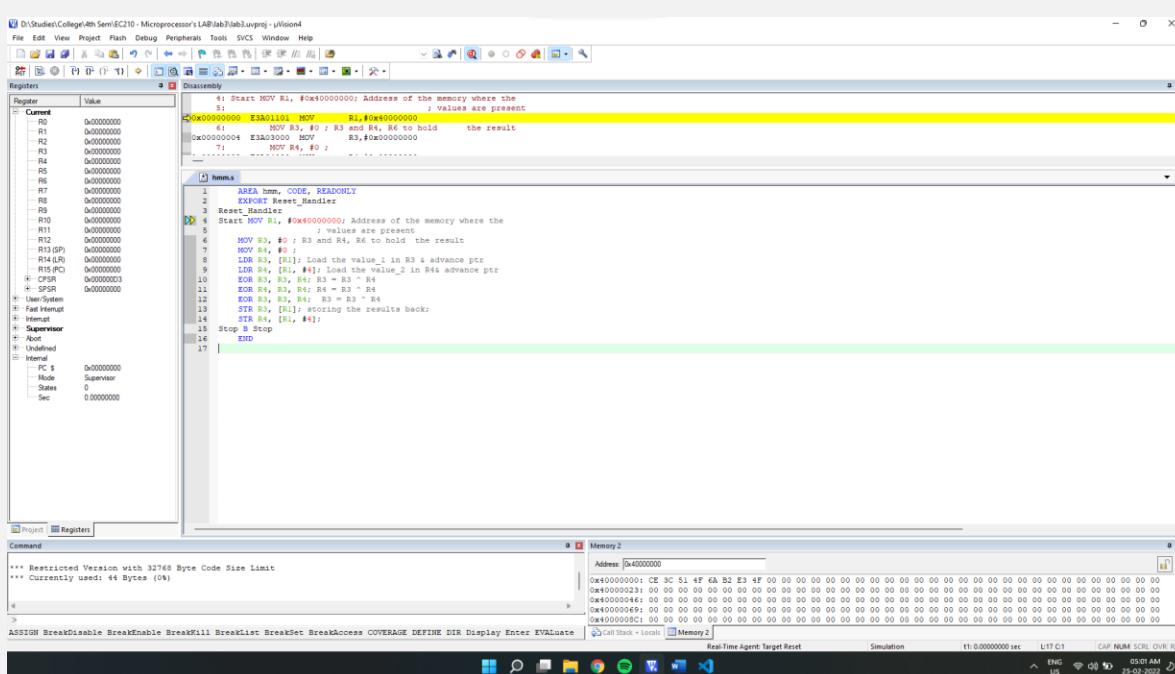
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; values are present
    MOV R3, #0 ; R3 and R4, R6 to hold the result
    MOV R4, #0 ;
    LDR R3, [R1]; Load the value_1 in R3 & advance ptr
    LDR R4, [R1, #4]; Load the value_2 in R4& advance ptr
    EOR R3, R3, R4; R3 = R3 ^ R4
    EOR R4, R3, R4; R4 = R3 ^ R4
    EOR R3, R3, R4; R3 = R3 ^ R4
    STR R3, [R1]; storing the results back;
    STR R4, [R1, #4];
Stop B Stop
END
```

## Debugging:

### Initial Memory:



## Setup:



## After Execution:

## Final Output:

The screenshot shows the uVision IDE interface with the following details:

- Title Bar:** D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - uVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Registers Window:** Shows the current register values for R0-R15 and CPSR.
- Disassembly Window:** Displays the assembly code for the Reset Handler. The instruction at address 0x00000028 is highlighted in yellow: `0x00000028 EAFFFFFF B 0x00000028`.
- Code Window:** Shows the source code for the file `hmm.s`. The assembly code corresponds to the following C-like pseudocode:

```
AREA hmm, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
    Start: MOV R1, #0x40000000; Address of the memory where the
           ; values are present
    1: MOV R3, #0 ; R3 and R4, R6 to hold the result
    2: MOV R4, #0 ;
    3: LDR R3, [R1]; Load the value_1 in R3 & advance ptr
    4: LDR R4, [R1, #4]; Load the value_2 in R4 & advance ptr
    5: EOR R3, R3, R4; R3 = R3 ^ R4
    6: MOV R6, #1 ;
    7: STR R3, [R1]; storing the results back;
    8: STR R4, [R1, #8];
    9: Stop B Stop
   10: END
```
- Memory Window:** Shows the memory dump starting at address 0x40000000. The first few bytes are: 6A B2 E3 4F CE 3C S1 4F.
- Command Window:** Displays compiler messages:

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 44 Bytes (0%)
```
- Bottom Status Bar:** Real-Time Agent: Target Reset, Simulation, t1: 000000027 sec, L16 C1, CAP NUM SCR LVR RW, ENG US, 05/03 AM, 25-02-2022.

## Final Memory:

Observation: we can see that the memory values are swapped for addresses 0x40000000 and 0x40000004.

**3.5] SWP to add a constant value 0xFF to a block of 10 words stored in the consecutive memory locations using LDM and STM instructions.**

->

We will make use of LDMIA to load multiple registers and then use ADD instruction to add the required constant 0xFF. Then store all those values with STMIA.

## Source code:

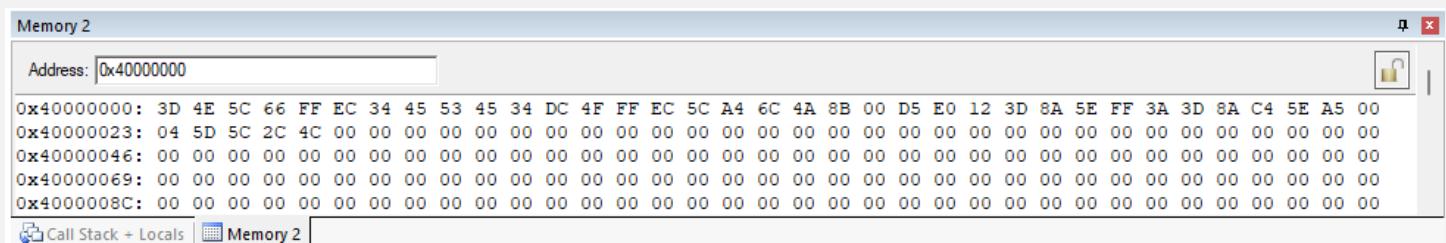
```
AREA hue, CODE, READONLY
EXPORT Reset_Handler

Reset_Handler
Start MOV R0, #0x40000000; Address of the memory where the
                           ; values is present
        LDMIA R0, {R1-R10}; loading the 10 values into register
        ADD R1, #0xFF;
        ADD R2, #0xFF;
        ADD R3, #0xFF;
        ADD R4, #0xFF;
        ADD R5, #0xFF;
        ADD R6, #0xFF;
        ADD R7, #0xFF;
        ADD R8, #0xFF;
        ADD R9, #0xFF;
        ADD R10, #0xFF;
        STMIA R0, {R1-R10}; Storing all the values into memory;

Stop B Stop
END
```

## Debugging:

## Initial Memory:



## Setup:

**Registers**

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x00000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

**Disassembly**

```

4: Start MOV R0, #0x40000000; Address of the memory where the
5: ; values is present
6: E3A00101 MOV R0,#0x40000000
7: E89007FF LDmia R0, (R1-R10); loading the 10 values into register
8: E99007FF ADD R1, #0xFF;
9: E99007FF ADD R2, #0xFF;
10: E99007FF ADD R3, #0xFF;
11: E99007FF ADD R4, #0xFF;
12: E99007FF ADD R5, #0xFF;
13: E99007FF ADD R6, #0xFF;
14: E99007FF ADD R7, #0xFF;
15: E99007FF ADD R8, #0xFF;
16: E99007FF ADD R9, #0xFF;
17: E99007FF ADD R10, #0xFF;
18: STmia R0, (R1-R10); Storing all the values into memory;
19: Stop B Stop
20: END

```

**Memory 2**

Address: 0x40000000

```

0x40000000: 3D 4E 5C 66 FF EC 34 45 53 45 34 DC 4F FF EC 5C A4 6C 4A 8B 00 D5 E0 12 3D 8A SE FF 3D 3D 8A C4 5E A5 00
0x40000023: 04 SD SC 2C 4C 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 00 00 00 00 00
0x40000046: 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 00 00 00 00 00 00 00 00 00 00
0x40000069: 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 00 00 00 00 00 00 00 00 00 00
0x4000008C: 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 00 00 00 00 00 00 00 00 00 00

```

Real-Time Agent: Target Reset Simulation t1: 0.0000000 sec L18 C12 CAP NUM SCR LVR R/W ENG US 0516 AM 25-02-2022

## After Registers get loaded:

**Registers**

Register	Value
R0	0x40000000
R1	0x466C4C30
R2	0x4534ECFF
R3	0xDC344553
R4	0x5C1CF4F
R5	0x8BA6C4A
R6	0x12E00000
R7	0xF0F0F0F0
R8	0xC40A303A
R9	0x0400405E
R10	0x4C2C5C5D
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000008
CPSR	0x00000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000008
Mode	Supervisor
States	13
Sec	0.00000022

**Disassembly**

```

7: ADD R1, #0xFF;
8: E280000008 E28100FF ADD R1,R1,$0x000000FF
9: E28000000C E28220FF ADD R2,R2,$0x000000FF
10: E28000010 E28330FF ADD R3,R3,$0x000000FF
11: E28000018 E28440FF ADD R4,R4,$0x000000FF
12: E2800001C E28550FF ADD R5,R5,$0x000000FF
13: E28000020 E28660FF ADD R6,R6,$0x000000FF
14: E28000024 E28770FF ADD R7,R7,$0x000000FF
15: E28000028 E28880FF ADD R8,R8,$0x000000FF
16: E2800002C E28990FF ADD R9,R9,$0x000000FF
17: E28000030 E28A00FF ADD R10,R10,$0x000000FF
18: STmia R0, (R1-R10); Storing all the values into memory;
19: Stop B Stop
20: END

```

**Memory 2**

Address: 0x40000000

```

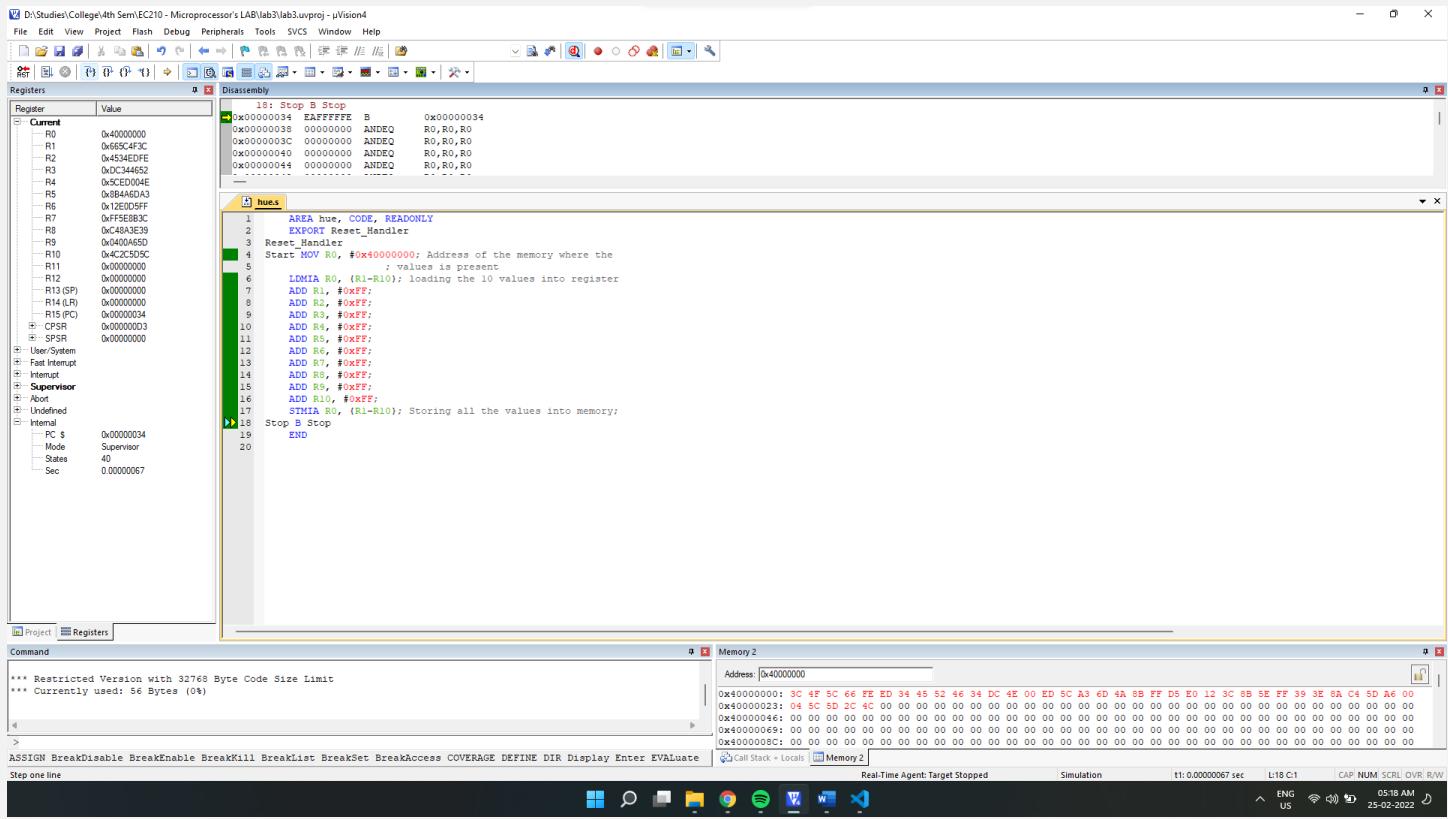
0x40000000: 3D 4E 5C 66 FF EC 34 45 53 45 34 DC 4F FF EC 5C A4 6C 4A 8B 00 D5 E0 12 3D 8A SE FF 3D 3D 8A C4 5E A5 00
0x40000023: 04 SD SC 2C 4C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 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 00 00 00 00 00 00 00 00 00 00
0x4000008C: 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 00 00 00 00 00 00 00 00 00 00

```

Real-Time Agent: Target Reset Simulation t1: 0.00000022 sec L18 C1 CAP NUM SCR LVR R/W ENG US 0517 AM 25-02-2022

Register	Value
R0	0x40000000
R1	0x665C4E3D
R2	0x4534ECFF
R3	0xDC344553
R4	0xCECFFF4F
R5	0x8B4A6CA4
R6	0x12E0D500
R7	0xFF5E8A3D
R8	0xC48A3D3A
R9	0x0400A55E
R10	0x4C2C5C5D
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000008
+ CPSR	0x000000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
+ Internal	
PC \$	0x00000008
Mode	Supervisor
States	13
Sec	0.00000022

## After complete Execution:



## Final Output:

Register	Value
R0	0x40000000
R1	0x665C4F3C
R2	0x4534EDFE
R3	0xDC344652
R4	0x5CED004E
R5	0x8B4A6DA3
R6	0x12E0D5FF
R7	0xFF5E8B3C
R8	0xC48A3E39
R9	0x0400A65D
R10	0x4C2C5D5C
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000034
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
Internal	
PC \$	0x00000034
Mode	Supervisor
States	40
Sec	0.00000067

### 3.6] Exchange a block of 32 bit data stored in the processor registers (R0- R7) with that in the memory starting at address 0x40000000.

We will first load values into registers r0-r7 (from different memory location 0x50000000) that are to be swapped from the memory at location starting from 0x40000000 using LDMIA. Then we will make use of **SWP** instruction to swap values between register and memory. For repeating the instruction for different registers, we will use **ADD** instruction to increment the register storing the address.

Source code:

```
AREA huihui, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
Start MOV R9, #0x50000000; for loading values into register
        LDMIA R9, {R0-R7}; Loading values into register that are to be swapped.
        MOV R8, #0x40000000; Address of the memory of which
; The values are to be swapped with the registers.
        SWP R0, R0, [R8];
        ADD R8, #4;
        SWP R1, R1, [R8];
        ADD R8, #4;
        SWP R2, R2, [R8];
        ADD R8, #4;
        SWP R3, R3, [R8];
        ADD R8, #4;
        SWP R4, R4, [R8];
        ADD R8, #4;
        SWP R5, R5, [R8];
        ADD R8, #4;
        SWP R6, R6, [R8];
        ADD R8, #4;
        SWP R7, R7, [R8];
Stop B Stop
END
```

Debugging:

Initial Memory:

# From 0x40000000

Memory 2

Address: 0x40000000

0x40000000: 4C 3E 01 57 3D EF F2 9C 4A 04 8C 3C 4A 66 98 9F EF 3C 4D 3D 5C 2D 45 A5 0B C3 75 15 C3 6A 3D EF 00 00 00
0x40000023: 00
0x40000046: 00
0x40000069: 00
0x4000008C: 00

Call Stack + Locals | Memory 2

# From 0x50000000 (for initializing register)

Memory 2

Address: 0x50000000

0x50000000: 3D 4E 64 20 5A 2F 9C 3D 8C FE AC FF 56 AD 45 0C 6C CD 03 C5 8C 0A A7 48 98 4C 00 3D FE 3E FC FE 00 00 00
0x50000023: 00
0x50000046: 00
0x50000069: 00
0x5000008C: 00

Call Stack + Locals | Memory 2

Real-Time Agent: Target Reset | Simulation | t1: 0.0000000 sec | L:19 C:22 | CAP NUM SCRL OVR R/W

# Setup:

D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x00000003
SPSR	0x00000000

User/System

Fast Interrupt

Interrupt

Supervisor

Not

Undefined

Internal

- PC \$ 0x00000000
- Mode Supervisor 0
- States 0
- Sec 0.00000000

Disassembly

```
Start MOV R0, #0x50000000; for loading values into register
    6: LDMIA R0, {R0-R7}; Loading values into register that are to be swapped.
0x00000004 E9990FF, LDMIA R0, {R0-R7}
    7: MOV R8, #0x40000000; Address of the memory of which
    8: ; The values are to be swapped with the registers.
    ...
4 Start MOV R0, #0x50000000; for loading values into register
    6: LDMIA R0, {R0-R7}; Loading values into register that are to be swapped.
    7: MOV R8, #0x40000000; Address of the memory of which
    8: ; The values are to be swapped with the registers.
    ...
10 LDM R0, {R0-R7}, [R8]
    11 SWP R1, R1, [R8];
    12 ADD R8, #4;
    13 SWP R2, R2, [R8];
    14 ADD R8, #4;
    15 SWP R3, R3, [R8];
    16 ADD R8, #4;
    17 SWP R4, R4, [R8];
    18 ADD R8, #4;
    19 SWP R5, R5, [R8];
    20 ADD R8, #4;
    21 SWP R6, R6, [R8];
    22 ADD R8, #4;
    23 SWP R7, R7, [R8];
    24 Stop ; Stop
    25 END
    26
```

Memory 2

Address: 0x50000000

0x50000000: 3D 4E 64 20 5A 2F 9C 3D 8C FE AC FF 56 AD 45 0C 6C CD 03 C5 8C 0A A7 48 98 4C 00 3D FE 3E FC FE 00 00 00
0x50000023: 00
0x50000046: 00
0x50000069: 00
0x5000008C: 00

Call Stack + Locals | Memory 2

Real-Time Agent: Target Reset | Simulation | t1: 0.0000000 sec | L:19 C:22 | CAP NUM SCRL OVR R/W

ENG US 0548 AM 25-02-2022

After registers get loaded:

Register	Value
<b>Current</b>	
R0	0x20644E3D
R1	0x3D9C2F5A
R2	0xFFACFE8C
R3	0x0C45AD56
R4	0xC503CD6C
R5	0x48A70A8C
R6	0x3D004C98
R7	0xFEFC3EFE
R8	0x00000000
R9	0x50000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000008</b>
+ CPSR	0x000000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
+ Internal	
PC \$	0x00000008
Mode	Supervisor
States	11
Sec	0.00000018

D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
<b>Current</b>	
R0	0x20644E3D
R1	0x3D9C2F5A
R2	0xFFACFE8C
R3	0x0C45AD56
R4	0xC503CD6C
R5	0x48A70A8C
R6	0x3D004C98
R7	0xFEFC3EFE
R8	0x00000000
R9	0x50000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
<b>R15 (PC)</b>	<b>0x00000008</b>
+ CPSR	0x000000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
+ Internal	
PC \$	0x00000008
Mode	Supervisor
States	11
Sec	0.00000018

```

7:      MOV R8, #0x40000000; Address of the memory of which
8: : The values are to be swapped with the registers.
+0x00000008: E3A09101 MOV     R8,#0x40000000
9:      SWP R0, R0, [R8];
0x0000000C: E1080000 SWP     R0,R0,[R8]
10:     ADD R8, #4;
11:     SWP R0, R0, [R8];
12:     ADD R8, #4;
13:     SWP R1, R1, [R8];
14:     ADD R8, #4;
15:     SWP R3, R3, [R8];
16:     ADD R8, #4;
17:     SWP R4, R4, [R8];
18:     ADD R8, #4;
19:     SWP R5, R5, [R8];
20:     ADD R8, #4;
21:     SWP R6, R6, [R8];
22:     ADD R8, #4;
23:     SWP R7, R7, [R8];
24: Stop B Stop
25: END
26

```

Memory 2

Address: 0x50000000

0x5000000001: 3D 4E 64 20 5A 2F 9C 3D BC FE AC FF 56 AD 45 OC 6C CD 03 C5 8C 0A A7 48 98 4C 00 00 00  
0x5000000023: 00  
0x5000000046: 00  
0x5000000069: 00  
0x500000008C: 00

Real-Time Agent: Target Reset Simulation Itt: 0.00000018 sec L7 C1 CAP NUM SCR1 OVR R/W

^ ENG US 0549 AM 25-02-2022

# After Execution:

## Final Output:

The screenshot shows the µVision4 IDE interface with the following windows open:

- Registers**: Shows the current register values.
- Assembly**: Displays the assembly code for the program, which includes a Reset Handler and a main loop for swapping registers R9-R15.
- Memory 2**: Shows a memory dump starting at address 0x40000000.
- Command**: Shows the command line interface with various options like BreakDisable, BreakEnable, BreakKill, etc.

```
0x00000040 E2888004 ADD    R8,R8,#0x00000004
 33:      SWP   R7,R7,[R8]
0x00000044 E1087097 SWP   R7,R7,[R8]
 24: Stop B Stop
0x00000048 EAEFFFFE B     0x00000048
0x0000004C 00000000 ANDEQ R0,R0,R0
...
1 AREA huimui, CODE, READONLY
2 EXPON Reset_Handler
3 Reset_Handler
4
5 Start MOV R9, #0x$00000000; for loading values into register
6 LDMA R9, (R0-R7); Loading values into register that are to be swapped.
7 MOV R9, #0x$00000000; Address of the memory of which
8 : The values are to be swapped with the registers.
9 SWP R9, R9, [R8];
10 ADD R9, #4;
11 SWP R1, R1, [R8];
12 ADD R9, #4;
13 SWP R2, R2, [R8];
14 ADD R9, #4;
15 SWP R3, R3, [R8];
16 ADD R9, #4;
17 SWP R4, R4, [R8];
18 ADD R9, #4;
19 SWP R5, R5, [R8];
20 ADD R9, #4;
21 SWP R6, R6, [R8];
22 ADD R9, #4;
23 SWP R7, R7, [R8];
24 Stop B Stop
25 END
```

## Final Values:

The Registers window displays the final values of all registers after execution:

Register	Value
<b>Current</b>	
R0	0x57013E4C
R1	0x9CF2EF3D
R2	0x3C8C044A
R3	0x9F98664A
R4	0x3D4D3CEF
R5	0xA5452D5C
R6	0x1575C30B
R7	0xEF3D6AC3
R8	0x4000001C
R9	0x50000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000048
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
<b>Internal</b>	
PC \$	0x00000048
Mode	Supervisor
States	57
Sec	0.00000095

## From 0x40000000

Memory 2	
Address:	0x40000000
0x40000000:	3D 4E 64 20 5A 2F 9C 3D 8C FE AC FF 56 AD 45 0C 6C CD 03 C5 8C 0A A7 48 98 4C 00 3D FE 3E FC FE 00 00 00
0x40000023:	00 00
0x40000046:	00 00
0x40000069:	00 00
0x4000008C:	00 00

## Observation:

We can see that values are swapped.

3.7] Take two Nos from the memory, No1 and No2 as given below

No1= 0xBDF85433 and No2 =0x8FFF5FF3

No1= 0x967ff500 and No2 =0x50000000

No1 =0x74FFF000 and No2 =0x6800AFFF

No1 =0x80000000 and No2 =0xFFFF0000

(A) For each case perform 32 bit signed comparison and set the register R3 to 0xFF if No1 is greater than No2. Else set R3 to 0x00 using conditional execution of instructions.

(B) For each case perform 32 bit unsigned comparison and set the register R3 to 0xFF if No1 is greater than No2. Else set R3 to 0x00 using conditional execution of instructions.

->

a) We will use LDR for first loading both the values into register, then compare them with CMP and then make use of condition execution code **GT(for signed numbers)** as suffix with ADD instruction made to add 0xFF in R3 for required condition(greater than).

## Source Code:

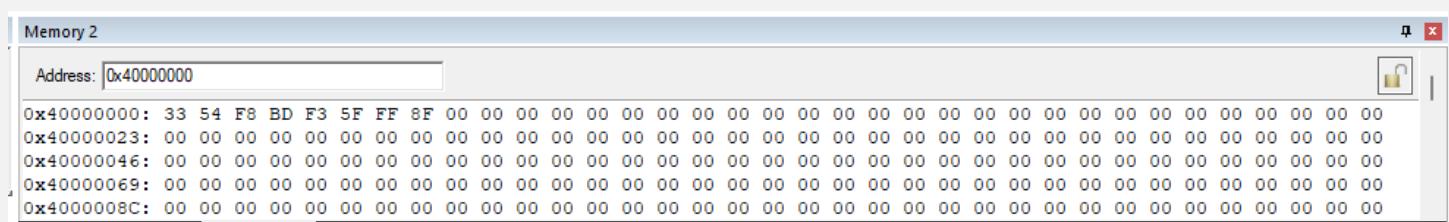
```
AREA qthree, CODE, READONLY
EXPORT Reset_Handler

Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
          MOV R3, #0 ; R3 holds the result
          LDR R0, [R1], #4 ; Load the first number & advance ptr
          LDR R2, [R1], #4 ; Load the second number & advance ptr
          CMP R0, R2; compare
          ADDGT R3, #0xFF; add if greater than(signed)
Stop B Stop
END
```

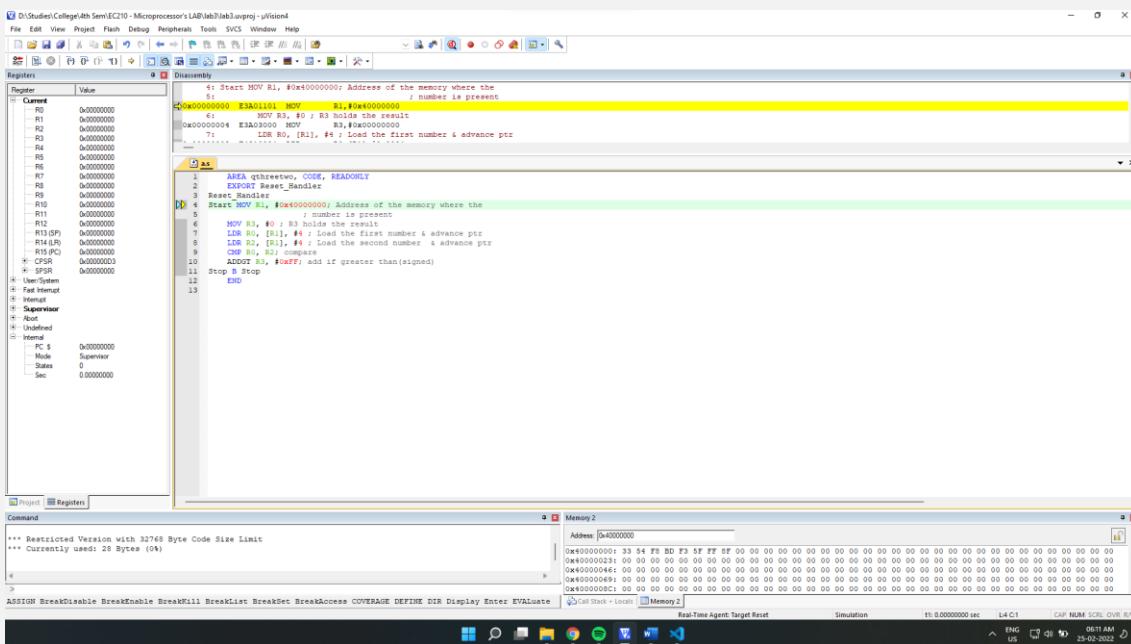
No1=0xBDF85433 and No2 =0x8FFF5FF3

## Debugging:

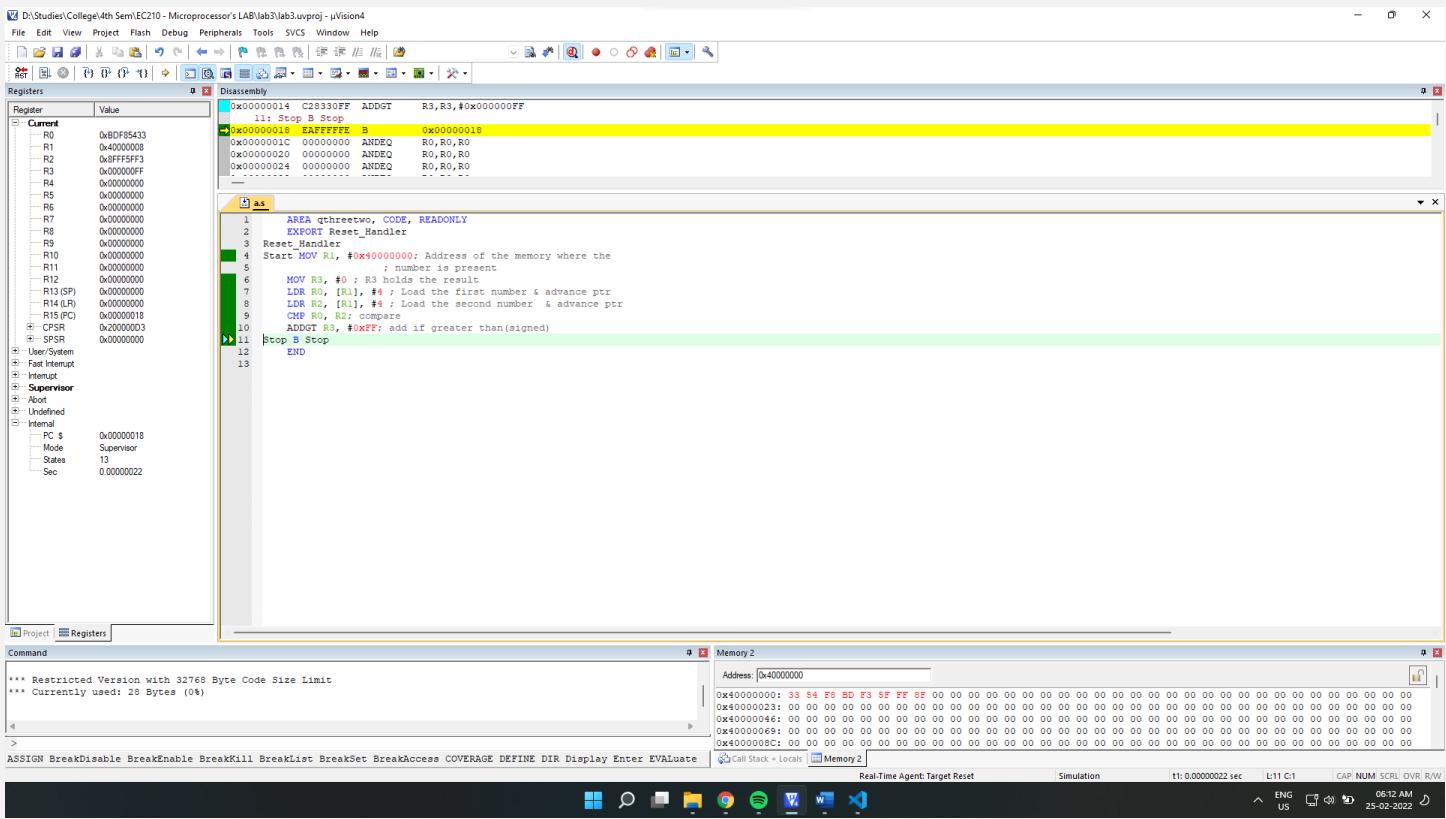
## Initial Memory:



# Setup:



# Final Output:



## Final Register Values:

Register	Value
R0	0xBDF85433
R1	0x40000008
R2	0x8FFFF5FF3
R3	0x000000FF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
<b>Internal</b>	
PC \$	0x00000018
Mode	Supervisor
States	13
Sec	0.00000022

No1= 0x967ff500 and No2 =0x50000000

## Debugging:

## Initial Memory:

**Memory 2**

Address: 0x40000000

```

0x40000000: 00 F5 7F 96 00 00 00 50 00 00 00 00 00 00 00 00
0x40000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000008C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

Call Stack + Locals | Memory 2

**Setup:**

D:\Studies\College 4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - μVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers | Disassembly

Registers

Register	Value
Current	0x00000000
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x00000003
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abaro	
Undefined	
Internal	
PC \$	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

Disassembly

```

        4: Start MOV R1, #0x40000000; Address of the memory where the
      5:                                     ; number is present
        0x00000000: E3A01101 MOV    R1,#0x40000000
      6:                                     ; R3 holds the result
        0x00000004: E3A03000 MOV    R3,#0x00000000
      7:                                     ; LDR R0, [R1], #4 ; Load the first number & advance ptr
        ...
        1: AREA qthreetwo, CODE, READONLY
      2: EXPORT Reset_Handler
      3: Reset_Handler
      4: Start MOV R1, #0x40000000; Address of the memory where the
      5:                                     ; number is present
      6:                                     ; R3 holds the result
      7: LDR R0, [R1], #4 ; Load the first number & advance ptr
      8: LDR R2, [R1], #4 ; Load the second number & advance ptr
      9: CMP R0, R2, comp
     10: ADDGT R3, #0xFF: add if greater than(signed)
     11: Stop B Stop
     12: END
     13:

```

Project | Registers

Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 28 Bytes (0%)

```

```

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

```

Real-Time Agent: Target Reset Simulation t1: 0.0000000 sec L4 C1 CAP NUM SCR L OV
 ENG US 06:15 AM 25-02-2022

After Execution:

S D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
Current	
R0	0x967FFF50
R1	0x40000008
R2	0x50000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (GP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x30000003
SPSR	0x00000000

User/System  
+ User  
  + Reset  
  + Interm  
  + Supervisor  
  + Abort  
  + Undefined  
  + Interm  
    + S  
      + Mode  
      + States  
      + Sec  
        0x00000018  
        Supervisor  
        13  
        0.00000022

```

0x00000014 C28330FF ADDGT R3,R3,#0x000000FF
11: Stop B Stop
0x00000018 EAFFFFFF B 0x00000018
0x0000001C 00000000 ANDEQ R0,R0,R0
0x00000020 00000000 ANDEQ R0,R0,R0
0x00000024 00000000 ANDEQ R0,R0,R0
.....
1 AREA qthreetwo, CODE, READONLY
2 EXPORT Reset_Handler
3 Reset_Handler
4 Start MOV R1, #0x40000000; Address of the memory where the
   ; number is present
5 MOV R3, #0 ; R3 hold the result
6 LDR R0, [R1], #4 ; Load the first number & advance ptr
7 LDR R2, [R1], #4 ; Load the second number & advance ptr
8 CMP R0, R2; compare
9 ADDGT R3, #0xFF; add if greater than(signed)
10 ADDGT R3, #0xFF; add if greater than(signed)
11 Stop B Stop
12 END
13

```

Project Registers Command

\*\*\* Restricted Version with 32768 Byte Code Size Limit  
\*\*\* Currently used: 28 Bytes (0%)

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

Memory 2

Address: 0x40000000

0x40000000: 00 F5 7F 96 00 00 00 50 00 00 00 00 00 00 00 00
0x40000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000008C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 2

Real-Time Agent: Target Stopped Simulation t1: 0.00000022 sec L:11 C:1 CAP NUM SCRL OVR R/W

ENGLISH US 06:15 AM 25-02-2022

## Final Output:

Memory 2

Address: 0x40000000

0x40000000: 00 F5 7F 96 00 00 00 50 00 00 00 00 00 00 00 00
0x40000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000008C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Call Stack + Locals Memory 2

Real-Time Agent: Target Stopped Simulation t1: 0.00000022 sec L:11 C:1 CAP NUM SCRL OVR R/W

## Final Register Values:

Register	Value
R0	0x967FF500
R1	0x40000008
R2	0x50000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x300000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
<b>Internal</b>	
PC \$	0x00000018
Mode	Supervisor
States	13
Sec.	0.00000022

No1 =0x74FFF000 and No2 =0x6800AFFF

## Debugging:

## Initial Memory:

## Setup:

D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
Current	0x00000000
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
+ CPSR	0x00000003
+ SPSR	0x00000000

User/System

- Fast Interrupt
- Internal
- Supervisor
- Abort
- Undefined
- Internal
- PC \$ 0x00000000
- Mode Supervisor
- States 0
- Sec 0.00000000

```

4: Start MOV R1, #0x40000000; Address of the memory where the
5:             ; number is present
6: E3A01101 MOV     R1,#0x40000000
7: E3A03000 MOV     R3,#0x00000000
8: LDR R0, [R1], #4 ; Load the first number & advance ptr
9: CMP R0, R2; compare
10 ADDGT R3, #0xFF; add if greater than(signed)
11 Stop B Stop
12 END
13

```

Project Registers Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 28 Bytes (0%)

```

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

Real-Time Agent:Target Reset Simulation t1: 0.0000000 sec L4 C1 CAP NUM SCR1 OVR1 R/W

EN US 0619 AM 25-02-2022

## After Execution:

## Final Output:

D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - µVision

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

Register	Value
Current	0x00000014 C0330FF ADDGT R3,R3,#0x000000FF
R0	0x40FF0000
R1	0x00000000
R2	0x000000FF
R3	0x00000020
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000018
R14 (LR)	0x00000000
R15 (PC)	0x00000003
+ CPSR	0x00000000

User/System

- Fast Interrupt
- Internal
- Supervisor
- Abort
- Undefined
- Internal
- PC \$ 0x00000018
- Mode Supervisor
- States 16
- Sec 0.0000027

```

1 AREA qthreetwo, CODE, READONLY
2 EXPORT Reset_Handler
3 Reset_Handler
4 Start MOV R1, #0x40000000; Address of the memory where the
5             ; number is present
6 MOV R3, #0 ; R3 holds the result
7 LDR R0, [R1], #4 ; Load the first number & advance ptr
8 LDR R2, [R1], #4 ; Load the second number & advance ptr
9 CMP R0, R2; compare
10 ADDGT R3, #0xFF; add if greater than(signed)
11 Stop B Stop
12 END
13

```

Project Registers Command

```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 28 Bytes (0%)

```

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

Step code execution

Real-Time Agent Target Reset Simulation t1: 0.0000027 sec L11 C1 CAP NUM SCR1 OVR1 R/W

EN US 0619 AM 25-02-2022

## Final Register Values:

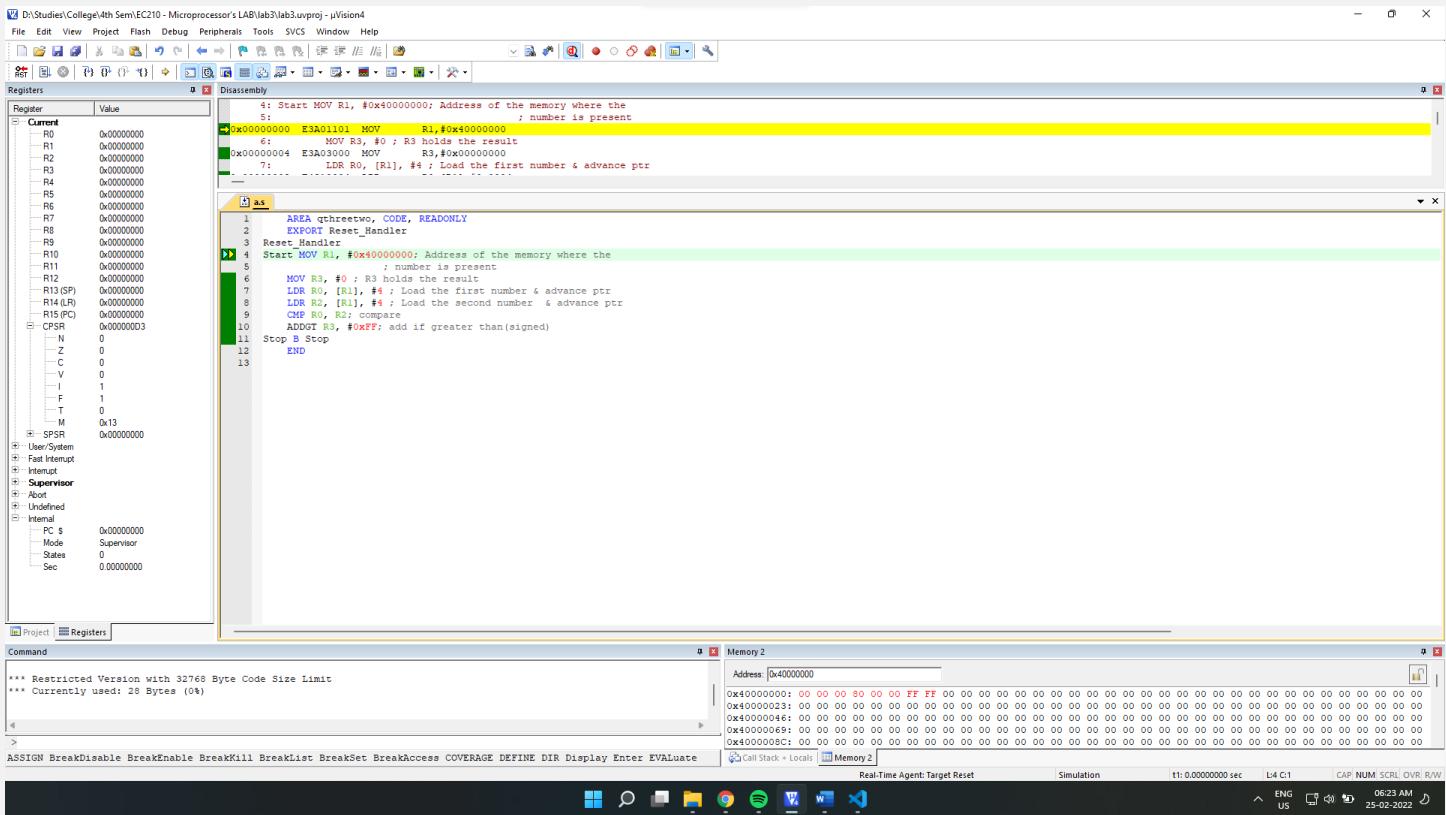
Register	Value
Current	
R0	0x74FFF000
R1	0x40000008
R2	0x6800AFFF
R3	0x000000FF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x200000D3
N	0
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
- Internal	
PC \$	0x00000018
Mode	Supervisor
States	16
Sec	0.00000027

No1 =0x80000000 and No2 =0xFFFF0000

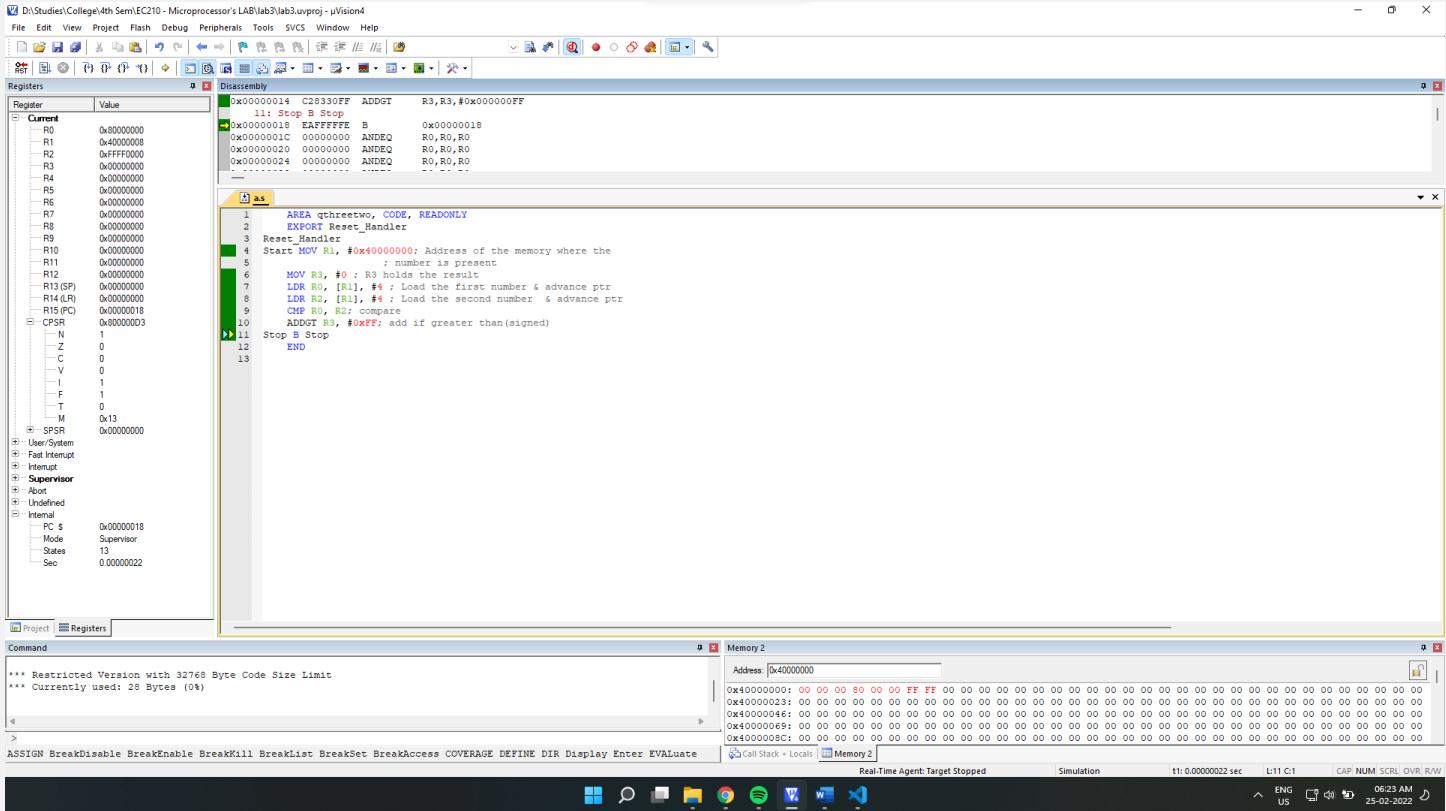
## Debugging:

## Initial Memory:

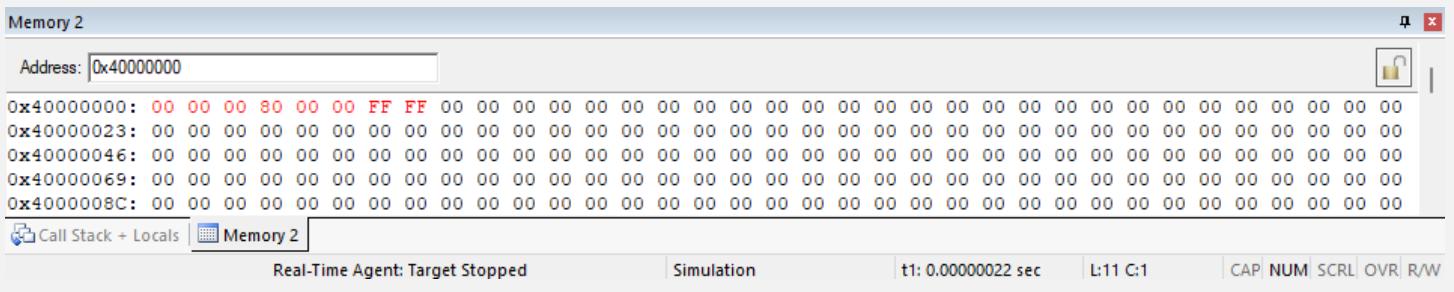
## Setup:



## After Execution:



## Final Output:



## Final Register Values:

Register	Value
R0	0x80000000
R1	0x40000008
R2	0xFFFF0000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000018
Mode	Supervisor
States	13
Sec	0.00000022

b) We will use LDR for first loading both the values into register, then compare them with CMP and then make use of condition execution code **HI**(for unsigned numbers) as suffix with ADD instruction made to add 0xFF in R3 for required condition(higher than/ greater than).

->

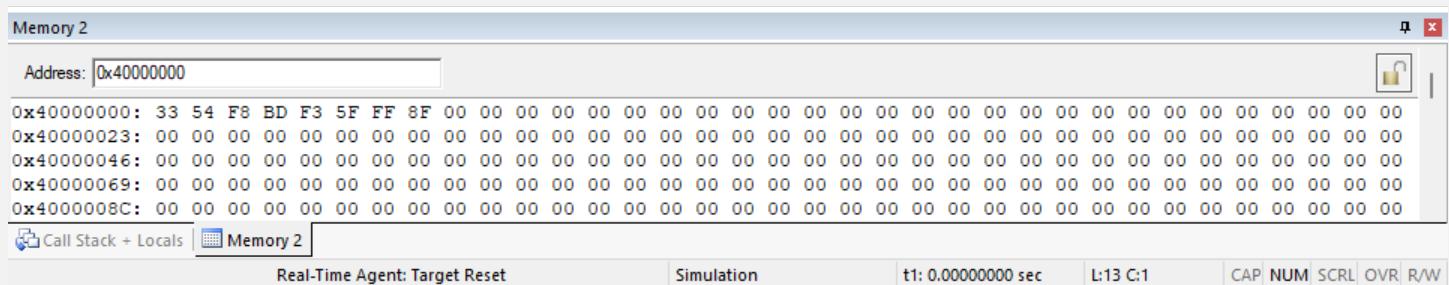
Source Code:

```
AREA qthree, CODE, READONLY
EXPORT Reset_Handler
Reset_Handler
Start MOV R1, #0x40000000; Address of the memory where the
                           ; number is present
    MOV R3, #0 ; R3 holds the result
    LDR R0, [R1], #4 ; Load the first number & advance ptr
    LDR R2, [R1], #4 ; Load the second number & advance ptr
    CMP R0, R2; compare
    ADDHI R3, #0xFF; add if higher/greater than(unsigned)
Stop B Stop
END
```

No1= 0xBDF85433 and No2 =0xFFFF5FF3

Debugging:

Initial Memory:



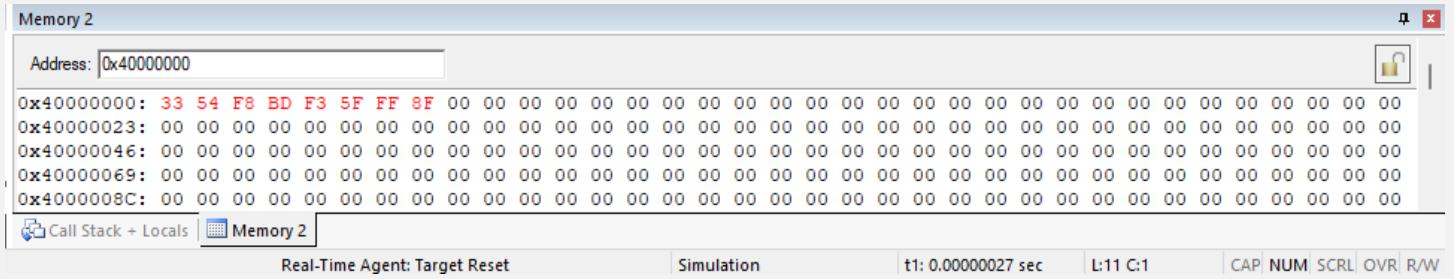
Setup:

The screenshot shows a software interface for a microcontroller project named "lab3.lab3.uvproj". The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripheral, Tools, SVCS, Window, Help. The toolbar contains icons for Open, Save, Build, Run, Stop, and others. The left sidebar displays the Register window with various processor registers like R0-R15, CPSR, and PC, along with their current values. The main workspace is divided into several panes: Disassembly, Registers, Stack, Memory, Command, and Simulation. The Disassembly pane shows assembly code for a Reset Handler, including instructions like MOV R1, #0x40000000; MOV R1, R0; LDR R0, [R1], #4; CMP R0, R2; ADDHI R3, #0xFF; and a Stop B Stop instruction. The Registers pane shows the state of all registers. The Stack pane shows the stack pointer (R13) at address 0x00000004. The Memory pane shows a memory dump starting at address 0x40000000, which is mostly filled with zeros. The Command pane at the bottom lists various commands and their descriptions. The status bar at the bottom right shows the date and time as 25-02-2022 06:34 AM.

## After Execution:

The screenshot shows the J-View software interface for a Microprocessor's LAB3 project. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripheral, Tools, SVCS, Window, Help. The Registers window on the left lists CPU registers R0 through R15, CPSR, and various interrupt flags. The Disassembly window in the center shows assembly code for a Reset Handler, including instructions like AREA, EXPORT, MOV, LDR, CMP, ADDHI, and Stop B Stop. The Memory window at the bottom displays memory starting at address 0x40000000 with various bytes of data. The bottom status bar shows the current time as 06:34 AM and the date as 25-02-2022.

## Final Output:



## Final Register Values:

Register	Value
R0	0xBDF85433
R1	0x40000008
R2	0x8FFF5FF3
R3	0x000000FF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000018
Mode	Supervisor
States	16
Sec	0.00000027

No1= 0x967ff500 and No2 =0x50000000

Debugging:

# Initial Memory:

The screenshot shows the uVision IDE interface with the 'Memory 2' window open. The address bar is set to 0x40000000. The memory dump area displays the following data:

```
0x40000000: 00 F5 7F 96 00 00 00 50 00 00 00 00 00 00 00 00  
0x40000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x4000008C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Below the memory dump are two tabs: 'Call Stack + Locals' and 'Memory 2', with 'Memory 2' being the active tab.

# Setup:

The screenshot shows the uVision IDE interface with the 'Registers' and 'Disassembly' windows open. The assembly code in the Disassembly window is:

```
; Start MOV R1, #0x40000000; Address of the memory where the  
; number is present  
E3A0101 MOV R1,$0x40000000  
; MOV R3, #0 ; R3 holds the result  
E3A0300 MOV R3,$0x00000000  
; LDR R0, [R1], #4 ; Load the first number & advance ptr  
LDR R0, [R1], #4  
; AREA qhretwo, CODE, READONLY  
; EXPORT Reset_Handler  
Reset_Handler:  
; Start MOV R1, #0x40000000; Address of the memory where the  
; number is present  
MOV R3, #0 ; R3 holds the result  
LDR R0, [R1], #4 ; Load the first number & advance ptr  
LDR R2, [R1], #4 ; Load the second number & advance ptr  
CMN R2, R3, compare  
ADDHI R2, #0xFF; add if higher/greater than(unsigned)  
Stop_B Stop  
END
```

The Registers window shows the current values for various registers:

Register	Value
Current	0x00000000
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000003
CPSR	0x00000000
SPSR	0x00000000

The Command window at the bottom shows the following output:

```
*** Restricted Version with 32768 Byte Code Size Limit  
*** Currently used: 28 Bytes (0%)  
4  
>  
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter EVALuate
```

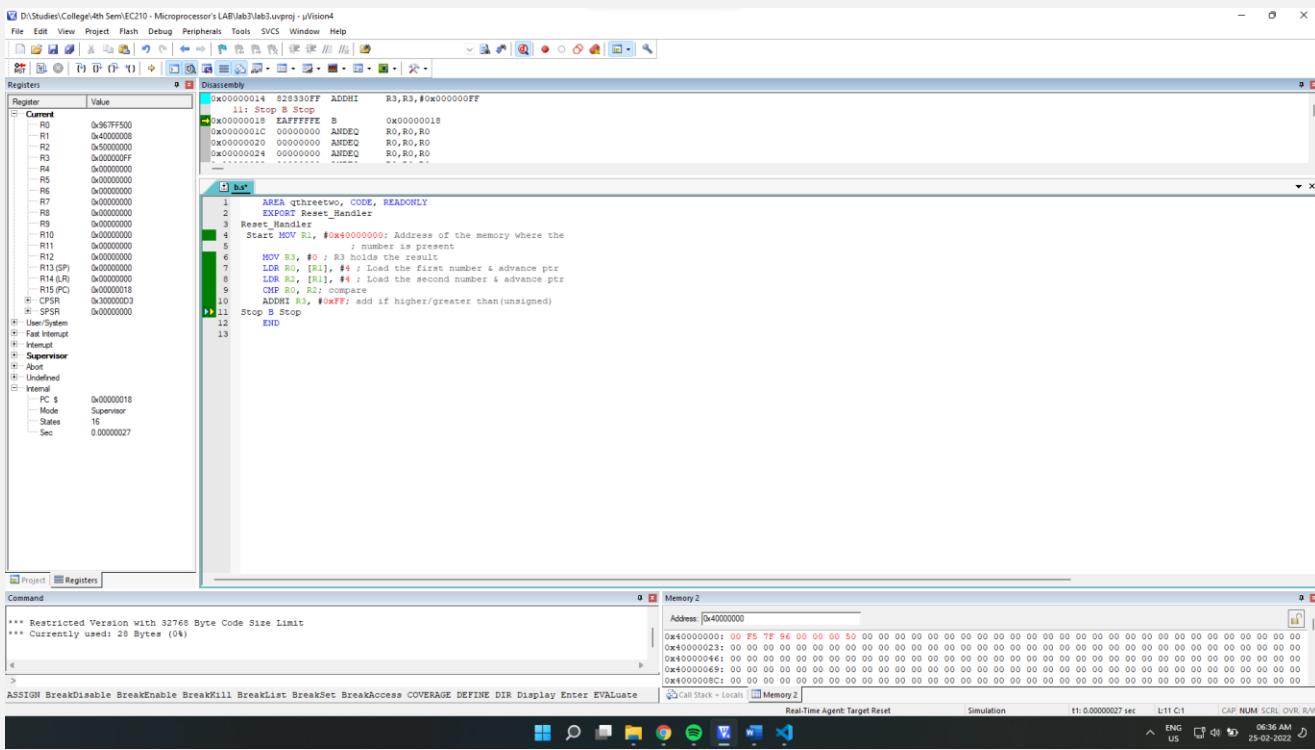
# After Execution:

## Final Output:

The screenshot shows the uVision IDE interface with the 'Memory 2' window open. The address bar is set to 0x40000000. The memory dump area displays the following data, identical to the initial state:

```
0x40000000: 00 F5 7F 96 00 00 00 50 00 00 00 00 00 00 00 00  
0x40000023: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x40000046: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x40000069: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
0x4000008C: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Below the memory dump are two tabs: 'Call Stack + Locals' and 'Memory 2', with 'Memory 2' being the active tab. The status bar at the bottom shows 'Real-Time Agent: Target Reset'.



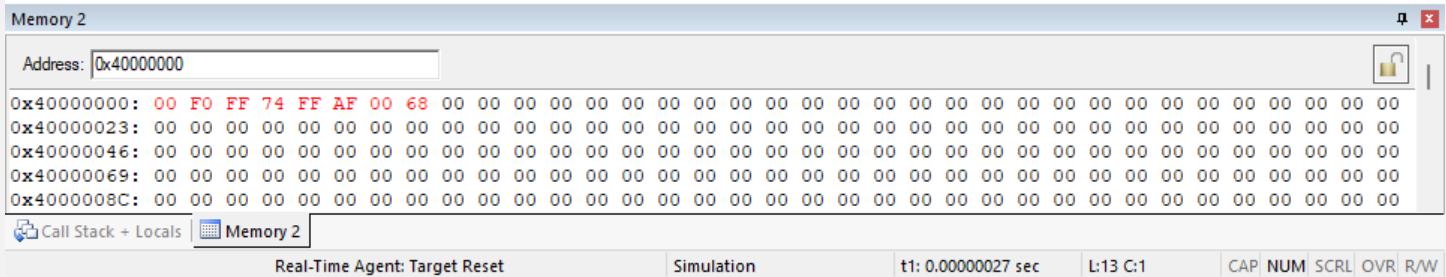
## Final Register Values:

Register	Value
Current	
R0	0x967FF500
R1	0x40000008
R2	0x50000000
R3	0x000000FF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x300000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC \$	0x00000018
Mode	Supervisor
States	16
Sec	0.00000027

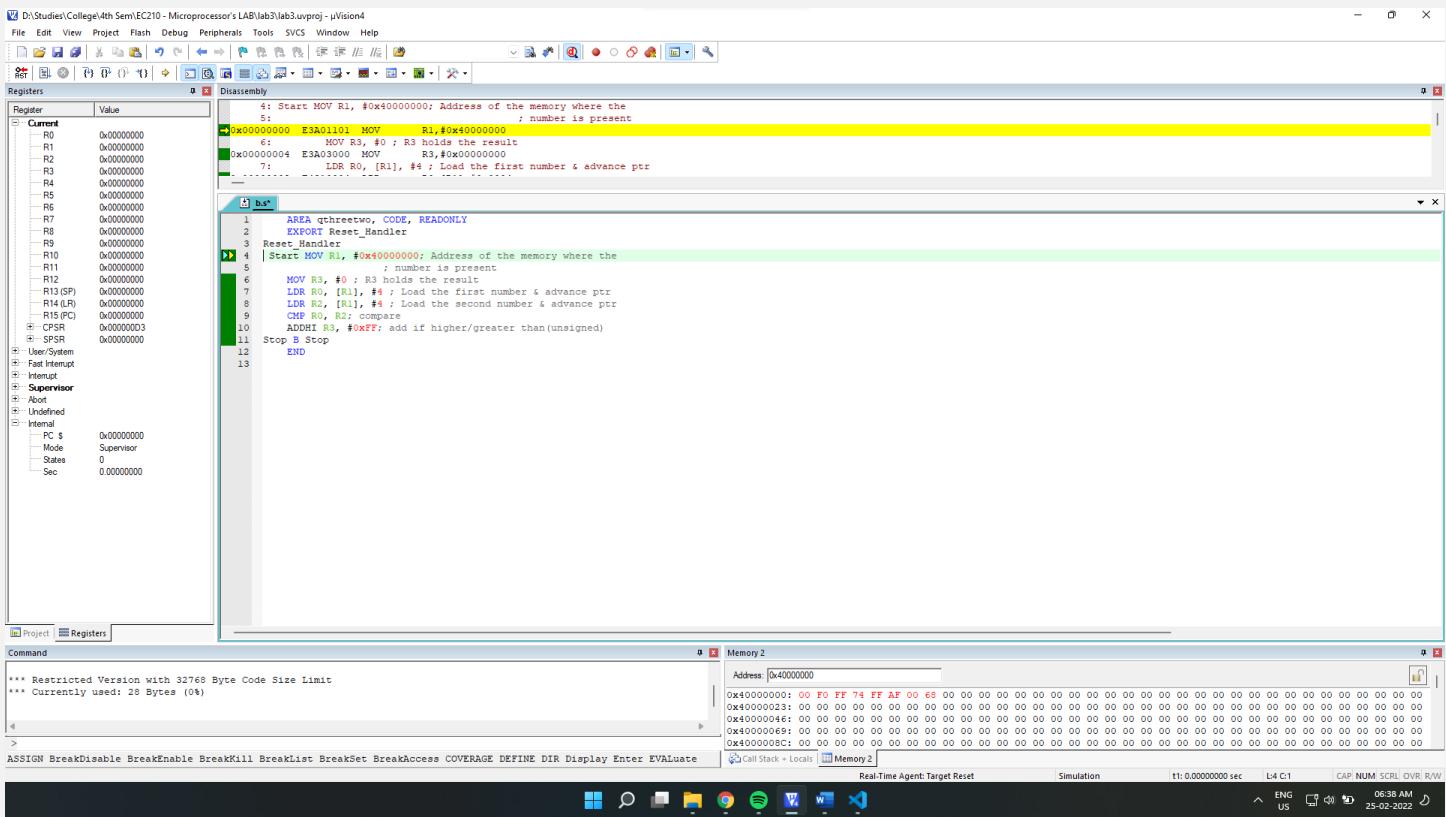
# No1 =0x74FFF000 and No2 =0x6800AFFF

## Debugging:

### Initial Memory:

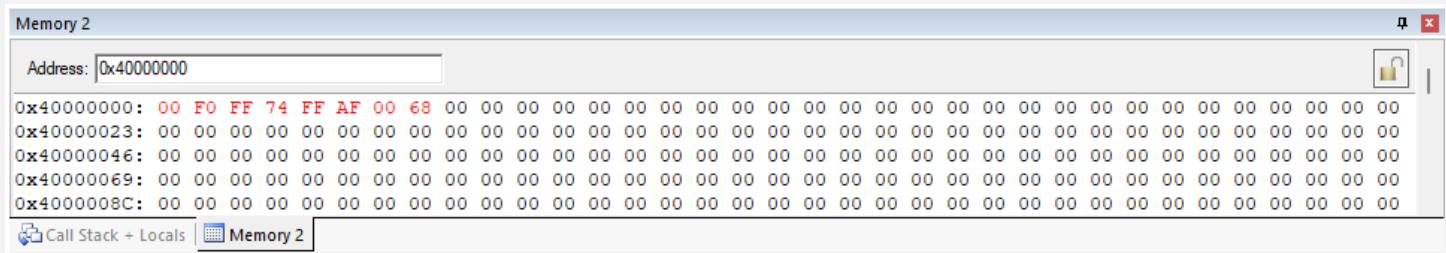


## Setup:



## After Execution:

### Final Output:



The screenshot shows the uVision IDE interface with the following details:

- Title Bar:** D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab3\lab3.uvproj - uVision4
- Menu Bar:** File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, Help
- Toolbars:** Standard, Project, Registers, Stack, Memory, Call Stack, Locals, Simulation, Breakpoints, Watch, Registers, Stack, Memory, Call Stack, Locals, Simulation, Breakpoints, Watch.
- Registers Window:** Shows CPU registers R0 through R15, CPSR, and PSR. Values are mostly zero, except for CPSR = 0x20000003 and PSR = 0x00000000.
- Assembly Window:** Displays the assembly code for the Reset Handler:

```
 AREA qhreetwo, CODE, READONLY
 EXPORT Reset_Handler
Reset_Handler
    Start MOV R1, #0x40000000; Address of the memory where the
                                ; number is present
    MOV R3, #0 ; R3 holds the result
    LDR R0, [R1], #4 ; Load the first number & advance ptr
    LDR R2, [R1], #4 ; Load the second number & advance ptr
    CMP R0, R2 ; Compare
    ADDH R3, R3, #0xFFFF; add if higher/greater than(unsigned)
    Stop B Stop
END
```
- Memory Window:** Shows memory starting at address 0x40000000. The first few bytes are FF FF FF FF, followed by 00 00 00 00, and then 00 00 00 00 for several pages.
- Command Window:** Displays compiler messages:

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 28 Bytes (0%)
>
```
- Status Bar:** Real-Time Agent: Target Reset, Simulation, t1: 0.000000027 sec, L11 C1, CAP NUM SCR OVR, ENG US, 25-02-2022, 06:39 AM.

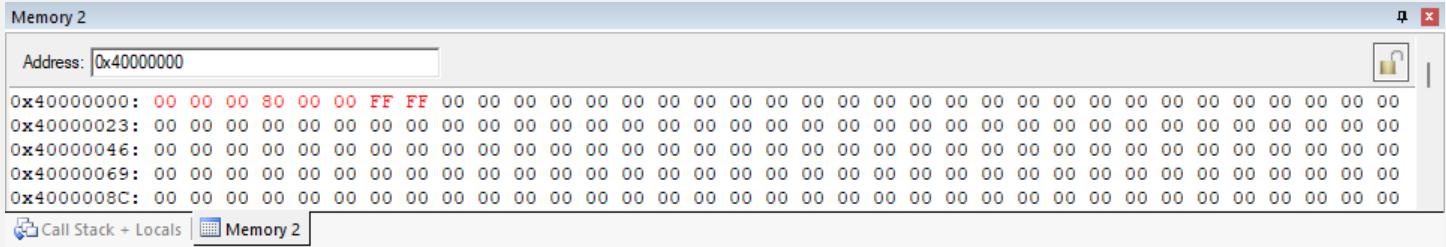
## Final Register Values:

Register	Value
R0	0x74FFF000
R1	0x40000008
R2	0x6800AFFF
R3	0x000000FF
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
+ CPSR	0x200000D3
+ SPSR	0x00000000
+ User/System	
+ Fast Interrupt	
+ Interrupt	
+ Supervisor	
+ Abort	
+ Undefined	
+ Internal	
PC \$	0x00000018
Mode	Supervisor
States	16
Sec	0.00000027

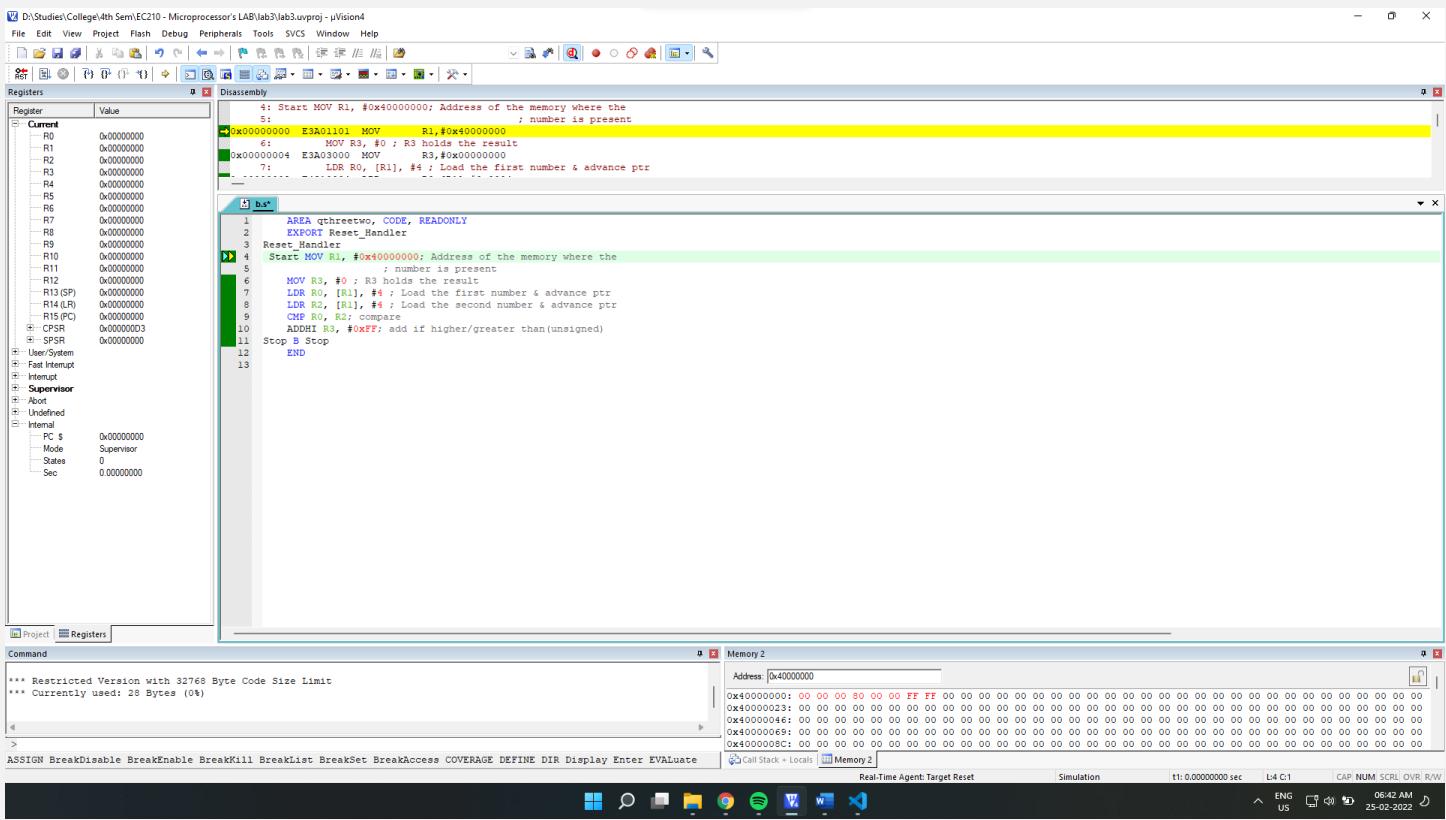
# No1 =0x80000000 and No2 =0xFFFF0000

## Debugging:

### Initial Memory:

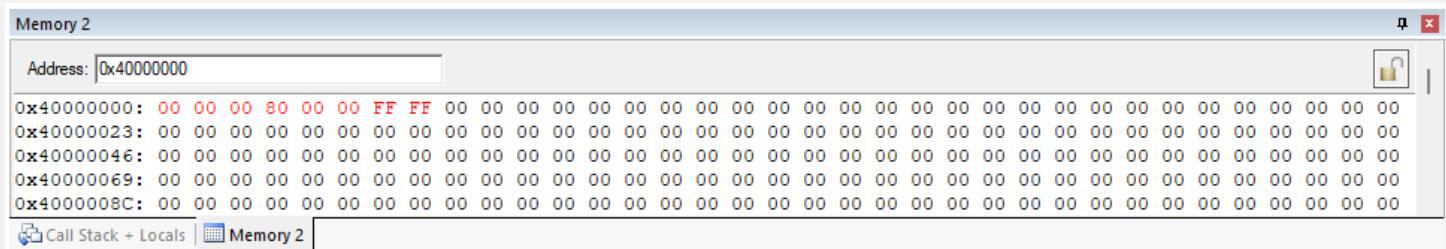


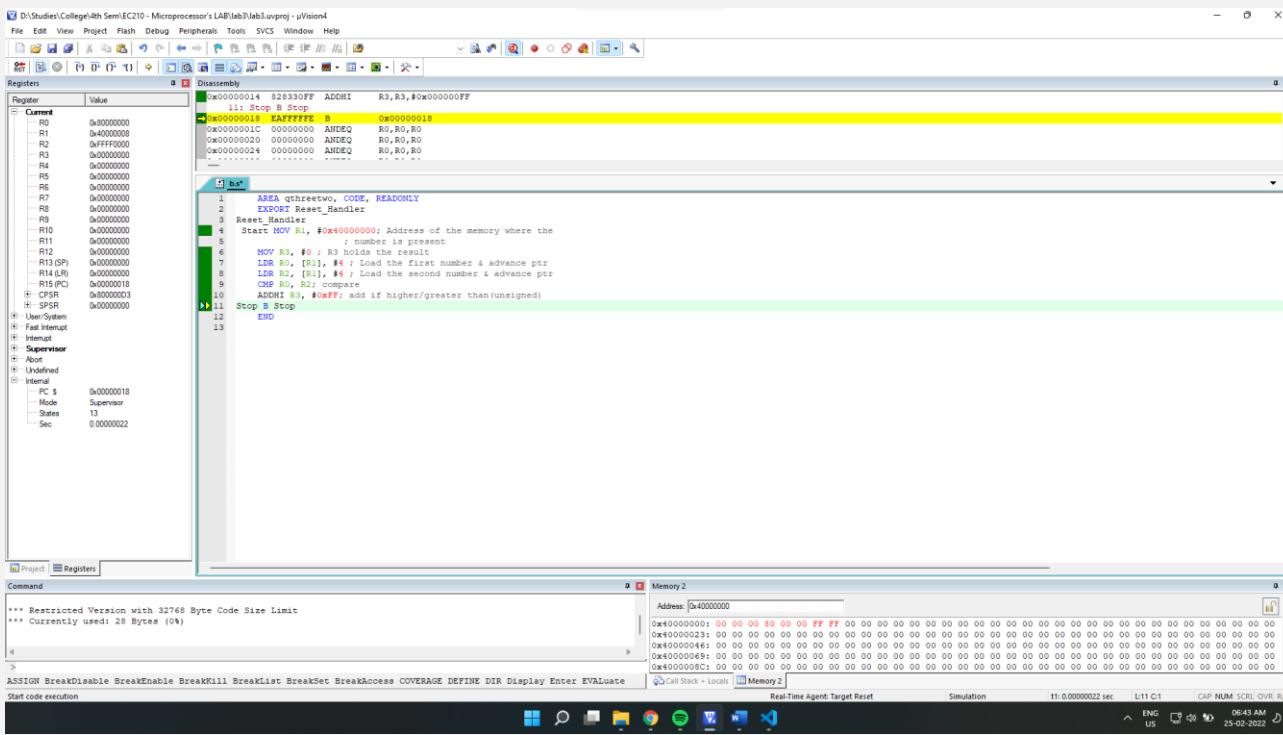
### Setup:



### After Execution:

### Final Output:





## Final Register Values:

Register	Value
<b>Current</b>	
R0	0x80000000
R1	0x40000008
R2	0xFFFFF000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
<b>Supervisor</b>	
Abort	
Undefined	
<b>Internal</b>	
PC \$	0x00000018
Mode	Supervisor
States	13
Sec	0.00000022