EC-210

MICROPROCESSORS LAB

LAB-2

UTKARSH MAHAJAN 201EC164

ARNAV RAJ 201EC109

<u>Objective:</u> To demonstrate the use of arithmetic instructions

Exercise:

2.1]Write an assembly program to take two 32 bit unsigned numbers in to the registers using MOV or MVN instructions and perform the following

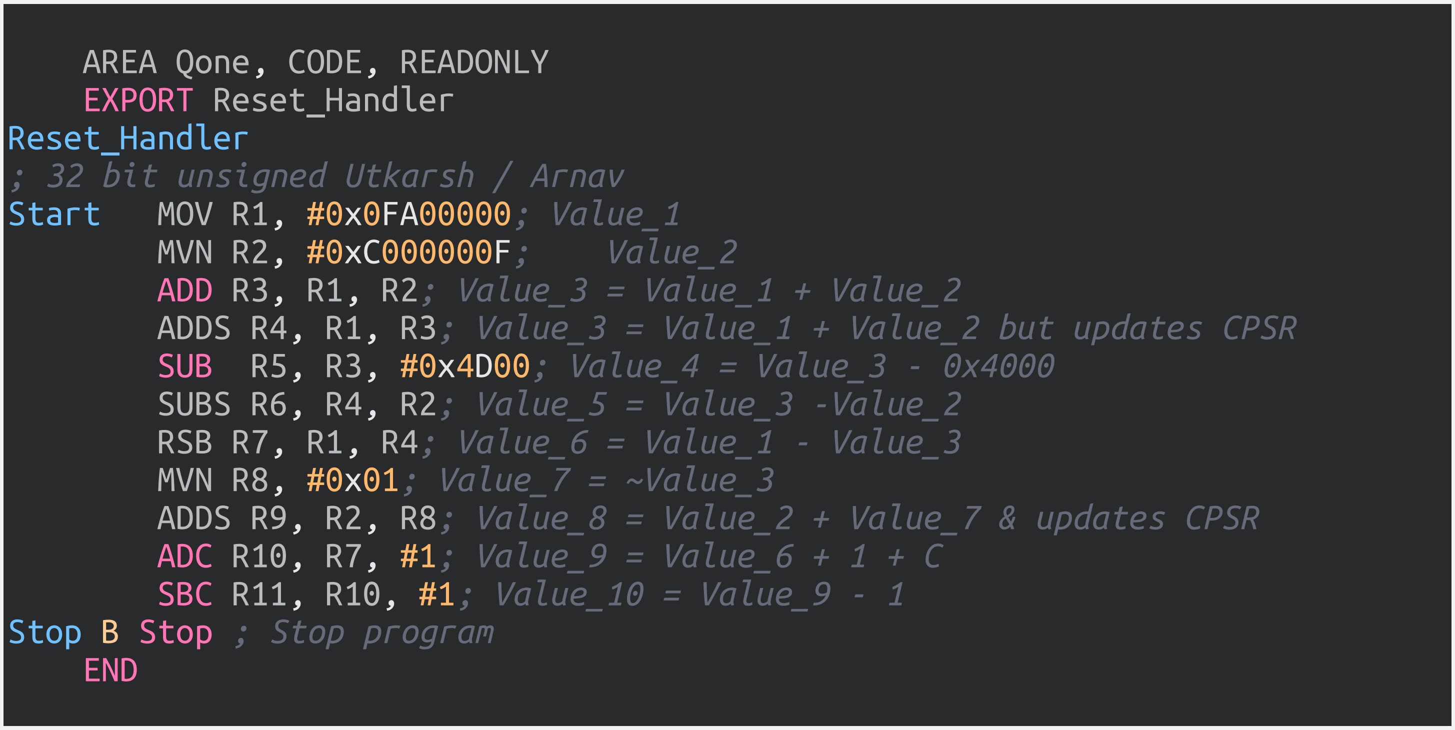
(a) Add using ADD, ADDS, ADC

(b) Subtract using SUB, SUBS, SBC

(c) Reverse subtract using RSB, RSC

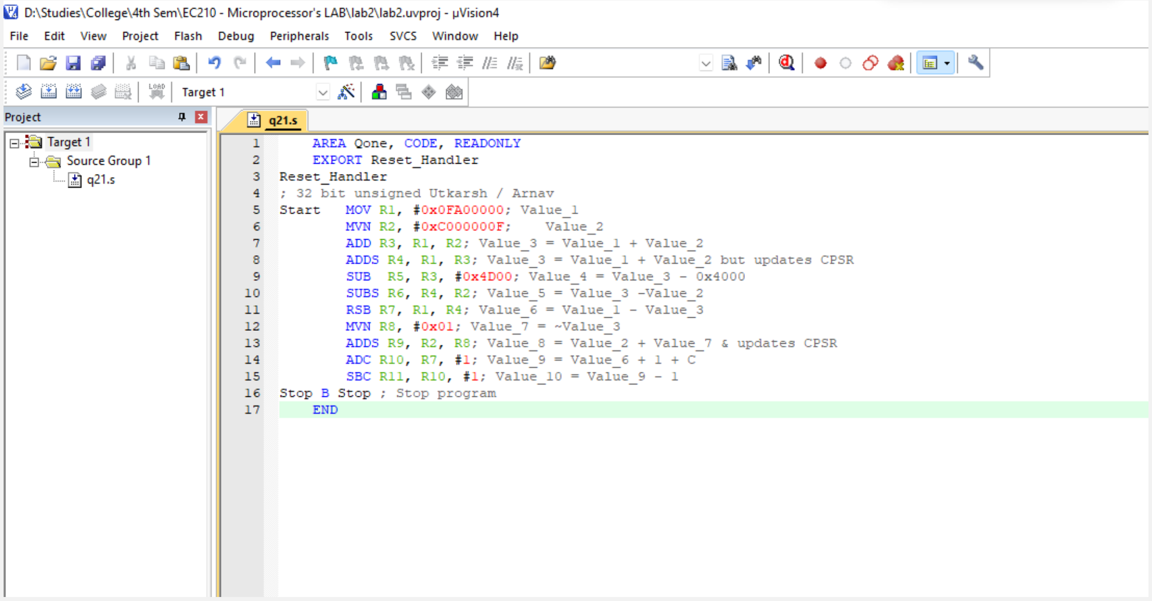
-> using all of the above instructions in a single assembly file.

Source Code:



```
    AREA Qone, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; 32 bit unsigned Utkarsh / Arnav
Start   MOV R1, #0x0FA00000; Value_1
        MVN R2, #0xC000000F;    Value_2
        ADD R3, R1, R2; Value_3 = Value_1 + Value_2
        ADDS R4, R1, R3; Value_3 = Value_1 + Value_2 but updates CPSR
        SUB  R5, R3, #0x4D00; Value_4 = Value_3 - 0x4000
        SUBS R6, R4, R2; Value_5 = Value_3 -Value_2
        RSB R7, R1, R4; Value_6 = Value_1 - Value_3
        MVN R8, #0x01; Value_7 = ~Value_3
        ADDS R9, R2, R8; Value_8 = Value_2 + Value_7 & updates CPSR
        ADC R10, R7, #1; Value_9 = Value_6 + 1 + C
        SBC R11, R10, #1; Value_10 = Value_9 - 1
Stop B Stop ; Stop program
    END
```

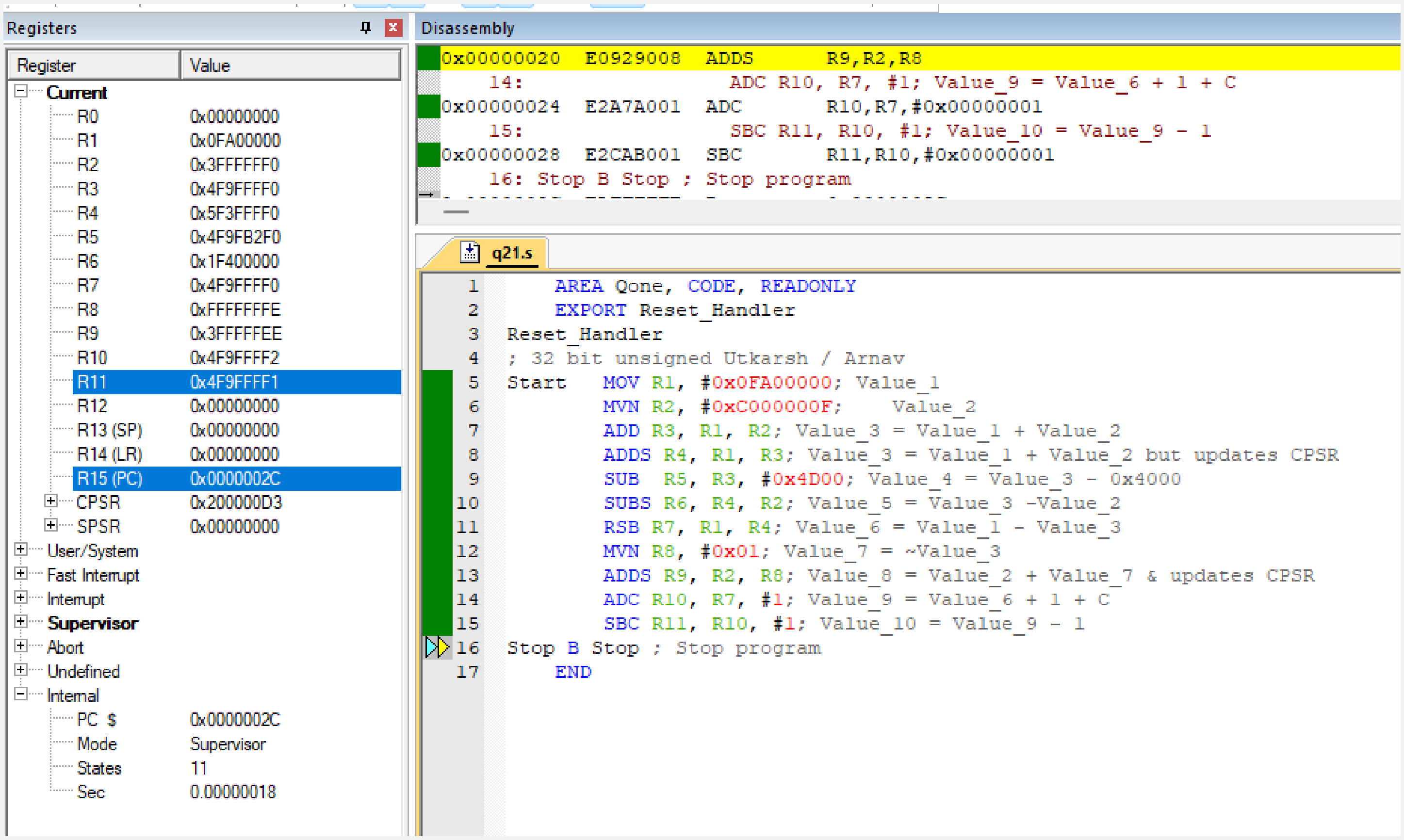
Setup:


D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
q21.s
q21.s

```
    AREA Qone, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; 32 bit unsigned Utkarsh / Arnav
Start   MOV R1, #0x0FA00000; Value_1
        MVN R2, #0xC000000F;    Value_2
        ADD R3, R1, R2; Value_3 = Value_1 + Value_2
        ADDS R4, R1, R3; Value_3 = Value_1 + Value_2 but updates CPSR
        SUB  R5, R3, #0x4D00; Value_4 = Value_3 - 0x4000
        SUBS R6, R4, R2; Value_5 = Value_3 -Value_2
        RSB R7, R1, R4; Value_6 = Value_1 - Value_3
        MVN R8, #0x01; Value_7 = ~Value_3
        ADDS R9, R2, R8; Value_8 = Value_2 + Value_7 & updates CPSR
        ADC R10, R7, #1; Value_9 = Value_6 + 1 + C
        SBC R11, R10, #1; Value_10 = Value_9 - 1
Stop B Stop ; Stop program
    END
```

Debugging:

End Result:



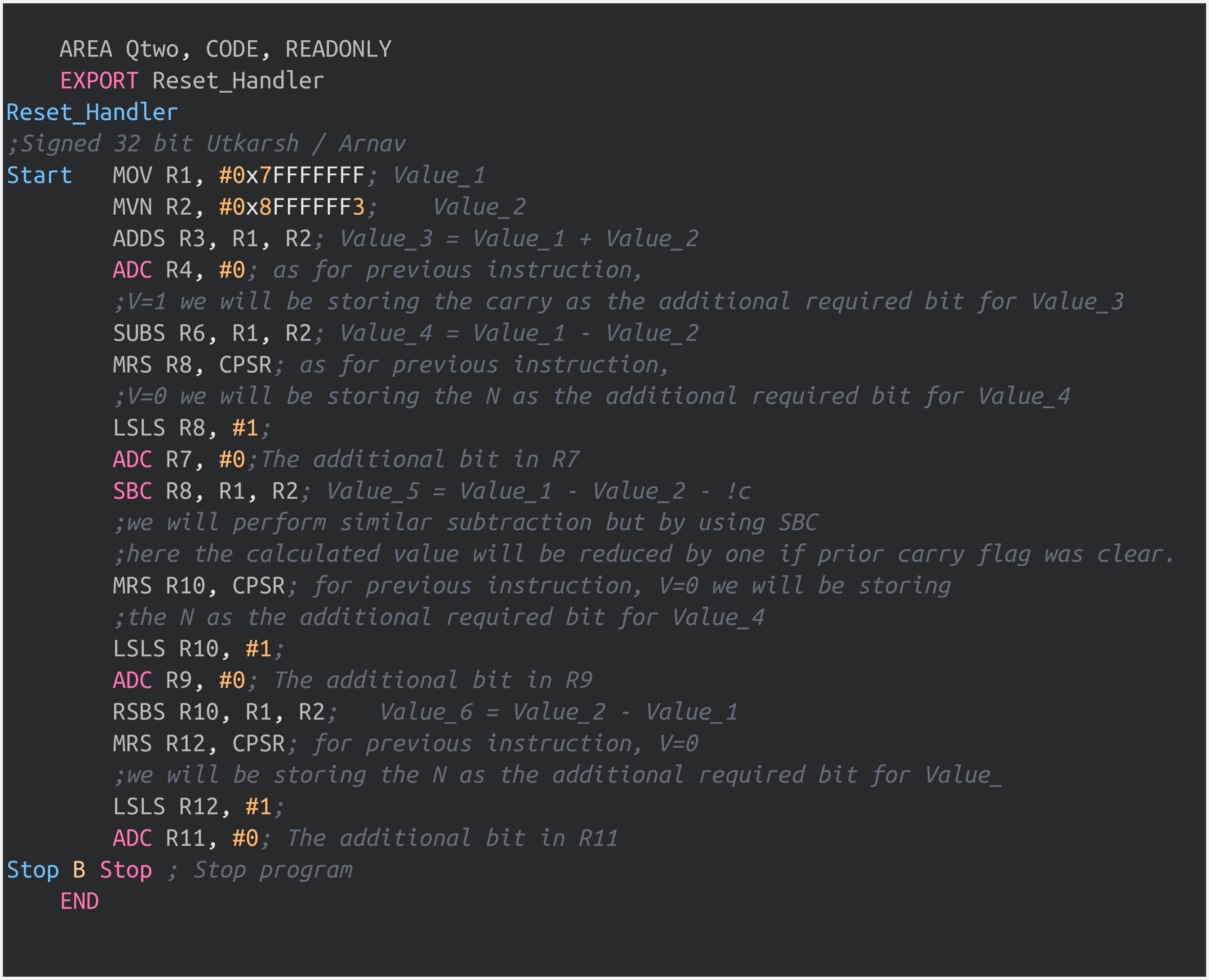
Register	Value
Current	
R0	0x00000000
R1	0x0FA00000
R2	0x3FFFFFF0
R3	0x4F9FFFF0
R4	0x5F3FFFF0
R5	0x4F9FB2F0
R6	0x1F400000
R7	0x4F9FFFF0
R8	0xFFFFFFFE
R9	0x3FFFFFEE
R10	0x4F9FFFF2
R11	0x4F9FFFF1
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000002C
CPSR	0x200000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x0000002C
Mode	Supervisor
States	11
Sec	0.00000018

```
0x00000020  E0929008  ADDS      R9,R2,R8
    14:                  ADC R10, R7, #1; Value_9 = Value_6 + 1 + C
0x00000024  E2A7A001  ADC       R10,R7,#0x00000001
    15:                  SBC R11, R10, #1; Value_10 = Value_9 - 1
0x00000028  E2CAB001  SBC       R11,R10,#0x00000001
    16: Stop B Stop ; Stop program
```

```
    AREA Qone, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; 32 bit unsigned Utkarsh / Arnav
Start   MOV R1, #0x0FA00000; Value_1
        MVN R2, #0xC000000F;    Value_2
        ADD R3, R1, R2; Value_3 = Value_1 + Value_2
        ADDS R4, R1, R3; Value_3 = Value_1 + Value_2 but updates CPSR
        SUB  R5, R3, #0x4D00; Value_4 = Value_3 - 0x4000
        SUBS R6, R4, R2; Value_5 = Value_3 -Value_2
        RSB R7, R1, R4; Value_6 = Value_1 - Value_3
        MVN R8, #0x01; Value_7 = ~Value_3
        ADDS R9, R2, R8; Value_8 = Value_2 + Value_7 & updates CPSR
        ADC R10, R7, #1; Value_9 = Value_6 + 1 + C
        SBC R11, R10, #1; Value_10 = Value_9 - 1
Stop B Stop ; Stop program
    END
```

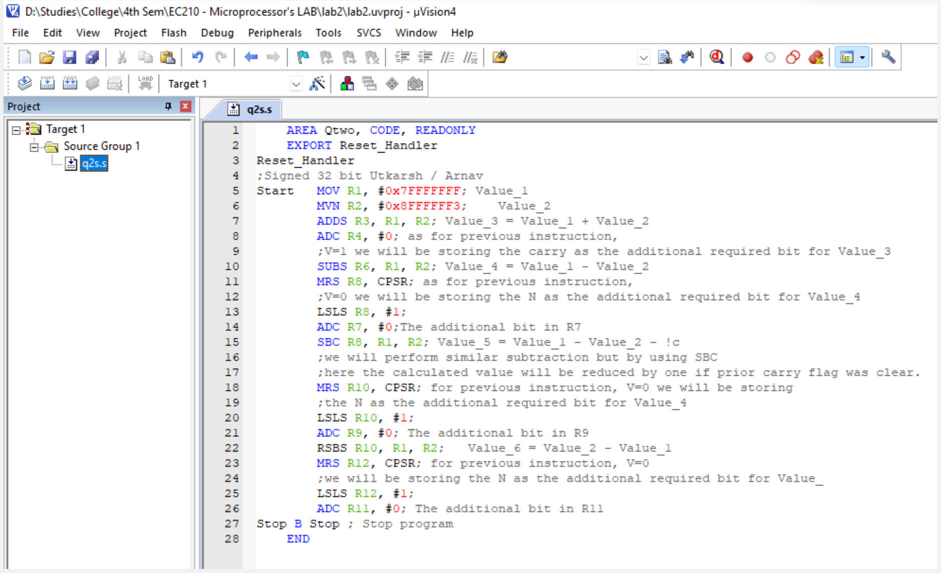
2.2 (a) Repeat ex 2.1 a,b,c for 32 bit signed numbers.

-> Source Code:



```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
;Signed 32 bit Utkarsh / Arnav
Start   MOV R1, #0x7FFFFFFF; Value_1
        MVN R2, #0x8FFFFFF3;    Value_2
        ADDS R3, R1, R2; Value_3 = Value_1 + Value_2
        ADC R4, #0; as for previous instruction,
        ;V=1 we will be storing the carry as the additional required bit for Value_3
        SUBS R6, R1, R2; Value_4 = Value_1 - Value_2
        MRS R8, CPSR; as for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R8, #1;
        ADC R7, #0;The additional bit in R7
        SBC R8, R1, R2; Value_5 = Value_1 - Value_2 - !c
        ;we will perform similar subtraction but by using SBC
        ;here the calculated value will be reduced by one if prior carry flag was clear.
        MRS R10, CPSR; for previous instruction, V=0 we will be storing
        ;the N as the additional required bit for Value_4
        LSLS R10, #1;
        ADC R9, #0; The additional bit in R9
        RSBS R10, R1, R2;   Value_6 = Value_2 - Value_1
        MRS R12, CPSR; for previous instruction, V=0
        ;we will be storing the N as the additional required bit for Value_
        LSLS R12, #1;
        ADC R11, #0; The additional bit in R11
Stop B Stop ; Stop program
    END
```

Setup:



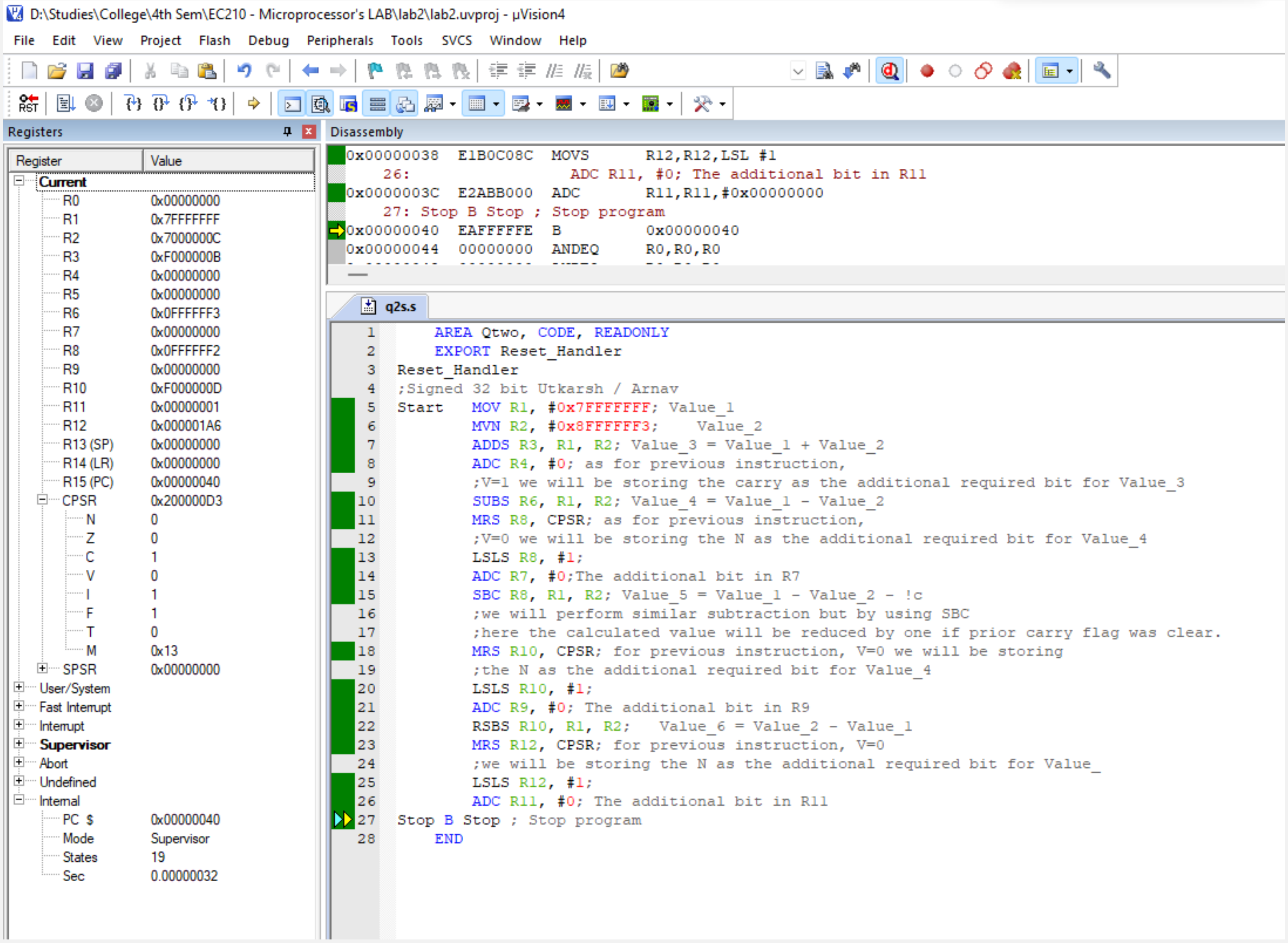
D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
q2s.s
q2s.s

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
;Signed 32 bit Utkarsh / Arnav
Start   MOV R1, #0x7FFFFFFF; Value_1
        MVN R2, #0x8FFFFFF3;    Value_2
        ADDS R3, R1, R2; Value_3 = Value_1 + Value_2
        ADC R4, #0; as for previous instruction,
        ;V=1 we will be storing the carry as the additional required bit for Value_3
        SUBS R6, R1, R2; Value_4 = Value_1 - Value_2
        MRS R8, CPSR; as for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R8, #1;
        ADC R7, #0;The additional bit in R7
        SBC R8, R1, R2; Value_5 = Value_1 - Value_2 - !c
        ;we will perform similar subtraction but by using SBC
        ;here the calculated value will be reduced by one if prior carry flag was clear.
        MRS R10, CPSR; for previous instruction, V=0 we will be storing
        ;the N as the additional required bit for Value_4
        LSLS R10, #1;
        ADC R9, #0; The additional bit in R9
        RSBS R10, R1, R2;   Value_6 = Value_2 - Value_1
        MRS R12, CPSR; for previous instruction, V=0
        ;we will be storing the N as the additional required bit for Value_
        LSLS R12, #1;
        ADC R11, #0; The additional bit in R11
Stop B Stop ; Stop program
    END
```

Debugging:

End Result:



D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RST
Registers
Disassembly
q2s.s

Register	Value
Current	
R0	0x00000000
R1	0x7FFFFFFF
R2	0x7000000C
R3	0xF000000B
R4	0x00000000
R5	0x00000000
R6	0x0FFFFFF3
R7	0x00000000
R8	0x0FFFFFF2
R9	0x00000000
R10	0xF000000D
R11	0x00000001
R12	0x000001A6
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000040
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 $	0x00000040
Mode	Supervisor
States	19
Sec	0.00000032

```
0x00000038  E1B0C08C  MOVS      R12,R12,LSL #1
    26:                 ADC R11, #0; The additional bit in R11
0x0000003C  E2ABB000  ADC       R11,R11,#0x00000000
    27: Stop B Stop ; Stop program
0x00000040  EAFFFFFE  B         0x00000040
0x00000044  00000000  ANDEQ     R0,R0,R0
```

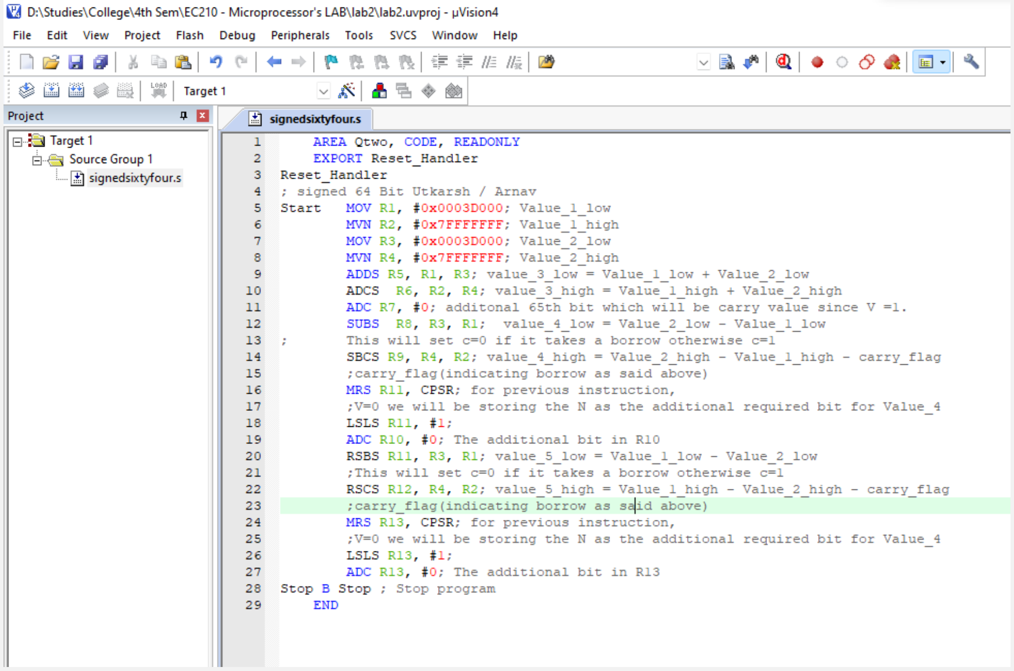
```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
;Signed 32 bit Utkarsh / Arnav
Start   MOV R1, #0x7FFFFFFF; Value_1
        MVN R2, #0x8FFFFFF3;    Value_2
        ADDS R3, R1, R2; Value_3 = Value_1 + Value_2
        ADC R4, #0; as for previous instruction,
        ;V=1 we will be storing the carry as the additional required bit for Value_3
        SUBS R6, R1, R2; Value_4 = Value_1 - Value_2
        MRS R8, CPSR; as for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R8, #1;
        ADC R7, #0;The additional bit in R7
        SBC R8, R1, R2; Value_5 = Value_1 - Value_2 - !c
        ;we will perform similar subtraction but by using SBC
        ;here the calculated value will be reduced by one if prior carry flag was clear.
        MRS R10, CPSR; for previous instruction, V=0 we will be storing
        ;the N as the additional required bit for Value_4
        LSLS R10, #1;
        ADC R9, #0; The additional bit in R9
        RSBS R10, R1, R2;   Value_6 = Value_2 - Value_1
        MRS R12, CPSR; for previous instruction, V=0
        ;we will be storing the N as the additional required bit for Value_
        LSLS R12, #1;
        ADC R11, #0; The additional bit in R11
Stop B Stop ; Stop program
    END
```

(b) Repeat ex 2.1 a,b,c for 64 bit signed numbers.

-> Source Code:

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; signed 64 Bit Utkarsh / Arnav
Start   MOV R1, #0x0003D000; Value_1_low
        MVN R2, #0x7FFFFFFF; Value_1_high
        MOV R3, #0x0003D000; Value_2_low
        MVN R4, #0x7FFFFFFF; Value_2_high
        ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
        ADCS  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
        ADC R7, #0; additonal 65th bit which will be carry value since V =1.
        SUBS  R8, R3, R1;  value_4_low = Value_2_low - Value_1_low
;       This will set c=0 if it takes a borrow otherwise c=1
        SBCS R9, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R11, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R11, #1;
        ADC R10, #0; The additional bit in R10
        RSBS R11, R3, R1; value_5_low = Value_1_low - Value_2_low
        ;This will set c=0 if it takes a borrow otherwise c=1
        RSCS R12, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R13, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R13, #1;
        ADC R13, #0; The additional bit in R13
Stop B Stop ; Stop program
    END
```

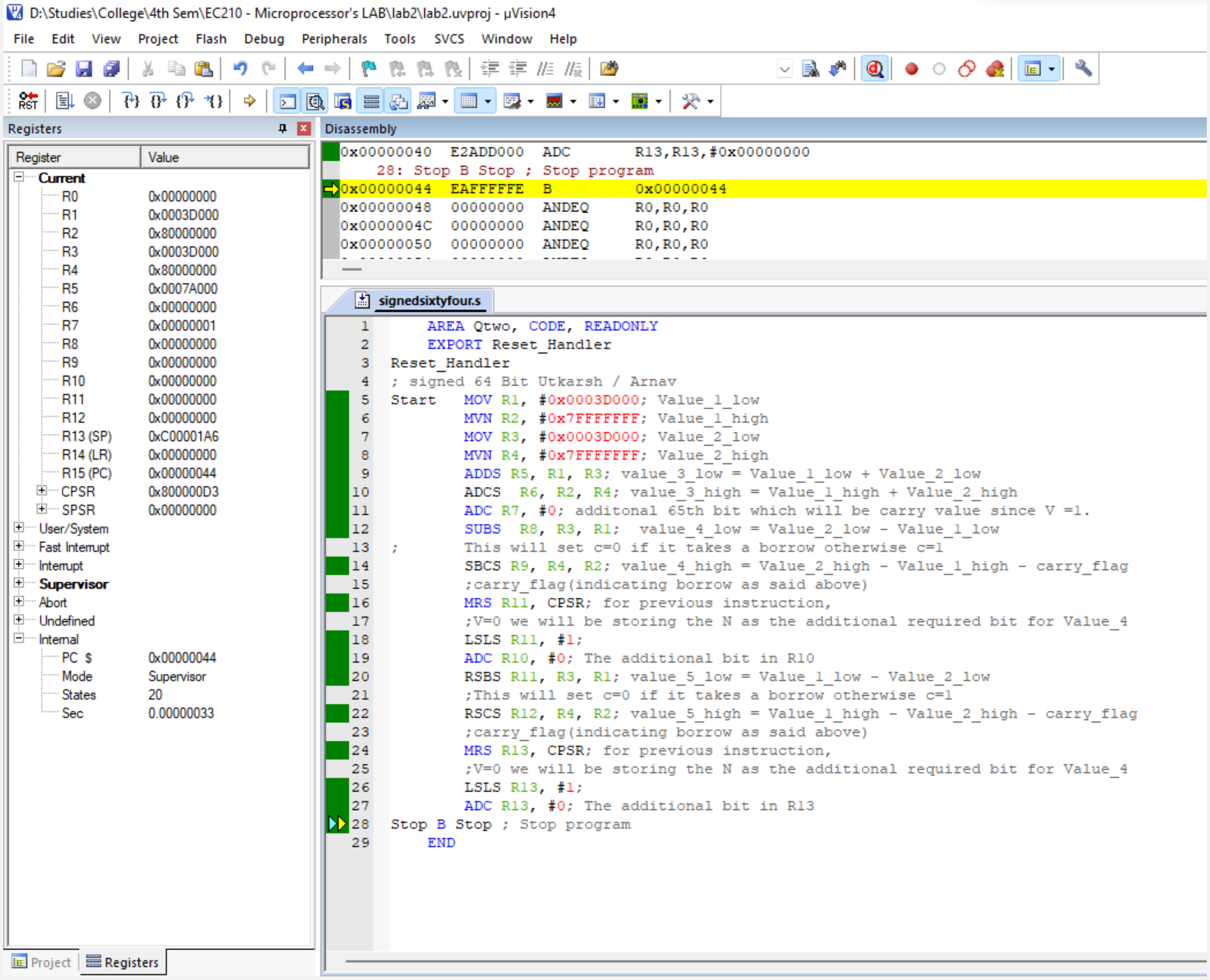
Setup:


D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
signedsixtyfour.s
signedsixtyfour.s

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; signed 64 Bit Utkarsh / Arnav
Start   MOV R1, #0x0003D000; Value_1_low
        MVN R2, #0x7FFFFFFF; Value_1_high
        MOV R3, #0x0003D000; Value_2_low
        MVN R4, #0x7FFFFFFF; Value_2_high
        ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
        ADCS  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
        ADC R7, #0; additonal 65th bit which will be carry value since V =1.
        SUBS  R8, R3, R1;  value_4_low = Value_2_low - Value_1_low
;       This will set c=0 if it takes a borrow otherwise c=1
        SBCS R9, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R11, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R11, #1;
        ADC R10, #0; The additional bit in R10
        RSBS R11, R3, R1; value_5_low = Value_1_low - Value_2_low
        ;This will set c=0 if it takes a borrow otherwise c=1
        RSCS R12, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R13, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R13, #1;
        ADC R13, #0; The additional bit in R13
Stop B Stop ; Stop program
    END
```

Debugging:

End Result:


D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers
Disassembly
signedsixtyfour.s
Project
Registers

Register	Value
Current	
R0	0x00000000
R1	0x0003D000
R2	0x80000000
R3	0x0003D000
R4	0x80000000
R5	0x0007A000
R6	0x00000000
R7	0x00000001
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0xC00001A6
R14 (LR)	0x00000000
R15 (PC)	0x00000044
CPSR	0x800000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000044
Mode	Supervisor
States	20
Sec	0.00000033

```
0x00000040  E2ADD000  ADC       R13,R13,#0x00000000
     28: Stop B Stop ; Stop program
0x00000044  EAFFFFFE  B         0x00000044
0x00000048  00000000  ANDEQ     R0,R0,R0
0x0000004C  00000000  ANDEQ     R0,R0,R0
0x00000050  00000000  ANDEQ     R0,R0,R0
```

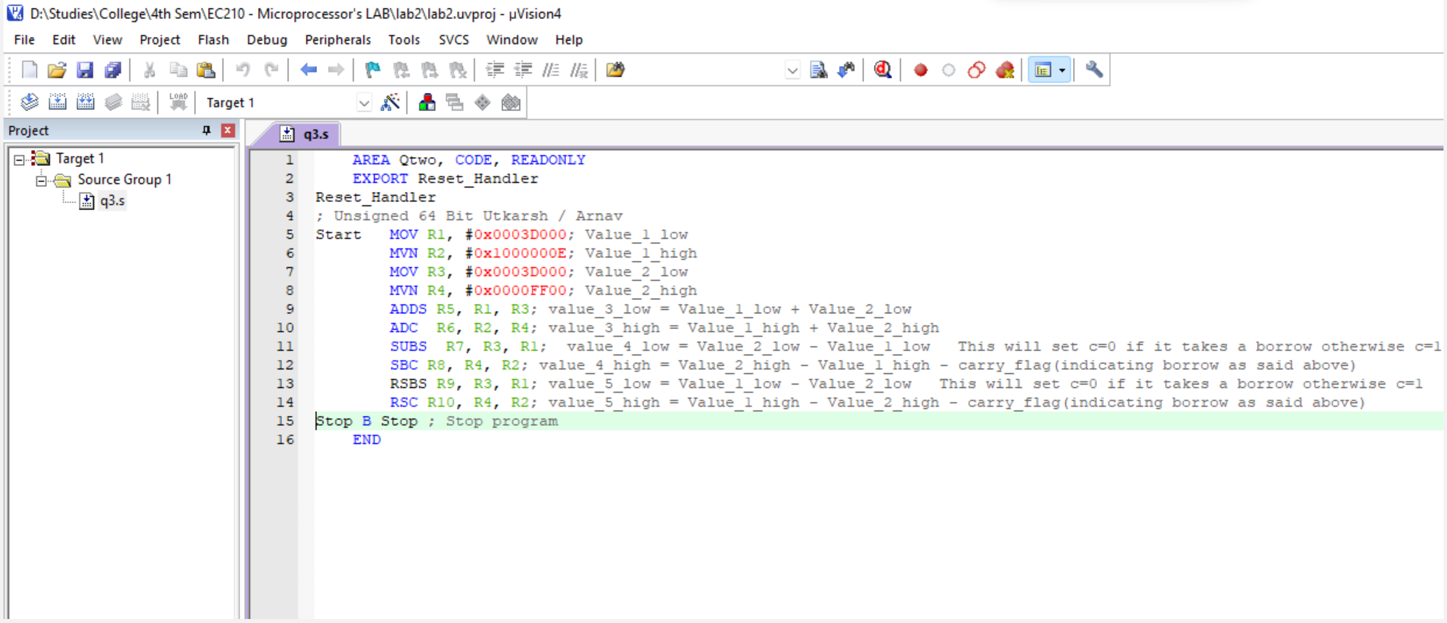
```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; signed 64 Bit Utkarsh / Arnav
Start   MOV R1, #0x0003D000; Value_1_low
        MVN R2, #0x7FFFFFFF; Value_1_high
        MOV R3, #0x0003D000; Value_2_low
        MVN R4, #0x7FFFFFFF; Value_2_high
        ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
        ADCS  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
        ADC R7, #0; additonal 65th bit which will be carry value since V =1.
        SUBS  R8, R3, R1;  value_4_low = Value_2_low - Value_1_low
;       This will set c=0 if it takes a borrow otherwise c=1
        SBCS R9, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R11, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R11, #1;
        ADC R10, #0; The additional bit in R10
        RSBS R11, R3, R1; value_5_low = Value_1_low - Value_2_low
        ;This will set c=0 if it takes a borrow otherwise c=1
        RSCS R12, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag
        ;carry_flag(indicating borrow as said above)
        MRS R13, CPSR; for previous instruction,
        ;V=0 we will be storing the N as the additional required bit for Value_4
        LSLS R13, #1;
        ADC R13, #0; The additional bit in R13
Stop B Stop ; Stop program
    END
```

(c) Repeat ex 2.1 a,b,c for 64 bit unsigned numbers.

-> Source Code:

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; Unsigned 64 Bit Utkarsh / Arnav
Start    MOV R1, #0x0003D000; Value_1_low
         MVN R2, #0x1000000E; Value_1_high
         MOV R3, #0x0003D000; Value_2_low
         MVN R4, #0x0000FF00; Value_2_high
         ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
         ADC  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
         SUBS  R7, R3, R1;  value_4_low = Value_2_low - Value_1_low
         ;  This will set c=0 if it takes a borrow otherwise c=1
         SBC R8, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag
         ;carry_flag(indicating borrow as said above)
         RSBS R9, R3, R1; value_5_low = Value_1_low - Value_2_low
         ; This will set c=0 if it takes a borrow otherwise c=1
         RSC R10, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag
         ; This will set c=0 if it takes a borrow otherwise c=1
Stop B Stop ; Stop program
    END
```

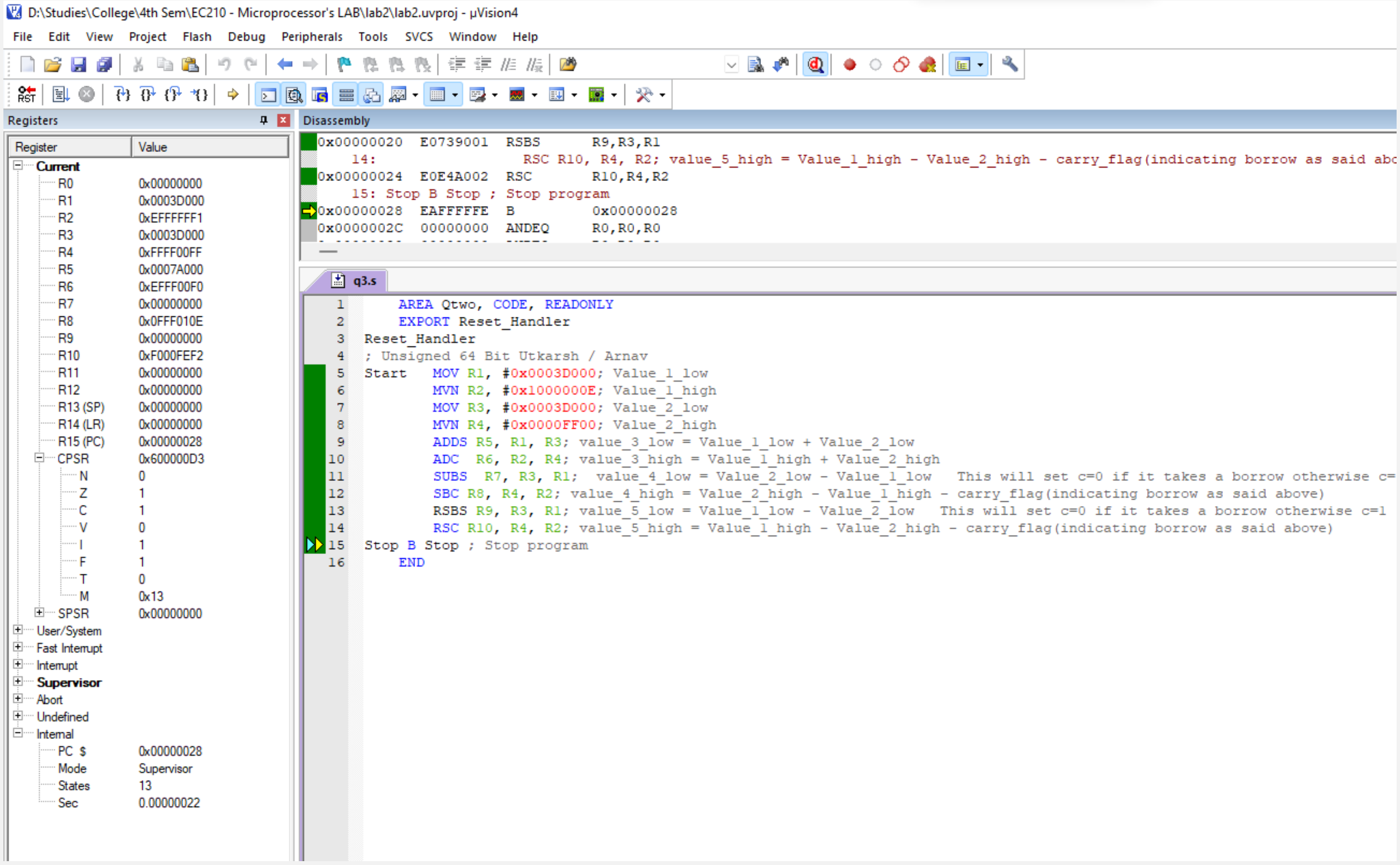
Setup:


D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Source Group 1
q3.s
q3.s

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; Unsigned 64 Bit Utkarsh / Arnav
Start   MOV R1, #0x0003D000; Value_1_low
        MVN R2, #0x1000000E; Value_1_high
        MOV R3, #0x0003D000; Value_2_low
        MVN R4, #0x0000FF00; Value_2_high
        ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
        ADC  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
        SUBS  R7, R3, R1;  value_4_low = Value_2_low - Value_1_low   This will set c=0 if it takes a borrow otherwise c=1
        SBC R8, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag(indicating borrow as said above)
        RSBS R9, R3, R1; value_5_low = Value_1_low - Value_2_low   This will set c=0 if it takes a borrow otherwise c=1
        RSC R10, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag(indicating borrow as said above)
Stop B Stop ; Stop program
    END
```

Debugging:

End Result:



D:\Studies\College\4th Sem\EC210 - Microprocessor's LAB\lab2\lab2.uvproj - µVision4
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
Registers
Disassembly
q3.s

```
Register       Value
Current
  R0           0x00000000
  R1           0x0003D000
  R2           0xEFFFFFF1
  R3           0x0003D000
  R4           0xFFFF00FF
  R5           0x0007A000
  R6           0xEFFF00F0
  R7           0x00000000
  R8           0x0FFF010E
  R9           0x00000000
  R10          0xF000FEF2
  R11          0x00000000
  R12          0x00000000
  R13 (SP)     0x00000000
  R14 (LR)     0x00000000
  R15 (PC)     0x00000028
  CPSR         0x600000D3
    N          0
    Z          1
    C          1
    V          0
    I          1
    F          1
    T          0
    M          0x13
  SPSR         0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC $         0x00000028
  Mode         Supervisor
  States       13
  Sec          0.00000022
```

```
0x00000020  E0739001  RSBS      R9,R3,R1
    14:               RSC R10, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag(indicating borrow as said abo
0x00000024  E0E4A002  RSC       R10,R4,R2
    15: Stop B Stop ; Stop program
0x00000028  EAFFFFFE  B         0x00000028
0x0000002C  00000000  ANDEQ     R0,R0,R0
```

```
    AREA Qtwo, CODE, READONLY
    EXPORT Reset_Handler
Reset_Handler
; Unsigned 64 Bit Utkarsh / Arnav
Start   MOV R1, #0x0003D000; Value_1_low
        MVN R2, #0x1000000E; Value_1_high
        MOV R3, #0x0003D000; Value_2_low
        MVN R4, #0x0000FF00; Value_2_high
        ADDS R5, R1, R3; value_3_low = Value_1_low + Value_2_low
        ADC  R6, R2, R4; value_3_high = Value_1_high + Value_2_high
        SUBS  R7, R3, R1;  value_4_low = Value_2_low - Value_1_low   This will set c=0 if it takes a borrow otherwise c=
        SBC R8, R4, R2; value_4_high = Value_2_high - Value_1_high - carry_flag(indicating borrow as said above)
        RSBS R9, R3, R1; value_5_low = Value_1_low - Value_2_low   This will set c=0 if it takes a borrow otherwise c=1
        RSC R10, R4, R2; value_5_high = Value_1_high - Value_2_high - carry_flag(indicating borrow as said above)
Stop B Stop ; Stop program
    END
```