**EC-210**

**Microprocessors Lab**

**LAB-2**



Utkarsh Mahajan 201EC164

ARNAV RAJ 201EC109

Objective: 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*

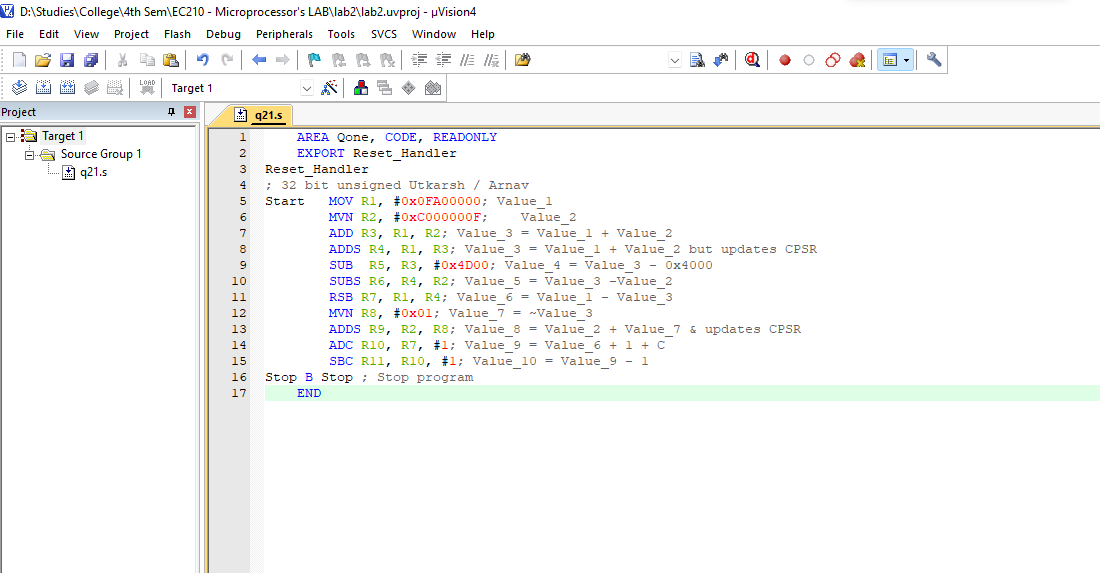
        ADC R10, R7, #1*; Value\_9 = Value\_6 + 1 + C*

        SBC R11, R10, #1*; Value\_10 = Value\_9 - 1*

Stop B Stop *; Stop program*

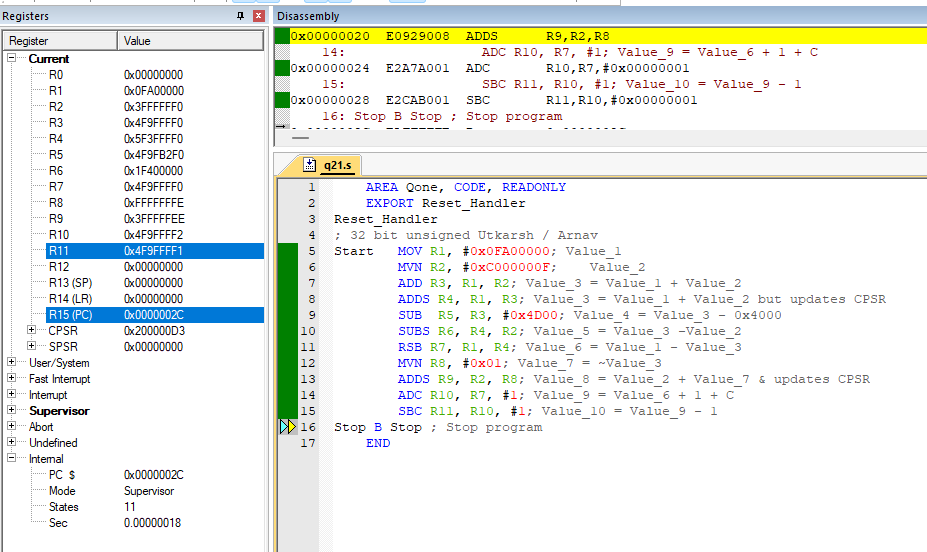
    END

Setup:



Debugging:

End Result:



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\_*

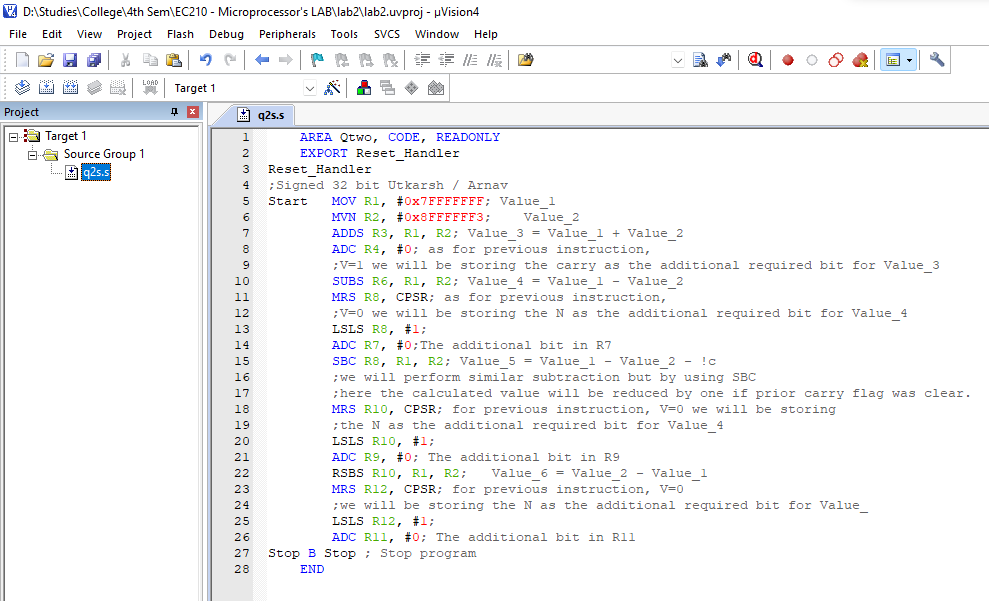
        LSLS R12, #1*;*

        ADC R11, #0*; The additional bit in R11*

Stop B Stop *; Stop program*

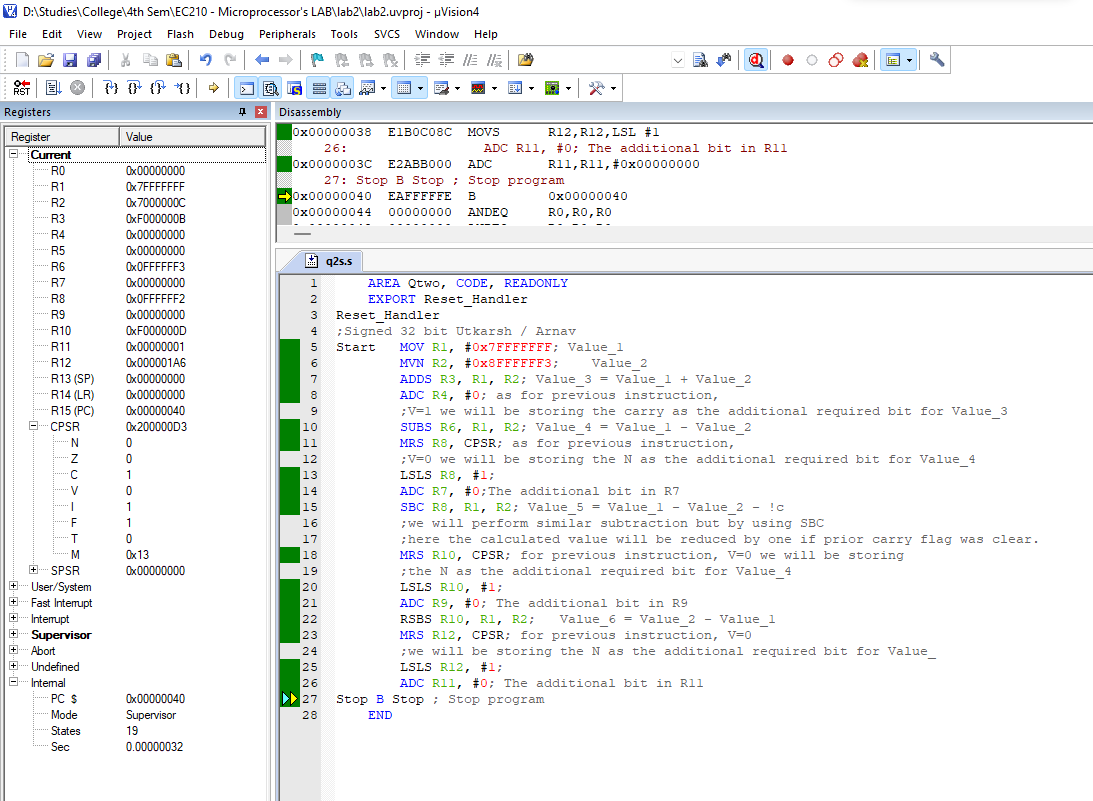
    END

Setup:



Debugging:

End Result:



(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*

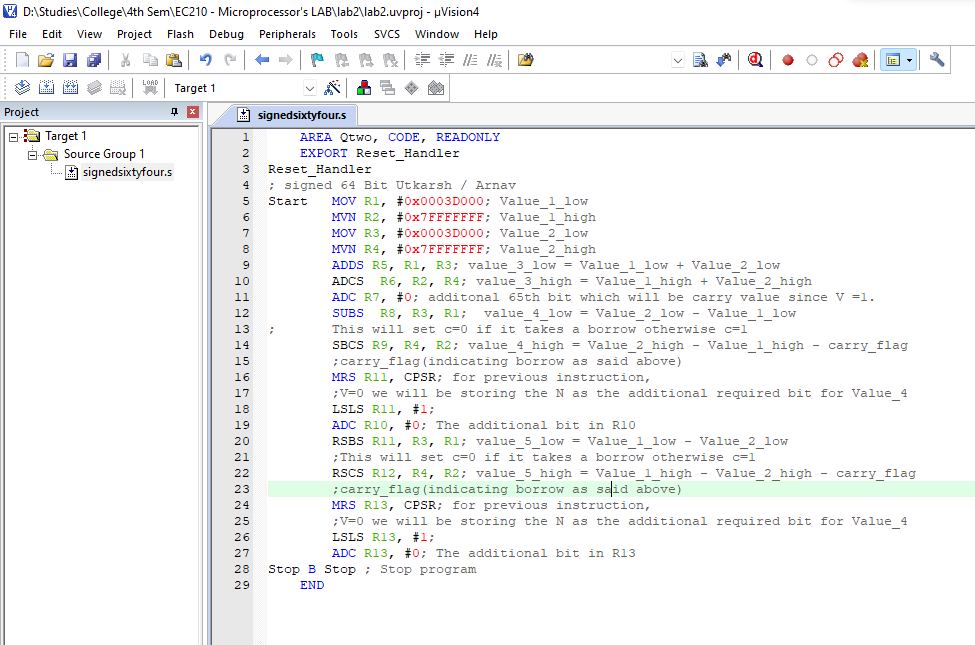
        LSLS R13, #1*;*

        ADC R13, #0*; The additional bit in R13*

Stop B Stop *; Stop program*

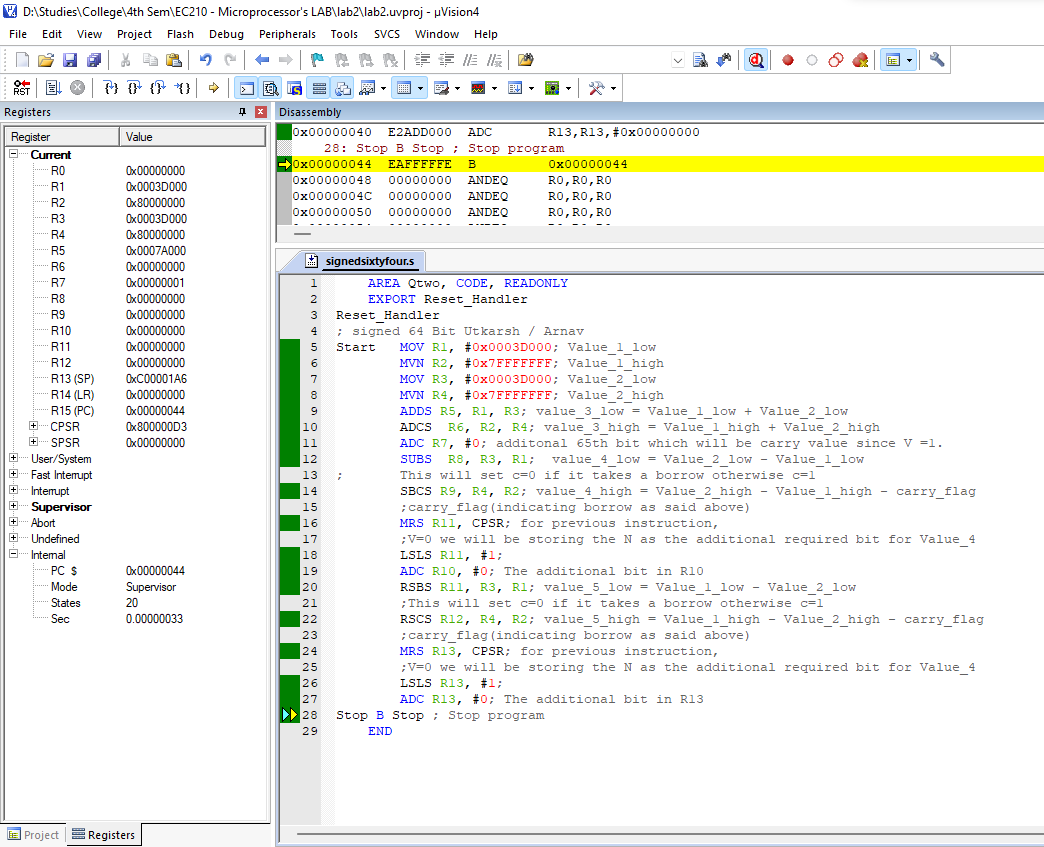
    END

Setup:



Debugging:

End Result:



(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*

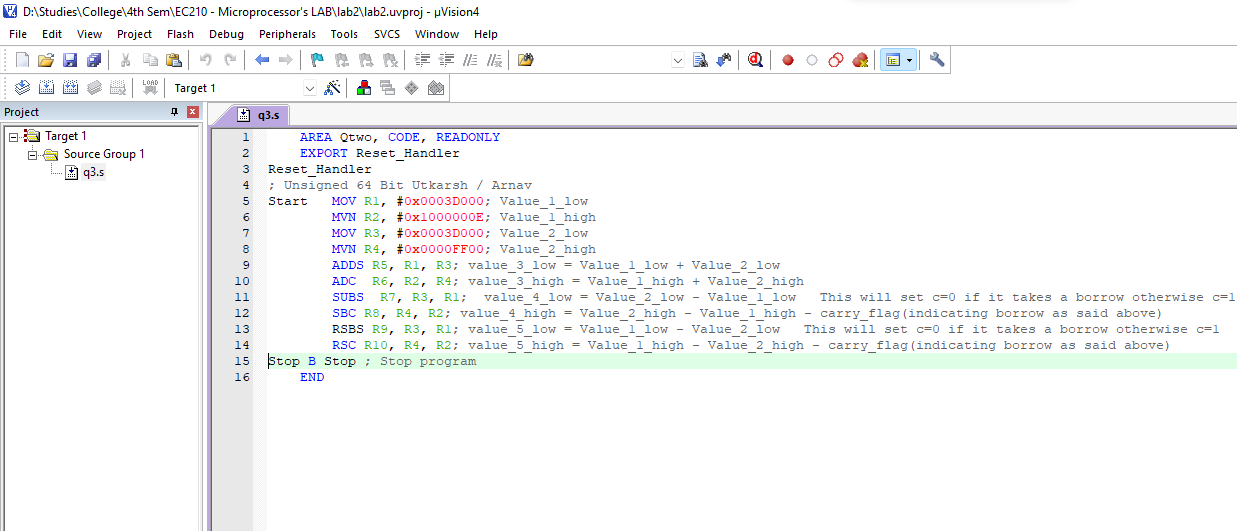
        RSC R10, R4, R2*; value\_5\_high = Value\_1\_high - Value\_2\_high - carry\_flag*

*; This will set c=0 if it takes a borrow otherwise c=1*

Stop B Stop *; Stop program*

    END

Setup:



Debugging:

End Result:

