

# Microprocessors LAB

## EC210

### Assignment 2

Submitted By:

Suhas S G

201EC263

Nikhil P Reddy

201EC241

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

ADD R3, R1, R2 => addition of R2 and R1 stored in R3 without updating the cpsr/(flags Z,C,V,N)

R3 : R1 + R2

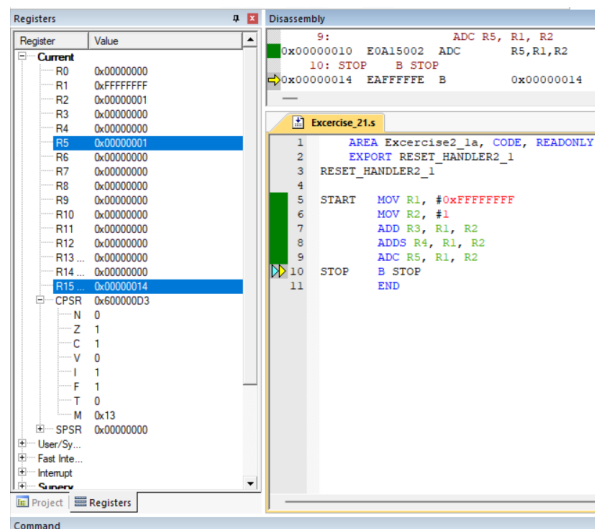
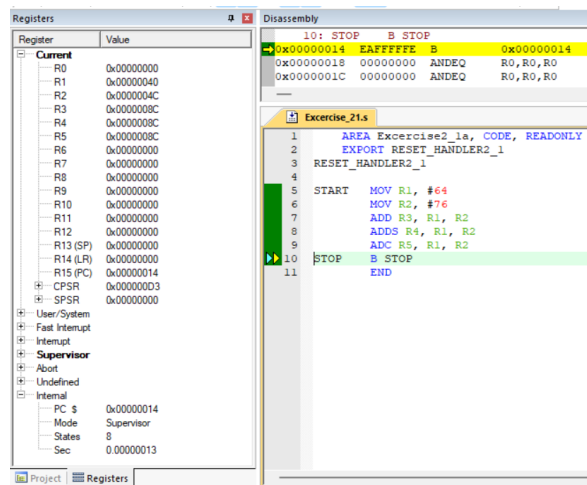
ADDS R3, R1, R2 => addition of R2 and R1 stored in R3 and updates the cpsr/(flags Z,C,V,N)

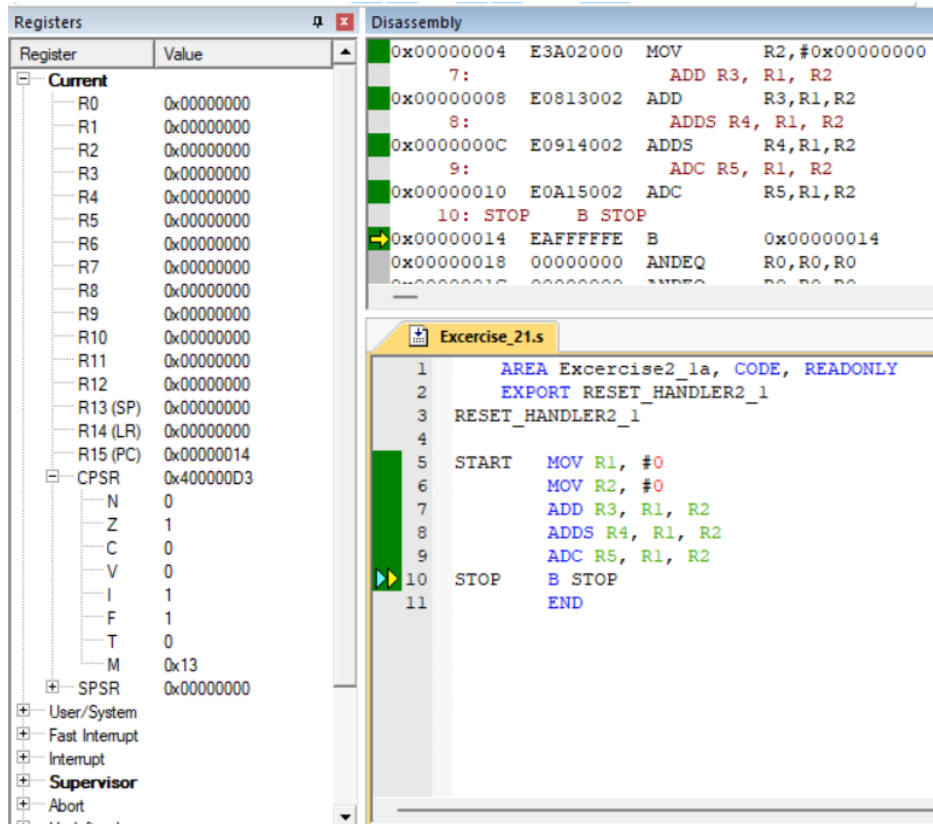
R3: R1 + R2

ADDC R3, R1, R2 => addition of R2, R1 and carry C stored in R3

R3: R1 + R2 + C

## OUTPUTS





As you can notice from the above examples the flags are only updated while using ADDS, and the carry is added to the sum using ADC

### (b) Subtract using SUB, SUBS, SBC

SUB R3, R1, R2 => subtraction of R1 from R2 stored in R3 without updating the cpsr/(flags Z,C,V,N)

R3 : R1 - R2

SUBS R3, R1, R2 => subtraction of R1 from R2 stored in R3 and updates the cpsr/(flags Z,C,V,N)

R3: R1 - R2

SBC R3, R1, R2 => subtraction of R1 and carry C from R2 stored in R3

R3: R1 - R2 - C

## OUTPUTS

Registers		Disassembly	
Register	Value		
<b>Current</b>		9:	SBC R5, R1, R2
R0	0x00000000	0x00000010	E0C15002 SBC R5,R1,R2
R1	0x0000005A	10: STOP	B STOP
R2	0x00000040	0x00000014	EFFFFFFF B 0x00000014
R3	0x0000001A	0x00000018	00000000 ANDEQ R0,R0,R0
R4	0x0000001A	0x0000001C	00000000 ANDEQ R0,R0,R0
R5	0x0000001A	0x00000020	00000000 ANDEQ R0,R0,R0
R6	0x00000000	0x00000024	00000000 ANDEQ R0,R0,R0
R7	0x00000000	0x00000028	00000000 ANDEQ R0,R0,R0
R8	0x00000000	0x0000002C	00000000 ANDEQ R0,R0,R0
R9	0x00000000		
R10	0x00000000		
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000014		
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			

Exercise_21.s	
1	AREA Exercise2_1b, CODE, READONLY
2	EXPORT RESET_HANDLER2_1
3	RESET_HANDLER2_1
4	
5	START MOV R1, #90
6	MOV R2, #64
7	SUB R3, R1, R2
8	SUBS R4, R1, R2
9	SBC R5, R1, R2
10	STOP B STOP
11	END

Registers		Disassembly	
Register	Value		
<b>Current</b>		9:	SBC R5, R1, R2
R0	0x00000000	0x00000010	E0C15002 SBC R5,R1,R2
R1	0x0000005A	10: STOP	B STOP
R2	0x0000005A	0x00000014	EFFFFFFF B 0x00000014
R3	0x00000000	0x00000018	00000000 ANDEQ R0,R0,R0
R4	0x00000000	0x0000001C	00000000 ANDEQ R0,R0,R0
R5	0x00000000	0x00000020	00000000 ANDEQ R0,R0,R0
R6	0x00000000	0x00000024	00000000 ANDEQ R0,R0,R0
R7	0x00000000	0x00000028	00000000 ANDEQ R0,R0,R0
R8	0x00000000	0x0000002C	00000000 ANDEQ R0,R0,R0
R9	0x00000000		
R10	0x00000000		
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000014		
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			

Exercise_21.s	
1	AREA Exercise2_1b, CODE, READONLY
2	EXPORT RESET_HANDLER2_1
3	RESET_HANDLER2_1
4	
5	START MOV R1, #90
6	MOV R2, #90
7	SUB R3, R1, R2
8	SUBS R4, R1, R2
9	SBC R5, R1, R2
10	STOP B STOP
11	END

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	0x00000000
R14	0x00000000
R15	0x00000014
CPSR	0x400000D3
N	0
Z	1
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

Address	Disassembly
9:	SBC R5, R1, R2
0x00000010	E0A15002 ADC R5, R1, R2
10: STOP	B STOP
0x00000014	EAFFFFFFE B 0x00000014
0x00000018	00000000 ANDEQ R0, R0, R0
0x0000001C	00000000 ANDEQ R0, R0, R0
0x00000020	00000000 ANDEQ R0, R0, R0
0x00000024	00000000 ANDEQ R0, R0, R0
0x00000028	00000000 ANDEQ R0, R0, R0
0x0000002C	00000000 ANDEQ R0, R0, R0

```

1 AREA Exercise2_1b, CODE, READONLY
2 EXPORT RESET_HANDLER2_1
3 RESET_HANDLER2_1
4
5 START MOV R1, #123
6       MOV R2, #123123
7       SUB R3, R1, R2
8       SUBS R4, R1, R2
9       SBC R5, R1, R2
10      STOP B STOP
11      END
  
```

Register	Value
R0	0x00000000
R1	0x0000005A
R2	0x0000005B
R3	0xFFFFFFFF
R4	0xFFFFFFFF
R5	0xFFFFFFFF
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13	0x00000000
R14	0x00000000
R15	0x00000014
CPSR	0x800000D3
N	1
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

Address	Disassembly
9:	SBC R5, R1, R2
0x00000010	E0C15002 SBC R5, R1, R2
10: STOP	B STOP
0x00000014	EAFFFFFFE B 0x00000014
0x00000018	00000000 ANDEQ R0, R0, R0
0x0000001C	00000000 ANDEQ R0, R0, R0
0x00000020	00000000 ANDEQ R0, R0, R0
0x00000024	00000000 ANDEQ R0, R0, R0
0x00000028	00000000 ANDEQ R0, R0, R0
0x0000002C	00000000 ANDEQ R0, R0, R0

```

1 AREA Exercise2_1b, CODE, READONLY
2 EXPORT RESET_HANDLER2_1
3 RESET_HANDLER2_1
4
5 START MOV R1, #90
6       MOV R2, #91
7       SUB R3, R1, R2
8       SUBS R4, R1, R2
9       SBC R5, R1, R2
10      STOP B STOP
11      END
  
```

### (c) Reverse subtract using RSB, RSC

RSB R3, R1, R2 => subtraction of R2 from R1 stored in R3 without updating the cpsr/(flags Z,C,V,N)

R3 : R2-R1

RSC R3, R1, R2 => subtraction of R2 and carry C from R1 stored in R3

R3: R1 - R2 - C

## OUTPUTS

The screenshot shows the Keil uVision IDE with the 'Registers' window on the left and the 'Disassembly' window on the right. The 'Registers' window displays the current state of registers R0 through R15, with R0 at 0x00000000, R1 at 0x0000005A, R2 at 0x0000005B, R3 at 0x00000001, R4 at 0x00000000, R5 at 0x00000000, R6 at 0x00000000, R7 at 0x00000000, R8 at 0x00000000, R9 at 0x00000000, R10 at 0x00000000, R11 at 0x00000000, R12 at 0x00000000, R13 at 0x00000000, R14 at 0x00000000, R15 at 0x00000000, and the CPSR at 0x000000D3. The 'Disassembly' window shows the assembly code for 'Exercise\_21.s', starting with '5: START MOV R1, #90' and ending with '10: STOP B STOP'. The code is currently at line 5.

Register	Value
R0	0x00000000
R1	0x0000005A
R2	0x0000005B
R3	0x00000001
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13	0x00000000
R14	0x00000000
R15	0x00000000
CPSR	0x000000D3

```
5: START MOV R1, #90
0x00000000 E3A0105A MOV R1, #0x0000005A
6: MOV R2, #91
0x00000004 E3A0205B MOV R2, #0x0000005B
7: RSB R3, R1, R2
0x00000008 E0613002 RSB R3, R1, R2
8: RSC R4, R1, R2
0x0000000C E0E14002 RSC R4, R1, R2
9: STOP B STOP
0x00000010 EAffffff B 0x00000010
```

Exercise\_21.s

```
1 AREA Exercise2_1b, CODE, READONLY
2 EXPORT RESET_HANDLER2_1
3 RESET_HANDLER2_1
4
5 START MOV R1, #90
6 MOV R2, #91
7 RSB R3, R1, R2
8 RSC R4, R1, R2
9 STOP B STOP
10 END
```

The screenshot shows the Keil uVision IDE with the 'Registers' window on the left and the 'Disassembly' window on the right. The 'Registers' window displays the current state of registers R0 through R15, with R0 at 0x00000000, R1 at 0x0000005B, R2 at 0x0000005A, R3 at 0xFFFFFFFF, R4 at 0xFFFFFFFF, R5 at 0x00000000, R6 at 0x00000000, R7 at 0x00000000, R8 at 0x00000000, R9 at 0x00000000, R10 at 0x00000000, R11 at 0x00000000, R12 at 0x00000000, R13 (SP) at 0x00000000, R14 (LR) at 0x00000000, R15 (PC) at 0x00000010, and the CPSR at 0x000000D3. The 'Disassembly' window shows the assembly code for 'Exercise\_21.s', starting with '5: START MOV R1, #91' and ending with '10: STOP B STOP'. The code is currently at line 5.

Register	Value
R0	0x00000000
R1	0x0000005B
R2	0x0000005A
R3	0xFFFFFFFF
R4	0xFFFFFFFF
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000010
CPSR	0x000000D3

```
5: START MOV R1, #91
0x00000000 E3A0105B MOV R1, #0x0000005B
6: MOV R2, #90
0x00000004 E3A0205A MOV R2, #0x0000005A
7: RSB R3, R1, R2
0x00000008 E0613002 RSB R3, R1, R2
8: RSC R4, R1, R2
0x0000000C E0E14002 RSC R4, R1, R2
9: STOP B STOP
0x00000010 EAffffff B 0x00000010
```

Exercise\_21.s

```
1 AREA Exercise2_1b, CODE, READONLY
2 EXPORT RESET_HANDLER2_1
3 RESET_HANDLER2_1
4
5 START MOV R1, #91
6 MOV R2, #90
7 RSB R3, R1, R2
8 RSC R4, R1, R2
9 STOP B STOP
10 END
```

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

Add using ADD, ADDS, ADC

The screenshot shows a debugger window with two panes. The left pane, titled 'Registers', displays the current state of 16 registers (R0-R15) and the CPSR. R0 is 0x00000000, R1 is 0xFFFFFFFF, R2 is 0xFFFFFFFF, R3 is 0xFFFFFFFF, R4 is 0xFFFFFFFF, R5 is 0xFFFFFFFF, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 is 0x00000000, R14 is 0x00000000, and R15 is 0x00000014. The CPSR shows N=1, Z=0, C=1, V=0, I=1, F=1, T=0, and M=0x13. The right pane, titled 'Disassembly', shows the assembly code for 'Exercise\_21.s'. The code starts with an area definition and an export statement, followed by a reset handler. The handler starts with MOV R1, #-91 and MOV R2, #-90, then performs ADD R3, R1, R2 and ADDS R4, R1, R2, followed by ADC R5, R1, R2. The code ends with a STOP instruction and B STOP. The current instruction being executed is ADC R5, R1, R2 at address 0x00000010.

Register	Value
R0	0x00000000
R1	0xFFFFFFFF
R2	0xFFFFFFFF
R3	0xFFFFFFFF
R4	0xFFFFFFFF
R5	0xFFFFFFFF
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13	0x00000000
R14	0x00000000
R15	0x00000014
CPSR	0xA00000D3
N	1
Z	0
C	1
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

```
9:          ADC R5, R1, R2
0x00000010 E0A15002 ADC      R5,R1,R2
10: STOP    B STOP
0x00000014 EAffffff B      0x00000014
0x00000018 00000000 ANDEQ   R0,R0,R0
0x0000001C 00000000 ANDEQ   R0,R0,R0
0x00000020 00000000 ANDEQ   R0,R0,R0
0x00000024 00000000 ANDEQ   R0,R0,R0
0x00000028 00000000 ANDEQ   R0,R0,R0
0x0000002C 00000000 ANDEQ   R0,R0,R0
```

Exercise\_21.s

```
1  AREA Exercise2_1b, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  MOV R1, #-91
6         MOV R2, #-90
7         ADD R3, R1, R2
8         ADDS R4, R1, R2
9         ADC R5, R1, R2
10        STOP    B STOP
11        END
```

Subtract using SUB, SUBS, SBC

The screenshot shows a debugger window with two panes. The left pane, titled 'Registers', displays the current state of 16 registers (R0-R15) and the CPSR. R0 is 0x00000000, R1 is 0xFFFFFFFF, R2 is 0xFFFFFFFF, R3 is 0xFFFFFFFF, R4 is 0xFFFFFFFF, R5 is 0xFFFFFFFF, R6 is 0x00000000, R7 is 0x00000000, R8 is 0x00000000, R9 is 0x00000000, R10 is 0x00000000, R11 is 0x00000000, R12 is 0x00000000, R13 (SP) is 0x00000000, R14 (LR) is 0x00000000, and R15 (PC) is 0x00000014. The CPSR shows N=1, Z=0, C=1, V=0, I=1, F=1, T=0, and M=0x13. The right pane, titled 'Disassembly', shows the assembly code for 'Exercise\_21.s'. The code starts with an area definition and an export statement, followed by a reset handler. The handler starts with MOV R1, #-91 and MOV R2, #-90, then performs SUB R3, R1, R2 and SUBS R4, R1, R2, followed by SBC R5, R1, R2. The code ends with a STOP instruction and B STOP. The current instruction being executed is SBC R5, R1, R2 at address 0x00000010.

Register	Value
R0	0x00000000
R1	0xFFFFFFFF
R2	0xFFFFFFFF
R3	0xFFFFFFFF
R4	0xFFFFFFFF
R5	0xFFFFFFFF
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x800000D3
SPSR	0x00000000

```
9:          SBC R5, R1, R2
0x00000010 E0C15002 SBC      R5,R1,R2
10: STOP    B STOP
0x00000014 EAffffff B      0x00000014
0x00000018 00000000 ANDEQ   R0,R0,R0
0x0000001C 00000000 ANDEQ   R0,R0,R0
0x00000020 00000000 ANDEQ   R0,R0,R0
0x00000024 00000000 ANDEQ   R0,R0,R0
0x00000028 00000000 ANDEQ   R0,R0,R0
0x0000002C 00000000 ANDEQ   R0,R0,R0
```

Exercise\_21.s

```
1  AREA Exercise2_1b, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  MOV R1, #-91
6         MOV R2, #-90
7         SUB R3, R1, R2
8         SUBS R4, R1, R2
9         SBC R5, R1, R2
10        STOP    B STOP
11        END
```

## Reverse subtract using RSB, RSC

The screenshot shows a debugger window with two panes: 'Registers' and 'Disassembly'.

**Registers Pane:**

Register	Value
R0	0x00000000
R1	0xFFFFFA5
R2	0xFFFFFA6
R3	0x00000001
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)	0x00000010
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

**Disassembly Pane:**

```

5: START  MOV R1, #-91
0x00000000 E3E0105A MVN    R1,#0x0000005A
6:        MOV R2, #-90
0x00000004 E3E02059 MVN    R2,#0x00000059
7:        RSB R3, R1, R2
0x00000008 E0613002 RSB    R3,R1,R2
8:        RSC R4, R1, R2
0x0000000C E0E14002 RSC    R4,R1,R2
9: STOP   B STOP
0x00000010 EAFFFFFFE B      0x00000010

```

**Exercise\_21.s:**

```

1  AREA Exercise2_1b, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  MOV R1, #-91
6         MOV R2, #-90
7         RSB R3, R1, R2
8         RSC R4, R1, R2
9  STOP   B STOP
10        END

```

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

For 64 bit unsigned numbers we use two different registers to store each number and perform operations by using the carry and overflow flag in the CPSR

## Add using ADD, ADDS, ADC

The screenshot shows a debugger window with two panes: 'Registers' and 'Disassembly'.

**Registers Pane:**

Register	Value
R0	0x00000000
R1	0x000004AF
R2	0x32ADBC14
R3	0xAFF123F0
R4	0xF1264923
R5	0x500EE0BE
R6	0x418772F1
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

**Disassembly Pane:**

```

9:        ADDS R6, R2, R4; add the least significant words
0x00000010 E0526004 SUBS    R6,R2,R4
10:       ADC R5, R1, R3; add the most significant words
0x00000014 E0C15003 SBC    R5,R1,R3
11: STOP   B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
0x00000018 EAFFFFFFE B      0x00000018
0x0000001C 000004AF ANDEQ  R0,R0,PC,LSR #9
0x00000020 32ADBC14 ADCCC  R11,R13,#0x00001400
0x00000024 AFF123F0 SWIGE  0x00F123F0
0x00000028 F1264923 (???)

```

**Exercise\_21.s:**

```

1  AREA Exercise, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  LDR R1, =0x000004AF; R1,R2 together form one 64 bit number(MSB is in R1)
6         LDR R2, =0x32ADBC14;
7         LDR R3, =0xAFF123F0; R3,R4 together form another 64 bit number(MSB is in R3)
8         LDR R4, =0xF1264923;
9         ADDS R6, R2, R4; add the least significant words
10        ADC R5, R1, R3; add the most significant words
11 STOP   B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
12        END

```



## Subtract using SUB, SUBS, SBC

The screenshot shows a debugger window with two panes: 'Registers' and 'Disassembly'.

**Registers Pane:**

Register	Value
R0	0x00000000
R1	0x000004AF
R2	0x32ADBC14
R3	0xAFF123F0
R4	0xF1264923
R5	0x500EE0BE
R6	0x418772F1
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x000000D3
N	0
Z	0
C	0
V	0
I	1
F	1
T	0
M	0x13
SPSR	0x00000000

**Disassembly Pane:**

```

9:          SUBS R6, R2, R4; add the least significant words
0x00000010 E0526004 SUBS    R6,R2,R4
10:         SBC R5, R1, R3; add the most significant words
0x00000014 E0C15003 SBC     R5,R1,R3
11: STOP    B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
0x00000018 EAF00000 B       0x00000018
0x0000001C 000004AF ANDEQ   R0,R0,PC,LSR #9
0x00000020 32ADBC14 ADCCC   R11,R13,#0x00001400
0x00000024 AFF123F0 SWIGE   0x00F123F0
0x00000028 F1264923 (???)

```

**Exercise\_21.s:**

```

1  AREA Exercise, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  LDR R1, =0x000004AF; R1,R2 together form one 64 bit number(MSB is in R1)
6         LDR R2, =0x32ADBC14;
7         LDR R3, =0xAFF123F0; R3,R4 together form another 64 bit number(MSB is in R3)
8         LDR R4, =0xF1264923;
9         SUBS R6, R2, R4; add the least significant words
10        SBC R5, R1, R3; add the most significant words
11 STOP   B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
12        END

```

## Reverse subtract using RSB, RSC

The screenshot shows a debugger window with two panes: 'Registers' and 'Disassembly'.

**Registers Pane:**

Register	Value
R0	0x00000000
R1	0x000004AF
R2	0x32ADBC14
R3	0xAFF123F0
R4	0xF1264923
R5	0x500EE0BE
R6	0x418772F1
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000018
CPSR	0x000000D3
SPSR	0x00000000

**Disassembly Pane:**

```

9:          RSB R6, R4, R2; add the least significant words
0x00000010 E0746002 RSB    R6,R4,R2
10:         RSC R5, R3, R1; add the most significant words
0x00000014 E0E35001 RSC     R5,R3,R1
11: STOP    B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
0x00000018 EAF00000 B       0x00000018
0x0000001C 000004AF ANDEQ   R0,R0,PC,LSR #9
0x00000020 32ADBC14 ADCCC   R11,R13,#0x00001400
0x00000024 AFF123F0 SWIGE   0x00F123F0
0x00000028 F1264923 (???)

```

**Exercise\_21.s:**

```

1  AREA Exercise, CODE, READONLY
2  EXPORT RESET_HANDLER2_1
3  RESET_HANDLER2_1
4
5  START  LDR R1, =0x000004AF; R1,R2 together form one 64 bit number(MSB is in R1)
6         LDR R2, =0x32ADBC14;
7         LDR R3, =0xAFF123F0; R3,R4 together form another 64 bit number(MSB is in R3)
8         LDR R4, =0xF1264923;
9         RSB R6, R4, R2; add the least significant words
10        RSC R5, R3, R1; add the most significant words
11 STOP   B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
12        END

```

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

For 64 bit signed numbers we use two different registers to store each number and perform operations by using the carry and overflow flag in the CPSR.

And the numbers are represented using 2's complement method otherwise the process is same as the signed numbers

### Add using ADD, ADDS, ADC

The screenshot displays the Keil uVision IDE interface. On the left, the 'Registers' window shows the current state of the ARM registers. The 'Disassembly' window on the right shows the assembly code for 'Exercise\_21.s'. The code is as follows:

```
6: LDR R2, =0x32ADBC14; note here numbers are represented using 2's compliment
0x00000004 E59F2014 LDR R2, [PC, #0x0014]
7: LDR R3, =0xAFF123F0; R3,R4 together form another 64 bit number(MSB is in R3)
0x00000008 E59F3014 LDR R3, [PC, #0x0014]
8: LDR R4, =0xF1264923;
0x0000000C E59F4014 LDR R4, [PC, #0x0014]
9: ADDS R6, R2, R4; add the least significant words
0x00000010 E0746002 RSBS R6, R4, R2
10: ADC R5, R1, R3; add the most significant words
0x00000014 E0E35001 RSC R5, R3, R1
11: STOP B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
12: END
```

### Subtract using SUB, SUBS, SBC

The screenshot displays the Keil uVision IDE interface. On the left, the 'Registers' window shows the current state of the ARM registers. The 'Disassembly' window on the right shows the assembly code for 'Exercise\_21.s'. The code is as follows:

```
9: SUBS R6, R2, R4; add the least significant words
0x00000010 E0746002 RSBS R6, R4, R2
10: SBC R5, R1, R3; add the most significant words
0x00000014 E0E35001 RSC R5, R3, R1
11: STOP B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
0x00000018 EAF00000 B 0x00000018
0x0000001C 000004AF ANDEQ R0, R0, PC, LSR #9
0x00000020 32ADBC14 ADCCC R11, R13, #0x00001400
0x00000024 AFF123F0 SWIG R0, R0, R0
0x00000028 F1264923 (???)
0x0000002C 00000000 MVEB R0, R0, R0
12: END
```

## Reverse subtract using RSB, RSC

Registers		Disassembly	
Register	Value		
R0	0x00000000	9:	RSBS R6, R4, R2; sub the least significant words
R1	0x000004AF	0x00000010	E0746002 RSBS R6,R4,R2
R2	0x32ADBC14	10:	RSC R5, R3, R1; sub the most significant words
R3	0xAFF123F0	0x00000014	E0E35001 RSC R5,R3,R1
R4	0xF1264923	11: STOP	B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
R5	0x500EE0BE	0x00000018	EFFFFFFE B 0x00000018
R6	0x418772F1	0x0000001C	000004AF ANDEQ R0,R0,PC,LSR #9
R7	0x00000000	0x00000020	32ADBC14 ADCCC R11,R13,#0x00001400
R8	0x00000000	0x00000024	AFF123F0 SWIGE 0x00F123F0
R9	0x00000000	0x00000028	F1264923 (???)
R...	0x00000000		
R...	0x00000000		
R...	0x00000000		
R...	0x00000000		
R...	0x00000000		
R...	0x00000018		
C...	0x000000D3		
	0		
	0		
	0		
	0		
	1		
	1		
	0		
	0x13		
S...	0x00000000		
User/...			
Fast I...			
Interr...			
Sup...			
Abort			

Exercise_21.s	
1	AREA Excercise, CODE, READONLY
2	EXPORT RESET_HANDLER
3	RESET_HANDLER
4	
5	START LDR R1, =0x000004AF; R1,R2 together form one 64 bit number(MSB is in R1)
6	LDR R2, =0x32ADBC14; note here numbers are represented using 2's complement
7	LDR R3, =0xAFF123F0; R3,R4 together form another 64 bit number(MSB is in R3)
8	LDR R4, =0xF1264923;
9	RSBS R6, R4, R2; sub the least significant words
10	RSC R5, R3, R1; sub the most significant words
11	STOP B STOP ; R6,R5 together form one 64 bit sum (MSB is in R6)
12	END