Praktikum 3:

Aufgabe 1:

Der Befehl LSD R0, R1, #3 ist eine arithmetische function welche "Logical schift Right" heißt, also werden alle bits um einen bestimmten Wert #x nach rechts verschoben.
Wenn ein Bit aus dem register geschoben wird, dann wir dieser veloren. Von dem most significant bit aus werden bits nachgeschoben, in diesem Fall werden immer NULL nachgeschoben.

In kurz, Die 32 Bitabfolge wird um 3 bit verschoben und der rest wird mit Nullen nachgeschoben

RSBS R0,R1,R1,LSL #2
LSL #2 ist eine arithmetische funktion welche die 32 bitreihenfolge, um 2 bits nach links verschiebt aber dabei die statusbits erhält indem 1 oder 0 nachgeschoben wird, je statusbits.

Diese nach Links verschobene zahl ist praktisch das 2^2 multiplizierte der originalen Zahl, welche in R1 war.

Diese Zahl wird wir in umgekehrter reihenfolge mit R1 subtrahiert und in R0 gespeichert, dabei werden die Statusbits in den CSRP gespeichert

Aufgabe 2:

N: Negativ
Das negativ Statusbit wird bei einer rechnung oder logischen operation gesetzt, wenn sich der Sign des Wertes verändert
Z.b: 1-2 = -1
In dem BSP wird das signbit durch die operation von 0 auf 1 verändert

Z:
Das zero Statusbit wird bei rechnungen oder logischen operationen ausgegeben, wenn eine operation NULL ergibt
Z.b. 0+0

C:
Das carry bit wird gesetzt bei operationen, wenn diese eine nicht anzeigbare, zu große Zahl ergeben. Mann kann es aber auch ohne overflow ausgeben, indem man 7FFFFFFF + FFFFFFFF miteinader verrechnet, die Zahl ist anzeigbar aber setzt den Carry, da bei der operation zwar ein overflow aber die Zahl die angezeigt wird ist noch anzeigbar.

V:
Das overflow bit zeigt das eine operation eine zu große oder zu kleine Zahl nicht angezeigt werden kann und nicht in ihrerer vollen größe angezeigt werden kann.
BSP:
ADDS R0, R1, R2
LSRS R0, #1
Als erst wird ein Overflow und das Negativ Flag gesetzt, dann mit shift das negativ Flag entfernt:

Aufgabe 3:

Aufgabe 4:

Um das zweier.compliment einer Zahl zu bilden, so muss man Alle bits flippen und die gebitflippete Zahl um 1 addieren.

5 Methoden, um dies zu simulieren:

Multiplizieren mit -1
R4, #1
R9 = /
R8 = /

Mov R8, R4
Mov R9, R1
Mul R4,R8,R9

Das doppelte der zahl von sich abziehen

Sub R5, R5, R5, LSL #1;

Die Zahl von 0 abziehen

RSB R6, R6, #0;

Das 1er kompliment bilden und diese dann um 1 addieren

Mvn R7,R7
ADDS R7,#1

Zusatzaufgabe 2:

Idee:
Man nehme sich die beiden Zahlen und addiert sie jeweils mit einander und zieht dann bei dem jeweils anderen um die gleiche Zahl wieder ab

```
main:
        ldr     r1, =0x1234ABCD
        ldr     r2, =0xDCBA4321
@ hier Ihre Loesung ergaenzen
        ADD     r1, r1, r2      @ Addiere beide Zahlen
        SUB     r2, r1, r2      @ Ziehe die eine Zahl von der Summe ab,
        SUB     r1, r1, r2      @ um die andere Zahl zu erhalten


        bx      lr      @ Ruecksprung zum aufrufenden Programm
```