Aufgaben

1) Die vorzeichenlose Zahl in r3 soll durch 8 geteilt werden. Dass Ergebnis soll in r4 stehen. Geben Sie den Befehl an.

2) Das .data-Feld beginne bei Adresse 0x6000. Welcher Wert (Hex) steht für Var1 in der Symboltabelle?

```
.section        .data
Var1:           .byte   12,'A',0xC,'4'                  Var1:_____
```

3) Das .data-Feld beginne bei Adresse 0x6000. Geben Sie die Speicherinalte (Hex.) von Adresse 0x6000 - 0x6003 an.

```
.section        .data                   |   0x6000          0x6001          0x6002
                0x6003
Tab:            .word   12,'A',0xC,'4'  |   ______          ______          ______
_____
```

4) Folgendes Datenfeld sei gegeben. Geben Sie die Assemblerbefehle an, um das dritte Datenwort des Feldes nach r1 zu kopieren.

```
.section        .data
Var1:           .byte    0x12, 0xAA, 0xA2
```

5) Was steht in r0 nach folgendem Befehl? (Hex.)

```
ldr r0, =0x12345678
```

6) In welchem Wertebereich muss r0 liegen, damit ein Sprung nach LOOP erfolgt?

```
cmp    r0, #16
bls    LOOP
```

7) In welchem Wertebereich muss r0 liegen, damit ein Sprung nach LOOP erfolgt?

```
mov    r1, #-16
cmp    r0,r1
bgt    LOOP
```

8) In welchem Wertebereich muss r0 liegen, damit ein Sprung nach LOOP erfolgt?

```
ldr    r1, #0xFFFFFFF0
ands   r0,r1
beq    LOOP
```

Hilfe: '0' = 30h, 'A' = 41h, 'a' = 61h1

Lösungen

Aufgabe 1

mov r4, r3, LSR #3

Ausführlich:

Bei den meisten Registeroperationen kann der letzte Operand vor der Operation um bis zu 32 bit verschoben oder rotiert werden. Fünf Typen von Schiebeoperationen stehen zur Verfügung:

1. ASR Arithmetisches Schieben nach rechts
2. LSL Logisches Schieben nach links <-
3. **LSR Logisches Schieben nach rechts ->**
4. ROR Rotieren nach rechts
5. RRX Rotieren nach rechts über carry

Schieben nach rechts entspricht dem Teilen durch 2, das schieben nach links einer Multiplikation mit 2. ((x/2)/2)/2 = x/8, deshalb wird um 3 Stellen geschoben. Vorsicht bei dieser Art von Division und ungeraden Zahlen:

43: 0010 1011 wird um 1 nach rechts verschoben, also durch 2 geteilt:
0001 0101 (1) = 21 wobei 21.5 korrekt wäre

Auch möglich ist es, um den Wert von Registern zu Shiften:

mov r2, r0, LSL **r3** [r2] [r0] linksverschoben um den Wert in r3

Weitere Beispiele:

add r9, r5, r5, LSL #2 [r9] [r5] + LSL2(r5) d.h. [r5] * 5
sub r0, r4, r5, LSL #2 [r0] [r4] - LSL2(r5) d.h. [r4] – [r5]*4

Aufgabe 2

Var1: 0x6000

Ausführlich:

Symboltabelle beinhaltet nur die Adressen auf die Daten, nicht die Daten selbst. Unser .data Feld beginnt bei 0x6000 und Var1 steht unmittelbar am Anfang, daher stehen auch die Daten ab Adresse 0x6000.

Aufgabe 3

0x6000	0x6001	0x6002	0x6003
0x0C	0x00	0x00	0x00

Ausführlich:

.section .data

Hilfe: ‘0’ = 30h, ‘A’ = 41h, ‘a’ = 61h2

Tab: .byte 12,'A',0xC,'4'
reserviert und initialisiert 4 Wörter (á 4 Byte)
Tab bezeichnet die Anfangsadresse des Wort-Feldes

12d = **0x0C**, da .., 9d = 0x9, 10d = 0xA, 11d = 0xB, 12d = 0xC, ...
da jedes Element ein ganzes .word darstellen soll sind die ersten 4 Byte, also 0x6000 - 0x6003, von der 0x0C belegt, die nächsten 4 für das 'A' (41h) usw..
Die LittleEndian Speicherung bewirkt, dass unser Wort gedreht wird. Würden wir es in ein Register laden, würde es wieder gedreht werden, und wir erhielten 0x00....00C

Aufgabe 4

```
ldr     r0, =Var1       Adresse von Var1 in r0 laden
mov     r1,[r0,#8]      r0 + 8 Byte, also (8/4=) 2 Wörter weiter
```

Ausführlich:

ldr: Laden von Speicherinhalten in Register
ldr Zielregister, [Basisadressregister, #+/-12-bit-Konstante] Konstante liegt also zwischen – 4095 … +4095

Weitere Beispiele:

```
ldr r0, [r1, #4]        Wort laden
ldrb r0, [r1, #4]       Byte laden, die vorderen 3 Byte des Registers werden mit 0 besetzt
ldrh r0, [r1, #4]       Halbwort laden
```

Aufgabe 5

0x12345678

Aufgabe 6

$0 \leq r0 \leq 16$ (dezimal)

Aufgabe 7

$-15 \leq r0 \leq 0x7FFFFFFF$ (dezimal)

Aufgabe 8

$0x0 \leq r0 \leq 0xF$ (hexdezimal)

Hilfe: '0' = 30h, 'A' = 41h, 'a' = 61h3