ÜBUNG: Angabe von Konstanten

Wie könnte man die Befehle noch schreiben ?
Welche Schreibweise ist am zweckmäßigsten (am besten lesbar).

```
a)  mov   r1, #41             ; Alter des Studenten
b)  mov   r2, #0x44           ; Lade Steuerbits
c)  mov   r3, #2_10000111     ; Lade Steuerbits
d)  mov   r4, #'C'            ; Lade den zu suchenden Buchstaben
e)  mov   r5, # -5            ; Lade Kontostand
f)  mov   r6, #65             ; Lade Alter des Großvaters
```

a) mov r1, #0x29
mov r1, #0b 0010 1001 } in allen Fällen steht das gleiche im Speicher !

b) mov r2, #68
mov r2, #2_0100 0100
mov r2, #'D'

c) ....   d) ....

e) -5 : ... 0000 0101
... 1111 1010   Kom1
1
... 1111 1011   Kom2

mov r5, #0xFFFFFFFB
mov r5, #4294967291 } geht nicht, da der ARM so große Konstanten nicht "immediate" laden kann !

Ausweg : statt mov → mvn (move negated)
(s.u.)

ÜBUNG: Adressierungsart immediate - erlaubte Konstanten

Welche Konstanten sind bei der Adressierungsart „*immediate*“ erlaubt ?

a)	234
b)	2357
c)	0xfe02
d)	0xfe0000
e)	0xfe0002
f)	‘A‘

a) 234 < 255 ⇒ erlaubt

b) bei Cortex M4 ⇒ erlaubt $(< 2^{16})$

c) " " " " "

d) durch Linksverschiebung von 0xfe um 16 Binärstellen erzeugbar ⇒ erlaubt (obwohl $> 2^{16}$)

$$\boxed{1111\ 1110}\ 0000\ 0000\ 0000\ 0000 \longleftarrow \boxed{1111\ 1110}$$

e) nicht erlaubt, da $> 2^{16}$ und nicht durch Linksverschiebung einer 8-bit-Zahl erzeugbar

f) 'A' ≙ 0x41 ⇒ erlaubt

ÜBUNG: Adressierungsarten "register, immediate"

Adr.	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009
Inh.	**12**	**23**	**34**	**45**	**56**	**67**	**78**	**89**	**9A**	**AB**

				3 +0	**2** +1	**1** +2	**0** +3	
	mov	r0, #0xaa	[r0] =	00	00	00	AA	→ immediate
	mov	r1, #170	[r1] =	00	00	00	AA	→ "
z=0	movs	r2, #0xaa00	[r2] =	00	00	AA	00	→ "
if(z=1)	moveq	r3, #0x0aa000	[r3] =	FF	FF	FF	FF	→ " (nicht ausgeführt)
	mov	r4, #0x1000	[r4] =	00	00	10	00	→ "
	ldrb	r5, [r4]	[r5] =	00	00	00	12	→ indirect, byte
*	ldr	r6, [r4]	[r6] =	45	34	23	12	→ indirect
	add	r4, #4	[r4] =	00	00	10	04	→ immediate
*	str	r6, [r4]	M[0x1004] =	12	23	34	45	→ indirect
	add	r4, #4	[r4] =	00	00	10	08	→ immediate
*	strh	r6, [r4]	M[0x1008] =	12	23			→ indirect, Halbword

$$170_{10} = (128 + 32 + 8 + 2)_{10} = 10101010_2 = AA_H$$

* wg. "little endian" (byte-swapping!) liegen die Bytes in umgekehrter Reihenfolge im Speicher!