8. ARM-Cortex-Befehle im Detail

- Ganzzahlarithmetik
- Kontrollstrukturen
- Bitmanipulation
- Schieben und Rotieren

8.2 Ganzzahl-Arithmetik

8.2.1 Übersicht

Aufruf: <**instruction**>{<cond>}{S} Rd, Rn, N

add	Binäre Addition
adc	Binäre Addition mit Berücksichtigung von Überträgen
sub	Binäre Subtraktion
sbc	Binäre Subtraktion mit Berücksichtigung von Überträgen
rsb	Umgekehrte binäre Subtr.
rsc	Umgekehrte binäre Subtr. mit Berücksichtigung von Überträgen
cmp	Vergleich zweier Integer-Operanden (verändert nur die Flags)
cmn	Vergleiche negiert (verändert nur die Flags)
neg	Negieren gemäß 2-er-Komplement
mul	Multiplikation zweier Zahlen (verschiedene Varianten)

8.1 Registeroperationen: Zur Notation des letzten Operators (**N**)

Aufruf: <instruction>{<cond>}{S} Rd, Rn, **N**

Bei den Registeroperationen (mov, add, sub, and, ….) hat der letzte Operator eine der folgenden Formen:

Operation N	**Syntax des letzten Operanden**	
Immediate	*# imm*	*(max. 8 Bit und Linksverschiebungen 1..31)*
Register	Rm	
Log. shift left by immediate	Rm, LSL *# shift*	
Log. shift left by register	Rm, LSL Rs	
Log. shift right by immediate	Rm, LSR *# shift*	
Log. shift right by register	Rm, LSR Rs	
Arith. shift right by immediate	Rm, ASR *# shift*	
Arith. shift right by register	Rm, ASR Rs	
Rotate right by immediate	Rm, ROR *# shift*	
Rotate right by register	Rm, ROR Rs	
Rotate right with extend	Rm, RRX	

8.2.2 Addition und Subtraktion

Aufruf: <instruction>{<cond>}{S} Rd, Rn, N

ADD	Addieren zweier 32-bit Werte	*Rd = Rn + N*
ADC	Addieren zweier 32-bit Werte mit carry	*Rd = Rn + N + carry*
SUB	Subtraktion zweier 32-bit Werte	*Rd = Rn – N*
SBC	Subtraktion zweier 32-bit Werte mit carry	*Rd = Rn – N - ~carry*
RSB	inverse Subtraktion zweier 32-bit Werte	*Rd = N – Rn*
RSC	inverse Subtraktion zweier 32-bit Werte mit carry	*Rd = N – Rn - ~carry*

Beispiele:

- add r0, **r1** [r0] ← [r0] +[r1]
- subeq r0, r1, **#2** [r0] ← [r1] – 2, falls Z=1
- subs r0, r1, r2, LSL #2 [r0] ← [r1] – [r2]*4, Flags passend setzen

8.2.3 Vergleich

Aufruf: <instruction>{<cond>} Rn, N

Bei den Vergleichsoperationen werden **nur die Flags** verändert,
d.h. es gibt **kein Zielregister** !

CMP	Vergleiche	*Rn - N*
CMN	Vergleiche negiert	*Rn + N*
TEQ	Vergleiche auf Gleichheit zweier 32-bit Werte → Z=1 , wenn Rn = N	*Rn xor N*
TST	Teste Bits 32-bit Werte → Z=1 , wenn es keine 1-Übereinstimmungen gibt	*Rn and N*

Beispiele:

- cmp r0, r1 [r0] - [r1], Flags entsprechend dem Ergebnis
- teq r0, #0xf0 [r0] xor 0xF0, Flags entsprechend dem Ergebnis
- tst r0, #2_11110000 [r0] and 0xF0, Flags entsprechend dem Ergebnis

ÜBUNG: Vergleiche

Wie sind die Flags N, Z, C, V nach folgenden Befehlen gesetzt?

```
mov r0, #15
mov r1, #30
mov r2, #-34            ; wird ersetzt durch:  mvn r2, #33
mov r3, #2_10100001     ; 0xA1 =  161

cmp r0, r1
cmp r1, r0
cmp r1, r0, LSL #1

teq r3, #161
teq r3, #0xFF

tst r3, #2_01011110
tst r3, #2_00000001
tst r3, #0
```

ERGÄNZUNG / EINSCHUB

5.3 Binäre Multiplikation

5.3.1 Multiplikation mit 2

Für Binärzahlen gilt: Multiplikation mit 2 = um eine Stelle nach links schieben

verdeutlichendes Beispiel: 45*2 = 90

45

Wertigkeit	128	64	**32**	16	**8**	**4**	2	**1**
Binärzahl	0	0	**1**	0	**1**	**1**	0	**1**

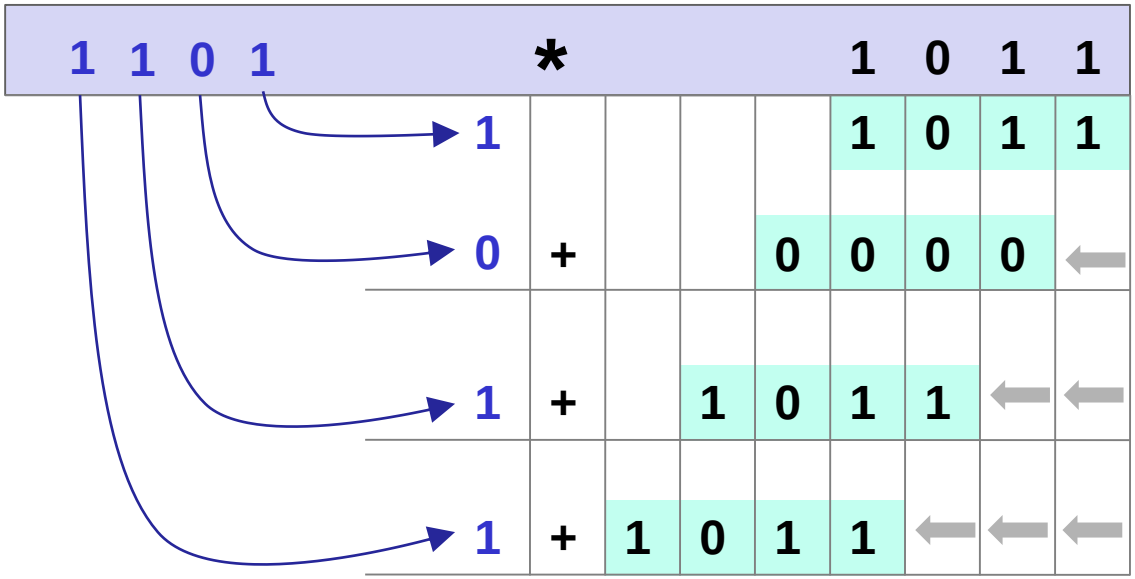
45 * 2

Wertigkeit	128	**64**	32	**16**	**8**	4	**2**	1
Binärzahl	0	**1**	0	**1**	**1**	0	**1**	0

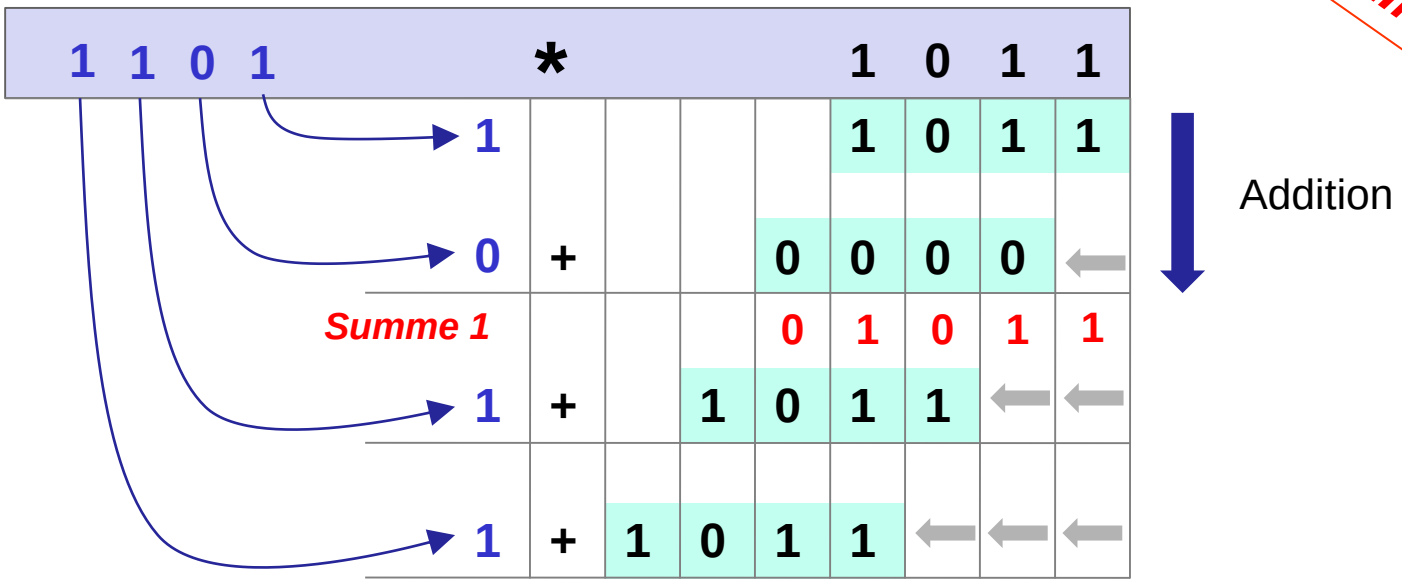
ERGÄNZUNG / EINSCHUB

5.3.2 Multiplikation von beliebigen Binärzahlen durch Schieben und Addieren

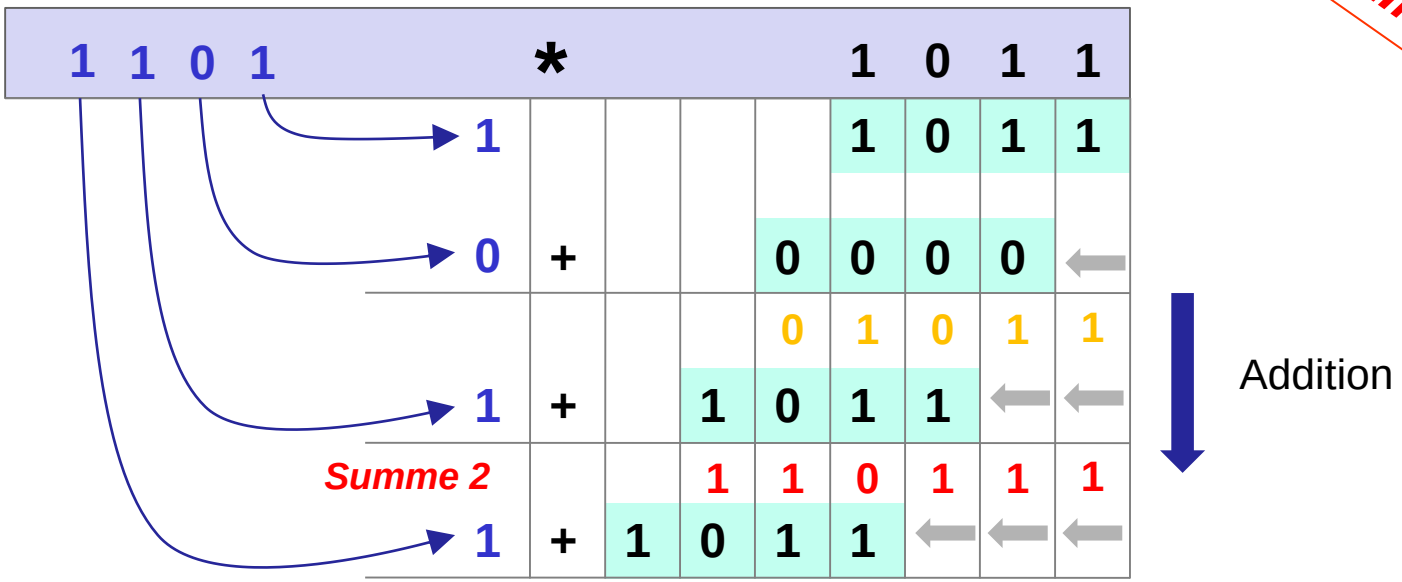
.



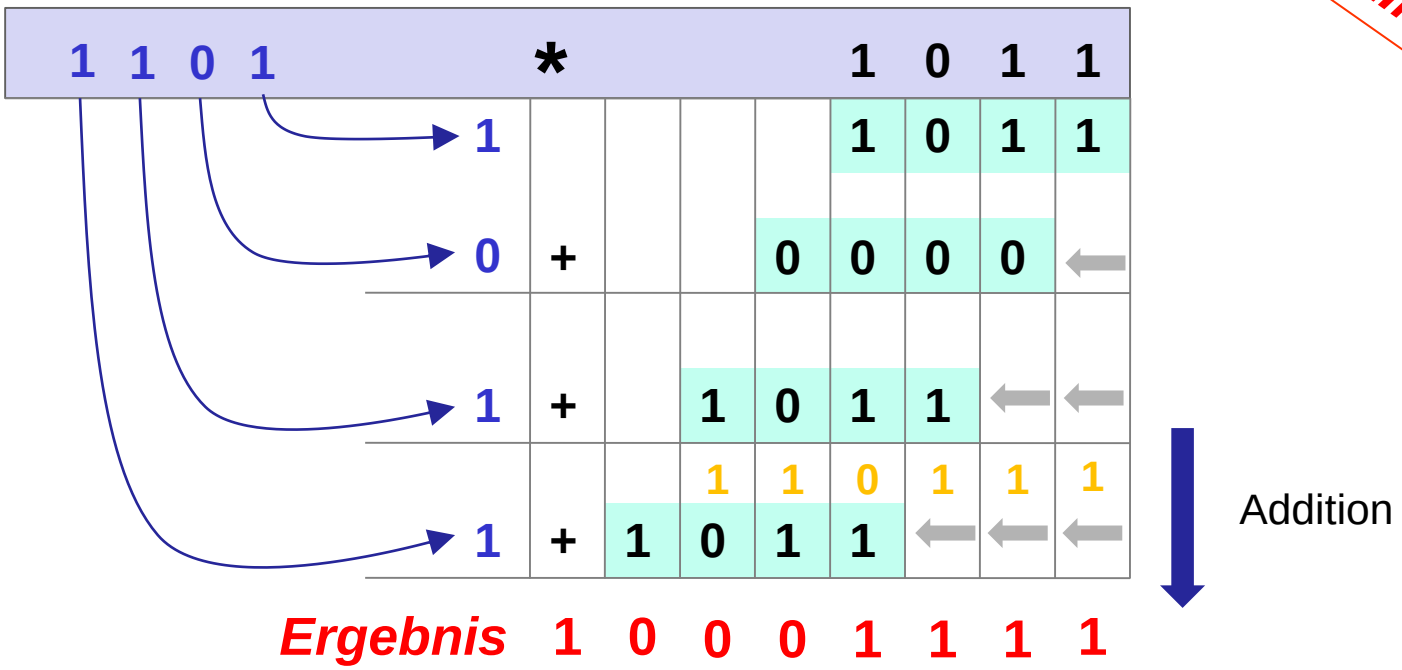
ERGÄNZUNG / EINSCHUB



ERGÄNZUNG / EINSCHUB



ERGÄNZUNG / EINSCHUB



ERGÄNZUNG / EINSCHUB

[Binäre Division]

[Division durch 2-er Potenzen]

Für Binärzahlen gilt: Division durch 2 = um eine Stelle nach **rechts** schieben

Achtung: Inhalt der letzten Stelle(n) vor Division = **Rest** der Division

wieder ein Beispiel: 45 DIV 4 = 11 Rest 1

Rest = 1 → 1/4

45

Wertigkeit	128	64	**32**	16	**8**	**4**	2	**1**
Binärzahl	0	0	**1**	0	**1**	**1**	0	**1**

45 DIV 4

Wertigkeit	128	64	32	16	**8**	4	**2**	**1**
Binärzahl	0	0	0	0	**1**	0	**1**	**1**

Beliebiger Divisor: Deutlich aufwändigere Rechnung

8.2.4 Multiplikation von unsigned und signed Werten

Aufruf:

```
mla {<cond>}{S}   Rd,   Rm,   Rs,  Rn
mul {<cond>}{S}   Rd,   Rm,   Rs
```

MLA	Multiplikation mit anschließender Addition	*Rd = Rm*Rs + Rn*
MUL	Multiplikation zweier 32-bit Werte	*Rd = Rm * Rs*

Achtung: Die Multiplikation zweier 32-Bit-Werte kann ein 64-Bit-Ergebnis zur Folge haben. Ein 32-Bit-Register reicht daher oft für das Ergebnis nicht aus.

Es gibt weitere Multiplikationsbefehle (*long multiply*) für signed und unsigned Integerzahlen. Dort wird das Ergebnis auf 2 Register (=64 Bit) aufgeteilt.

Parameter: NUR REGISTER!

Division von unsigned und signed Werten

UDIV	Division unsigned	*Rd = Rm / Rs*
SDIV	Division signed	*Rd = Rm / Rs*

Hinweis: Multiplikation/Division von 2er-Potenzen: LSL und LSR/ASR !

ÜBUNG: Arithm. Ausdruck

Schreiben Sie ein Programm, welches den folgenden Ausdruck berechnet:

$$(X^2 + Y^2)-(X+Y)^3$$

mit X=+5, Y=-7.

Worauf ist zu achten?