Fortsetzung: Control Program Status Register (CSPR)

- Negative Flag (31), Zero Flag (30), Carry Flag (29) und Overflow Flag (28) = Condition Code Flags (→ CCF)

Beeinflussung der CCF

- ARM-7 → Beeinflussung der CCF muss explizit gewünscht werden
- Beispiel:

```
ADD   R0, R1, R2    ;    R0 = R1 + R2, kein Einfluss auf CCF
ADDS  R0, R1, R2    ;    R0 = R1 + R2, angehängtes S bewirkt Beeinflussung der CCF
```

Bedingte Befehlsausführung

- Vermeidung von Sprüngen → Code kann effizienter ausgeführt werden
- CMP → compare (Verwendung CMP A,B) → beeinflusst immer die CCF
- Schreibweise für bedingte Ausführung → Befehl{Bedingung}{S} R_d, R_n, Operand 2
- Beispiele:

(1)

```
CMP   3,5           ;    Subtraktion 3 – 5 zum Vergleich
```

$$\left.\begin{array}{lll} & 0000 & 0011 \\ & 1111 & 1011 \\ C=0 & 1111 & 1110 \end{array}\right\} N=1, Z=0, V=0$$

```
ADDLT     R0, R1, R2    ;    wird ausgeführt, weil N ≠ V
ADDGTS    R0, R1, R2    ;    wird nicht ausgeführt, weil N ≠ V
```

(2)

```
CMP   5,3
```

$$\left.\begin{array}{lll} & 0000 & 0101 \\ & 1111 & 1101 \\ C=1 & 0000 & 0010 \end{array}\right\} N=1, Z=0, V=0$$

```
SUBMI     R3, R2, R0    ;    wird nicht ausgeführt, weil N = 0
SUBHS     R4, R1, R2    ;    wird ausgeführt, weil C = 1
ADDEQS    R0, R2, R3    ;    wird nicht ausgeführt, weil Z = 0
```

(3)

C-Programm übersetzen in Assembler:

Durchlauf 1 → a = 4, b = 3 → 4 != 3

Durchlauf 2 → a = 1, b = 3 → 1 != 3

Durchlauf 3 → a = 1, b = 2 → 1 != 2

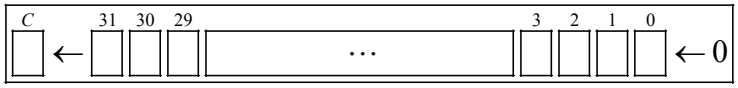
Durchlauf 4 → a = 1, b = 1 → END

Assembler-Programm:

```
start   CMP     a,b     ;   Vergleichen von a und b
        SUBGT   c, a, b ;   Subtraktion a = a – b (wenn N = V und Z = 0)
        SUBLT   b, b, a ;   Subtraktion b = b – a (wenn N ≠ V)
        B NE    start   ;   Sprung zurück zu start (wenn Z ≠ 0)
```

Barrel-Shifter

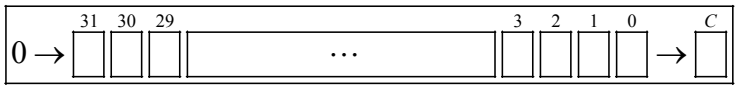
- Verschieben oder Rotieren des zweiten Operanden
- Befehl wird mit Anzahl der zu verschiebenden Bits versehen
- LSL (Logic Shift Left):



  - Um n Bit nach links schieben
  - Multiplikation mit 2^n
  - Im Carry steht Bit 32 – n
  - Beispiel LSL um 3 Bit:

$$\begin{array}{cccc} \ldots & 0000 & 0000 & 0011 & 1010 \\ & 0 & 0 & 3 & A \\ \ldots & 0000 & 0001 & 1101 & 0000 \\ & 0 & 1 & D & 0 \end{array}$$

- LSR (Logic Shift Right):

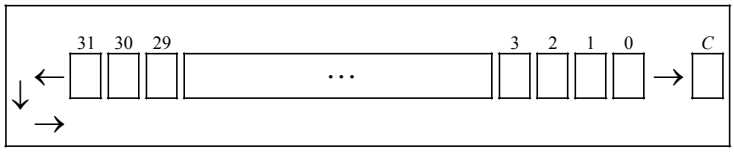


  - Um n Bit nach rechts schieben
  - Division mit 2^n
  - Im Carry steht Bit n – 1
  - Beispiel LSL um 2 Bit:

$$\begin{array}{cccc} \ldots & 0000 & 0000 & 0011 & 1010 \quad 3\cdot 16+10\cdot 1=58 \\ & 0 & 0 & 3 & A \\ \ldots & 0000 & 0000 & 0000 & 1110\rightarrow 1 \quad 14 \\ & 0 & 0 & 0 & E \end{array}$$

  - Fixed Point → grundsätzlich abrunden (korrektes Ergebnis wäre 14,5)

- ASR (Arithmetic Shift Right):



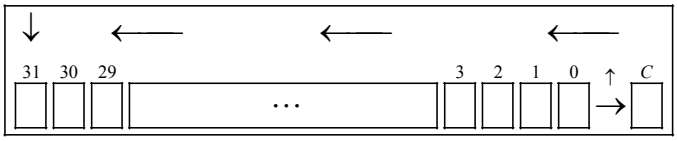
  - Um n Bit nach rechts verschieben
  - Vorzeichenrichtige Division mit 2^n
  - Carry = n - 1

- Beispiel ASR um 2 Bits:

$$\ldots\ 1111\quad 1111\quad 1100\quad 0110 \qquad 3\cdot 16+10\cdot 1=-58$$
$$F\qquad F\qquad C\qquad 6$$
$$\ldots\ 1111\quad 1111\quad 1111\quad 0001\rightarrow 1 \qquad -15$$
$$F\qquad F\qquad F\qquad 1$$
$$-\frac{58}{4}=-\frac{29}{2}=-14,5\Rightarrow -15$$

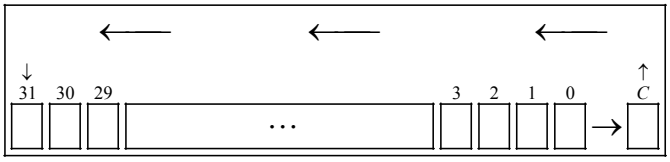
Abrunden = neg. Aufrunden

- ROR (Rotate Right)



  - Rotation um n Bits nach rechts
  - Carry = Bit 31 = n – 1

- RXR (Rotate Extended Right)



  - Rotieren um 1(!) Bit nach rechts
  - Carry als 33. Bit
  - C = n – 1 bzw. Bit 0

- Beispiel:

SUB	R1, R2, R3, LSL #2	;	R1 = R2 – (R3 * 2^2) = R2 – 4 * R3
MOV	R2, R5, LSL R3	;	MOV verschiebt 8 Bit, R2 = R5 * 2^{R3}
MOV	R1, R1, ASR #3	;	R1 = R1 – 2^3 (vorzeichenrichtig!)

- Aufgabe:
  R1 = 0x7060E2AD, R2 = 0xBC3

  ADD R0, R1, R2, LSL #4

R1:	0111	0000	0110	0000	1110	0010	1010	1101
R2:	0000	0000	0000	0000	0000	1011	1100	0011
LSL#4:	0000	0000	0000	0000	1011	1100	0011	0000
	0111	0000	0110	0001	1001	1110	1101	1101
R0 =	7	0	6	1	9	*E*	*D*	*D*

Arithmetik/Logik Befehle

- Allgemeine Instruktionsform: Befehl{Bedingung}{S} R_d, R_n, Operand 2
- Beispiele:

(1) RSB (reverse subtraction)

```
RSB     R0, R1, R2          ;   R0 = R2 – R1
RSB     R0, R1, R2, LSL #2  ;   R0 = (R2 * 2^2) – R1
```

(2) R_n = 0x8, Operand = 0x5, C = 0, C_{inv} = 1

```
SBC     Rd, Rn, Operand 2   ;   Rd = 0x8 – 0x5 – 1 = 2
```

(3) R_n = 0x4C, Operand = 0x55

```
BIC     Rd, Rn, Operand 2   ;   Rd = Rn AND NOT (Op 2)
```

$Op2$	0101	0101
$\overline{Op2}$	1010	1010

R_n	0100	1100
$\overline{Op2}$	1010	1010
R_d	0000	1000

(4) R1 = 0x19 (25), R2 = 0x39 (57)

```
ADD     R3, R1, R2          ;   R3 = 57 + 25 = 82
SUB     R3, R1, R2          ;   R3 = 25 – 57 = -32
RSB     R3, R1, R2          ;   R3 = 57 – 25 = 32
AND     R3, R1, R2          ;   R3 = 25
```

$R1$	0001	1001
$R2$	0011	1001
$R3$	0001	1001

```
EOR     R3, R1, R2          ;   R3 = 32
```

$R1$	0001	1001
$R2$	0011	1001
$R3$	0010	0000