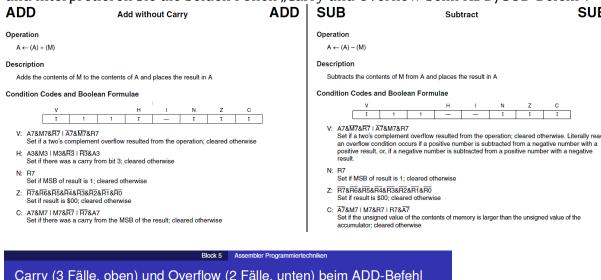
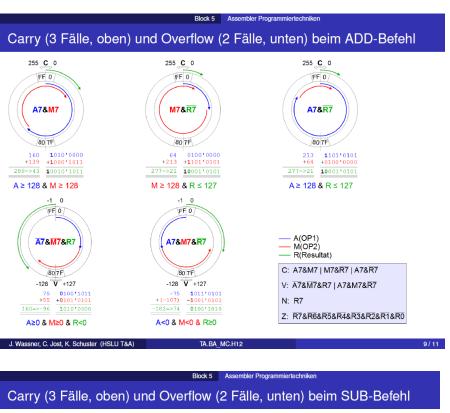
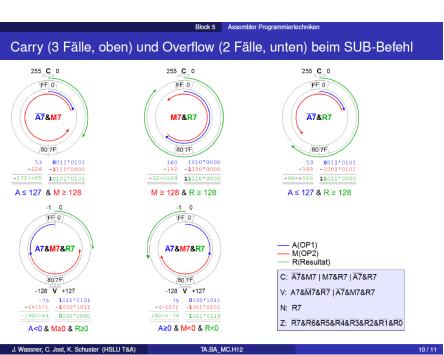
## Arithmetik mit 8-Bit Zahlen

1. Studieren Sie die Befehle ADD und SUB im Freescale "HCS08 Family Reference Manual" und interpretieren Sie die beiden Folien "Carry und Overflow beim ADD/SUB-Befehl".







2. Schreiben Sie einen Programmteil, welcher 2 Operanden von den Adressen \$00C0 und \$00C1 einliesst und nach einer noch anzugebenden arithmetischen Verknüpfung (Aufgabe 1.3 und 1.4) das Resultat ins X-Register schreibt und die aktuellen Werte der Flags VNZC auf 4 LEDs wie folgt ausgibt:

```
C \rightarrow PTF0 = LED B FL

Z \rightarrow PTF1 = LED R FL

V \rightarrow PTE7 = LED B FR
```

 $N \rightarrow PTF2 = LED R FR$ 

Die LED sind schön 1:1 wie die Flags im CCR sind. D.h. man muss die Flags / Bits nicht schieben und kann sie direkt auf den Port F resp. E ausgeben.

```
; Rear-LED ON, damit sichtbar ist, dass der MC-Car noch läuft (Save Battery)
         LDA
                                     ; LED Rear On --> uP still running ;-)
              PTDDD
         STA
         STA
              PTDD
; Start des Benutzer-Codes
userMain: LDA
               #$07
         STA
               PTFDD
                                      ; Pin0-3, Port F DataDirection = Output
         LDA
               #$80
                                      : Pin 7. Port E DataDirection
         STA
               PTEDD
         LDA
               #$FF
                                       ; All LED OFF
         STA
               PTED
         STA
               PTFD
         JMP
               AUFGABE1
                                       ; Jump to Aufgabe X
 ;---- Aufgabe 4.1 ------
              #$00
AUFGABE1: LDA
                                     ; Default-Wert 00 in Speicher C0 und C1 speichern
         STA
               $00C0
         STA
               $00C1
         LDA
               $00C0
                                       ; Load OP1
Loop:
         ADD
                                      ; ADD OP2
         ;SUB
                                       ; SUB OP2
                                       ; Result -> X // Transfer Accumulator to X
         TAX
         TPA
                                      ; Transfer Processor Status Byte to Accumulator
                                      ; (CCR -> Akku)
                                      ; Invertiere Akku (CCR) because of low-active LED
                                      ; V -> PTE7
               PTED
         STA
                                      ; NZC -> PTF2..0
         STA
               PTFD
         BRA
               Loop
                                       ; Endlosschleife
```

- 3. Benutzen Sie den unter 1.2 vorbereiteten Programmteil, um die Flagsetzung beim ADD Befehl zu analysieren. Legen Sie dazu vor dem Programmablauf zweckmässige Operanden auf die Adressen \$00C0 und \$00C1 (z.B mit dem Debugger Befehl >mem). Verifizieren Sie die erwarteten Flags auf der Anzeige der roten und blauen FrontLEDs und stellen Sie den Bezug zu den Zahlenkreisdiagrammen auf den Folien her. siehe oben!
- 4. Machen Sie dasselbe wie in Aufgabe 1.3 für den Befehl SUB. siehe oben!

## Bitmanipulationen

Schreiben Sie ein kleines Programm, welches das Carry-Flag im CCR mittels Bit-Maskierung löscht! Kontrollieren Sie Ihr Programm, indem Sie das Carry-Flag zuerst mit >reg setzen.

```
TPA
 AND #$FE
 TAP
; Rear-LED ON, damit sichtbar ist, dass der MC-Car noch läuft (Save Battery)
         LDA #$04
                                     ; LED Rear On --> uP still running ;-)
         STA
               PTDDD
         STA
              PTDD
; Start des Benutzer-Codes
               #$07
userMain: LDA
                                     ; Pin0-3, Port F DataDirection = Output
         STA
               PTFDD
         LDA
               #$80
              PTEDD
                                      ; Pin 7, Port E DataDirection
         T<sub>1</sub>DA
               #$FF
                                     ; All LED OFF
         STA
               PTED
                                      ; Jump to Aufgabe 1 / 2 or 3
         JMP
              AUFGARE2
;---- Aufgabe 4.2 -----
         ; Carry-Flag im CCR mittels Bit-Maskierung löschen
         ; vorher Flag setzen, z.B. mittels Debugger Shell: reg SR=0x01
AUFGABE2: NOP
                                      ; set Carry-Flag now --> SR=0X01
                                      ; CCR -> Akku (Processor Status Byte)
                                      ; Clear Carry-Flag (Bit0) %00000001
         AND
              #$FE
                                      ; Akku -> CCR
         TAP
         ; Overflow-Flag mittels Bit-Maskierung setzen: reg SR=0x00
                                      ; set Overflow-Flag now --> SR=0x00
                                       ; CCR -> Akku
         TPA
                                      ; Set Overflow-Flag (Bit7) %10000000
         ORA
              #$80
         TAP
                                       ; Akku -> CCR
         ; direkte setzen des Carry-Flag
                                       ; Set Carry-Flag
         ; direkte löschen des Carry-Flag
                                      ; Clear Carry-Flag
         CLC
EndLoop2: BRA AUFGABE2
                                      ; Endlos-Loop
```

2. Schreiben Sie ein kleines Programm, welches das Overflow-Flag mittels Bit-Maskierung setzt!

```
siehe oben!
 TPA
 ORA #$80
 TAP
```

Es gibt einige Befehle, welches das direkte Setzen (Sxx von SET) bzw. Löschen (Cxx von CLEAR) zulassen. Wie heissen die Befehle, welche dasselbe wie unter Aufgabe 2.1 und 2.2 tun?

```
; Set Carry-Flag
SEC
CLC
     : Clear Carry-Flag
```

## **BRANCH-Befehle**

1. Informieren Sie sich im HSLU\_HCS08\_Programming\_Guide über die BRANCH-Befehle.

```
Branch if Carry Bit Clear (C = 0
; BCS
            Branch if Carry Bit Set (C = 1)
            Branch if Not Equal (Z = 0)
; BNE
; BEQ
            Branch if Equal (Z = 1)
 BHCC
            Branch if Half Carry Bit Clear (H = 0)
; BHCS
            Branch if Half Carry Bit Set (H = 1)
: BMC
            Branch if Interrupt Mask Clear (I = 0)
; BMS
            Branch if Interrupt Mask Set (I = 1)
; BPL
            Branch if Plus (N = 0)
; BMI
            Branch if Minus (N = 1)
; BRCLR n
            Branch if Bit n in Memory Clear
; BRSET n Branch if Bit n in Memory Set
            Branch if Greater Than or Equal To (N \text{ and } V = 0)
; BGT
            Branch if Greater Than (Z \text{ or } (N \text{ and } V) = 0)
            Branch if Higher (C or Z = 0)
; BHT
; BHS
            Branch if Higher or Same (Same as BCC)
            Branch if Less Than or Equal To (Z or (N and V) = 1)
; BLO
            Branch if Lower (C = 1)
            Branch if Lower or Same (C or Z = 1)
; BLS
            Branch if Less Than (signed) (N and V = 1)
: BLT
```

Wie heissen und wie funktionieren die Branch-Befehle mit denen man gleichzeitig maskieren kann?

```
BRCLR n, Addr, Label ; Verzweigt nach Label, wenn Bit n der Speicherstelle ; an Adresse Addr nicht gesetzt ist (Addr nur DIR)

BRSET n, Addr, Label ; Verzweigt nach Label, wenn Bit n der Speicherstelle ; an Adresse Addr gesetzt ist (Addr nur DIR)
```

 Schreiben Sie ein kleines Programm, welches mittels eines Compare-Befehls zwei 8-Bit Operanden vergleicht und dann mittels verschiedener BRANCH Befehle verzweigt.
 Machen Sie sich so die Bedeutung verschiedener Branch Befehle klar (BCS, BHI, etc.).
 Signalisation: Kein Sprung Schalte LED B FL ein (PTF0)
 Sprung Schalte LED B FR ein (PTE7)

```
;---- Aufgabe 4.3 --
AUFGABE3: LDA
                #$FF
                                         ; All LED OFF
          STA
                PTED
          STA
                PTFD
          LDA
                #$0F
                                        ; Load $0F into Akku
          CMP
                                        ; Compare with $0F
                #$0F
                ISEOUAL
                                         ; Branch if Equal \rightarrow JMPEQUAL (Z = 0)
          BEO
         ; not Equal -> Schalte LED B FL ein (PTF0)
NOTEOUAL:
              #$FE
                                        ; LED Low-Active! 01 -> FE
          LDA
          STA
                PTFD
               AUFGABE3
                                        ; Loop
          BRA
         ; Equal -> Schalte LED B FR ein (PTE7)
ISEOUAL:
          LDA #$7F
                                        ; LED Low-Active! 80 -> 7F
          STA
                PTED
               AUFGABE3
          BRA
                                        : Tigop
```