Praktikum Mikrorechner 2 (Arithmetische Berechnungen und Makros)

G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

November 5, 2014

Addition mehrer Zahlen

- typischer Ausschnitt aus einem C-Programm

```
// Vereinbahrung von Variablen
// z.B. als 16-Bit positive ganze Zahlen
unsigned int x1, x2, x3, y;
main(){y = x1 + x2 + x3};
```

- beide Additionen erfordern dieselbe Befehlsfolge, nur mit anderen Adressen:

```
mov  a, x1_Byte0
add  a, x2_Byte0
mov  y_byte0, a
add  a, x1_Byte1
addc a, x2_Byte1
mov  y_Byte1, a
```

$y = x1 + x2$

```
mov  a, y_Byte0
add  a, x3_Byte0
mov  y_byte0, a
add  a, y_Byte1
addc a, x3_Byte1
mov  y_Byte1, a
```

$y = y + x3$

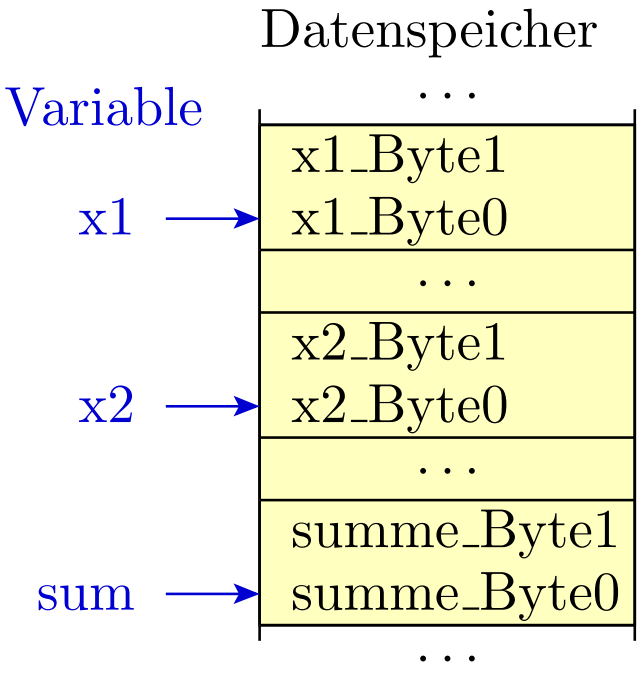
Macro: 1 x schreiben, mehrmals verwenden

```
<Macro-Name> MACRO <parameter1>, <parameter2>, ...
  <Anweisung 1>
  <Anweisung 2>
  <Anweisung 3>
  ...
  <Anweisung n>
ENDM
```

Macro für eine 16-Bit-Addition

```
 add16 MACRO summe, s1, s2
 ; Achtung: acc und cy werden verändert
  mov a, s1
  add a, s2
  mov summe, a
  mov a, s1 -1
  addc a, s2 -1
  mov summe -1, a
ENDM

x1   data 61h
x2   data 63h
sum  data 70h
...
add16 sum, x2, x1
```



Mehrfachverwendung des Macros

```
; Symbole vereinbaren

Start equ 100h
Monitor equ 0e300h
x1 data 61h
x2 data 63h
x3 data 65h
y  data 67h
org Start

; Anfangswerte einstellen
; Alternative zu "db 61=41"
mov x1,    #4fh
mov x1 -1, #23h
       ...
mov x3 -1, #0d2h
```

Datenspeicher

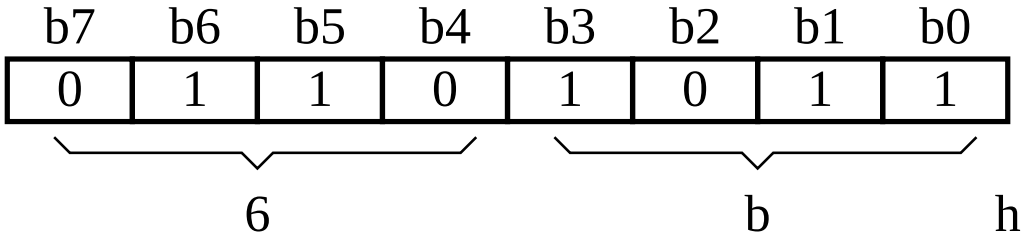
Variable	60h	23h	x1_Byte1
x1 →	61h	4fh	x1_Byte0
	62h	38h	x2_Byte1
x2 →	63h	12h	x2_Byte0
	64h	d2h	x3_Byte1
x3 →	65h	cfh	x3_Byte0
	66h	?	y_Byte1
y →	67h	?	y_Byte0

```
add16 y, x1, x2
add16 y, y, x3
ljmp Monitor
end
```

} 2 x 6 Befehle (mit less testen)

Unsigned Integer

Interpretation eines Bitvektors x



Bit
Byte

Hexadezimalzahl

als positive ganze Zahl mit dem Wert:

$$V_{\mathrm{UInt}}(\mathrm{x}) = \sum_{i=0}^{n-1} b_i \cdot 2^i$$

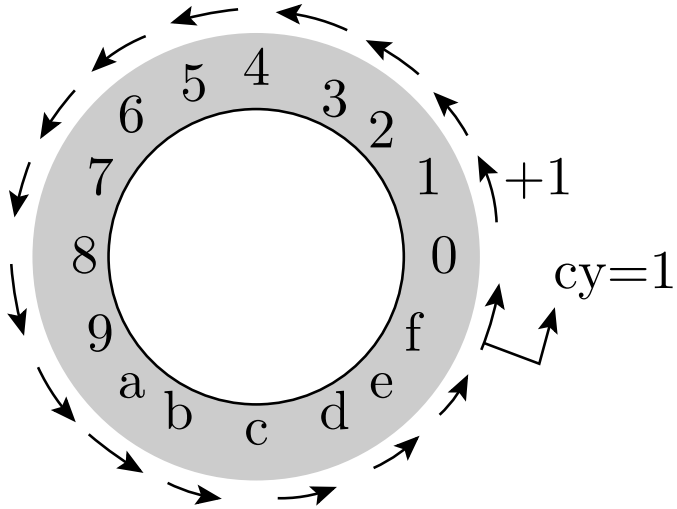
Wertebereich:

$$0 \leq V_{\mathrm{UInt}}(\mathrm{x}) \leq 2^n - 1$$

Wenn das Ergebnis einer Addition oder Subtraktion größer oder kleiner als der darstellbare Zahlenbereich ist $\Rightarrow$ Übertragsbit cy=1

Zahlenkreis und Additionsbefehle

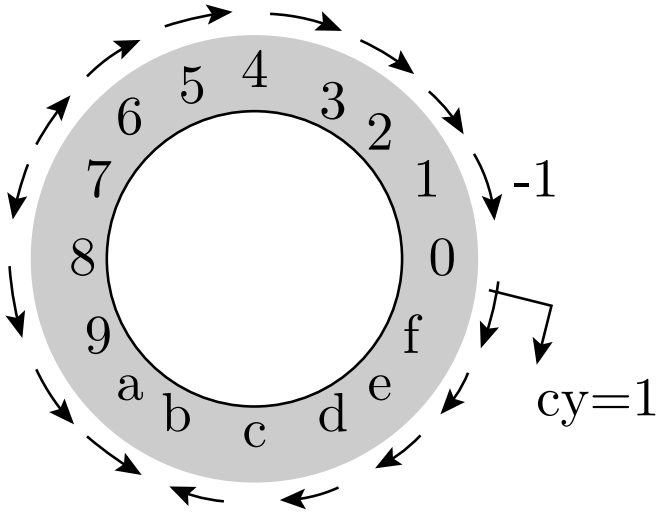
	...	3f	a2
+	...	1d	78
cy:		0 ← 1 ← 0	← 0
acc:		5d	1a




```
add  a, <Op2> ; acc := acc + <Op2>
addc a, <Op2> ; acc := acc + <Op2> + cy
inc <Op>      ; <Op> := <Op> + 1 (increment)
```

Subtraktionsbefehle

	...	5d	1a
-	...	1d	78
cy:	0 ←	1 ←	0 ← 0
acc:		3f	a2

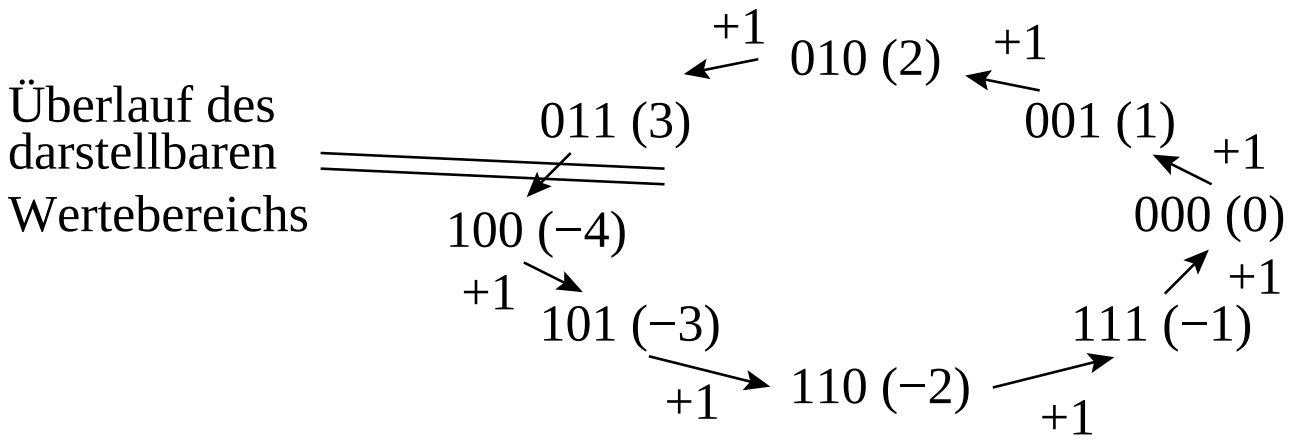



```
clr c           ; cy=0
subb a, <Op2>   ; acc := acc - <Op2> - cy
dec <Op>        ; <Op> = <Op> - 1 (decrement)
```

Integer

vorzeichenbehaftete ganze Zahlen

$$V_{\mathrm{Int}}(\mathrm{x}) = \begin{cases} V_{\mathrm{UInt}}(\mathrm{x}) & \text{wenn} \quad b^{n-1} = 0 \\ -2^n + V_{\mathrm{UInt}}(\mathrm{x}) & \text{wenn} \quad b^{n-1} = 1 \end{cases}$$



Vorzeichen: $b_{n-1} = 1 \Rightarrow$ negativ; $b_{n-1} = 0 \Rightarrow$ positiv
Überlauf- (O-)Flag: Wird gesetzt, wenn Ergebnis einer Addition oder Subtraktion kleiner min. oder größer max. ist.

Addition und Subtraktion

Die zirkulare Verschiebung des Überlaufpunktes auf dem Zahlenkreis hat keinen Einfluss auf den Algorithmus für Addition und die Subtraktion:

	Rechnung (hex)	positive Zahlen	vorzeichenbehaftet
	13a7h	5 031	5031
	+0b259h	+ 45 657	−19 879
Übertrag	0 0 1 1 0		
	c600h	50 688	−14 848

Wertebereich vorzeichenbehafteter Zahlen:

n	min.	max.
8	-128 (80h)	127 (7fh)
16	-32.768 (8000h)	32.767 (7fffh)

Negation: bitweise Negation + 1

Multiplikation (Unsigned)

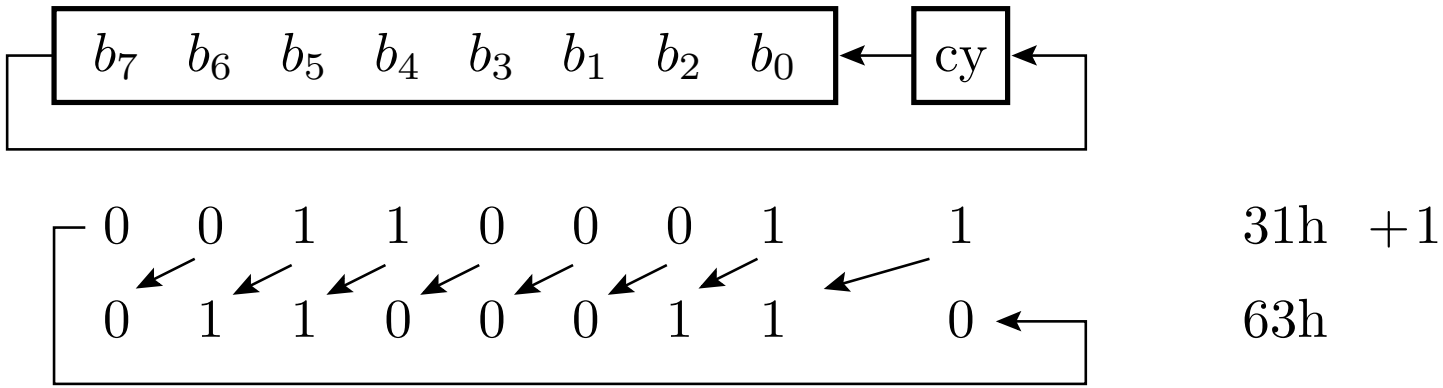
```
mul ab   ; acc := Byte0(acc*b), b   := Byte1(acc*b)

mult8 MACRO p, f1, f2
; genutzt: acc, b
  mov a, f1
  mov b, f2
  mul ab
  mov p, a    ; Lowbyte speichern
  mov p-1, b ; Highbyte speichern
ENDM
```

$$
\begin{array}{rcr}
32\text{h} & * & \text{f3h} \\
\hline
 & & 1\text{e6h} \\
 & + & 2\text{d90h} \\
\hline
 & & 2\text{f76h}
\end{array}
$$

Spezialbefehl für die Multiplikation mit 2

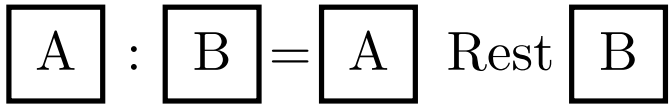
```
rlc a   ; acc := 2 * acc + cy (Rotation links)
```



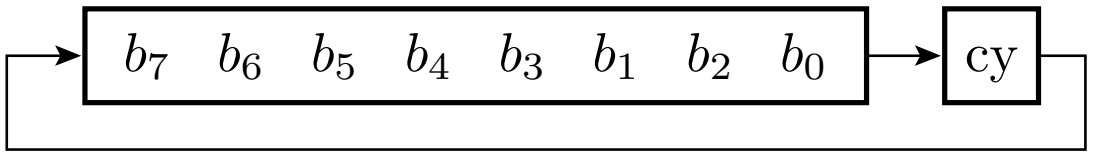
Achtung: cy vorher löschen

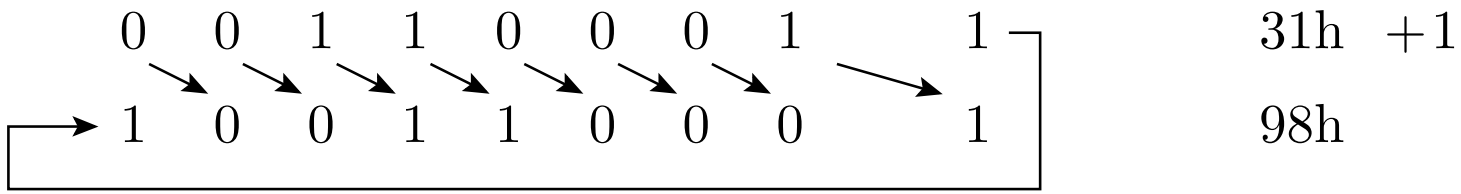
Division

```
div ab; acc := int(acc/b), b   := Rest(a/b)
```




```
rrc a ; acc := acc/2 + 256*cy (Rotation rechts)
```





Achtung: cy vorher löschen

Aufgabe 2.1: Die behandelten Macros ausprobieren

```
add16 MACRO summe, s1, s2
...
mult8 MACRO p, f1, f2
...
x1 equ 61h
x2 equ 63h
prod1 equ 65h
prod0 equ 67h
Monitor equ 0e300h
org 100h
add16 x1, x1, x2
mult8 prod0, prod1-1, prod1
ljmp Monitor
end
```

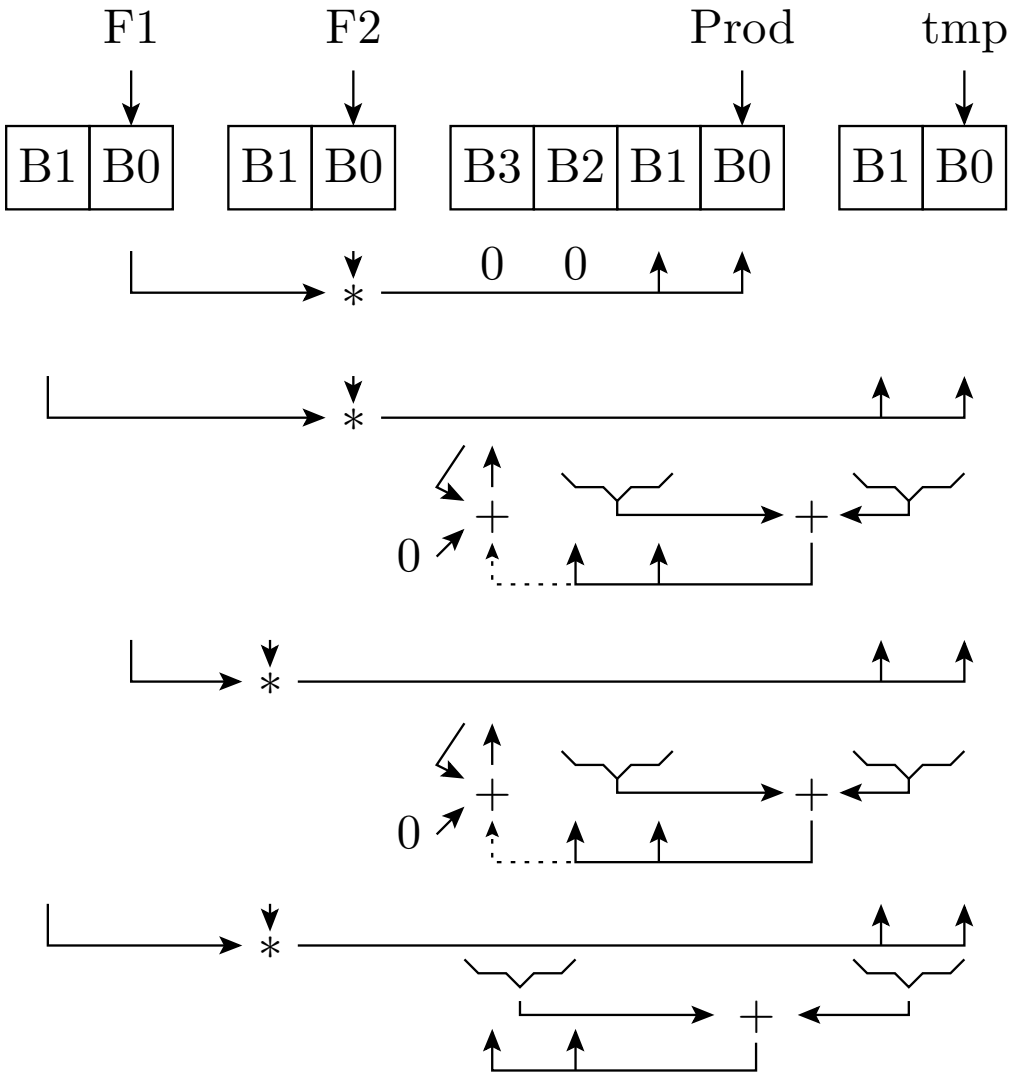
Variablen im Datenspeicher		
Sym.	Adr.	Inh.
	60h	13h
x1→	61h	45h
	62h	ffh
x2→	63h	feh
	64h	13h
p1→	65h	25h
	66h	?
p0→	67h	?

Aufgabe 2.2: Makro für 16-Bit-Multiplikation

- (Für Fortgeschrittene) Entwicklung eines Macros für eine 16-Bit-Multiplikation nach folgendem Algorithmus:

$$\begin{aligned} P &= (X_1 \cdot 2^8 + X_0) \cdot (Y_1 \cdot 2^8 + Y_0) \\ &= X_1 Y_1 \cdot 2^{16} + (X_1 Y_0 + X_0 Y_1) \cdot 2^8 + X_0 Y_0 \end{aligned}$$

unter Verwendung der Variablen- und Macro-Vereinbahrungen aus der Voraufgabe.




```
mult8 F1, F2, Prod
mov Prod-2, #0
mov Prod-3, #0

mult8 F1-1, F2, tmp

add16 tmp, Prod-1, Prod-1
mov a, Prod-3
addc a, #0
mov Prod-3, a

mult8 F1, F2-1, tmp

add16 tmp, Prod-1, Prod-1
mov a, Prod-3
addc a, #0
mov Prod-3, a

mult8 F1-1, F2-1, tmp

add16 tmp, Prod-2, Prod-2
```