UNIVERSITÄT LEIPZIG

Institut für Informatik

Seminaraufgaben

2.Semester – Sommersemester 2002

Abt. Technische Informatik
Gerätebeauftragter
Dr. rer.nat. Hans-Joachim Lieske
Tel.: [49]-0341-97 32213
Zimmer: HG 02-37
e-mail: lieske@informatik.uni-leipzig.de
www: http://www.ti-leipzig.de/~lieske/
Sprechstunde:Mi. 14^{00} – 15^{00} (Vorlesungszeit)

Aufgaben zur Übung
Grundlagen der Technische Informatik 2

5. Aufgabenkomplex - 1. Aufgabe

Assemblerprogramm für den Toy-Rechner

Entwickeln Sie ein Programm zur Multiplikation von zwei 16-Bit-Binärzahlen.

1. Entwickeln Sie einen Problemablaufplan
2. Schreiben Sie den Assemblercode und den Maschinencode entsprechend dem Beispielprogramm.
3. Probieren Sie das Programm an dem Beispiel: 36865 ● 12448 aus.

Bemerkungen:

Speicherbereich des Programms: 000h – 7FFh
Speicherbereich für die Daten: 800h – FFFh

1. Faktor ist auf Adresse: 800h
2. Faktor ist auf Adresse: 801h
   Die Adressen sind schon mit Zahlen belegt.

Das Ergebnis kommt auf Adresse: 802h, 803h
   Dabei ist in der Adresse 802h das niederwertige Datenwort des Ergebnisses (low-Datenwort)
   Und in der Adresse 803h das höherwertige Datenwort des Ergebnisses (high-Datenwort)

Das Programm beginne bei 000 h und gehe aufwärts.
Die Speicherzellen ab 804h können zum Zwischenspeichern benutzt werden.
Es können weitere Speicherzellen bzw. Variablen definiert werden.
Besonderheit: Mit einer Adresse wird immer ein Datenwort adressiert!
   Ein Datenwort ist hier eine 16 Bit-Speicherzelle.
   Der Vorsatz $ bezeichnet eine Hexadezimalzahl, hier wurde die Alternative, der Anhang h verwendet.
   ($800=800h)

z.B: <800>=25FDh=9725, <801>=B7A3h=47011, <803h,802h>=1640 0617h=457181975

9725 ● 47011=457181975

800h															
High-Byte								Low-Byte							
2				**5**				**F**				**D**			
0	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**

801h															
High-Byte								Low-Byte							
B				**7**				**A**				**3**			
1	**0**	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**

803h																802h															
High-Byte								Low-Byte								High-Byte								Low-Byte							
1				**6**				**4**				**0**				**0**				**6**				**1**				**7**			
0	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**1**

Hier einige Grundlagen:

```
; Program MULT16
;Variablen
; Faktor1   = 800h
; Faktor2   = 801h
; Erg-low  = 802h
; Erg-high = 803h
;Speicher1 = 804h          (Namen können frei gewählt werden)
;Speicher2 = 805h          (Namen können frei gewählt werden)
          ….

;Beginn des Programms

000h      …
          …

;Ende des Programms
```

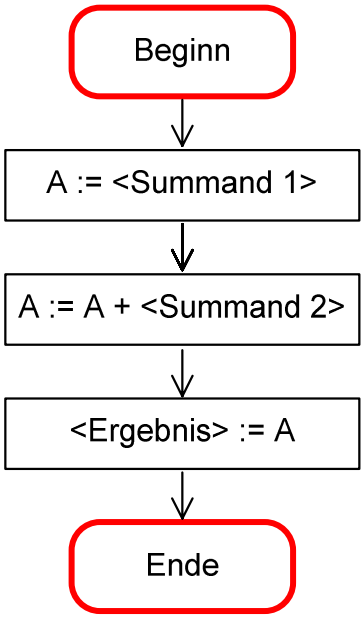
Beispiel für Problemablaufplan und Programm:

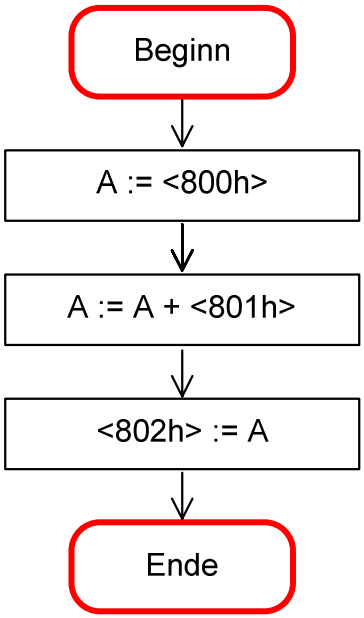
Addition von zwei 8-Bit Zahlen.

Die Zahl befindet sich im unteren Byte der 16-Bit Speicherzelle, das obere Byte ist Null.
Die Adressen sind schon mit Zahlen belegt.

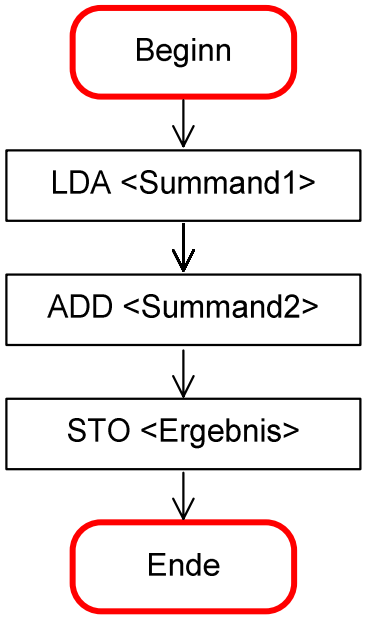
```
; Program ADD8
;Variablen
;Summand1 = 800h
;Summand2 = 801h
;Ergebnis     = 802h

…
```

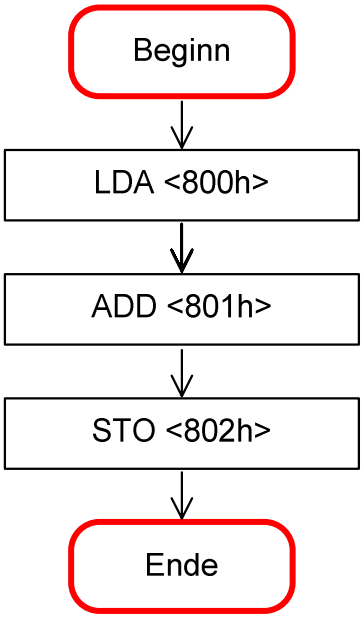
2 Möglichkeiten des Problemablaufplanes (1x vorgeschrieben):





Beispiele für Programmablaufplan (fakultativ):



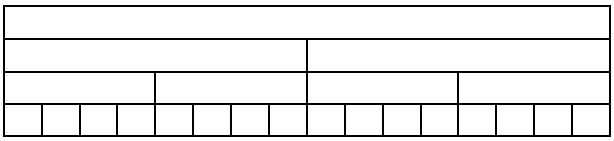


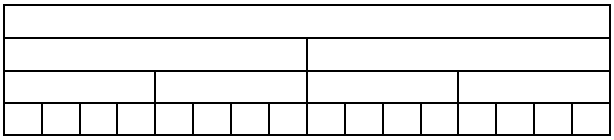
Beispiel für Programm (vorgeschrieben)

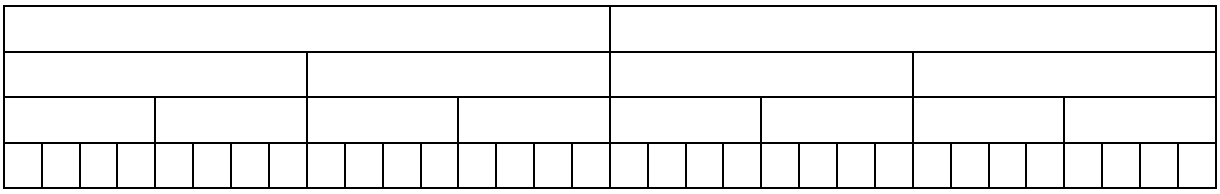
```
; Program ADD8
;Variablen
;Summand1= 800h
;Summand2= 801h
;Ergebnis   =  802h


;Beginn des Programms

000h     1800h   ;LDA <Summand1>      ;Lade Akkumulator mit 1., Summanden
001h     3801h   ;ADD <Summand2>      ;Addiere 2. Summanden zum Akkumulator
002h     0802h   ;STO <Ergebnis>      ;Speichere Akkumulator in Ergebnis

;Ende des Programms
```

Hier der Befehlssatz des TOY-Rechners

Opcode	Operation		Beschreibung
0	STO	<Adresse>	speichere den ACCU ins RAM an die Adresse
1	LDA	<Adresse>	lade ACCU mit dem Inhalt der Adresse
2	BRZ	<Adresse>	springe nach Adresse, wenn der ACCU Null ist
3	ADD	<Adresse>	addiere den Inhalt der Adresse zum ACCU
4	SUB	<Adresse>	subtrahiere den Inhalt der Adresse vom ACCU
5	OR	<Adresse>	logisches ODER des ACCUS mit dem Inhalt der Adresse
6	AND	<Adresse>	logisches UND des ACCUS mit dem Inhalt der Adresse
7	XOR	<Adresse>	logisches ExODER des ACCUS mit dem Inhalt der Adresse
8	NOT		logisches NICHT der Bits im ACCU
9	INC		inkrementiere den ACCU
10	DEC		dekrementiere den ACCU
11	ZRO		setze den ACCU auf NULL
12	NOP		nicht benutzt
13	NOP		nicht benutzt
14	NOP		nicht benutzt
15	NOP		nicht benutzt







Lösung

5. Aufgabenkomplex - 1. Aufgabe

Assemblerprogramm für den Toy-Rechner

(Gesamtpunktzahl=30 Punkte)

Entwickeln Sie ein Programm zur Multiplikation von zwei 16-Bit-Binärzahlen.

Viele Varianten möglich, hier eine:

1. Entwickeln Sie einen Problemablaufplan **15 Punkte**

1.1. Variante von Herrn Lars Geidel und Herrn Nils Michaelson

Das Programm beruht auf der Idee, dass eine Multiplikation der Zahlen A und B der B-fachen Addition der Zahl A entspricht.

Die Kontrolle auf carry (Übertrag) reduziert sich auf folgende Überlegung:

1. Bei der Addition tritt ein Übertrag auf, wenn die beiden höchstwertigen Bits eins sind.

$$\begin{aligned} (A) \quad & 1\, x_{14}x_{13}x_{12}\ x_{11}x_{10}x_9x_8\ x_7x_6x_5x_4\ x_3x_2x_1x_0 \\ (B) \quad & 1\, y_{14}y_{13}y_{12}\ y_{11}y_{10}y_9y_8\ y_7y_6y_5y_4\ y_3y_2y_1y_0 \end{aligned}$$

2. Bei der Addition tritt ein Übertrag auf, wenn das höchstwertige Bit der ersten Zahl eins ist und die Summe der beiden Restbits eins ergibt.

$$\begin{aligned} (A) \quad & 1\, x_{14}x_{13}x_{12}\ x_{11}x_{10}x_9x_8\ x_7x_6x_5x_4\ x_3x_2x_1x_0 \\ \textit{und} & \\ (\textit{rest}\ A) \quad & 0\, x_{14}x_{13}x_{12}\ x_{11}x_{10}x_9x_8\ x_7x_6x_5x_4\ x_3x_2x_1x_0 \\ (\textit{rest}\ B) \quad & +0\, y_{14}y_{13}y_{12}\ y_{11}y_{10}y_9y_8\ y_7y_6y_5y_4\ y_3y_2y_1y_0 \\ = \quad & 1\, u_{14}u_{13}u_{12}\ u_{11}u_{10}u_9u_8\ u_7u_6u_5u_4\ u_3u_2u_1u_0 \end{aligned}$$

3. Bei der Addition tritt ein Übertrag auf, wenn das höchstwertige Bit der zweiten Zahl eins ist und die Summe der beiden Restbits eins ergibt.

$$\begin{aligned} (B) \quad & 1\, y_{14}y_{13}y_{12}\ y_{11}y_{10}y_9y_8\ y_7y_6y_5y_4\ y_3y_2y_1y \\ \textit{und} & \\ (\textit{rest}\ A) \quad & 0\, x_{14}x_{13}x_{12}\ x_{11}x_{10}x_9x_8\ x_7x_6x_5x_4\ x_3x_2x_1x_0 \\ (\textit{rest}\ B) \quad & +0\, y_{14}y_{13}y_{12}\ y_{11}y_{10}y_9y_8\ y_7y_6y_5y_4\ y_3y_2y_1y_0 \\ = \quad & 1\, v_{14}v_{13}v_{12}\ v_{11}v_{10}v_9v_8\ v_7v_6v_5v_4\ v_3v_2v_1v_0 \end{aligned}$$

```
; Program MULT16

; Variablen:

; A   = 800h
; B   = 801h
; Erg-low  = 802h
; Erg-high = 803h
; Ma 1  = 804h              <Ma 1>  =  8000h= 1000 0000 0000 0000 B
; Ma 2  = 805h              <Ma 2 > =  EFFFh= 0111 1111 1111 1111 B
; HiV 1 = 806h
; HiV 2 = 807h

;Beginn des Programms

…

;Ende des Programms
```

			Befehl	Operand	Erklärung
Beginn	Acc:=0		ZRO		
	<Erg-low>:=Acc		STO	Erg-low	
	<Erg-high>:=Acc		STO	Erg-high	
	Acc:=A		LDA	A	
	Springe bei Acc=0 nach Ende		BRZ	Ende	
	Acc:=B		LDA	B	
	Springe bei Acc=0 nach Ende		BRZ	Ende	
					Test 1 auf Übertrag - höchstwertigsten Bits von A und unteren Ergebniszelle ist 1
Test 1	Acc:=A		LDA	A	
	Acc:=Acc AND <Ma 1>		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
	<HiV 1>:=Acc		STO	HiV 1	Abspeichern für später
	Acc:=<Erg-low>		LDA	Erg-low	
	Acc:=Acc AND <Ma 1>		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
	Acc:=Acc AND <HiV 1>		AND	HiV 1	Test hi-bit – Erg-low und A ergeben Carry
	Springe bei Acc=0 nach Test2		BRZ	Test2	Springe , wenn kein Carry zum Test 2 auf Carry
					Addition des Carry zur Erg-low
high-Add	Acc:= <Erg-high>		LDA	Erg-high	Lade oberen Ergebniszelle
	Acc:=Acc+1		INC		Addiere Carry
	<Erg-high>:=Acc		STO	Erg-high	speichere oberen Ergebniszelle
	Acc:=0		ZRO		Setze Akku auf 0 zum unbedingten Sprung
	Springe bei Acc=0 nach low-Add		BRZ	low-Add	Springe zur Addition der Erg-low mit A
					Test 2
Test 2	Acc:=<A>		LDA	A	
	Acc:= Acc AND <Ma2>		AND	Ma2	Ausfiltern der hinteren Bits von A
	<HiV 2>:=Acc		STO	HiV 2	Abspeichern für später
	Acc:= <Erg-low>		LDA	Erg-low	
	Acc:=Acc AND <Ma 2>		AND	Ma2	Ausfiltern der hinteren Bits von Erg-low
	Acc:=Acc + <HiV 2>		ADD	HiV 2	Addieren der hinteren Bits von Erg-low und A
	Acc:=Acc AND <Ma 1>		AND	Ma 1	Ausfiltern des höchstwertigsten Bits der Summe der hinteren Bits von Erg-low und A
	Springe bei Acc=0 nach low-Add		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
					weiter, wenn höchstes bit der Hinterteile=1
	Acc:=< Erg-low>		LDA	Erg-low	
	Acc:=Acc AND <Ma 1>		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von Erg-low
	Acc:=Acc OR <HiV 1>		OR	HiV 1	Carry, wenn höchstwertigsten Bits von Summe Hinterteil und von Erg-low oder A eins
	Springe bei Acc=0 nach low-Add		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
	Acc:=0		ZRO		Setze Akku auf 0 zum unbedingten Sprung
	Springe bei Acc=0 nach high-Add		BRZ	high-Add	Sprung zu high-Add wenn Carry
					Addition der Erg-low Zelle
low-Add	Acc:=<A>		LDA	A	
	Acc:= Acc + < Erg-low>		ADD	Erg-low	Addiere A zu Erg-low
	< Erg-low>:= Acc		STO	Erg-low	speichere
					Dekrementierung von B
	Acc:=<B>		LDA	B	
	Acc:=Acc - 1		DEC		Verringere B um 1
	<B> := Acc		STO	B	Beende das Programm, wenn B=0 ist
	Springe bei Acc=0 nach Ende		BRZ	Ende	
	Acc:=0		ZRO		Setze Akku auf 0 zum unbedingten Sprung
	Springe bei Acc=0 nach Test 1		BRZ	Test 1	
Ende					

2. Schreiben Sie den Assemblercode und den Maschinencode entsprechend dem Beispielprogramm.

10 Punkte

; Program MULT16

; Variablen:

; A = 800h
; B = 801h
; Erg-low = 802h
; Erg-high = 803h
; Ma 1 = 804h <Ma 1> = 8000h= 1000 0000 0000 0000 B
; Ma 2 = 805h <Ma 2 > = EFFFh= 0111 1111 1111 1111 B
; HiV 1 = 806h
; HiV 2 = 807h

;Beginn des Programms

…

;Ende des Programms

Adresse	Marken	Befehl	Operand	Erklärung
000 h	Beginn	ZRO		
001 h		STO	Erg-low	
002 h		STO	Erg-high	
003 h		LDA	A	
004 h		BRZ	Ende	
005 h		LDA	B	
006 h		BRZ	Ende	
				Test 1 auf Übertrag - höchstwertigsten Bits von A und unteren Ergebniszelle ist 1
007 h	Test 1	LDA	A	
008 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
009 h		STO	HiV 1	Abspeichern für später
00A h		LDA	Erg-low	
00B h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
00C h		AND	HiV 1	Test hi-bit – Erg-low und A ergeben Carry
00D h		BRZ	Test2	Springe , wenn kein Carry zum Test 2 auf Carry
				Addition des Carry zur Erg-low
00E h	high-Add	LDA	Erg-high	Lade oberen Ergebniszelle
00F h		INC		Addiere Carry
010 h		STO	Erg-high	speichere oberen Ergebniszelle
011 h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
012 h		BRZ	low-Add	Springe zur Addition der Erg-low mit A
				Test 2
013 h	Test 2	LDA	A	
014 h		AND	Ma2	Ausfiltern der hinteren Bits von A
015 h		STO	HiV 2	Abspeichern für später
016 h		LDA	Erg-low	
017 h		AND	Ma2	Ausfiltern der hinteren Bits von Erg-low
018 h		ADD	HiV 2	Addieren der hinteren Bits von Erg-low und A
019 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits der Summe der hinteren Bits von Erg-low und A
01A h		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
				weiter, wenn höchstes bit der Hinterteile=1
01B h		LDA	Erg-low	
01C h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von Erg-low
01D h		OR	HiV 1	Carry, wenn höchstwertigsten Bits von Summe Hinterteil und von Erg-low oder A eins
01E h		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
01F h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
020 h		BRZ	high-Add	Sprung zu high-Add wenn Carry
				Addition der Erg-low Zelle
021 h	low-Add	LDA	A	
022 h		ADD	Erg-low	Addiere A zu Erg-low
023 h		STO	Erg-low	speichere
				Dekrementierung von B
024 h		LDA	B	
025 h		DEC		Verringere B um 1
026 h		STO	B	Beende das Programm, wenn B=0 ist
027 h		BRZ	Ende	
028 h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
029 h		BRZ	Test 1	
02A h	Ende			

Adresse	Code	Marken	Befehl	Operand	Erklärung
000 h	B000 h	Beginn	ZRO		
001 h	0802 h		STO	Erg-low	
002 h	0803 h		STO	Erg-high	
003 h	1800 h		LDA	A	
004 h	202A h		BRZ	Ende	
005 h	1801 h		LDA	B	
006 h	202A h		BRZ	Ende	
					Test 1 auf Übertrag - höchstwertigsten Bits von A und unteren Ergebniszelle ist 1
007 h	1800 h	Test 1	LDA	A	
008 h	6804 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
009 h	0806 h		STO	HiV 1	Abspeichern für später
00A h	1802 h		LDA	Erg-low	
00B h	6804 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von A
00C h	6806 h		AND	HiV 1	Test hi-bit – Erg-low und A ergeben Carry
00D h	2013 h		BRZ	Test2	Springe , wenn kein Carry zum Test 2 auf Carry
					Addition des Carry zur Erg-low
00E h	1803 h	high-Add	LDA	Erg-high	Lade oberen Ergebniszelle
00F h	9000 h		INC		Addiere Carry
010 h	0803 h		STO	Erg-high	speichere oberen Ergebniszelle
011 h	B000 h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
012 h	2021 h		BRZ	low-Add	Springe zur Addition der Erg-low mit A
					Test 2
013 h	1800 h	Test 2	LDA	A	
014 h	6805 h		AND	Ma2	Ausfiltern der hinteren Bits von A
015 h	0807 h		STO	HiV 2	Abspeichern für später
016 h	1802 h		LDA	Erg-low	
017 h	6805 h		AND	Ma2	Ausfiltern der hinteren Bits von Erg-low
018 h	3807 h		ADD	HiV 2	Addieren der hinteren Bits von Erg-low und A
019 h	6804 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits der Summe der hinteren Bits von Erg-low und A
01A h	2021 h		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
					weiter, wenn höchstes bit der Hinterteile=1
01B h	1802 h		LDA	Erg-low	
01C h	6804 h		AND	Ma 1	Ausfiltern des höchstwertigsten Bits von Erg-low
01D h	5806 h		OR	HiV 1	Carry, wenn höchstwertigsten Bits von Summe Hinterteil und von Erg-low oder A eins
01E h	2021 h		BRZ	low-Add	Springe zur Addition der Erg-low mit A da kein Carry
01F h	B000 h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
020 h	200E h		BRZ	high-Add	Sprung zu high-Add wenn Carry
					Addition der Erg-low Zelle
021 h	1800 h	low-Add	LDA	A	
022 h	3802 h		ADD	Erg-low	Addiere A zu Erg-low
023 h	0802 h		STO	Erg-low	speichere
					Dekrementierung von B
024 h	1801 h		LDA	B	
025 h	A000 h		DEC		Verringere B um 1
026 h	0801 h		STO	B	Beende das Programm, wenn B=0 ist
027 h	202A h		BRZ	Ende	
028 h	B000 h		ZRO		Setze Akku auf 0 zum unbedingten Sprung
029 h	2007 h		BRZ	Test 1	
02A h		Ende			

3. Probieren Sie das Programm an dem Beispiel: 36865 ● 12448 aus.

5 Punkte

;Beginn des Programms

000h …
…

;Ende des Programms

Verschiedene Möglichkeiten je nach Algorithmus