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
Einheitlicher Abgabetermin: Di. der 02.Juli 2002

Korrigierte Version vom 25.Juni 2002

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
```

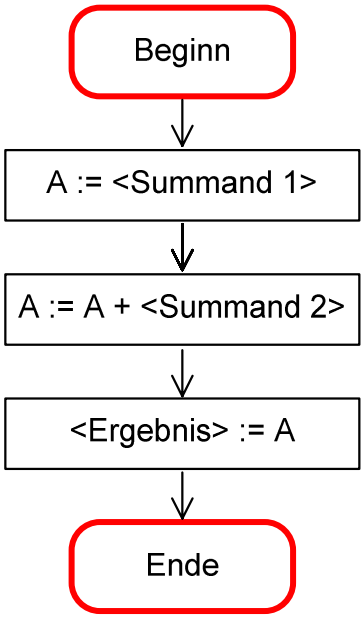
Beispiel für Problemablaufplan und Programm:

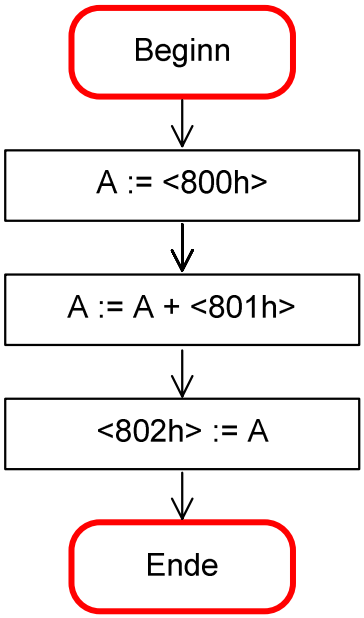
Addition von zwei 8-Bit Zahlen.

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

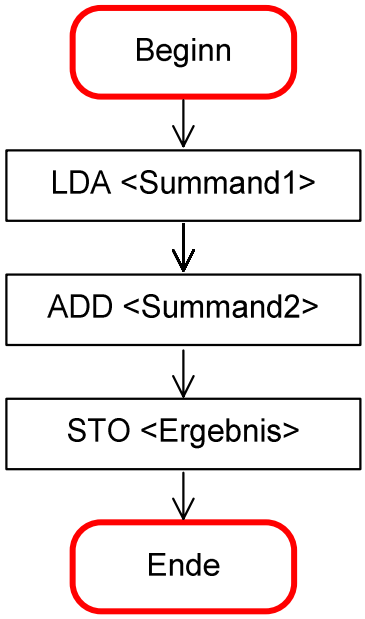
```
; Program ADD8
;Variablen
;Summand1 = 800h
;Summand2 = 801h
;Ergebnis     = 802h

…
```

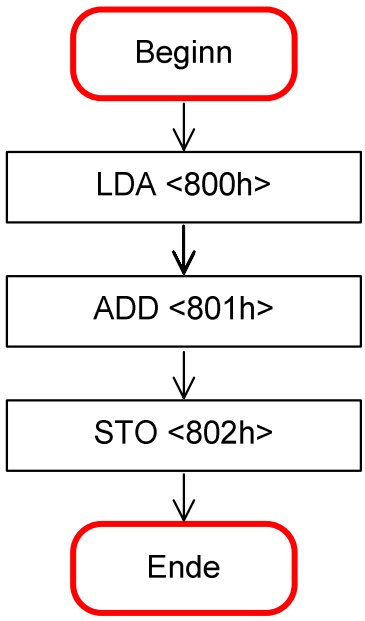
2 Möglichkeiten des Problemablaufplanes (1x vorgeschrieben):





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