

Übung 3 - Lösung

1. (A - B) + C

Entwickeln Sie ein Assemblerprogramm, das für verschiedene (32-Bit) Werte von A, B und C den folgenden Ausdruck berechnet:

$$\text{ERG} = (A - B) + C$$

Die Größen A, B und C sollen anschließend jeweils verdoppelt werden und die Berechnung soll erneut durchgeführt werden.

Falls bei der Berechnung ein „Overflow“ auftritt, soll das Programm beenden und der Betrag einer Zählvariablen, die die „Durchläufe“ mitzählt, soll in der Speicherzelle „Zahl“ sowie das Ergebnis ERG in einer Variablen „ERG“ abgespeichert werden. Außerdem sollen die Werte für A, B und C beim Abbruch gespeichert werden.

a) Erstellen Sie ein Flussdiagramm für den Algorithmus

b) Erstellen Sie das Assembler Programm

c) Füllen Sie die folgende Tabelle mit Hilfe des Programms aus:

A Start	B Start	C Start	A Ende	B Ende	C Ende	ERG	Zahl
1	2	3	4000 0000h	4000 0000h	6000 0000h	4000 0000h	1Dh
2	3	1	4000 0000h	6000 0000h	2000 0000h	0	1Dh
23h	87h	5h	2300 0000h	4380 0000h	280 0000h	D080 0000h	17h
5	0	3	5000 0000h	0	300 0000h	4000 0000h	1Ch

Folgende Codevorlage kann verwendet werden:

```
%include "io.inc"
```

```
section .data
```

```
A DD 5h
B DD 0h
C DD 3h
ERG DD 0
Zahl DD 0
```

```
section .text
```

```
global CMAIN
```

```
CMAIN:
```

```
mov ebp, esp; for correct debugging
;write your code here
mov edx,0 ; Anzahl der durchläufe = 0
```

Schleife:

```
mov eax,[A] ; verschiebe A nach AX
mov ebx,[B] ; verschiebe B nach BX
mov ecx,[C] ; verschiebe C nach CX
```

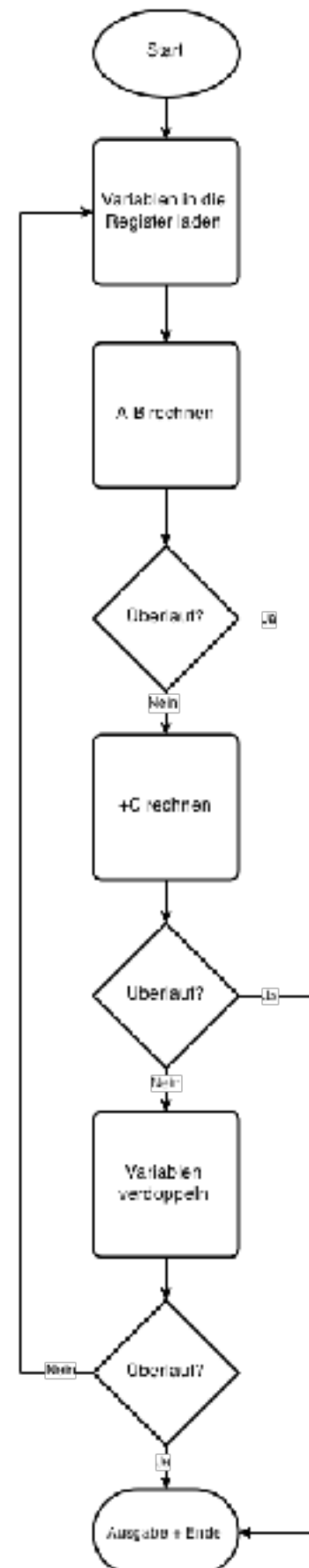
```
sub eax,ebx ; subtrahiere BX von AX -> AX
jo Abbruch
add eax,ecx ; addiere CX zu AX -> AX
jo Abbruch
mov [ERG],eax ; verschiebe von AX nach ERG
```

```
mov eax,[A] ; verschiebe A nach AX
mov ebx,[B] ; verschiebe B nach BX
mov ecx,[C] ; verschiebe C nach CX
add eax,eax ; AX+=AX
jo Abbruch
mov [A],eax ; verschiebe AX nach A
add ebx,ebx ; BX+=BX
jo Abbruch
mov [B],ebx ; verschiebe BX nach B
add ecx,ecx ; CX+=CX
jo Abbruch
mov [C],ecx ; verschiebe CX nach C
inc edx
jmp Schleife ; schleife
```

Abbruch:

```
mov [Zahl],edx ; schreibe anzahl der durchläufe in Zahl
```

```
PRINT_STRING "A: "
PRINT_HEX 4,A
NEWLINE
```



```
PRINT_STRING "B: "  
PRINT_HEX 4,B  
NEWLINE  
PRINT_STRING "C: "  
PRINT_HEX 4,C  
NEWLINE  
PRINT_STRING "Ergebnis: "  
PRINT_HEX 4,ERG  
NEWLINE  
PRINT_STRING "Zahl: "  
PRINT_HEX 4,Zahl
```

```
xor eax, eax  
ret
```