Übung 7 - Lösung

1. Taschenrechner

Erstellen Sie einen Taschenrechner als Assembler Programm. Es werden zwei Zahlen und die Rechenoperation eingegeben und dann das Ergebnis ausgegeben.

Beispiel: Eingabe:
+
5
7

Ausgabe:
12

Folgende Rechneroperationen sollen unterstützt werden: +, -, *, /, modulo (%)
Die Eingabe soll mit mit 32 Bit Zahlen und sowohl die Ein- als auch die Ausgabe mit Dezimalzahlen realisiert werden. Bei der Multiplikation muss die Ausgabe bis 64 Bit gehen! Das Makro PRINT_HEX kann nicht verwendet werden, da im 32 Bit Modus keine 64 Bit Zahlen ausgegeben werden können.
Das Horner Schema (die Hexadezimale Zahl durch 10 teilen und dann den Rest von unten nach oben lesen) und der Stack helfen hier!

a) Erstellen Sie ein Flussdiagramm für den Algorithmus

b) Erstellen Sie das Assembler Program

c) Testen Sie das Programm:

Eingabe:
*
4294967295
2345

Ausgabe:
10071698306775

```
%include "io.inc" section
.data
ERG DQ 0
DIV1 DD 10
```

```
section .text global
CMAIN CMAIN:
    mov ebp, esp; for correct debugging
;write your code here      xor edx, edx
GET_CHAR edx
    GET_DEC 4, eax
    GET_DEC 4, ebx
    CMP edx, 2Bh     JE
Addieren
    CMP edx, 2Dh
    JE Subtrahieren
    CMP edx, 2Ah
    JE Multiplizieren
    CMP edx, 2Fh
    JE Dividieren
    CMP edx, 25h
    JE Modulo

    jmp Fehler

Subtrahieren:
xor    edx,    edx
sub eax, ebx
jmp Ausgabe

Addieren:     xor
edx, edx      add
eax, ebx     jmp
Ausgabe

Multiplizieren:      mul
ebx     jmp
Ausgabe

Dividieren:
xor edx, edx
div ebx      xor
edx, edx     jmp
Ausgabe Modulo:
    xor edx, edx     div
ebx     mov eax, edx
xor edx, edx     jmp
```

Ausgabe

```
Ausgabe:
   mov    DWORD    [ERG],eax
mov DWORD [ERG+4],edx
   xor ecx,ecx

   .DIV:       xor  edx,
edx          mov  eax,
[ERG+4]    div dword
[DIV1]              mov
[ERG+4], eax    mov
eax,  [ERG]        div
dword [DIV1]     mov
[ERG],  eax      push
edx    inc ecx    mov
ebx,[ERG]         cmp
ebx,0          jnz  .DIV
mov     ebx,[ERG+4]
cmp ebx,0    jnz .DIV

   .Loop:    pop ebx
PRINT_HEX 4,ebx
loop .Loop

   xor eax, eax    ret

Fehler:    PRINT_STRING
"Fehler!"    xor
eax, eax    ret
```