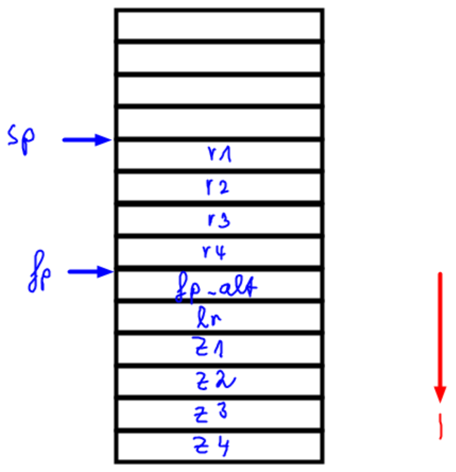
AsmTest_015

ÜBUNG: Parameterübergabe über Stack (1)



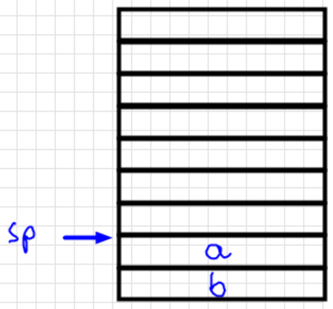
```
WORDSIZE    EQU    4

;*********************************************
; Data section, aligned on 4-byte boundery
;*********************************************

    AREA   MyData, DATA, align = 2
    GLOBAL MyData
    GLOBAL Base

Base
z1      DCD    16
z2      DCD    32
z3      DCD     8
z4      DCD     4
Erg     DCD     0

;*********************************************
; Code section, aligned on 8-byte boundery
;*********************************************

    AREA MyCode, CODE, readonly, align = 3

;---------------------------------------------
; main subroutine
;---------------------------------------------
    EXPORT main [CODE]

main    PROC
        ldr     r1, =Base

        ldr     r2, [r1, #z1-Base]      ; Input-Parameter z1
        ldr     r3, [r1, #z2-Base]      ; Input-Parameter z2
        ldr     r4, [r1, #z3-Base]      ; Input-Parameter z3
        ldr     r5, [r1, #z4-Base]      ; Input-Parameter z4
        push    {r2, r3, r4, r5}        ; Push

        bl      SumProd                 ; Call Subroutine
        add     sp, #4*WORDSIZE         ; Stack bereinigen

        str     r0, [r1, #Erg-Base]     ; Ergebnis speichern

; Programmende (Endlosschleife)
loop
     b  loop
; ------------------------------
; Ende : main
; ------------------------------

        ALIGN

; ----------------------------------------
; SumProd (Unterprogramm)
; IN:   Stack
; OUT:  r0
; ----------------------------------------
SumProd
        push    {fp, lr}                ; fp und lr retten
        mov     fp, sp                  ; Framepointer setzen
        push    {r1-r4}                 ; Register retten

        ldr     r1, [fp, #8]            ; r1 <-- z1
        ldr     r2, [fp, #12]           ; r2 <-- z2
        ldr     r3, [fp, #16]           ; r3 <-- z3
        ldr     r4, [fp, #20]           ; r4 <-- z4

        add     r1, r2                  ; z1+z2
        add     r3, r4                  ; z3+z4
        mul     r0, r1, r3              ; (z1+z2)*(z3+z4)

        pop     {r1-r4}                 ; Register restaurieren
        pop     {fp, lr}                ; fr und lr restaurieren
        bx      lr                      ; return zum aufrufenden Programm
; ------------------------------------------------------------------------------
; Ende : SumProd
; ------------------------------------------------------------------------------

        ENDP
        ALIGN
        END
```

ÜBUNG: Parameterübergabe über Stack (2) und lok. Speicher

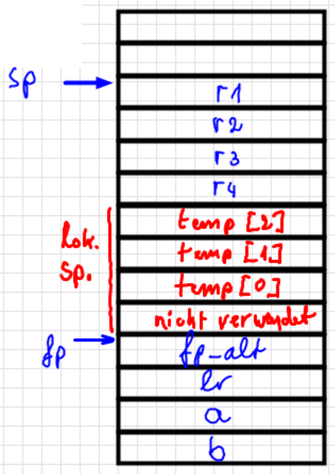
Asm Test_013

```
WORDSIZE    EQU    4

;*********************************************
; Data section, aligned on 4-byte boundery
;*********************************************

    AREA MyData, DATA, align = 2
    GLOBAL MyData,  Base

Base
a             DCD   16
b             DCD    4
Erg           DCD    0

;*********************************************
; Code section, aligned on 8-byte boundery
;*********************************************

    AREA MyCode, CODE, readonly, align = 3

;---------------------------------------------
; main subroutine
;---------------------------------------------
    EXPORT main [CODE]

main    PROC
        ldr     r1, =Base

        ldr     r2, [r1, #a-Base]        ; Input a
        ldr     r3, [r1, #b-Base]        ; Input b
        push    {r2, r3}
        bl      SumTerm                  ; Call Subroutine
        add     sp, #2*WORDSIZE          ; Stack bereinigen

        str     r0, [r1, #Erg-Base]      ; Ergebnis speichern

; Programmende (Endlosschleife)
loop
     b  loop
; ------------------------------
; Ende : main
; ------------------------------
```

Stack vor dem
Unterprogrammaufruf



```
; ----------------------------------------
; SumTerm (Unterprogramm)
; IN:   Stack
; OUT:  r0
; ----------------------------------------
SumTerm
        push    {fp, lr}          ; Framepointer und Linkregister retten
        mov     fp, sp            ; Framepointer setzen
        sub     sp, #4*WORDSIZE   ; lokalen Speicher reservieren
        push    {r1-r4}           ; Arbeitsregister retten

        ldr     r1, [fp, #8]      ; r1 <-- a
        ldr     r2, [fp, #12]     ; r2 <-- b
        mul     r3, r2, r2
        str     r3, [fp, #-8]     ; b*b --> temp[0]

        mul     r3, r2, r3
        str     r3, [fp, #-12]    ; b*b*b --> temp[1]

        mul     r3, r2, r3
        str     r3, [fp, #-16]    ; b*b*b*b --> temp[2]

        mov     r2, #1
        ldr     r3, [fp, #-8]
        add     r2, r3            ; 1+b^2

        ldr     r3, [fp, #-12]
        add     r2, r3            ; 1+b^2+b^3

        ldr     r3, [fp, #-16]
        add     r2, r3            ; 1+b^2+b^3+b^4

        mul     r0, r1, r2        ; a*(1+b^2+b^3+b^4)

        pop     {r1-r4}           ; Register restaurieren
        mov     sp, fp            ; sp unter lokalen Speicher setzen
        pop     {fp, lr}          ; Framepointer und Linkregister restaurieren
        bx      lr                ; return zum aufrufenden Programm
; ---------------------------------------------------------------------------
; Ende : SumTerm
; ---------------------------------------------------------------------------

       ENDP
       ALIGN

       END
```


sp
r1
r2
r3
r4
lok. Sp.
temp [2]
temp [1]
temp [0]
nicht verwendet
fp
fp-alt
lr
a
b