AREA MyData, DATA, align = 2

GLOBAL Ergebnis\_2

text DCB "RMP SS16",0

COEF DCD 0x3, 0x02, 0x1; a, b , c

X DCD 2

Y DCD 3

Ergebnis\_2 SPACE 32

; …

;--------------------------------------------

; main subroutine

;--------------------------------------------

EXPORT main [CODE]

main PROC

; Vorbereitung POLY mit Parameter per Stack

ldr r2,=X;

ldr r0,[r2];

ldr r2,=Y;

ldr r1,[r2];

push{r0, r1}; X in r0 oberhalb von Y in r1 auf Stack

bl POLY

add sp,#8; Korrektur für 2 Variablen auf Stack

ldr r3,=Ergebnis\_2;

str r0,[r3, #4]

forever b forever ; nowhere to retun if main ends

ENDP

POLY PROC

push{lr, fp}; fp = r11 , lr = r14

mov fp, sp; sp = r13

push{r1,r2, r3, r4};

ldr r1, [fp, #8]; X in r1

ldr r2, [fp, #12]; Y in r2

;z = y\*(a\*X + c) + b\*x

ldr r3,=COEF;

ldr r4,[r3],#8; a

mul r0, r4, r1; aX

ldr r4,[r3],#-4; c

add r0, r0, r4; aX+c

mul r0, r0, r2; Y(aX+c)

ldr r4,[r3]; b

mul r4, r1, r4; bX

add r0, r0, r4; z

pop{r1 - r4}

pop{lr, fp}

bx lr

;

ENDP

ALIGN

END

; Vorbereitung FACTORIAL mit Parameter per Stack

ldr r2,=X;

ldr r0,[r2];

ldr r1,[r2];

push{r0, r1}; X in r0 oberhalb von r1 auf Stack

bl FACTORIAL

add sp,#8; Korrektur für 2 Variablen auf Stack

ldr r3,=Ergebnis\_2;

str r0,[r3, #4]

; ….

ENDP

FACTORIAL push{lr, fp}; fp = r11 , lr = r14

mov fp, sp; sp = r13

push{r1,r2, r3, r4};

ldr r1, [fp, #8]; X in r1

;ldr r2, [fp, #12]; X in r2

mov r0,#0x01 ; Ergebnis

mov r2,#0x00 ; cnt

WHILE cmp r1,r2;

beq RET

add r2, r2, #0x01;

mul r0, r0, r2; Rueckgabe mit r0 lr

b WHILE

RET pop{r1 - r4}

pop{lr, fp}

bx lr

;

ALIGN

END

char \*K[] = {"do\_", "while\_\_", "sizeof\_"};

K[1] = “while\_\_“ bzw. 0x1004

\*(K +1)

K[2][4] = ‘o‘

\*(\*(K +2) + 4)

&K[2][4] = 0x100c +4= 0x1010

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **N** | **Z** | **V** | **C** |  | **3** | **2** | **1** | **0** |
| 0 | 0 | 0 | 0 | r0=r1=r2  =r3=r4= | ee | 55 | ee | 55 |
| ldr | r0, =L1 | 0 | 0 | 0 | 0 | r0 = | 20 | 00 | 00 | 00 |
| mov | r1, #133 | 0 | 0 | 0 | 0 | r1 = | 00 | 00 | 00 | 85 |
| ands | r2, r1 | 0 | 0 | 0 | 0 | r2 = | 00 | 00 | 00 | 05 |
| ldrb | r3, [r0, #8]! | 0 | 0 | 0 | 0 | r3 = | 00 | 00 | 00 | 44 |
| movs | r4, r1, LSL #24 | 1 | 0 | 0 | 0 | r4 = | 85 | 00 | 00 | 00 |
| subs | r2, r1 | 1 | 0 | 0 | 0 | r2 = | ff | ff | ff | 80 |
| eor | r1, r3 | 1 | 0 | 0 | 0 | r1 = | 00 | 00 | 00 | c1 |
| ldr | r2, [r0, #4] | 1 | 0 | 0 | 0 | r2 = | 00 | 00 | 20 | 16 |
| ldr | r3, =L4 | 1 | 0 | 0 | 0 | r3 = | 20 | 00 | 00 | 14 |
| ldr | r1, [r0] | 1 | 0 | 0 | 0 | r1 = | 00 | 00 | 45 | 44 |

struct Student {

char Name[25];

long int MatrikelNr;

char Studiengang[3];

} Std1, Std2, Std3;

struct Student Stud1 = {"Zuse", 125716, "TI"};

struct Student Stud2 = {"Gates", 125609, "AI"};

Student Std[20];

strcpy( Std[0].Name, "Winter");