Praktikum Mikrorechner 6 (Unterprogramme)

Prof. Kemnitz

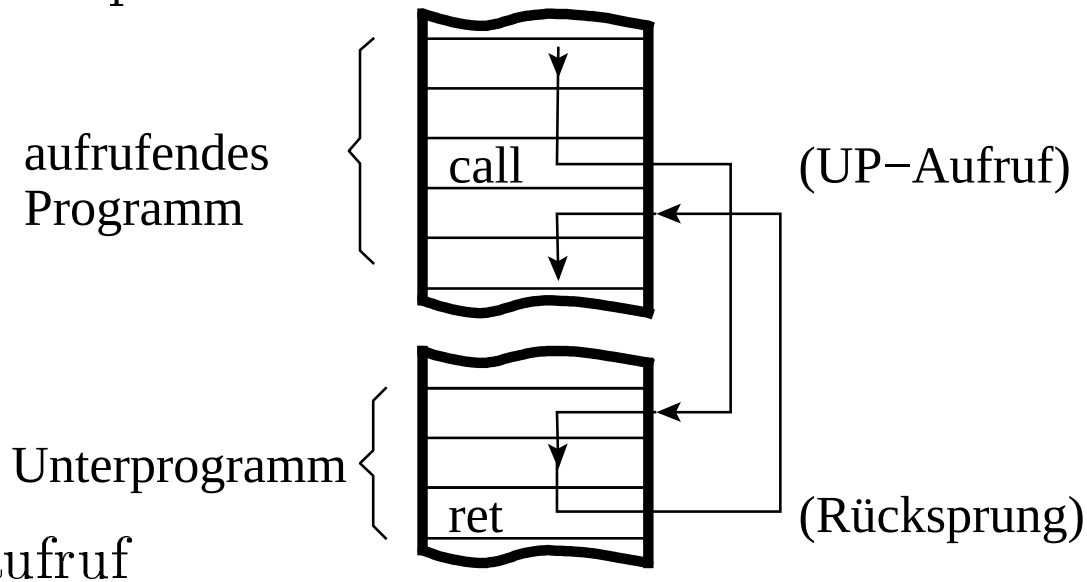
Institut für Informatik, Technische Universität Clausthal

5. November 2014

Unterprogramme

Unterprogramme

Einfügen einer Codefolge an mehreren Programmstellen, ohne die Codefolge mehrfach zu kopieren



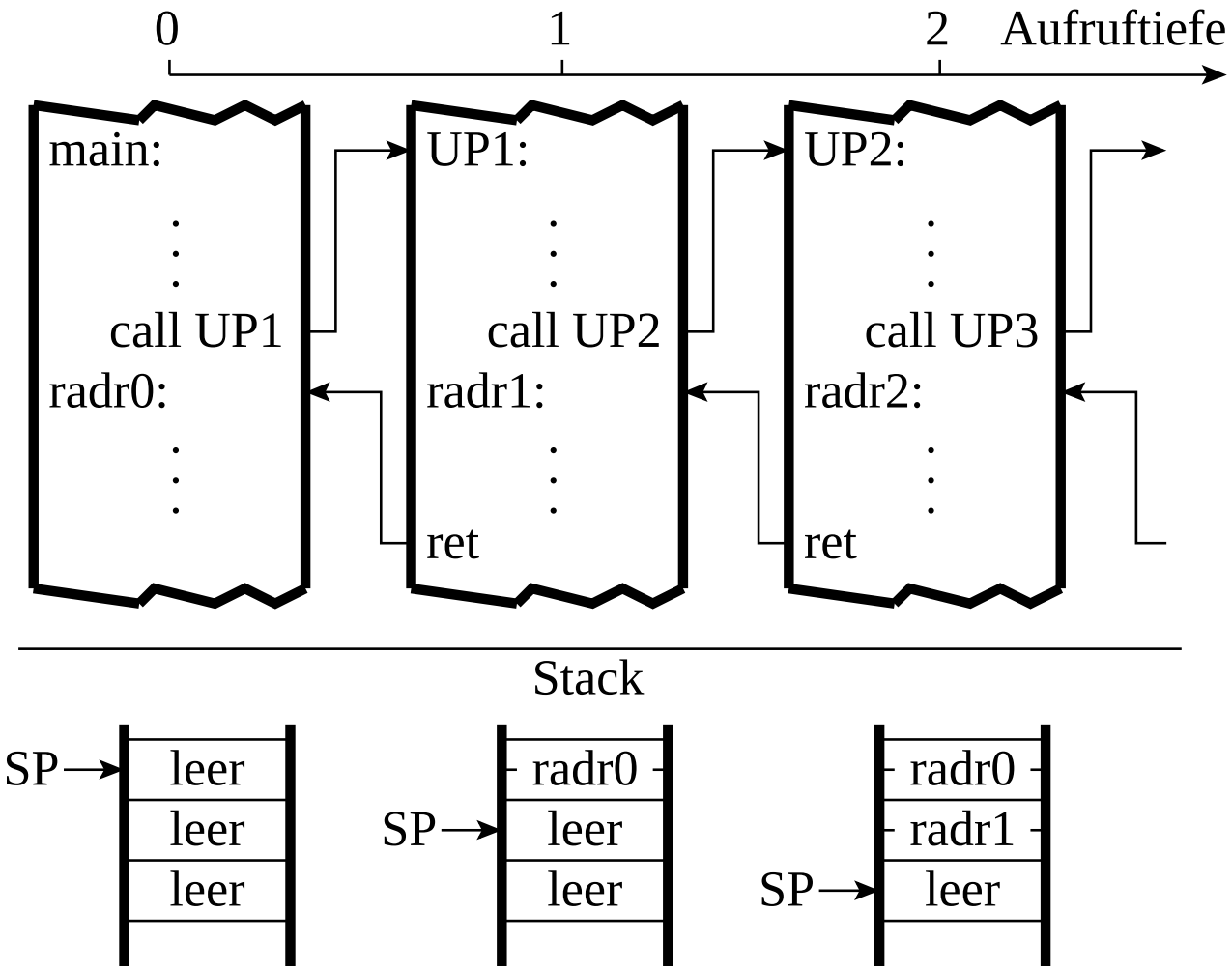
- Unterprogrammaufruf

```
lcall <adr16>
acall <adr11>
```

- Rücksprung

```
ret
```

Stack



Ablegen der Rückkehradressen nach dem LIFO- (last in first out) Prinzip

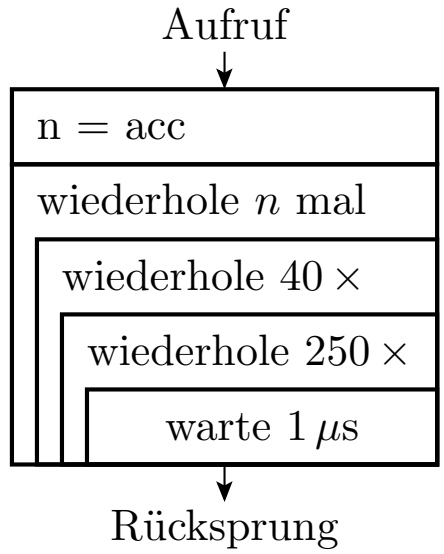
Stackpointer

- Spezialregister, Name sp, Adresse 81h
- Adresszeiger auf den nächsten freien Stackplatz
- Probleme/Fehlerquellen:
  - Risiko der Doppelnutzung von Speicherplätzen als Variable und Stackbereich
  - Stacküberlauf und Stackunterlauf
- Festlegen des Stackanfangs

```
mov sp, #f0h; Stack von f0 bis ff
```

```
Ct0_wnx10ms data 60h; lokale Variablen
Ct1_wnx10ms data 61h;
Ct2_wnx10ms data 62h;
```

```
w10ms:
; Unterprogramm Warte n x 10ms
; Übergabe der Wartezeit im Akku
    mov Ct2_wnx10ms, a
M0_wnx10ms:
    mov Ct1_wnx10ms, #20
M1_wnx10ms:
    mov Ct0_wnx10ms, #250
M2_wnx10ms:
      djnz Ct0_wnx10ms, M2_wnx10ms
     djnz Ct1_wnx10ms, M1_wnx10ms
    djnz Ct2_wnx10ms, M0_wnx10ms
    ret
```



Blinkprogramm

```
org 100h
    mov sp, #0c0h
    mov p1, #01h
wiederhole:
    mov a, #50
    lcall w10ms
    cpl P1.0
    ljmp wiederhole
org 200h
$include (w10ms.asm); Unterprogramm einbinden
end
```

Problem: include-Datei definiert globale Variablen; Gefahr der Mehrfachvergabe von Speicheradressen!

Lokale Variablen

Lokale Variablen

- Lokale Variablen sind nur in dem Programmteil definiert, in dem sie verwendet werden
- Schutz vor unbeabsichtigerter Mehrfachverwendung gleicher Adressen
- Voraussetzung für rekursive (sich selbst aufrufende) Unterprogramme

Prinzip:

- sichern der Inhalte der als lokale Variablen genutzten Speicherplätze zu Beginn des Unterprogramms auf den Stack
- Wiederherstellung vor dem Rücksprung

```
push <dadr>; auf den Stack kopieren
pop <dadr>;  vom Stack zurückkopieren
```

Wartezeitunterprogramm mit lokalen Variablen

```
Ct0_wnx10ms data 60h; globale Variablen
Ct1_wnx10ms data 61h;
Ct2_wnx10ms data 62h;
w10ms:
; Unterprogramm Warte n x 10ms
; Übergabe der Wartezeit im Akku
    push Ct0_wnx10ms; Sichern der Inhalte
    push Ct1_wnx10ms; lokaler Variablen
    push Ct2_wnx10ms
       <eigentliche Funktion>
    pop Ct2_wnx10ms
    pop Ct1_wnx10ms; wiederherstellen der Inhalte in
    pop Ct0_wnx10ms; umgekehrter Reihenfolge
ret
```

- die häufigsten lokalen Variablen: acc, psw, ...

Macro mit lokalen Variablen

```
add16 MACRO sum, s1, s2
;----------------------------------------
; E: s1, s2 (Summanden, je 2 Byte)
; A: sum (Summe, 2 Byte)
; Stack: 2
;----------------------------------------
    push acc
    push psw
        mov a, s1+1
        add a, s2+1
        mov sum + 1, a
        mov a, s1
        addc a, s2
        mov sum, a
    pop psw
    pop acc
ENDM
```

Gestaltungsrichtlinien für Unterprogramme

```
Unterprogramm:
;---------------------------------------
; E: ...  ; Eingabevariablen (Name/Adresse/Bedeutung)
; A: ...  ; Ergebnisvariablen (Name/Adresse/Bedeutung)
; V: -    ; veränderte Variablen (Nebeneffekte)
; Stack: 2 + n_LV  ; Stackbedarf
;---------------------------------------
    push acc
    push psw
     ... ; eigentliche Verarbeitung
    pop psw
    pop acc
ret
```

n_{LV} – Anzahl der Bytes für lokale Variablen

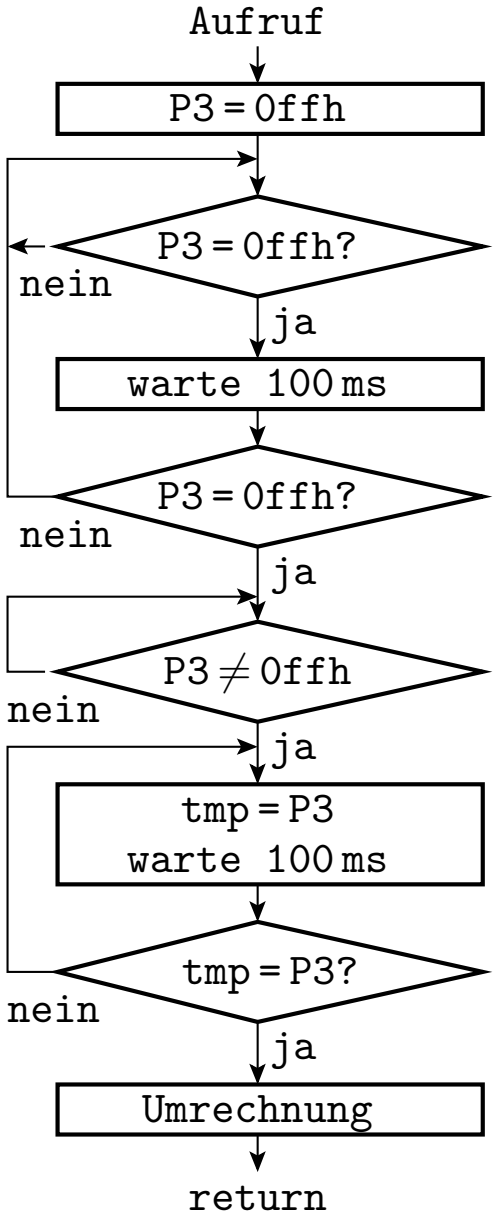
Aufgaben

Aufgabe 6.1: Tastaturabfrage

Entwickeln Sie ein Unterprogramm, das

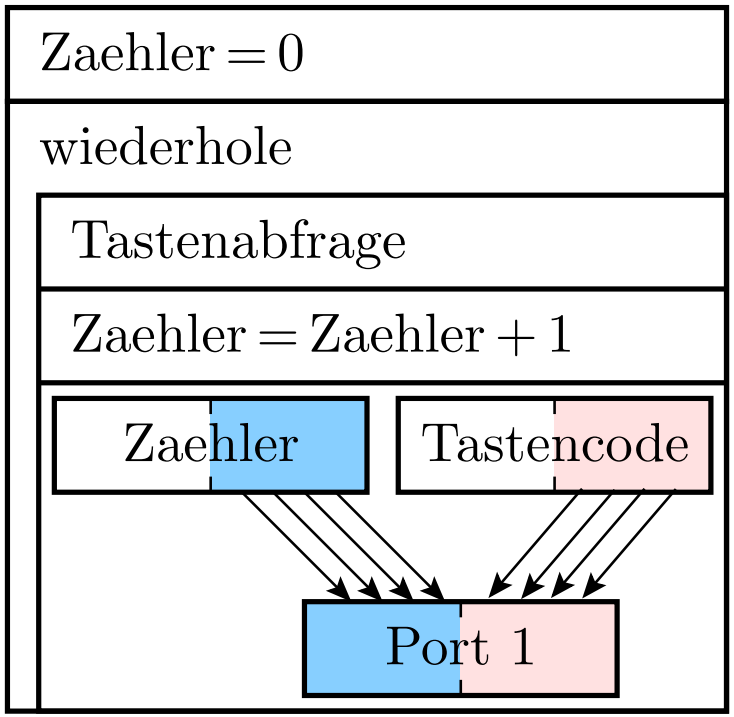
- wartet, bis alle drei Tasten für mindestens 100 ms losgelassen waren
- anschließend wartet, bis die Tasten für mindestens 100 ms in derselben Weise gedrückt waren und
- abschließend den Tastencodes im Akku in folgender Weise zurückgibt.

btn3.4	btn3.3	btn3.2	Rückgabewert
aus	aus	ein	1
aus	ein	aus	2
aus	ein	ein	3
ein	ein	ein	7



Entwickeln Sie zum Testen des Unterprogramms ein Hauptprogramm, das in einer Endlosshleife:

- die Tasten abfragt
- nach jeder Tastenabfrage einen Zähler erhöht
- die Bits 0 bis 3 des Tastencodes auf P1.0 bis P1.3 (LED1 bis LED4) und
- die Zählerbits 0 bis 3 auf P1.4 bis P1.7 (LED5 bis LED8) ausgibt.



Aufgabe 6.2: Rechts-Links-Lauflicht

Entwickeln Sie ein Programm, bei dem auf den Leutdioden ein Lichtpunkt bei Betätigung der Taste

- an P3.2 (oder mehrere Tasten) stehen bleibt
- an P3.3 nach rechts läuft
- an P3.4 nach links läuft.