Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Embedded Systems 1 & 2

a.13 – Interfacing Assembler - C

Klassen T-2/I-2 // 2018-2019

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Daniel Gachet | HEIA-FR/TIC
a.13 | 10.09.2018

Inhalt

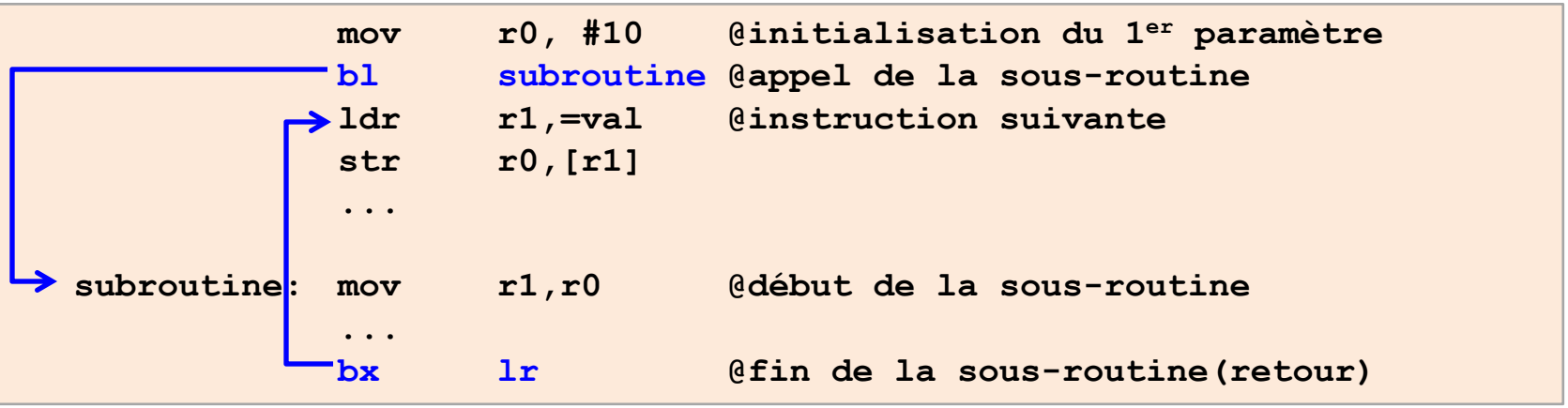
- Einleitung
- Aufruf von Unterprogrammen
- Übergabe von Parametern
- Änderung von Registern
- Konventionen

Die Programmiersprache C ermöglicht einen einfachen Zugriff auf die Register von Peripheriegeräten. Jedoch sind einige interne Ressourcen von Mikroprozessoren (spezielle Register, Interruptions, …) nur durch Assembler zugänglich. Es ist also üblich, dass in Assembler geschriebene Routinen als Schnittstelle für C Methoden (und umgekehrt) verwendet werden.

- **Zu lösende Punkte**
  - Aufruf von Unterprogrammen
  - Rückkehr zum Hauptprogramm (Speichern der Rückgabeadresse)
  - Übergeben der Ein- und Austrittsparameter, sowie des Resultates
  - Änderung von Registern
  - Bibliothekseinsatz durch Multitasking-Systeme

Aufruf von Unterprogrammen

- **Der ARM-Prozessor bietet zwei Instruktionen für den Aufruf von Unterroutinen**
  - **BL <target_address>** → direkte Adressierung ±32MB
  - **BLX <Rx>** → indirekte Adressierung
- **Diese Instruktionen speichern automatisch den Wert des PC, welcher auf die nächste auszuführende Instruktion im Register R14 zeigt. R14 wird auch « Link Register » oder « LR » genannt.**
- **Nachdem das Programm die Kontrolle dem Unterprogramm übergeben hat, folgt die Ausführung der normalen Reihenfolge der Instruktionen. Das Unterprogramm enthält als letzte Instruktion BX LR (oder MOV PC, LR, oder LDMFD SP!, {PC}) welche die Kontrolle dem Programm zurückgibt, um dessen Ausführung fortzusetzen.**



Techniken für die Übergabe von Parametern (Erinnerung)

- **Zwei Techniken werden verwendet wenn ein Programm die Parameter an ein Unterprogramm übergibt :**
  - Übergabe durch Wert
  - Übergabe durch Referenz (Adresse)

- **Die Technik der Übergabe durch Wert (call by value) gibt eine Kopie der aktuellen Daten:**
  - Das Unterprogramm erhält eine Kopie der Daten.
  - Es kann diese lesen und ändern, ohne das die Originaldatei verändert wird
  - Es gibt zwei verschiedene Speicherpositionen
  - Die Änderung einer Speicherposition hat keinen Einfluss auf die Andere.

- **Die Technik der Übergabe durch Referenz (call by reference) gibt die Adresse der Daten :**
  - Das Unterprogramm erhält die Adresse der Daten
  - Diese Daten werden von dem aufrufendem Programm und dem Unterprogramm geteilt.
  - Die Daten befinden sich nur in einer einzigen Speicherposition
  - Eine Änderung des Parameters durch das Unterprogramm ist für das aufrufende Programm sichtbar.

Ein wichtiger Teil der Erstellung eines Unterprogrammes ist das Bestimmen des zu verwendenden Mechanismus für die Übergabe der Parameter.

- **Die zwei wesentlichen Mechanismen**
  - Übergabe der Parameter durch Register
    - Die Register bieten eine schnelle Methode für die Übergabe der Parameter an. Jedoch ist diese Methode durch die Anzahl Register des Mikroprozessors limitiert.
  - Übergabe der Parameter durch den Stapel
    - Die Übergabe der Parameter durch den Stapel ist ein allgemeiner Ansatz. Jedoch ist dies langsamer, wegen dem Datenaustausch mit dem Speicher.
- **Die Wahl hängt stark von den Möglichkeiten des Mikroprozessors, der Programmiersprache und dem zu entwickelnden Programm selbst ab.**
- **Die modernen Compiler verwenden häufig beide Mechanismen**

Übergabe der Parameter durch die Register

- Die Parameter eines Unterprogrammes können durch die internen Register des Mikroprozessors übergeben werden (R0 bis R3)
- Das Unterprogramm verwendet die gleiche Methode für die Rückgabe des Resultates
- Bei der Übergabe der Parameter durch Register, muss sich der Programmiere vergewissern, dass jeder Parameter richtig im Register platziert ist. D.h. die Konventionen der Übergabe müssen eingehalten werden.
- Die für jeden Parameter zugeordneten Register sind in der Regel in den Kommentaren am Anfang des Unterprogrammes definiert, wie z.B.

```
int foo (const void* msg, void* dest);
```

  - R0 : Adresse von msg
  - R1 : Adresse von dest
  - R0 : Rückgabewert

Übergabe der Parameter durch den Stapel

- Eine der meistverwendeten Methoden zur Übergabe von Parametern an ein Unterprogramm ist das Platzieren dieser Parameter auf den Stapel
- Die Übergabe der Parameter durch den Stapel erfordert eine Konvention, welche die Reihenfolge dieser Parameter festlegt. Das aufrufende Programm legt die Parameter in der definierten Reihenfolge auf den Stapel. Das aufgerufene Programm greift durch indirekte Registeradressierung mit Verschiebung auf diese Werte zu.
- Um bei der den Stapel wiederherzustellen, müssen die Parameter danach gelöscht werden.

- Aufrufendes Programm :

```
main:   sub     sp, #4          @Reserviert Platz auf dem Stapel
        ldr     r0,=val
        str     r0,[sp, #0]     @Parameter auf Stapel platzieren
        bl      subroutine
        add     sp, #4          @Stellt den Stapel wieder her
```

- Programme appelé :

```
subroutine:     ...
        ldr     r0, [sp, #0]    @Nimmt den Wert des Parameters
        ...
        bx      lr
```

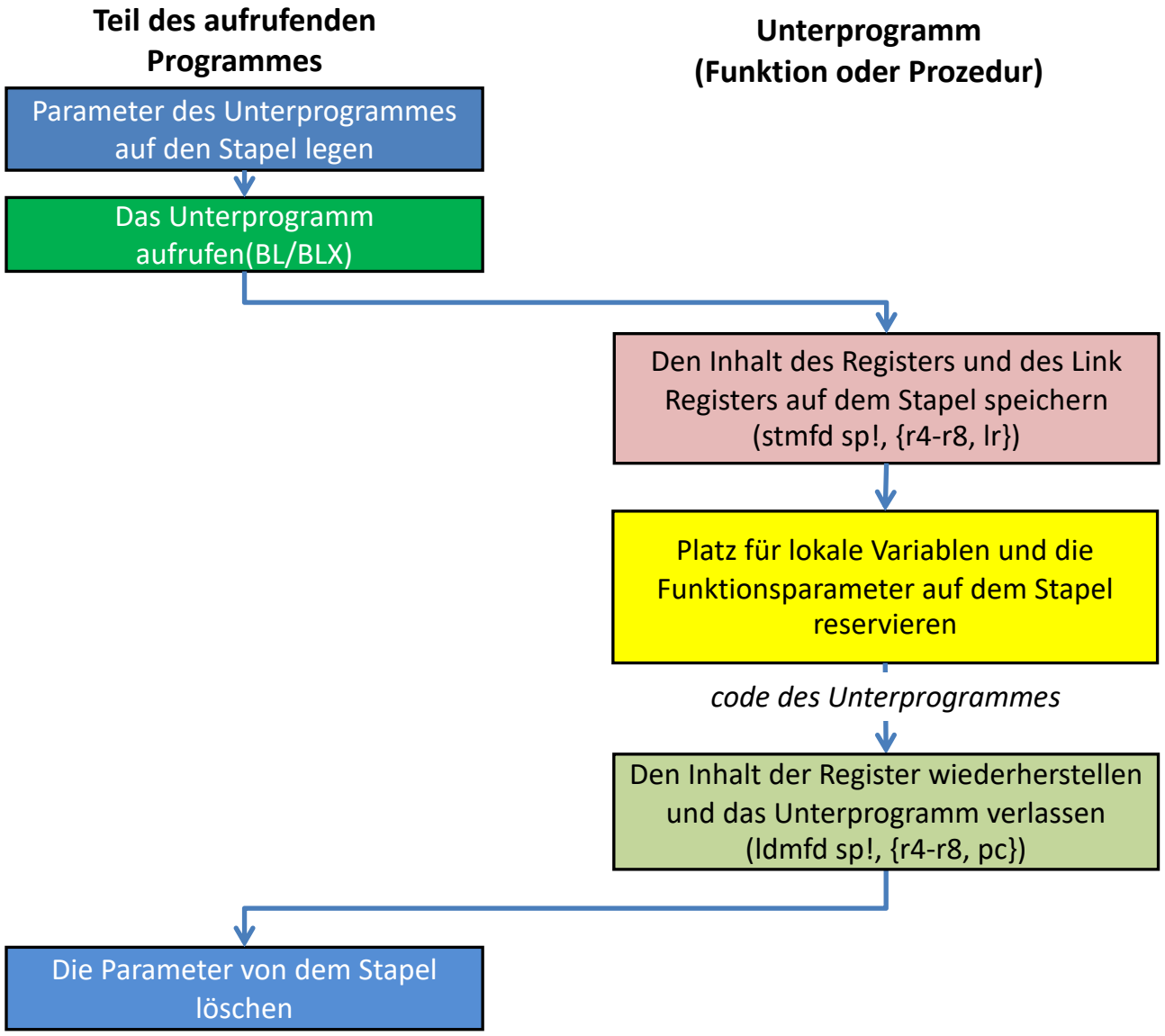
Änderung der Register

Ein Unterprogramm ist allgemein in einem grossen Instruktionenblock integriert. Es koexistiert mit dem Hauptprogramm und den anderen Unterprogrammen.

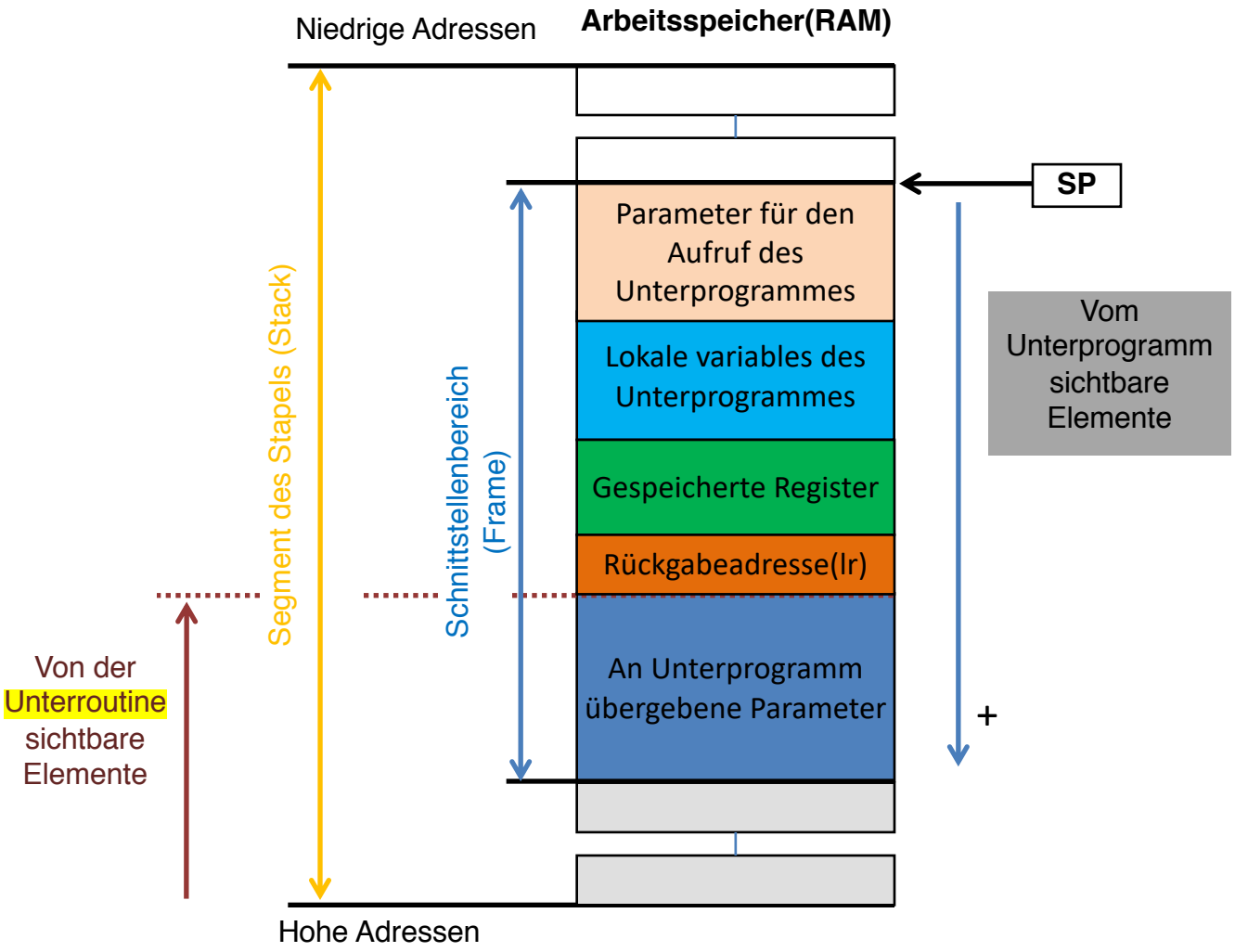
- **Da die Anzahl Register eines Prozessors begrenzt sind, kann ein Unterprogramm nicht mehrere Register für sich alleine reservieren.**
- **Ein Unterprogramm kann die von dem aufrufenden Programm verwendeten Register benutzen, solange es die Originalwerte der Register nicht verändert.**
- **Die vom Unterprogramm verwendeten Register werden oft vor einer Änderung gespeichert, und dann vor der Rückgabe wiederhergestellt.**
- **Der Stapelt dient als Speicherplatz.**
- **Um Register zu speichern, kann man die Instruktionen `stmfd/ldmfd` verwenden, zB um Inhalt der Register R4 bis R8 und LR zu speichern und wiederherzustellen**

```
stmfd   sp!,{r4-r8,lr}   // speichern      → push {r4-r8,lr}
ldmfd   sp!,{r4-r8,pc}   // wiederherst.   → pop  {r4-r8,pc}
```

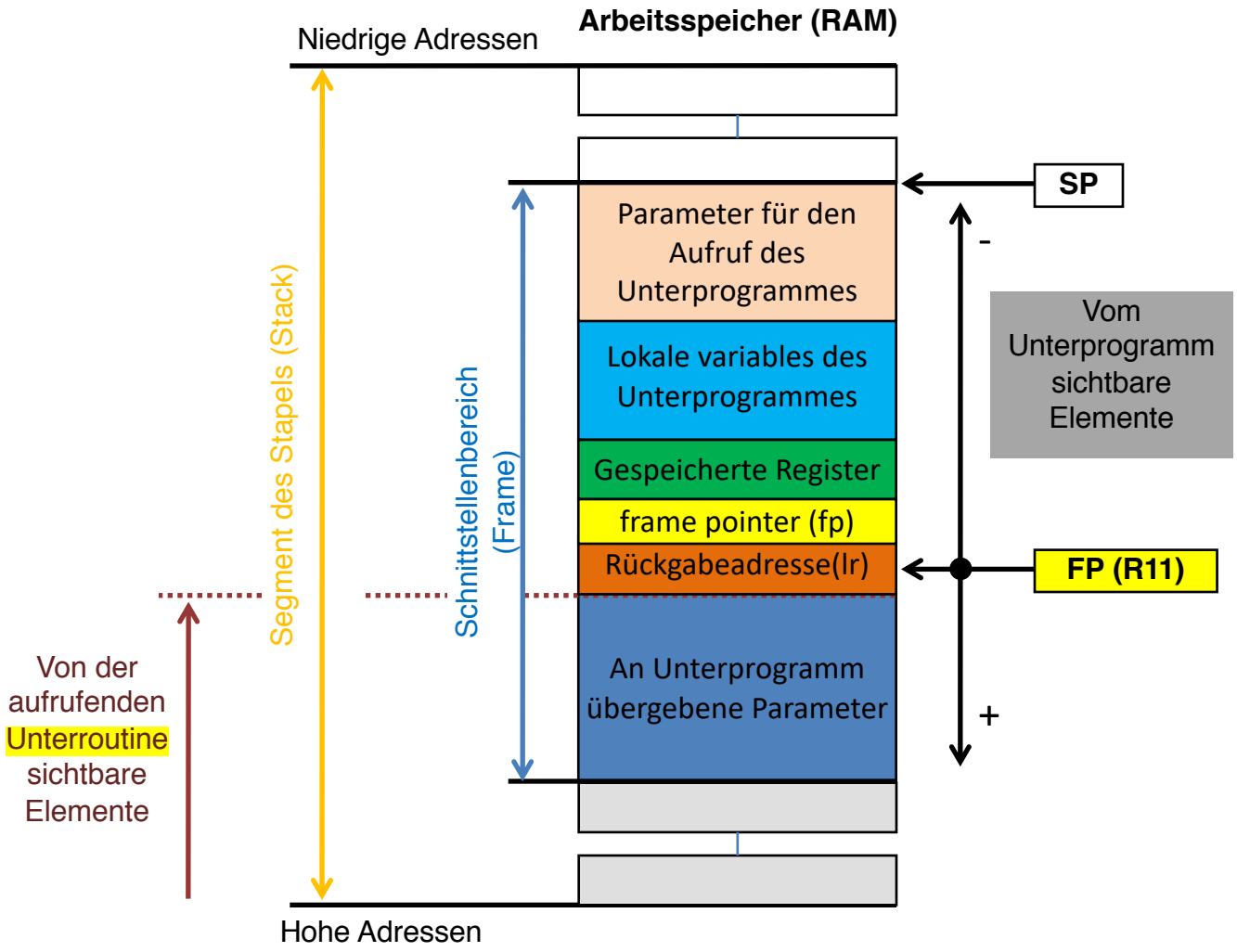
Reihenfolge der Operationen



Stack frame


Niedrige Adressen
Arbeitsspeicher(RAM)
SP
Parameter für den Aufruf des Unterprogrammes
Lokale variables des Unterprogrammes
Gespeicherte Register
Rückgabeadresse(lr)
An Unterprogramm übergebene Parameter
Segment des Stapels (Stack)
Schnittstellenbereich (Frame)
Vom Unterprogramm sichtbare Elemente
Von der Unterroutine sichtbare Elemente
+
Hohe Adressen

Frame pointer



C-Konventionen für ARM-Prozessoren

- Die ersten 4 Parameter werden in den Registern R0 bis R3 platziert.
- Die folgenden Parameter werden vom letzten bis zum 5. auf den Stapel gelegt (von links nach rechts)
- Auf einem 32-bit-Prozessor werden die Chars, die Ganzzahlen und aufgelistete Typen auf 32 bit erweitert, bevor sie auf den Stapel gelegt werden
- Die Anzahl auf dem Stapel gelegten/reservierten Bytes muss ein Vielfaches von 8 sein
- Die Register haben folgende Funktion *(siehe. 04_ARM_Architecture_Procedure_Call_Standard.pdf)*

Register	Synonym	Special	Role in the procedure call standard
r15		PC	The Program Counter.
r14		LR	The Link Register.
r13		SP	The Stack Pointer.
r12		IP	The Intra-Procedure-call scratch register.
r11	v8		Variable-register 8.
r10	v7		Variable-register 7.
r9		v6 SB TR	Platform register. The meaning of this register is defined by the platform standard.
r8	v5		Variable-register 5.
r7	v4		Variable register 4.
r6	v3		Variable register 3.
r5	v2		Variable register 2.
r4	v1		Variable register 1.
r3	a4		Argument / scratch register 4.
r2	a3		Argument / scratch register 3.
r1	a2		Argument / result / scratch register 2.
r0	a1		Argument / result / scratch register 1.

Table 2, Core registers and AAPCS usage

Wiedereintrittscode - Bibliotheken

- **In der Informatik ist der Wiedereintritt die Eigenschaft einer Funktion, von mehreren Funktionen gleichzeitig genutzt werden zu können. Der Wiedereintritt verhindert die Duplizierung eines von mehreren Benutzern gleichzeitig verwendeten Programmes im Arbeitsspeicher.**
- **Die Konditionen, dass ein Programm wiedereintrittsfähig ist, sind:**
  - Darf keine statischen (globale) nicht-konstante Daten verwenden
  - Darf keine Adresse von statischen (globalen) nicht-konstanten Daten zurückgeben
  - Darf nur von dem aufrufenden Programm stammende Daten verwenden
  - Darf nicht auf Singleton-Ressourcensperren basieren
  - Darf seinen eigenen Code nicht ändern
  - Darf keine nicht-wiedereintrittsfähigen Unterprogramme aufrufen