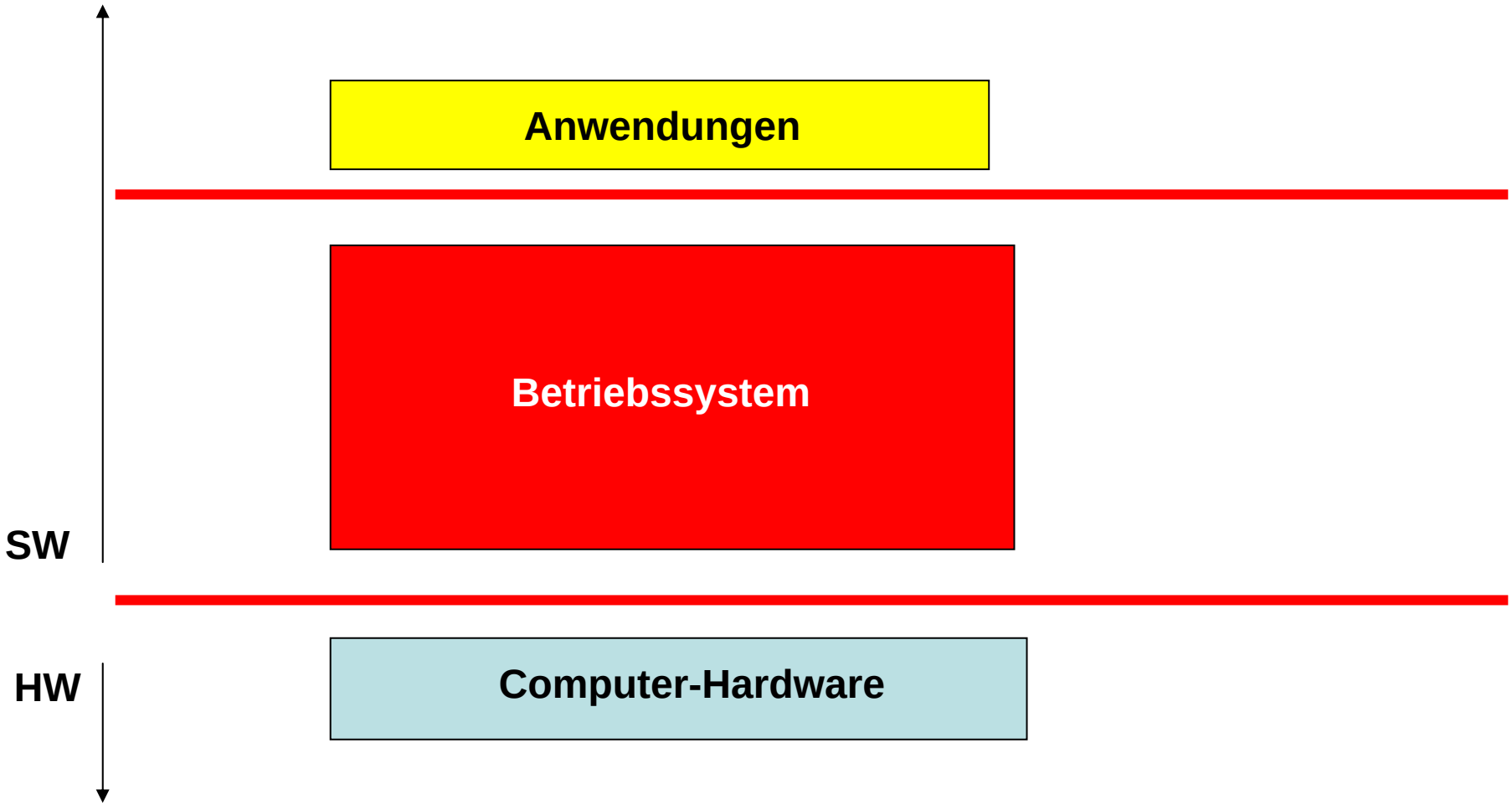
Betriebssysteme
WS 2006/2007

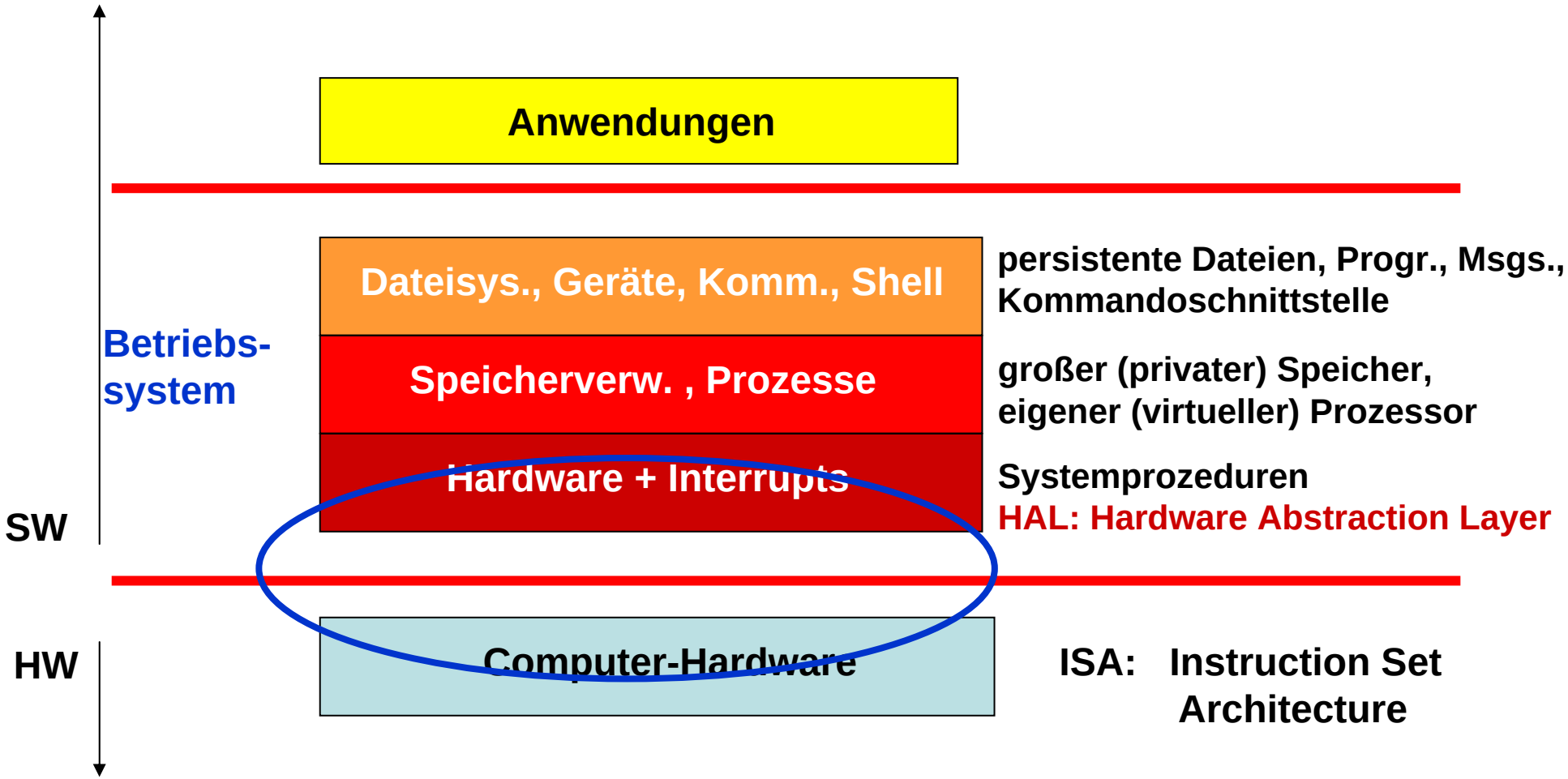
Jörg Kaiser
IVS – EOS

Otto-von-Guericke-Universität Magdeburg

Schichtenmodell

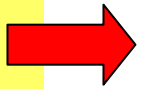


Schichtenmodell

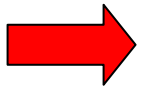


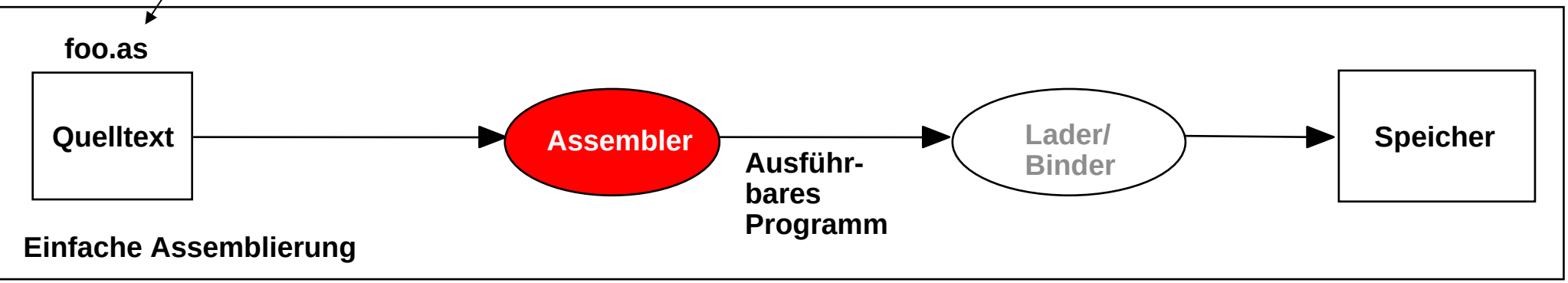
Einschub: Assemblieren, Binden und Laden

```
LOOP    LDA     5, PCR
        ADDA    LABEL
        BHI     LOOP

LABEL   some data
```



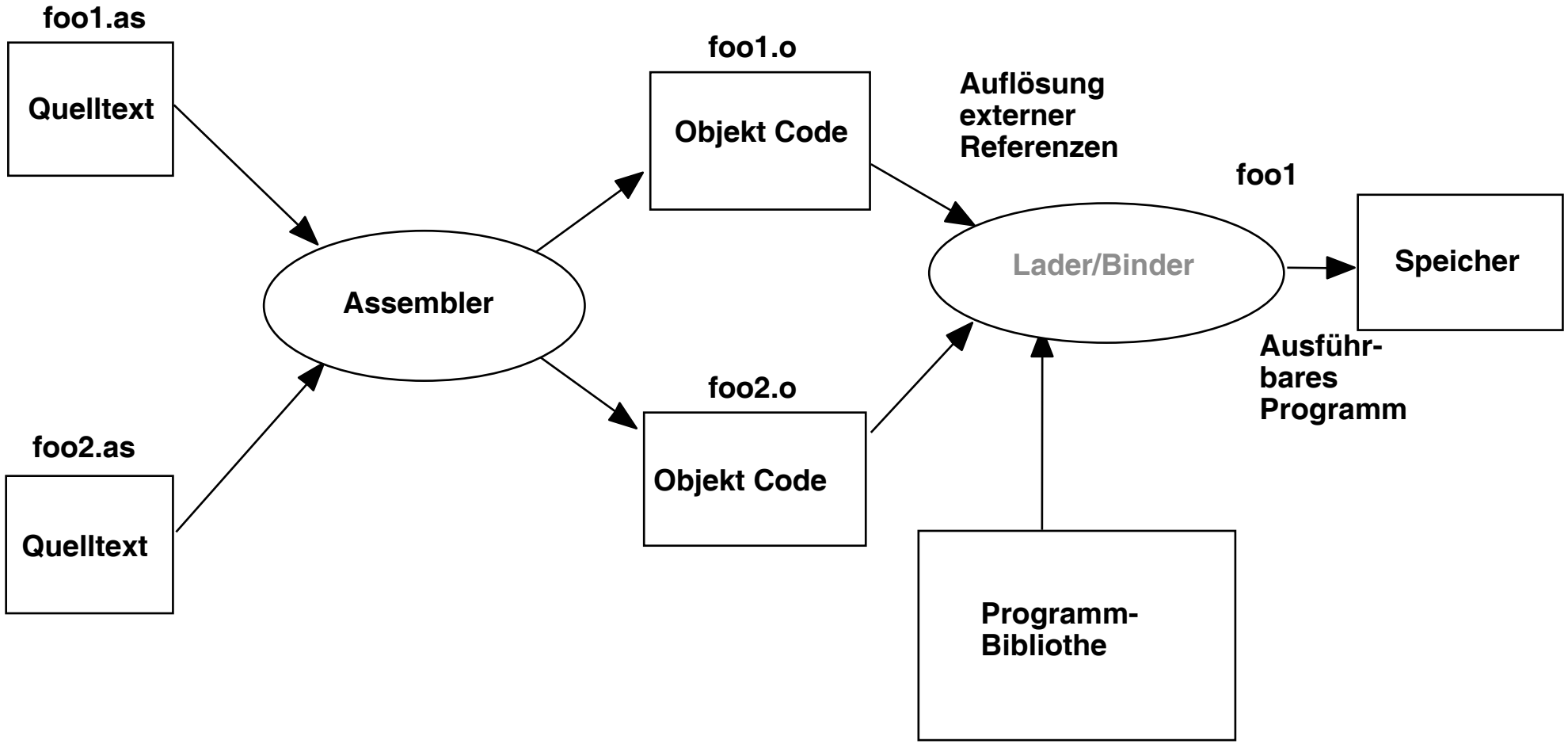
A6	10100110	10100110
8C	10001100	10001100
BB	10111011	10111011
label ?	xxxxxxxx	01101101
label ?	xxxxxxxx	01101101
22	00100010	00100010
FA	11111010	11111010



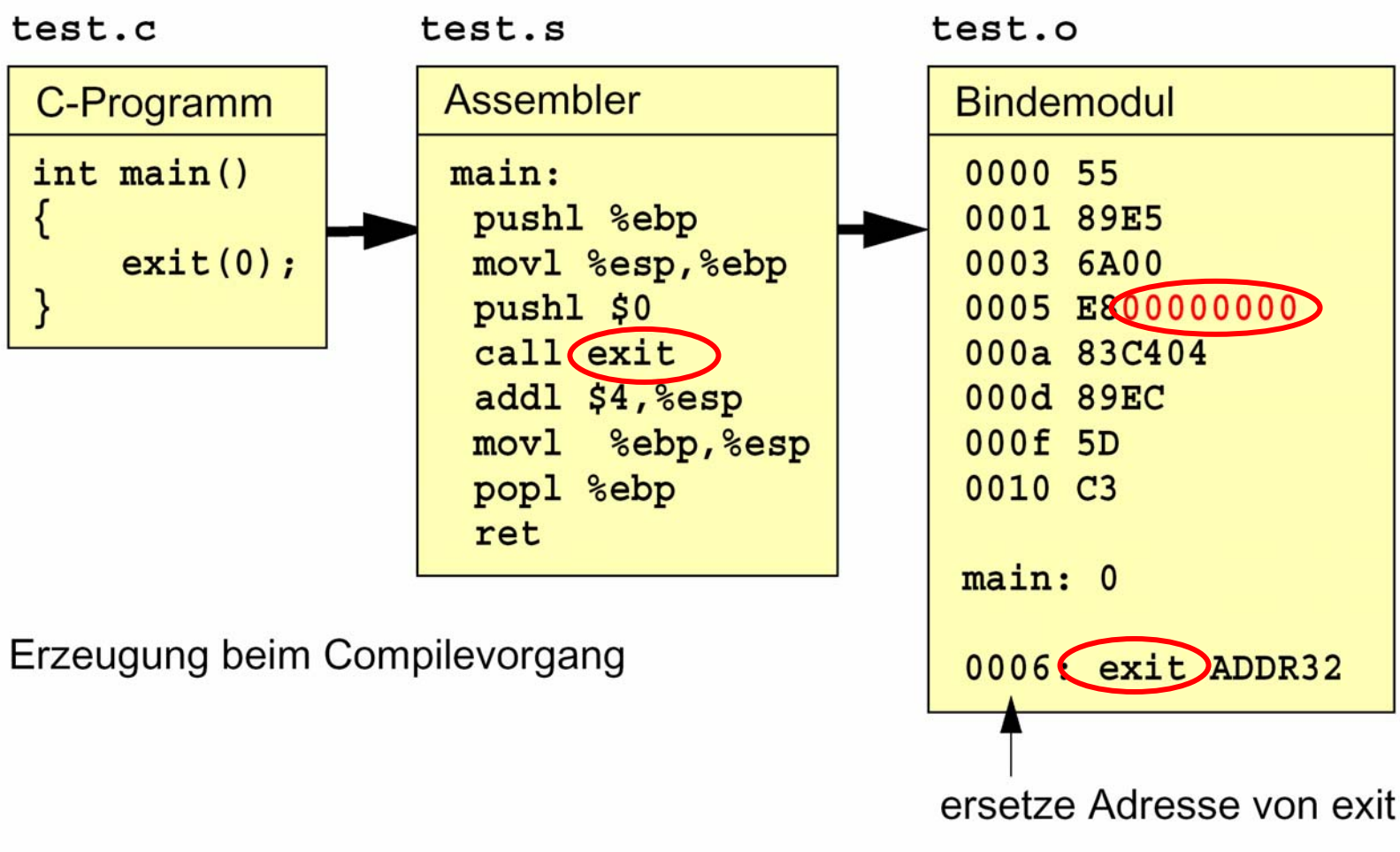


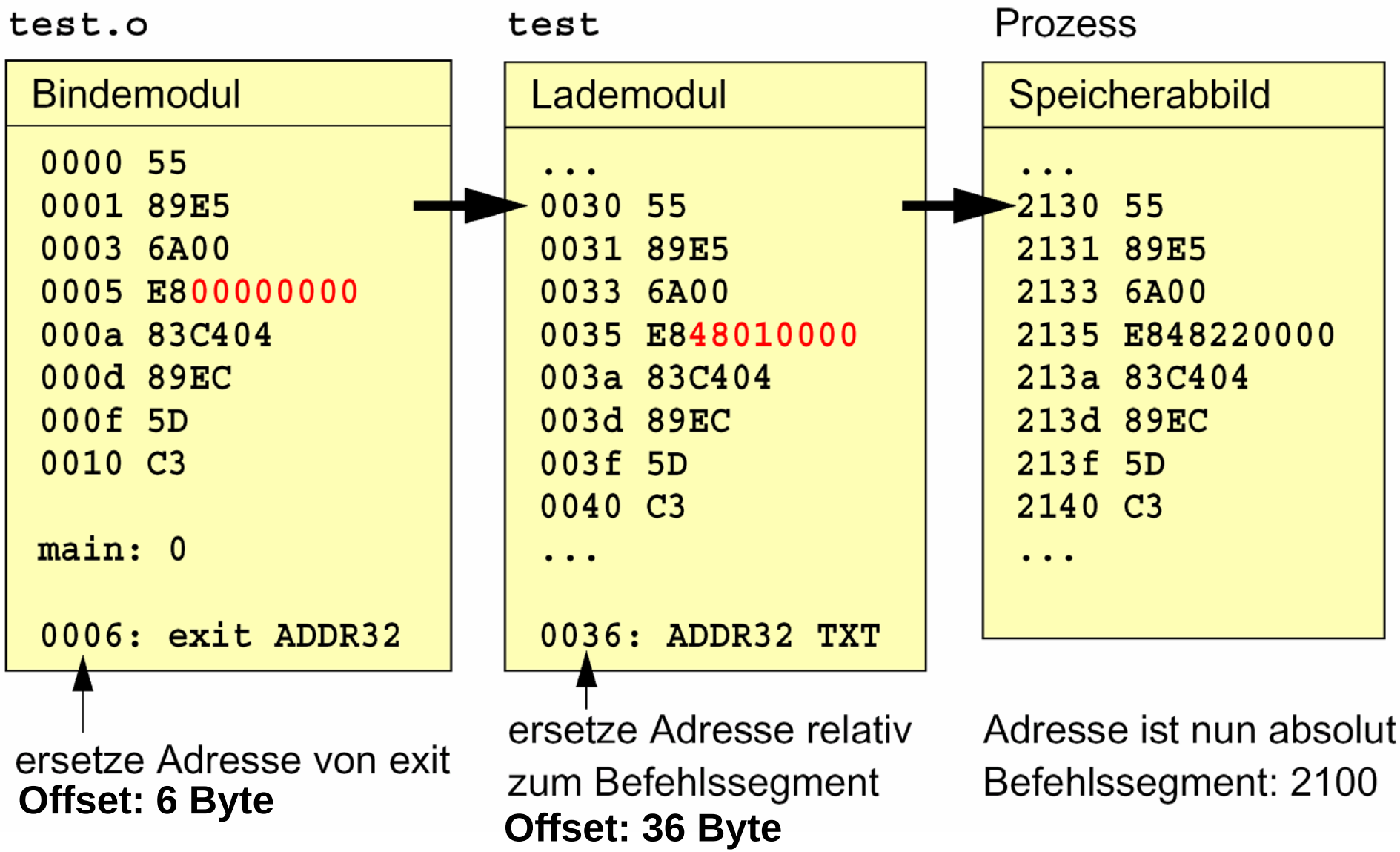
- Zuordnung von Namen zu Binärzahlen
- Mnemotechnische Codes
- Statische Reservierung von Speicherplatz
- Berechnung von Sprungadressen
- Erstellen der Speicherbelegung

Einschub: Assemblieren und Binden mehrerer Assemblermoduln



Compilierung von Hochsprachenprogrammen




test.o
Bindemodul
0000 55
0001 89E5
0003 6A00
0005 E800000000
000a 83C404
000d 89EC
000f 5D
0010 C3
main: 0
0006: exit ADDR32
ersetze Adresse von exit
Offset: 6 Byte
test
Lademodul
...
0030 55
0031 89E5
0033 6A00
0035 E848010000
003a 83C404
003d 89EC
003f 5D
0040 C3
...
0036: ADDR32 TXT
ersetze Adresse relativ
zum Befehlssegment
Offset: 36 Byte
Prozess
Speicherabbild
...
2130 55
2131 89E5
2133 6A00
2135 E848220000
213a 83C404
213d 89EC
213f 5D
2140 C3
...
Adresse ist nun absolut
Befehlssegment: 2100

Strukturierung von Aktivitäten und Nebenläufigkeit

Ein Programm "in Ausführung":

Benötigte Resourcen:

- Prozessor
- Speicher

Beschreibung eines Zustands während der Programmausführung:

- Befehlszähler
- Prozessorregister
- Speicherinhalt

Bekannte Programmierungskonzept:

- Prozedur, Routine, Unterprogramm, Funktion, ..

Strukturierung von Aktivitäten und Nebenläufigkeit

Progammierkonzept: Prozedur, Routine, Funktion, ..

Bestandteile:

- Programmcode
- Daten
- Dyn. Ausführungsumgebung
  - Parameter
  - Ausführungsstatus

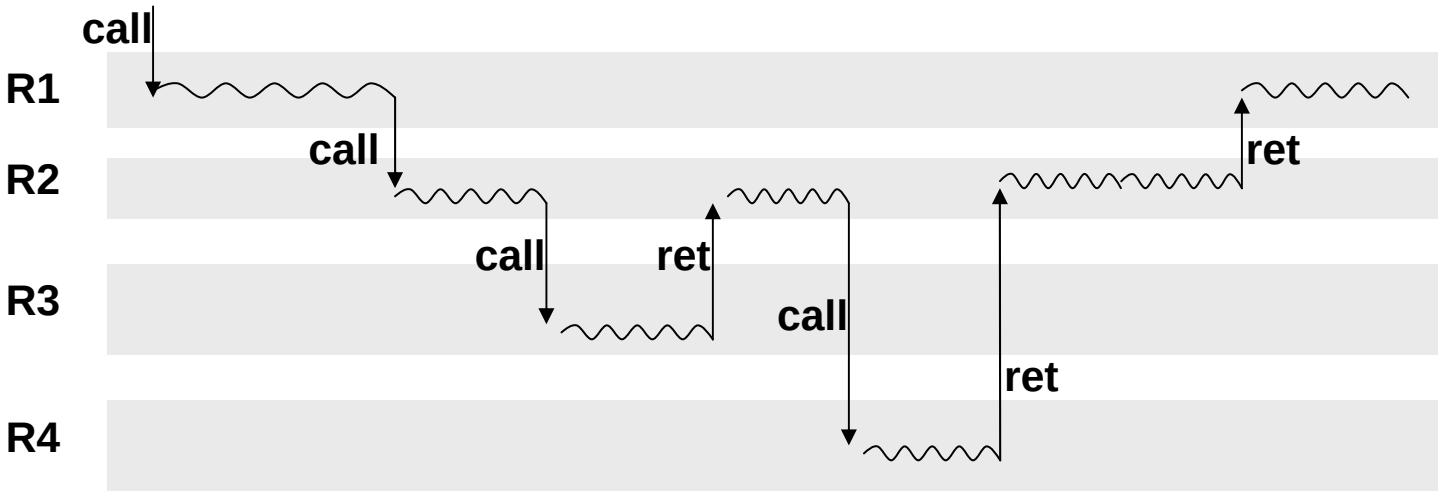
} wird auf dem Stack angelegt und verwaltet

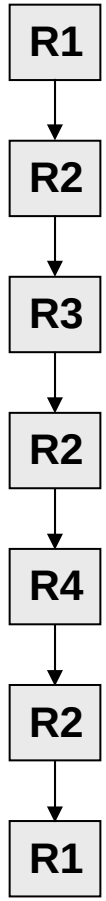
Aufruf: Programmierter, expliziter Aufruf aus dem Programm, Rückkehr zur Aufrufstelle nach Abarbeitung.

Systemunterstützung: Spezielle Instruktionen, Compiler

Strukturierung von Aktivitäten und Nebenläufigkeit

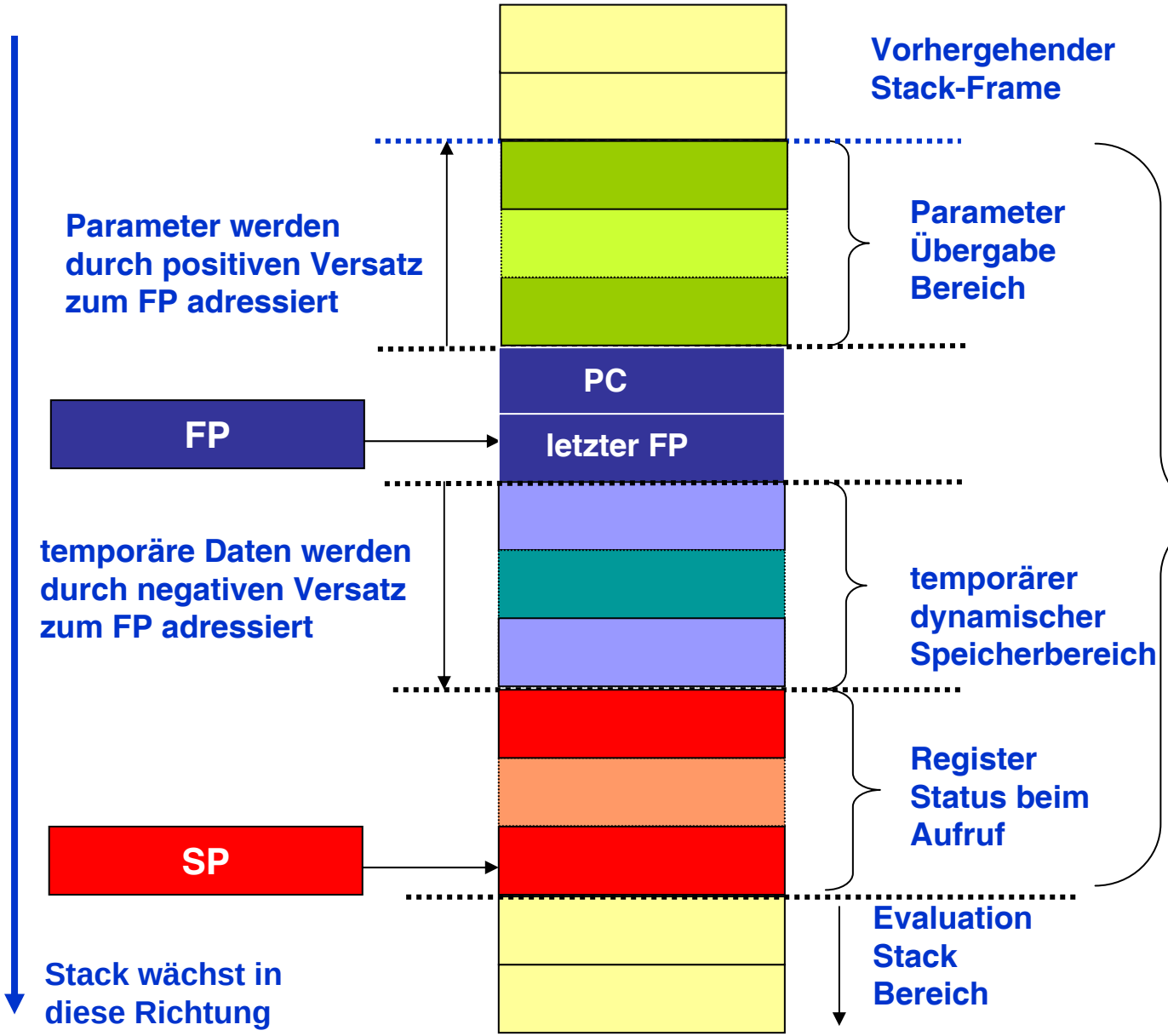
Aufrufhierarchie bei Routinen





feste, streng geschachtelte, hierarchische (asymmetrische) Aufruffolge.

Aktivierungs-Block



SP: Stack-Pointer
FP: Frame-Pointer

- activation record
- prozedure- context
- stack-frame

Strukturierung von Aktivitäten und Nebenläufigkeit

Der Programmcode einer Prozedur kann in mehreren Ausführungsumgebungen, repräsentiert durch einen Aktivierungsblock, unabhängig ausgeführt werden!

Beispiel: Rekursives C - Programm zur Berechnung der Fakultät

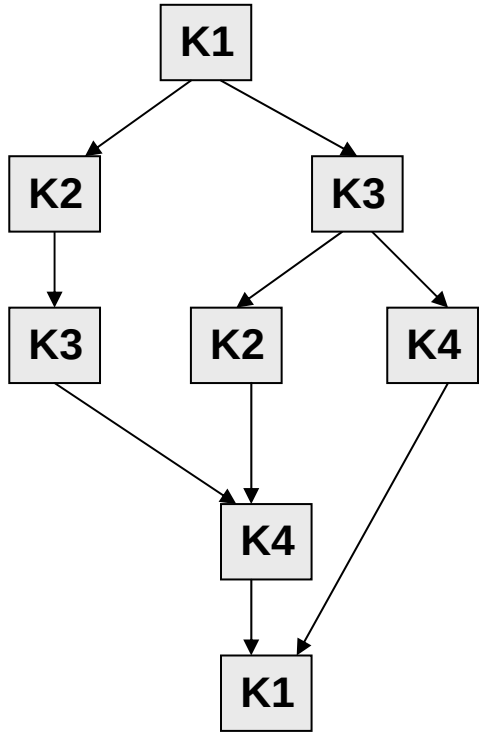
```
main ()
{
    printf ("die Fakultät von n ist: %d\n", fact (n));
}

int fact (int n)
{
    if (n < 1)
      return (1);

    else
      return (n * fact (n-1));
}
```

Das Konzept der Ko-Routine

- Mittel zur Strukturierung und expliziten Kontrolle nebenläufiger Aktivitäten.
- KR repräsentieren gleichberechtigte, autonome Kontrollflüsse.
- Gleichberechtigte (symmetrische) Beziehungen.
- Anstelle eines "Return" bei Routinen wird beim Wechsel der Koroutinen jedesmal explizit angegeben, wohin die Kontrolle transferiert werden soll.
- Abfolge der Aufrufe kann sich ändern.

Routinen

Koroutinen



Das Konzept der Ko-Routine

Primitive zur Steuerung von Koroutinen:

create: Erzeugung, nicht Aktivierung, einer neuen Koroutinen

resume: Supendierung der laufenden Koroutine und Übertragen der Kontrolle auf eine andere Koroutine. Wiederaufnahme einer suspendierten Koroutine an der Stelle, an der die Kontrolle abgegeben wurde.

Grundregeln:

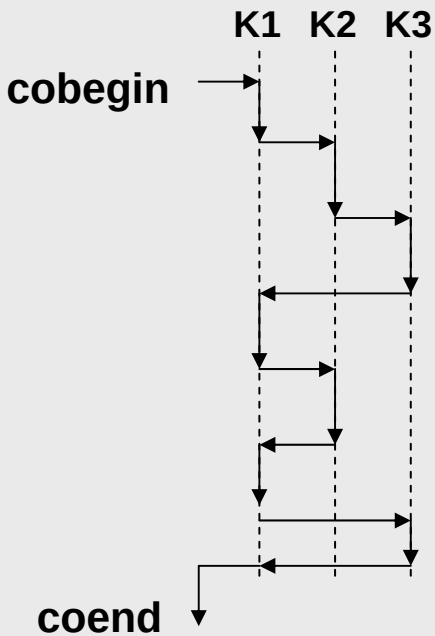
- die Ausführung einer Koroutine muss dort fortgesetzt werden, wo die Koroutine suspendiert wurde.
- jede Koroutine mus zu Ende kommen.

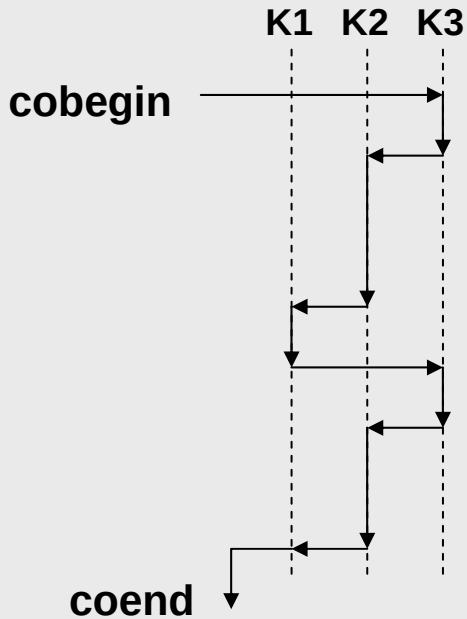
Das Konzept der Ko-Routine

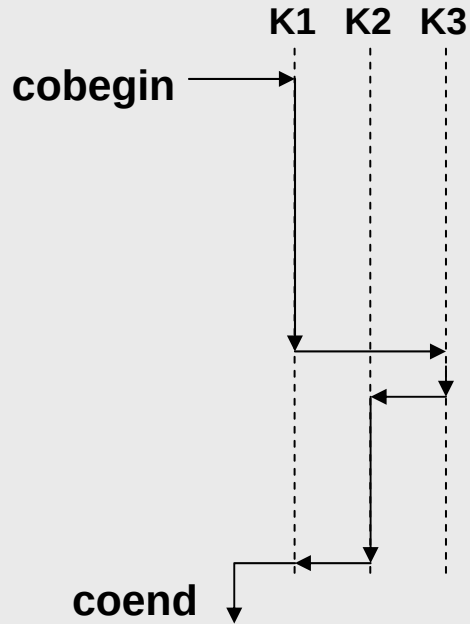
Beispiel:

```
Cobegin
    task K1 (n,m);
    task K2 (n, x, u);
    task K3 (f, g, h);
Coend.
```

Mögliche Aufrufabfolge der Koroutinen:







Eigenschaften der Koroutinen

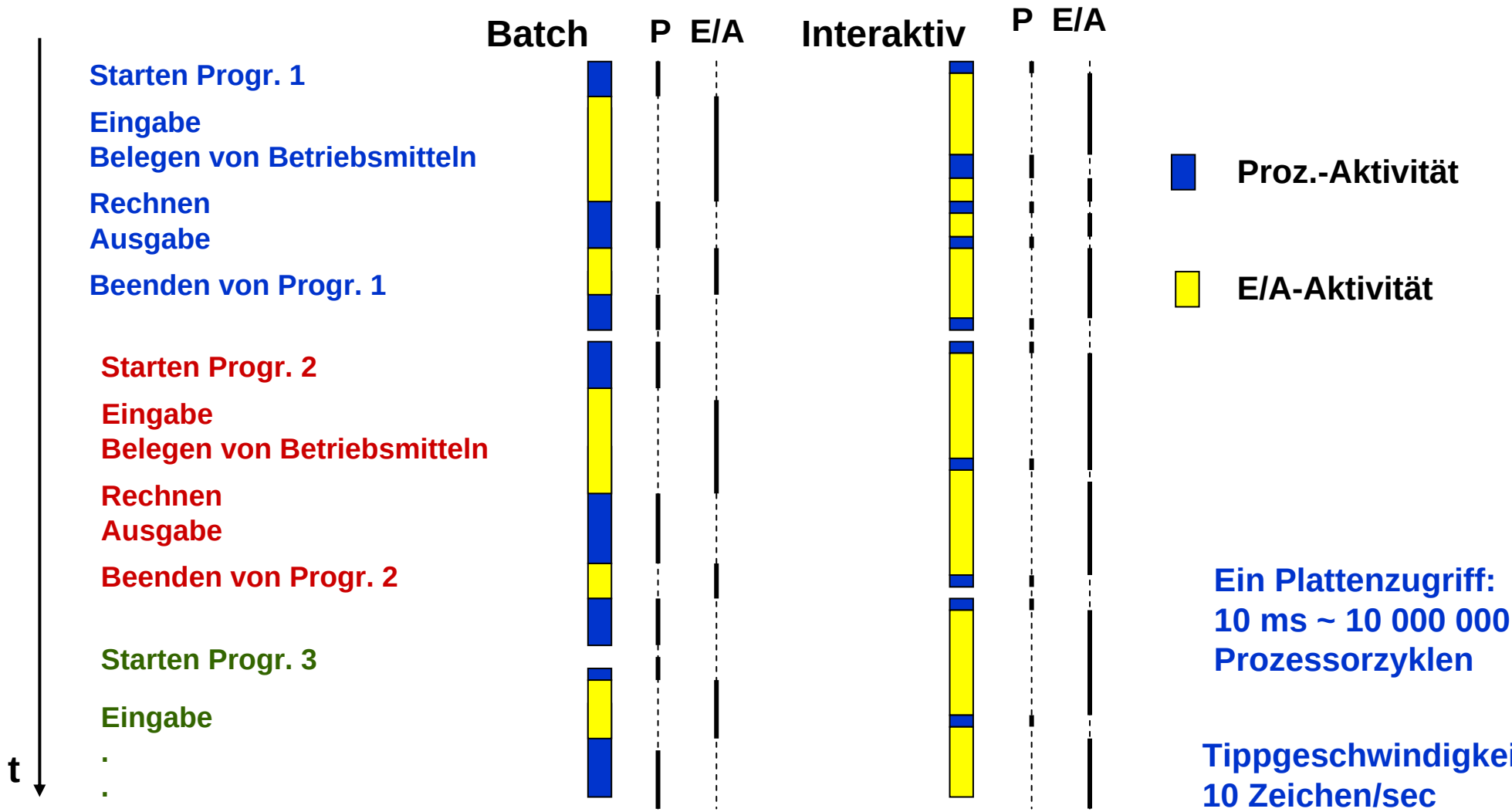
1. die Ausführung beginnt immer an der letzten ”Unterbrechungsstelle“, d. h., an der zuletzt die Kontrolle über den Prozessor abgegeben wurde.

2. Die Kontrollabgabe geschieht dabei grundsätzlich kooperativ (freiwillig).

3. Der Zustand ist invariant zwischen zwei aufeinanderfolgenden Ausführungen.

4. Eine Koroutine muss ihren Zustand bei Abgabe der Kontrolle speichern. Sie kann als ”zustandsbehaftete Prozedur“ aufgefasst werden.

"One program at a time"



Ein Plattenzugriff:
10 ms ~ 10 000 000 Prozessorzyklen

Tippgeschwindigkeit:
10 Zeichen/sec

"One program at a time"

Fragen:

1. Woher weiß das Programm, wann eine E/A Operation stattfindet oder beendet ist?

2. Was macht man mit der Wartezeit?

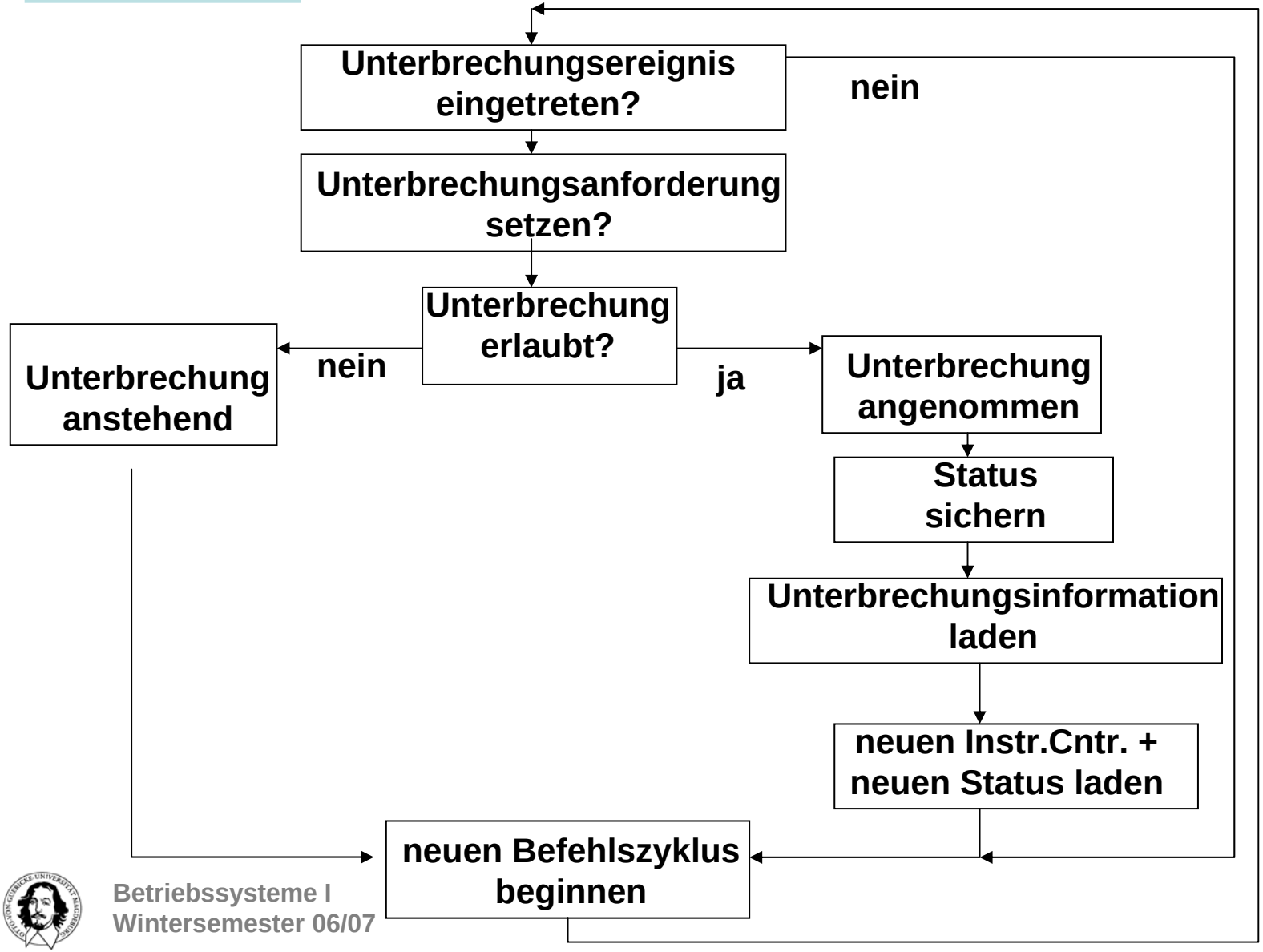
Effizienz

Synchronisation

Ad hoc Lösung zu 1:
Programmierte Ein/Ausgabe, periodische Abfrage von Kontrollregistern und Gerätezustand. ➡ Aktives Warten, busy waiting.

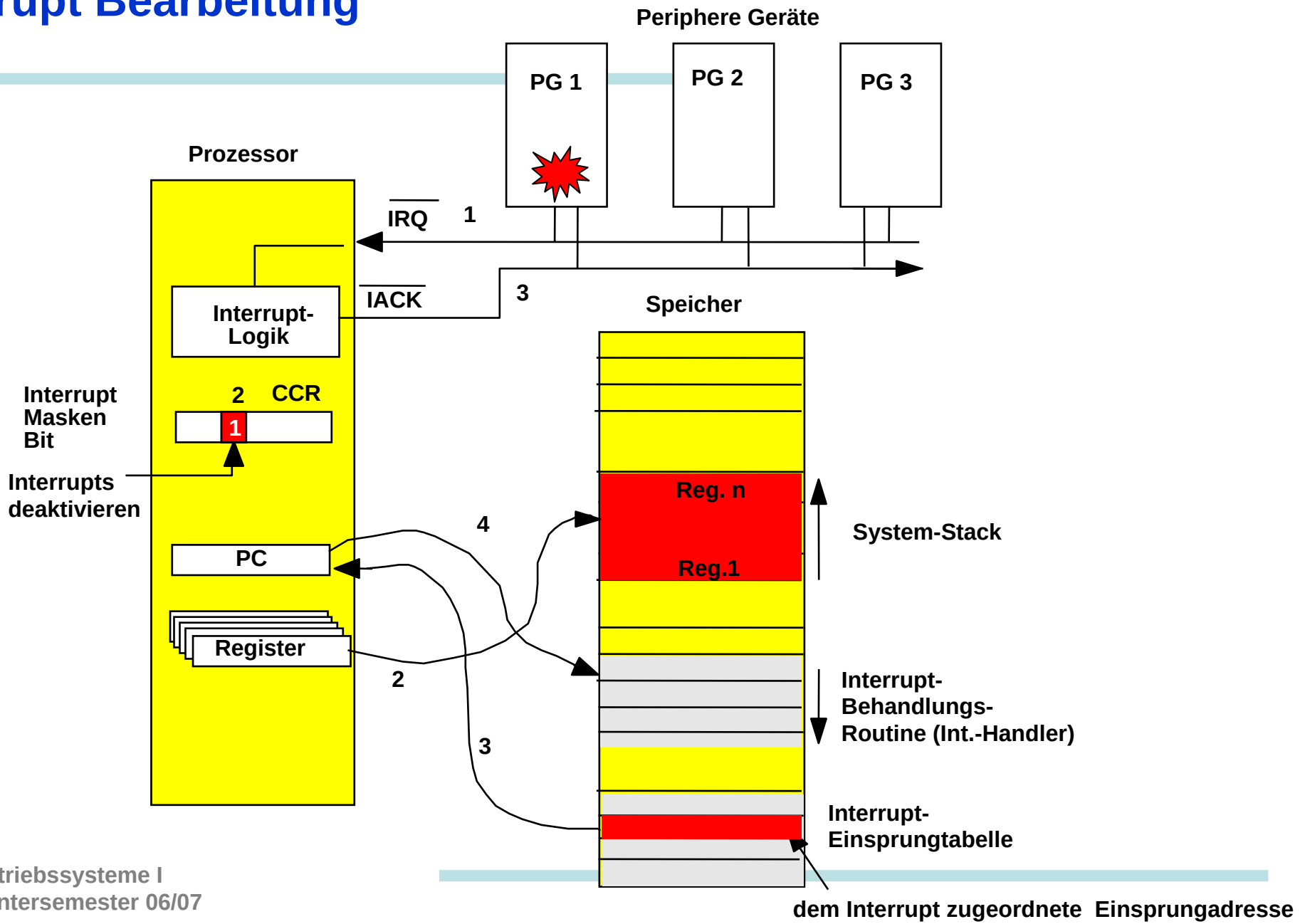
Ad hoc Lösung zu 2:
Explizite Programmierung der Ausführung mehrerer Aufgaben in einem sequentiellen Programmablauf.

Koordination mit Geräten: Systembezogene Unterbrechungen



Hardware-Unterstützung durch die CPU

Interrupt Bearbeitung



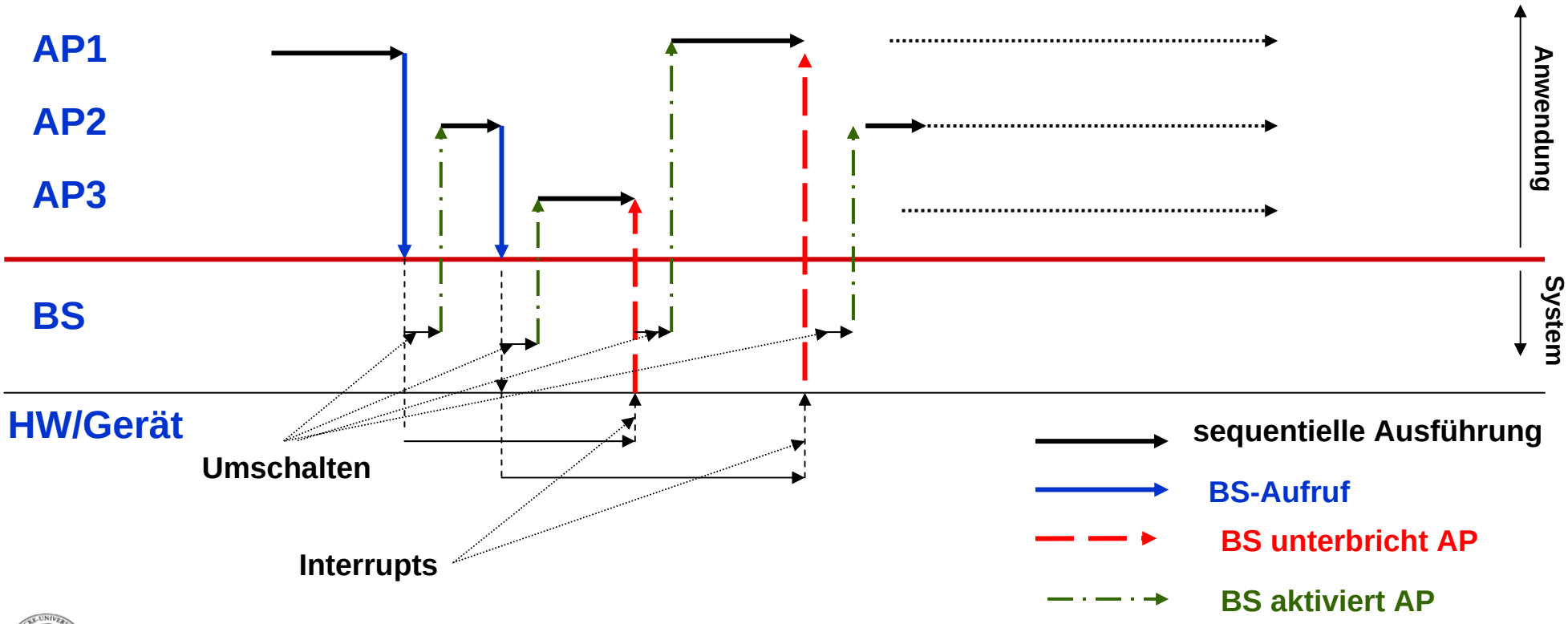
Strukturierung von Aktivitäten und Nebenläufigkeit

Eigenschaften von Unterbrechungen:

- **Unterbrechungen führen auf der Hardwareebene einen Kontrolltransfer durch.**
- **Unterbrechungen werden zwingend behandelt.**
- **Unterbrechungen führen häufig zu einem Wechsel der Schutzebene z.B. von der Anwendungsebene in die Systemebene.**

Strukturierung von Aktivitäten und Nebenläufigkeit

Unterbrechungsbearbeitung bietet die Möglichkeit zu einer für das Anwendungs-programm (AP) völlig transparenten Nebenläufigkeit.

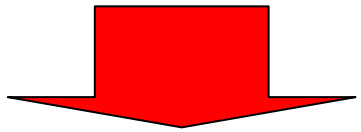


Strukturierung von Aktivitäten und Nebenläufigkeit

Gesucht: Mechanismus zum sicheren Aufruf von Funktionen/Diensten des Betriebssystems.

Problem: Trennung von Anwenderadressraum und Systemadressraum.
Häufig Unterstützung durch die Hardware: Supervisor/User Status, MMU

Anforderungen: Nur erlaubte Einsprungstellen in die Systemroutinen.
Transparenz, wo bestimmte Routinen tatsächlich liegen.



Programmbezogene Unterbrechung, Trap: synchrone, reproduzierbare Unterbrechung

- spezielle Instruktion der CPU oder Ausnahmebedingung der Hardware.
- Systemaufruf.
- Adreßraumverletzung.
- Unbekannter Befehl.
- Falsche Adressierungsart.
- Fehlerhafte Rechenoperation.

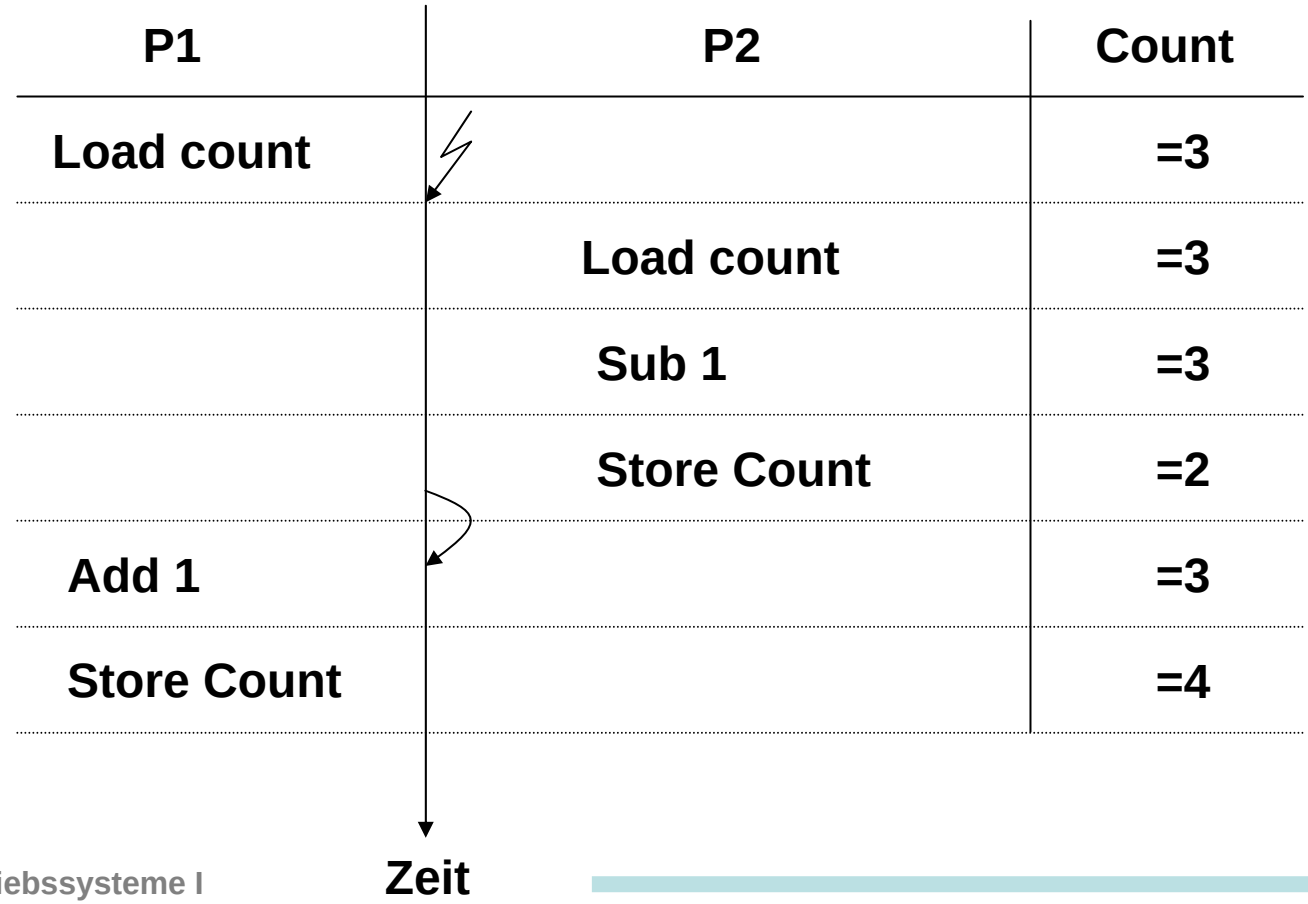
Strukturierung von Aktivitäten und Nebenläufigkeit

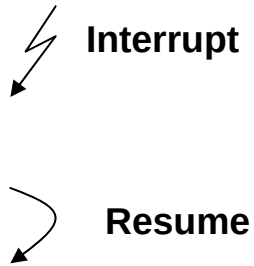
Vergleich Interrupt und Trap:

	Interrupt	Systemaufruf (Trap)	Ausnahme (Trap)
Quelle	extern	intern	intern
Synchronität	async.	sync.	sync.
Vorhersagbarkeit	nein	ja	nur bedingt
Reproduzierbarkeit	nein	ja	ja

Der Preis der Nebenläufigkeit: Konsistenz

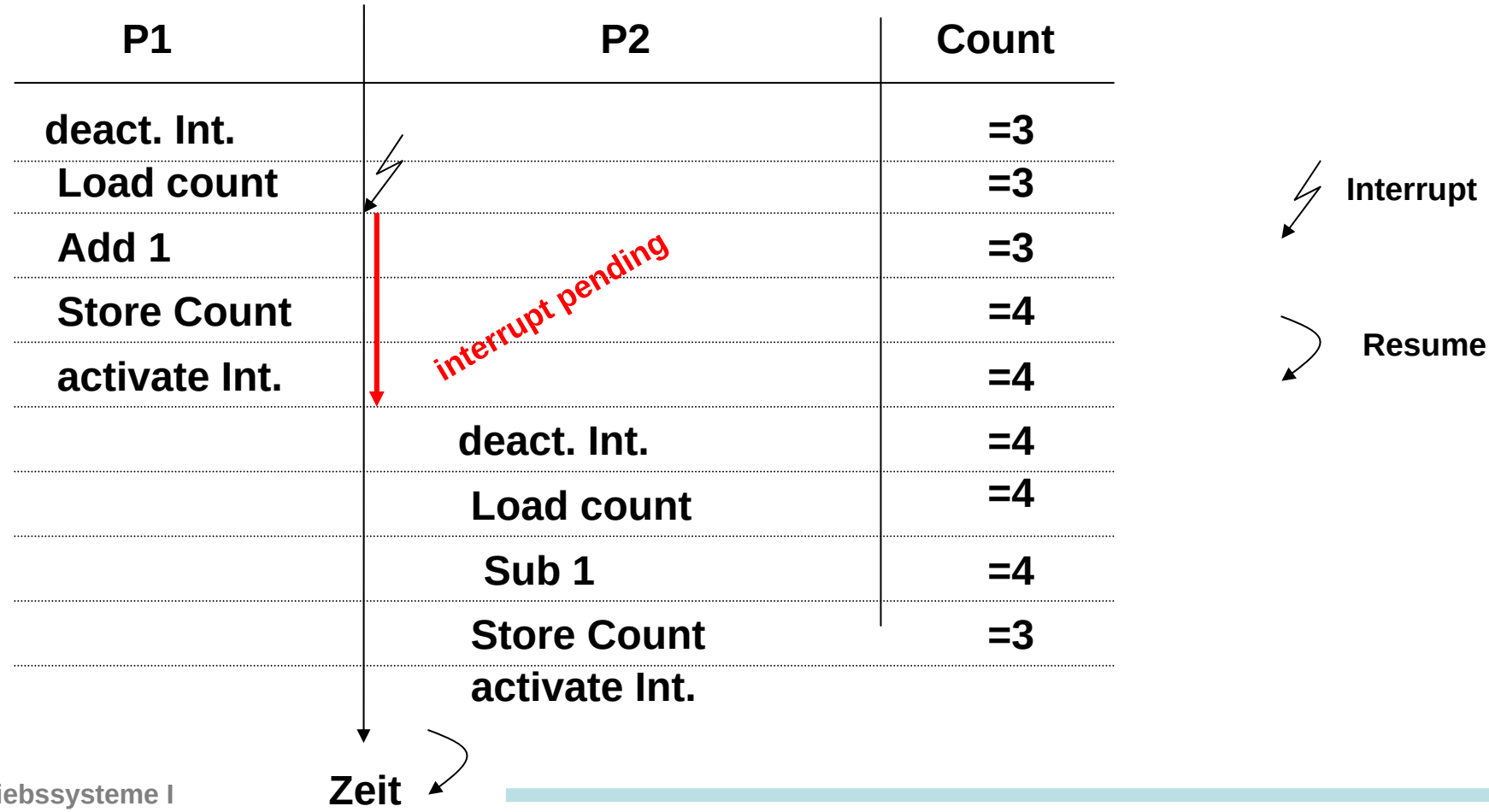
Beispiel: Gemeinsame Nutzung einer Zählvariablen.
Problem: Transparenz der nebenläufigen Aktivitäten.





Der Preis der Nebenläufigkeit: Konsistenz

Lösungsansatz 1: Abschalten der Interrupts



Probleme

Während der Zeitdauer der Interruptabschaltung können wichtige Interrupts nicht behandelt werden.

Deaktivierung betrifft auch völlig unabhängige Programme, die keine gemeinsame Variable nutzen.

Der Preis der Nebenläufigkeit: Konsistenz

Lösungsansatz 2: Unteilbare, atomare Operationen

- Unteilbare read-modify-write Buszyklen,
- Test and Set (TAS), Compare and Swap (CAS),
- Atomare Befehlssequenzen (z.B. ATOMIC (C167 Microcontroller))

Zur Erinnerung: Diese Lösungen werden von den unteren Schichten der Hardware im Zusammenspiel mit dem BS verwendet. Weitere Lösungen zu den Problemen der Nebenläufigkeit werden später behandelt.

Zusammenfassung

Programme werden in einer individuellen Umgebung ausgeführt.

Die Ausführungsumgebung wird durch einen Aktivierungsblock repräsentiert.

Ein Programm kann mehrfach ausgeführt werden.

Routinen bilden eine asymmetrische, synchrone Aufrufbeziehung.

Koroutinen repräsentieren gleichberechtigte, autonome Kontrollflüsse.

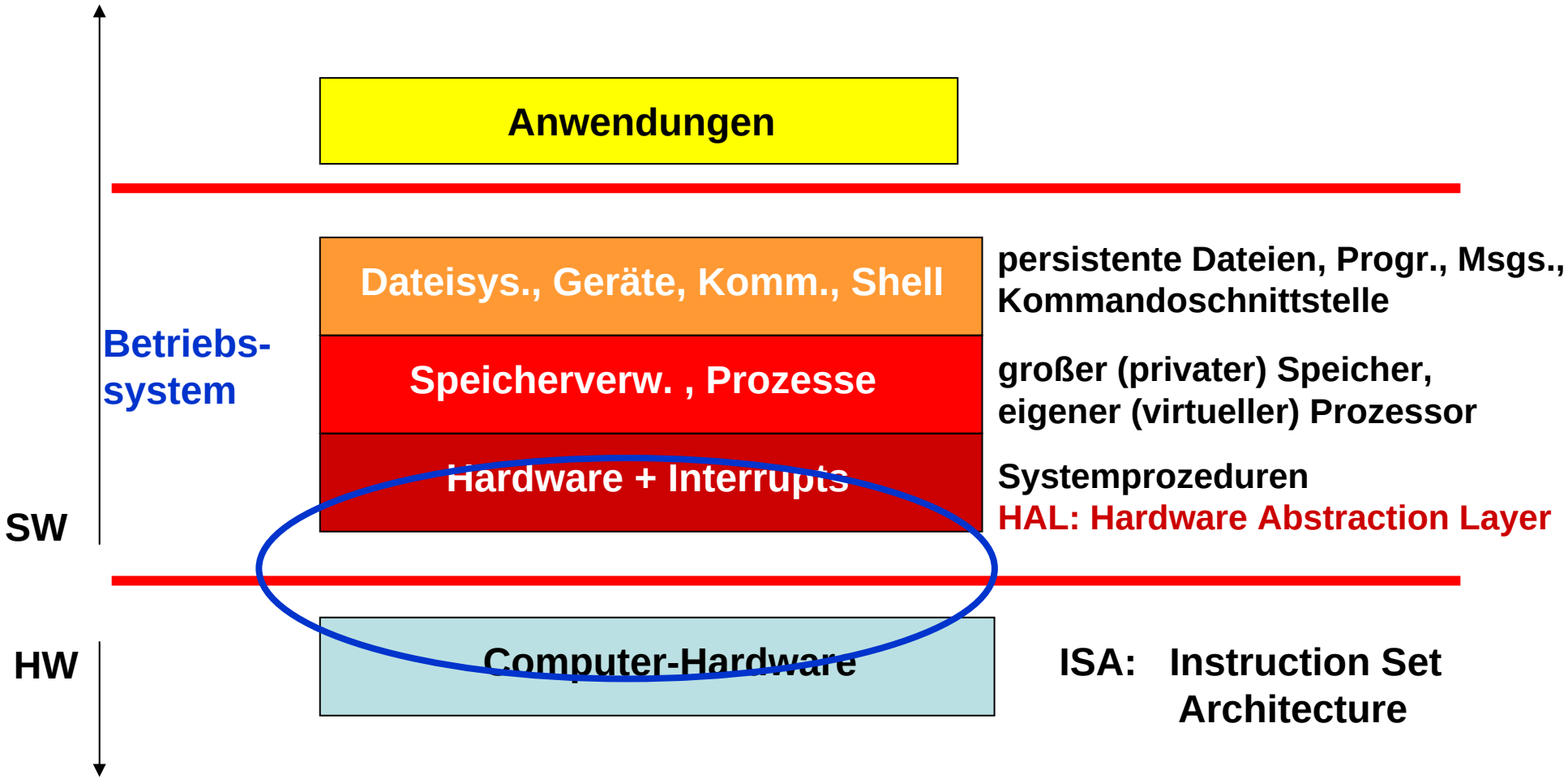
Koroutinen bilden eine symmetrische, synchrone Aufrufbeziehung.

Unterbrechungsbearbeitung ermöglicht eine transparente Ausführung mehrerer Programme.

Kontrolltransfer wird nicht mehr kooperativ organisiert, sondern übergeordnet.

Nebenläufigkeit erfordert Maßnahmen zur Konsistenzerhaltung.

Schichtenmodell



Aktivitäten und Zustand

	Aktivitätsträger	Zustandsträger
physische HW-Komponenten	Prozessor(en)	flüchtiger Speicher, persistenter Speicher, Geräte
BS- Abstraktionen	Prozesse, Threads (Ausführungsfäden)	Adreßräume, Dateien