3 Prozesse und Nebenläufigkeit

3.1 Das Prozeßkonzept

- Programm: Folge von Anweisungen (statisch), wird im Speicher hinterlegt
- Prozeß: In Ausführung befindliches Programm mit den zugehörigen Daten (dynamisch)
  hier Rechenprozeß, nicht technischer Prozeß

  Ein Programm kann sich mehrfach in Ausführung befinden!

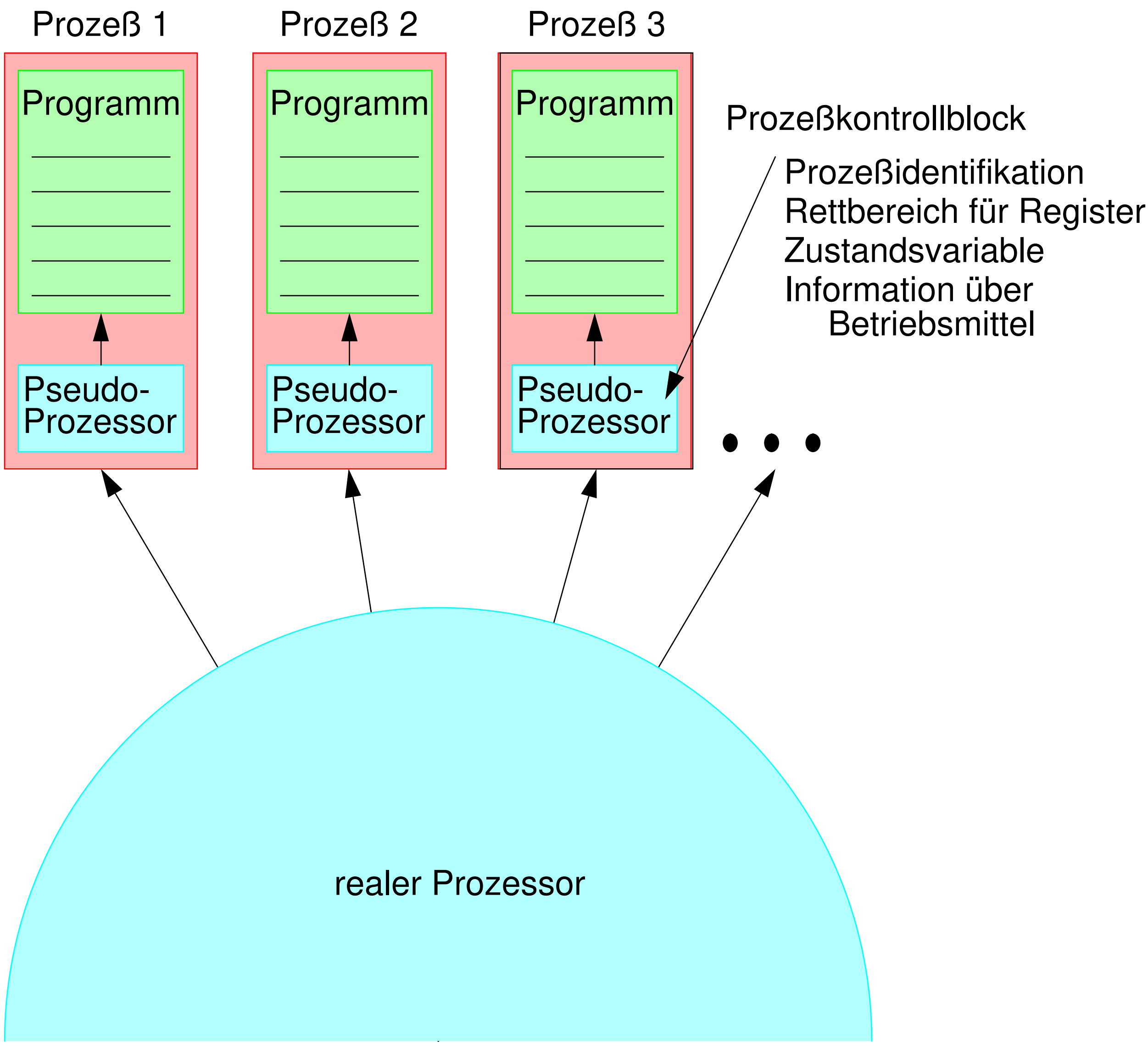
- Prozeßzustände
  - Bereit (*ready*)

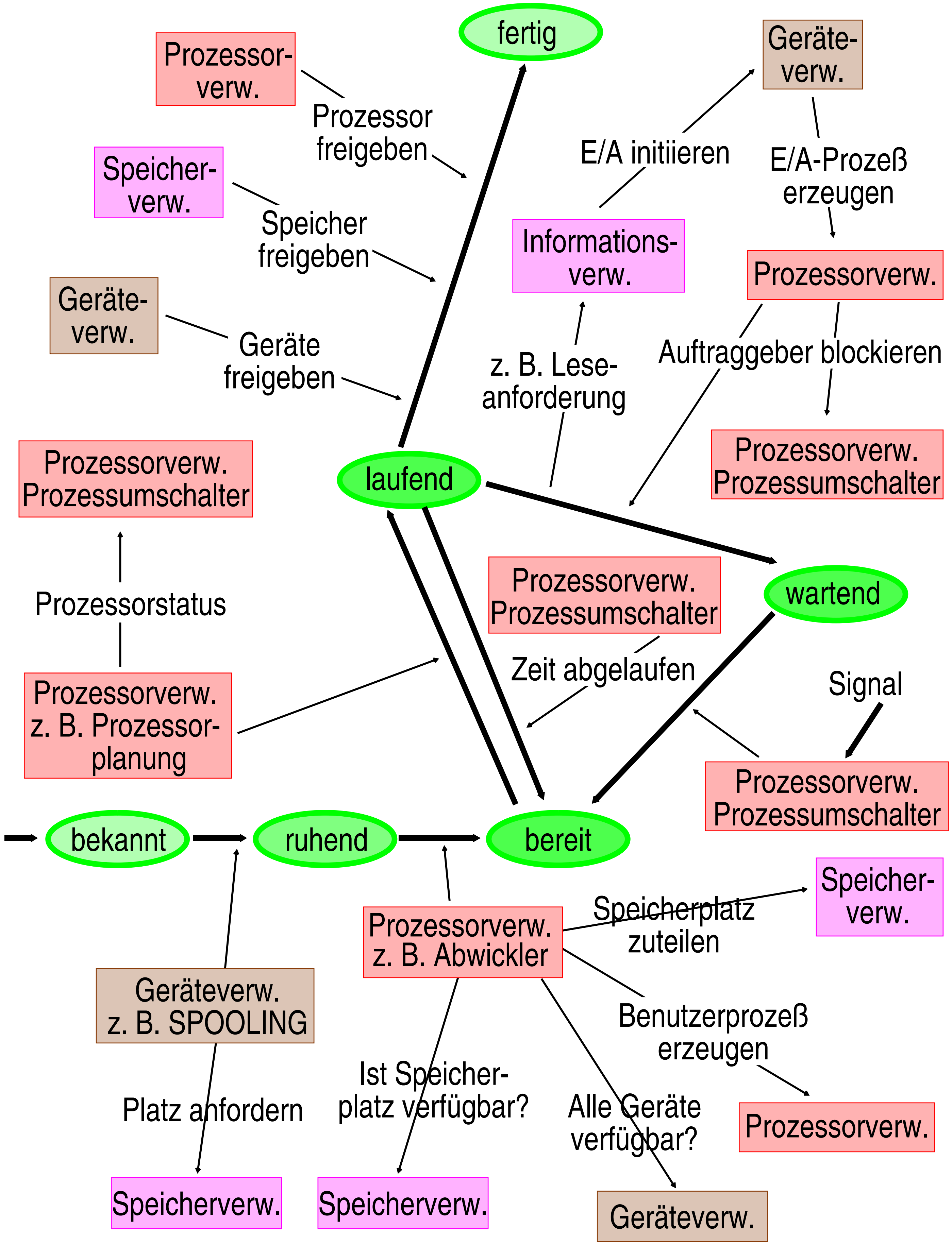
    Der Prozeß bewirbt sich um die Zuteilung des realen Prozessors und hat bereits alle anderen Betriebsmittel, die für seine Weiterarbeit erforderlich sind, zugeteilt bekommen.
  - Laufend (*running*)

    Der Prozeß wird vom realen Prozessor ausgeführt.
  - Blockiert (*blocked*)

    Der Prozeß bewirbt sich nicht um die Zuteilung des Prozessors, da er auf die Erfüllung einer Bedingung (z. B. dem Abschluß eines E/A-Transfers) wartet.
  - Beendet (*terminated*)

    Der Prozeß hat all seine Berechnungen beendet und die ihm zugeteilten Betriebsmittel freigegeben.


Prozeß 1
Prozeß 2
Prozeß 3
Programm
Programm
Programm
Pseudo-
Prozessor
Pseudo-
Prozessor
Pseudo-
Prozessor
Prozeßkontrollblock
Prozeßidentifikation
Rettbereich für Register
Zustandsvariable
Information über
Betriebsmittel
realer Prozessor

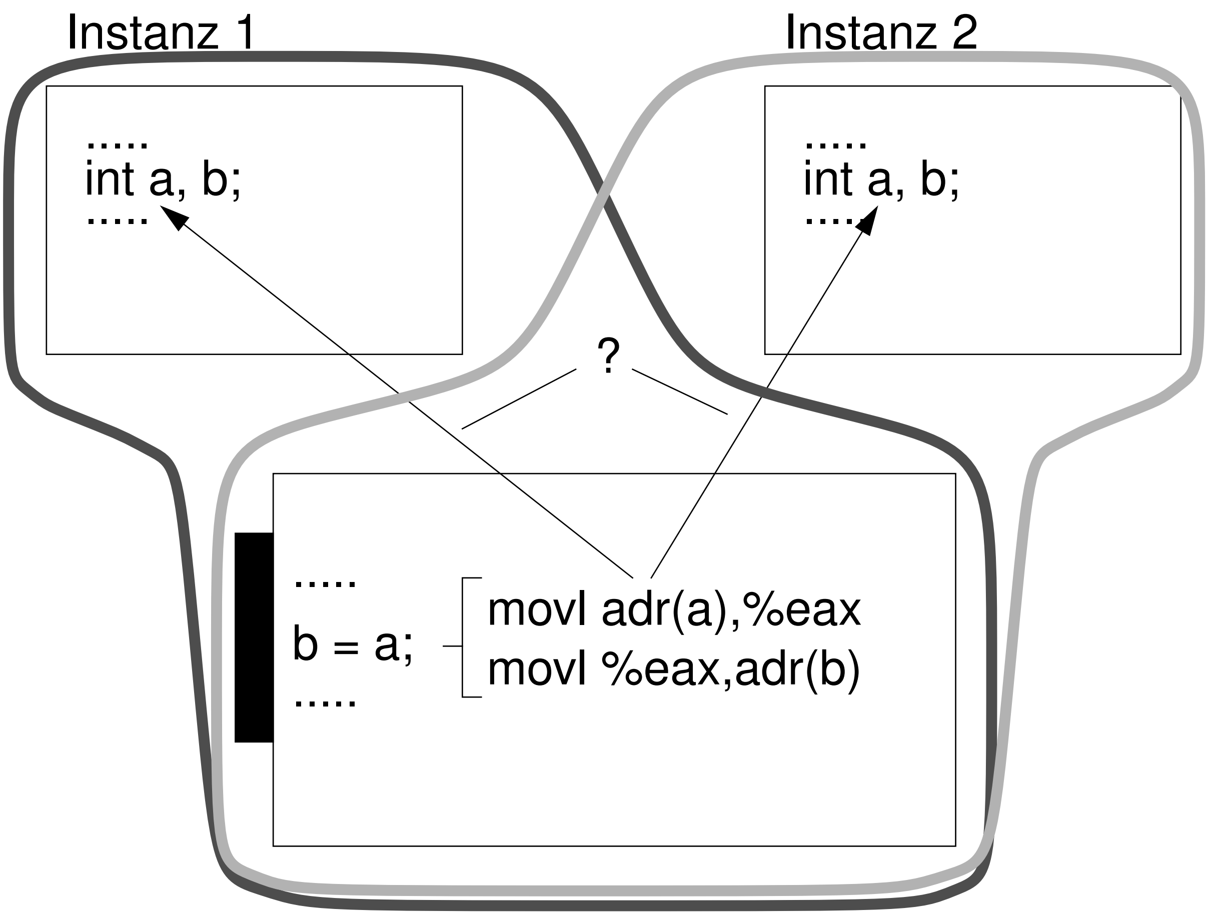

fertig
Prozessor-
verw.
Prozessor
freigeben
Speicher-
verw.
Speicher
freigeben
Geräte-
verw.
Geräte
freigeben
E/A initiieren
Geräte-
verw.
E/A-Prozeß
erzeugen
Informations-
verw.
Prozessorverw.
z. B. Lese-
anforderung
Auftraggeber blockieren
Prozessorverw.
Prozessumschalter
Prozessorverw.
Prozessumschalter
laufend
Prozessorstatus
Prozessorverw.
Prozessumschalter
wartend
Zeit abgelaufen
Signal
Prozessorverw.
z. B. Prozessor-
planung
Prozessorverw.
Prozessumschalter
bekannt
ruhend
bereit
Speicher-
verw.
Speicherplatz
zuteilen
Prozessorverw.
z. B. Abwickler
Benutzerprozeß
erzeugen
Geräteverw.
z. B. SPOOLING
Platz anfordern
Ist Speicher-
platz verfügbar?
Alle Geräte
verfügbar?
Prozessorverw.
Speicherverw.
Speicherverw.
Geräteverw.

- Bestandteile
  - Befehlssegment (pro Klasse)
  - Datensegment (pro Instanz)
  - Stapelsegment (pro Aktivitätsträger)

- Ablauf
  - Register: Befehlszähler, Index-/Basisregister
  - Methodenaufruf und -beendigung

- Ausnahmebehandlung
  - Programmstatuswort
  - Unterbrechungen, Signale
    - Traps
    - Interrupts

Bindung Befehlsadressen - Datensegment



Mögliche (wenn auch unbefriedigende) Lösung:

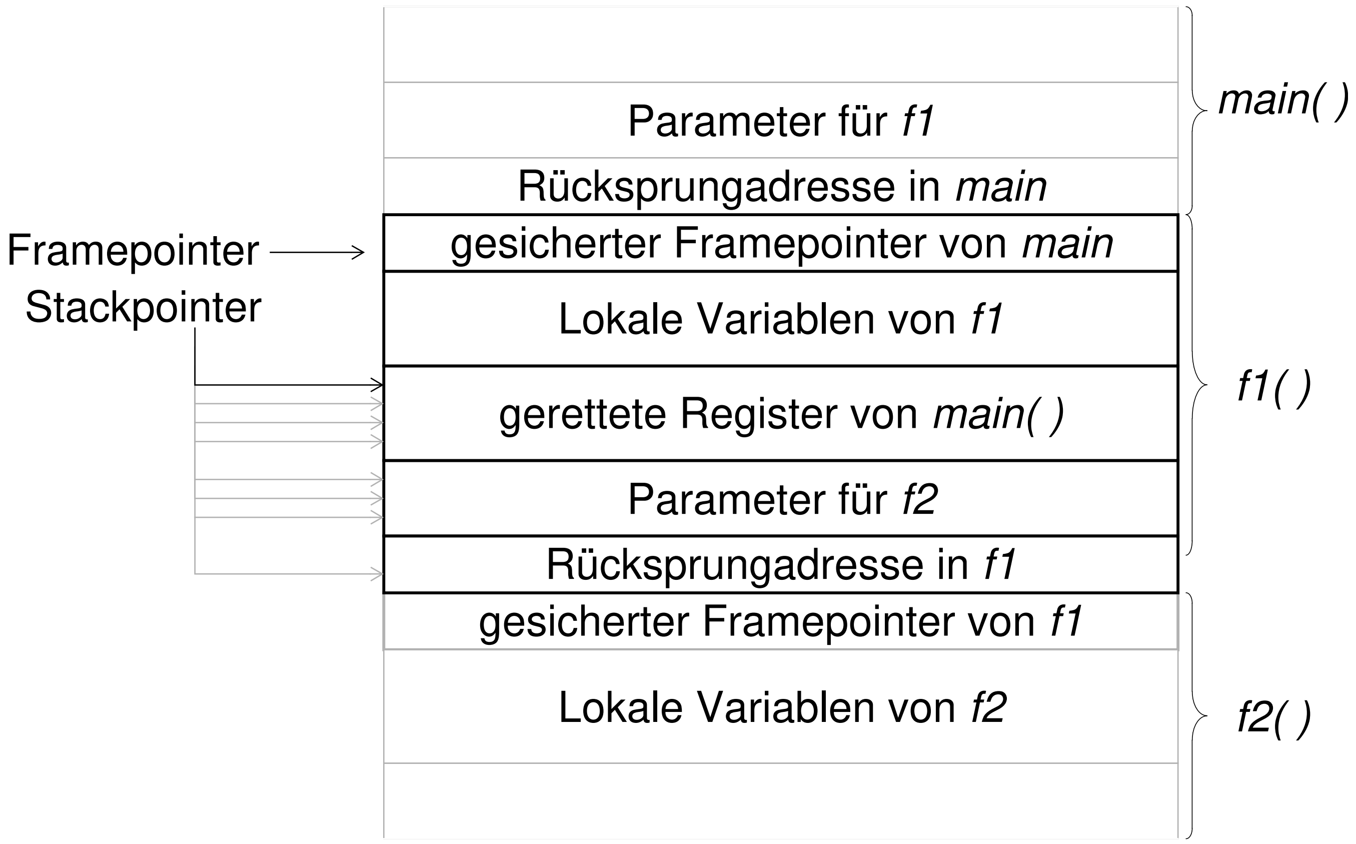
Benutzung indizierter Befehle

```
1                           basis:
2 0000 00000000             a:        .long   0
3 0004 00000000             b:        .long   0
4
5                           start:
6 0008 8D 2D 00000000  leal basis,%ebp
7 000e 8B 85 00000000  movl a-basis(%ebp),%eax
8 0014 89 85 04000000  movl %eax,b-basis(%ebp)
```

Stapelaufbau

- Bei jedem Methodenaufruf wird ein ***Stack-Frame*** angelegt, in dem
  - lokale Variablen der Methode
  - Aufrufparameter an weitere Methoden
  - Registerbelegung der Methode während des Aufrufs weiterer Methoden

  gespeichert werden.
- Stapelorganisation ist Prozessor-, Compiler- und Betriebssystem-abhängig (Beispiele aus einem UNIX auf Motorola 68k-Prozessor)
- Aufbau eines *Stack-Frame* (Funktionen *main( ), f1( ), f2( )*)

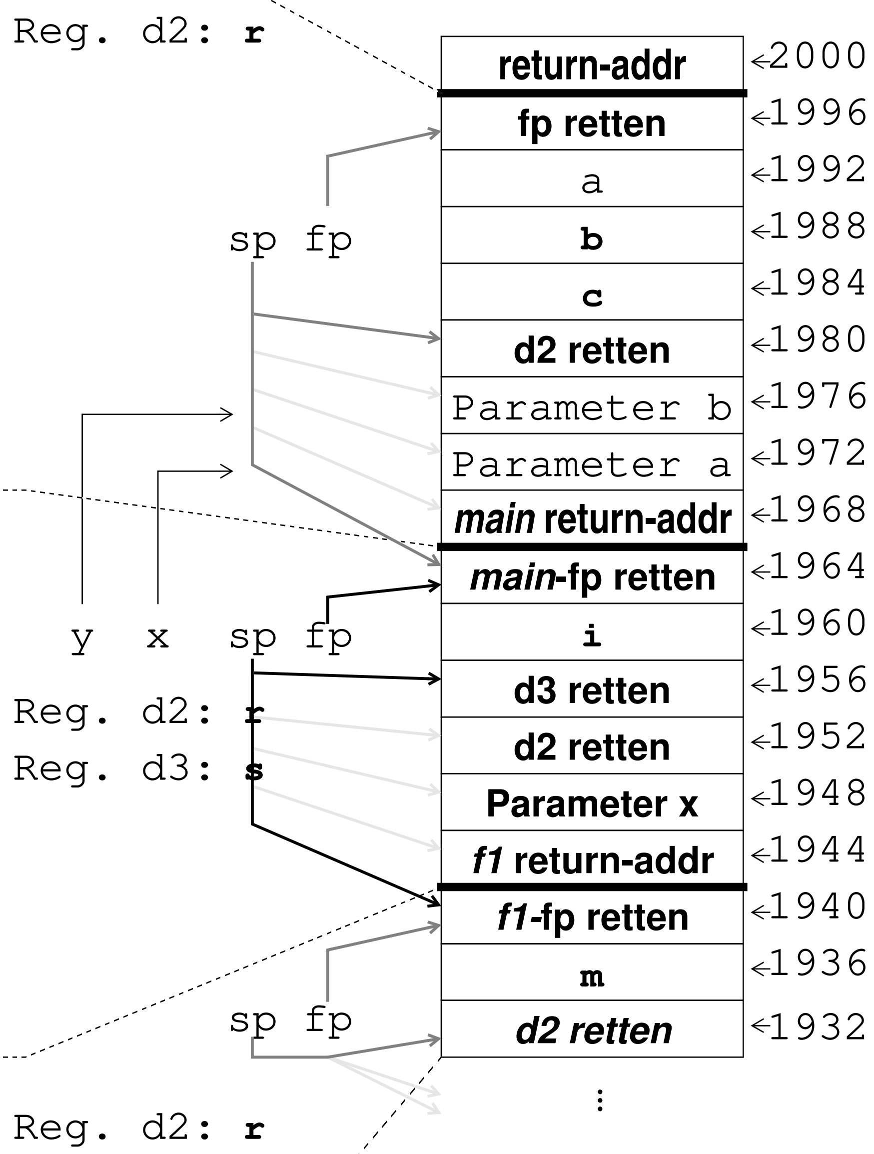


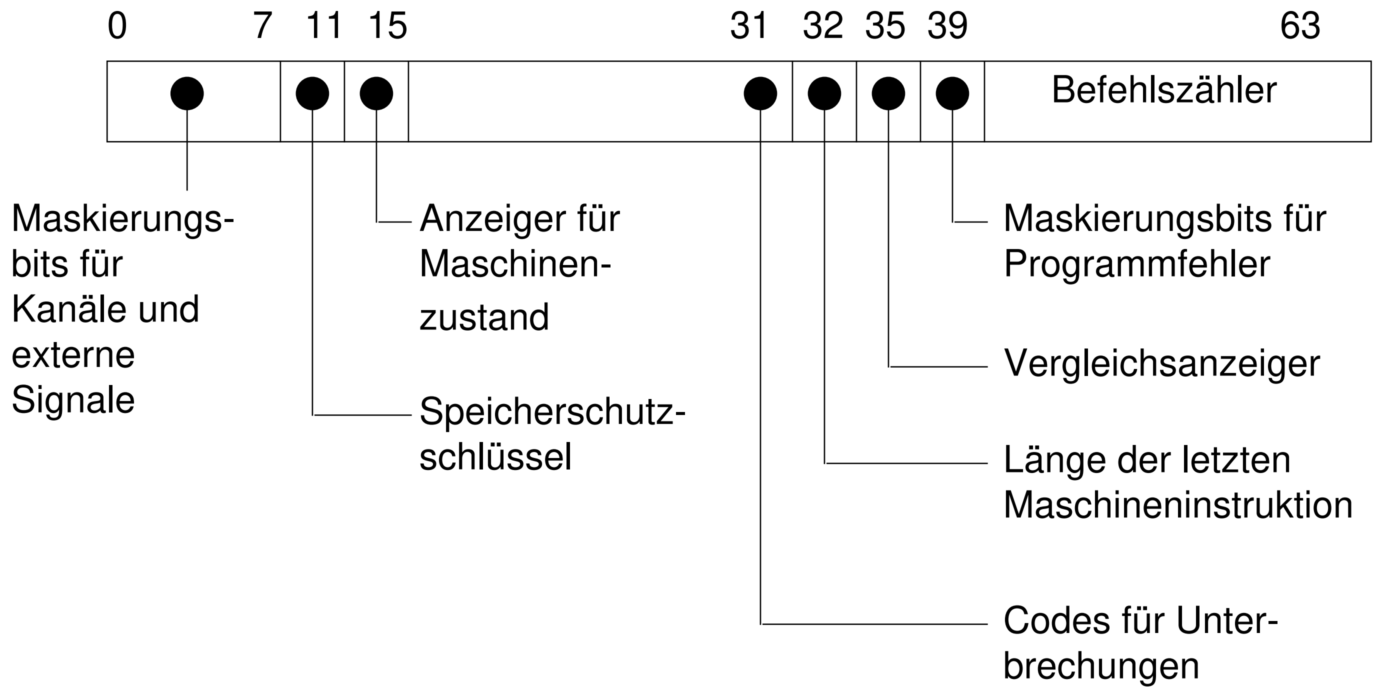
Stapel bei geschachtelten Methodenaufrufen

```
main()
{
        int a, b, c;
        register r;

        a = 10;
        b = 20;
        r = 40;

        f1(a, b);

        return(a);
}
```

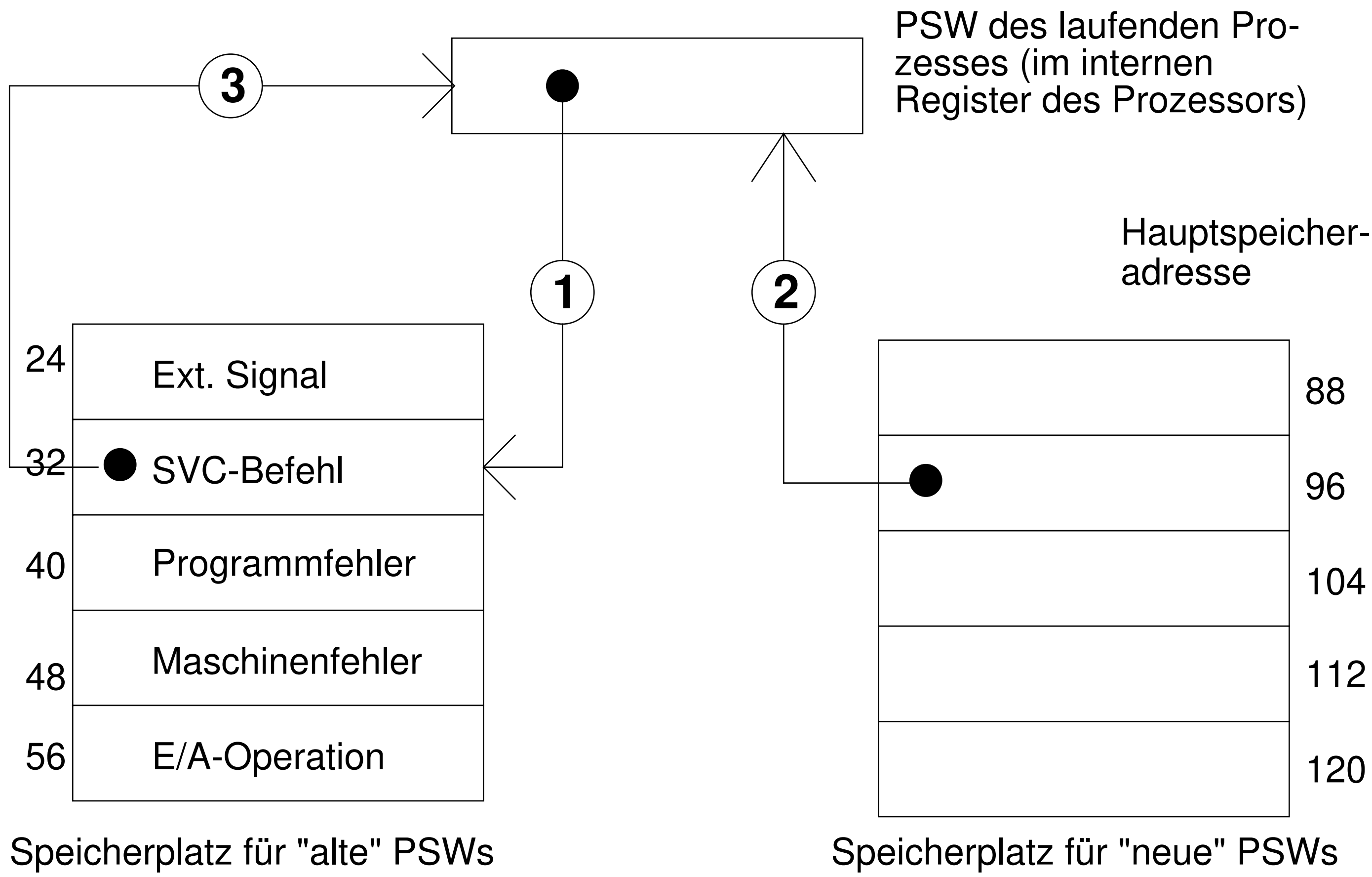
```
f1(x,y)
{
        int i;
        register r;
        register s;

        x++;

        i = f2(x);

        return(i);
}
```

```
f2(z)
{
        int m;
        register r;

        r = z;
        m = 100;

        return(z+1);
}
```





Programmstatuswort des Systems IBM 360



Austausch der PSWs im System IBM 360