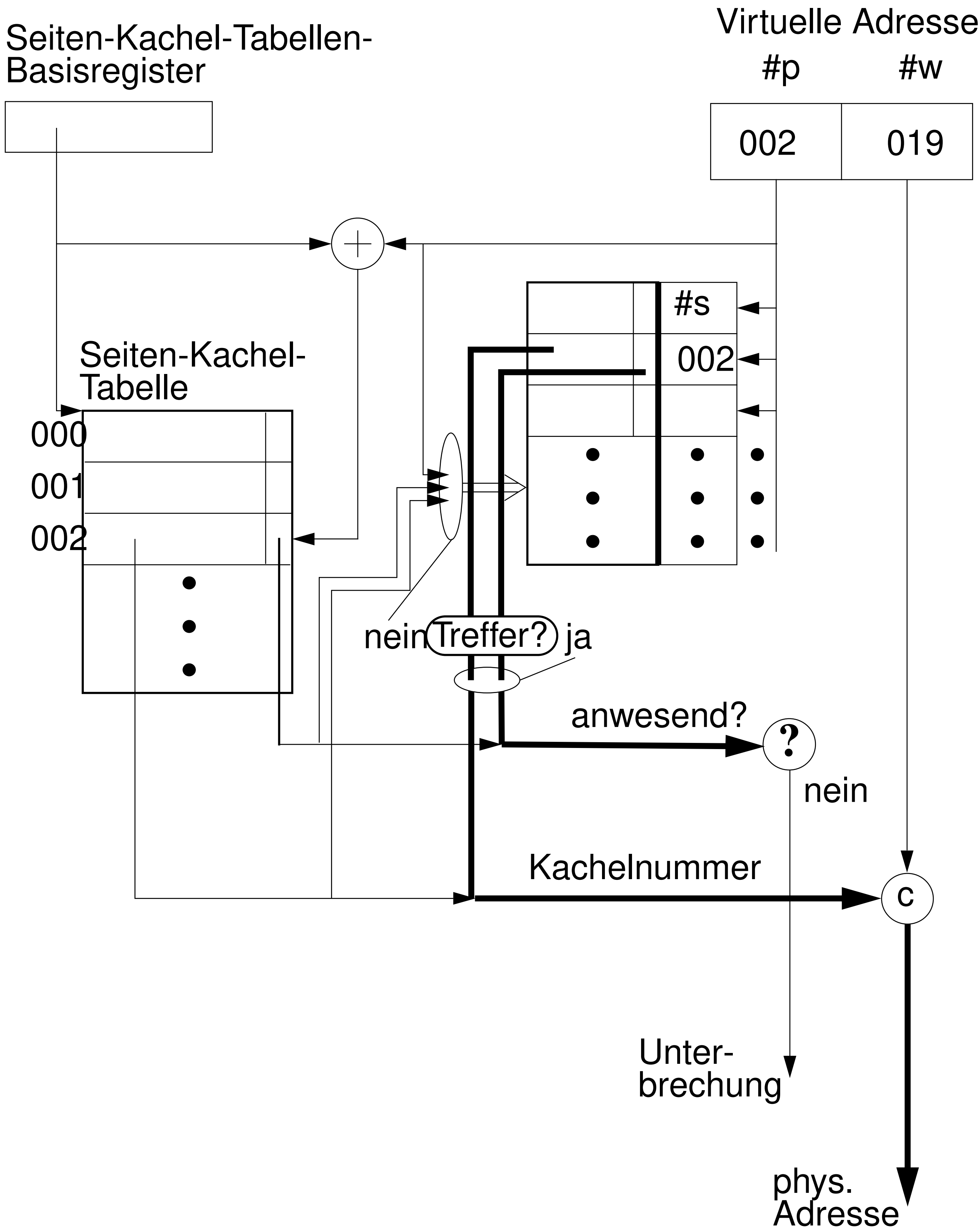
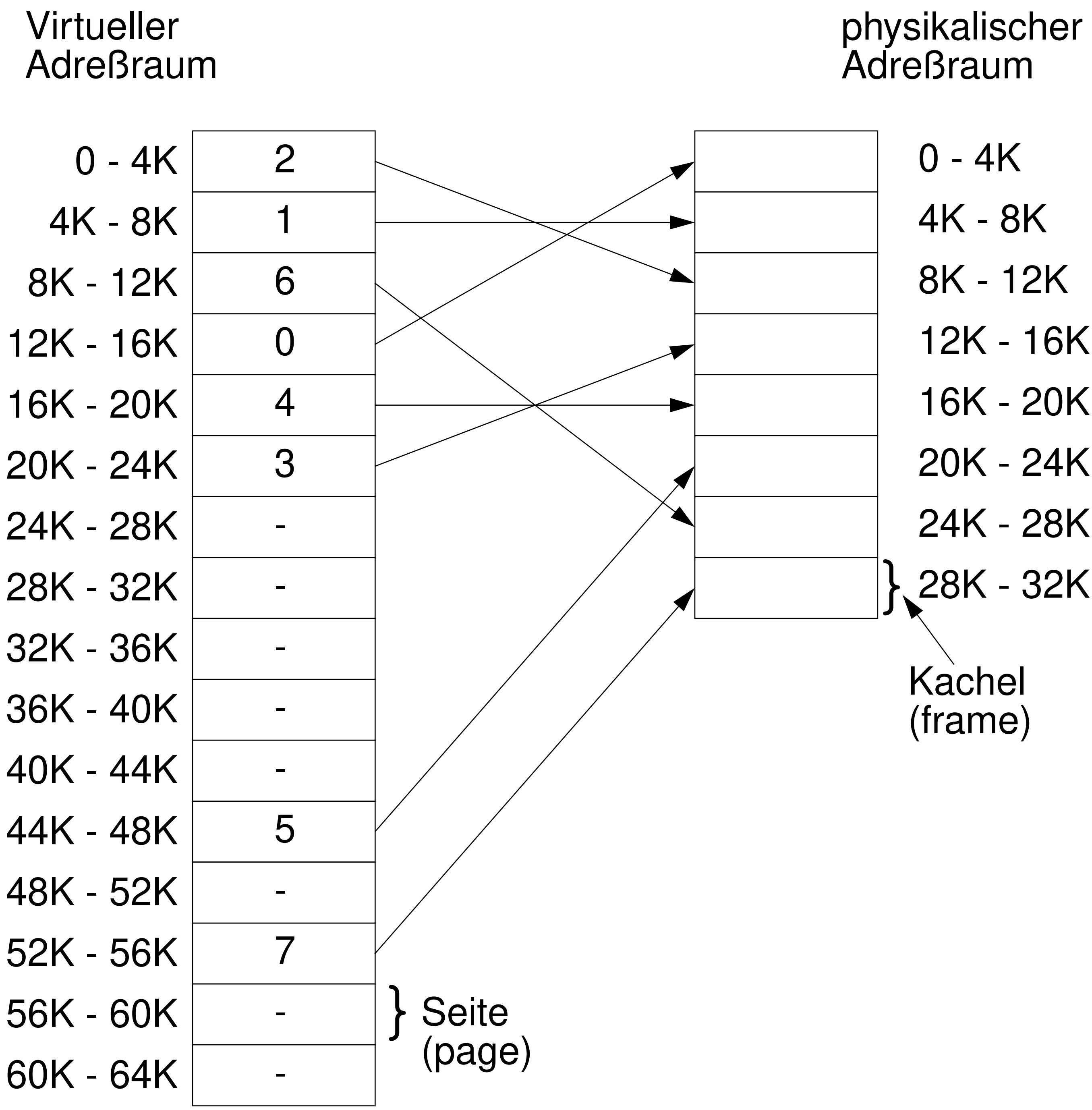
4.4 Seitenadressierung, gestreute Speicherung (paging)

- Mechanismus

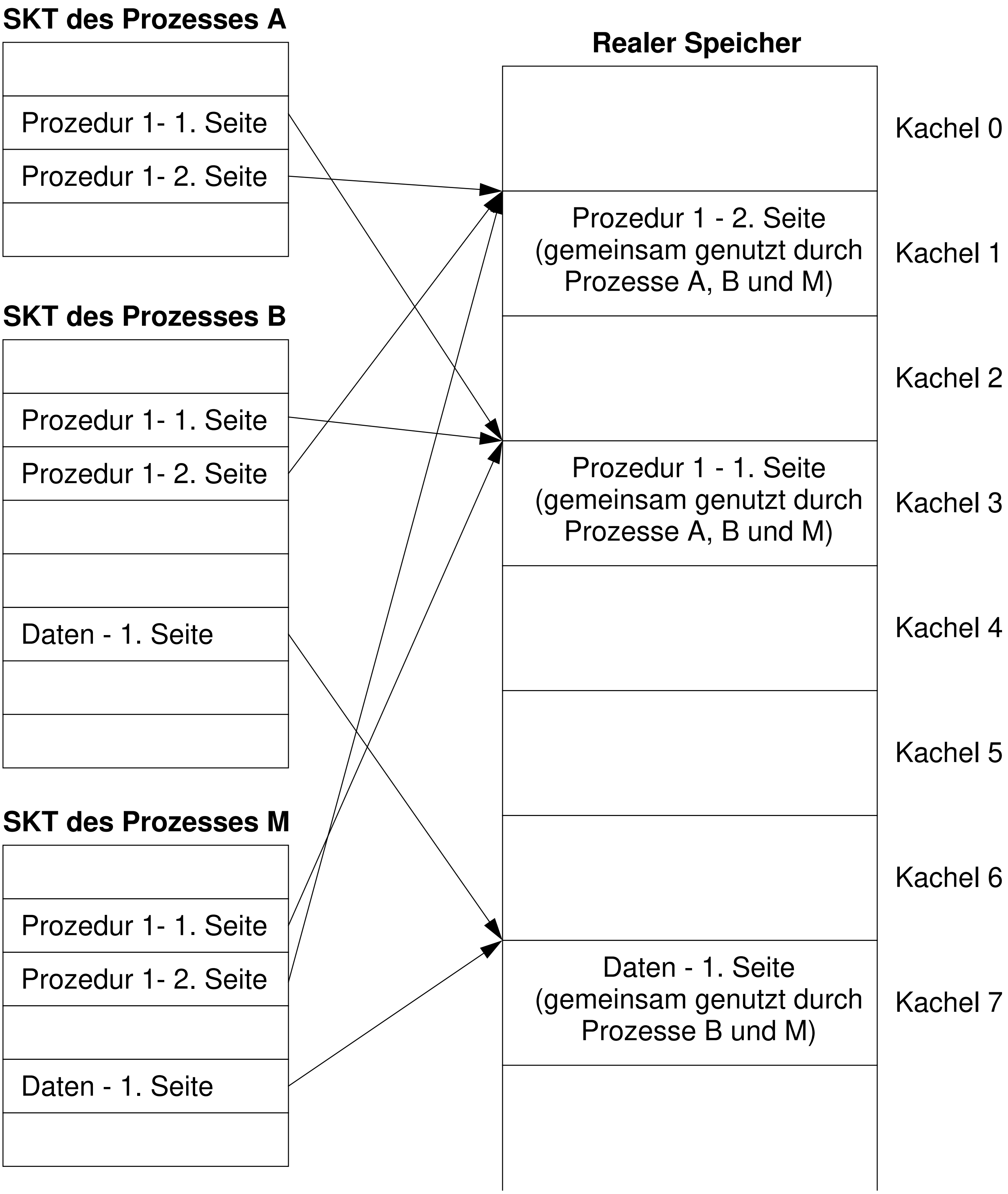


- Nutzung zur gestreuten Speicherung

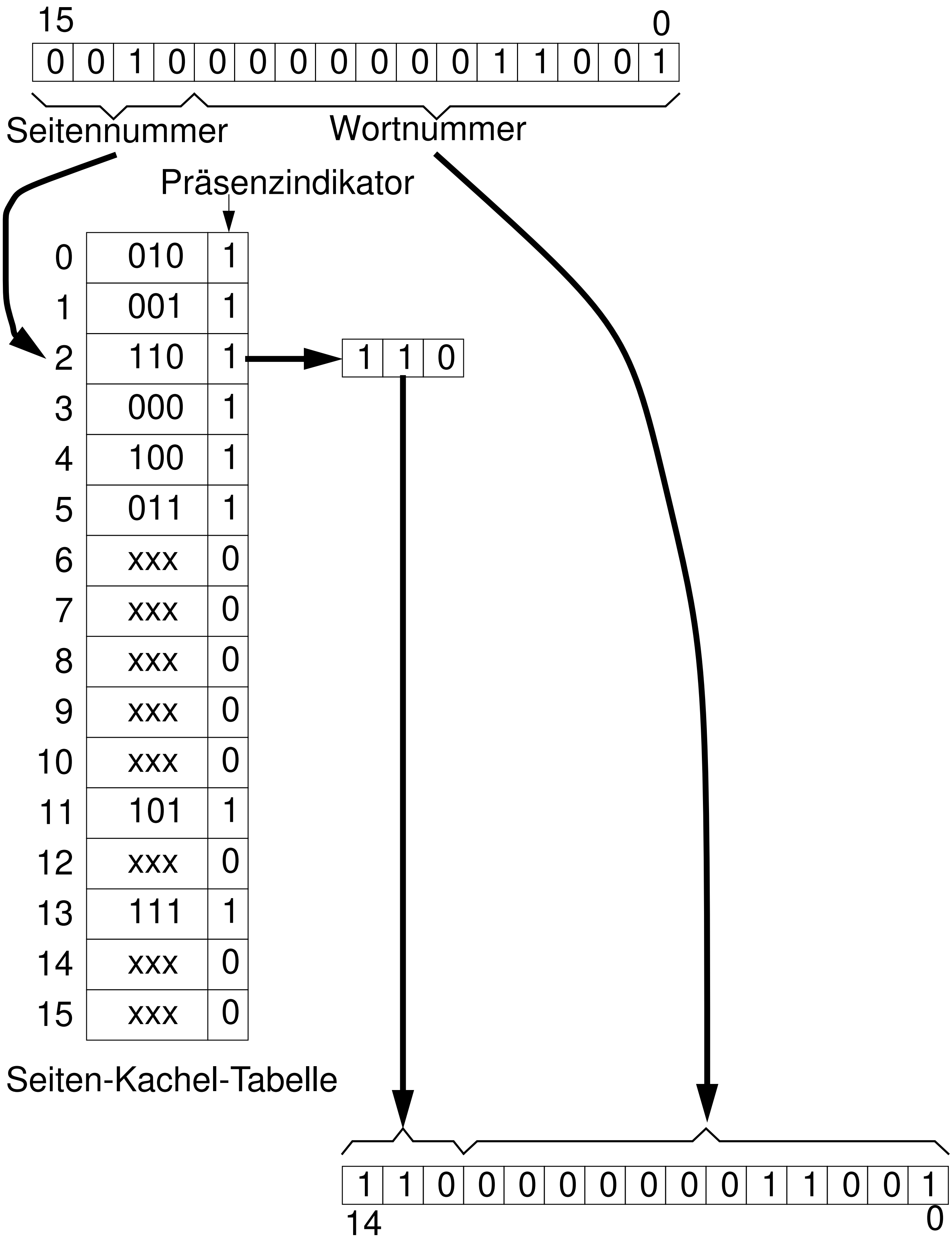


Vereinfacht das Plazierungsproblem bei Speicherbelegung

- Realisierung gemeinsamer Speicherbereiche



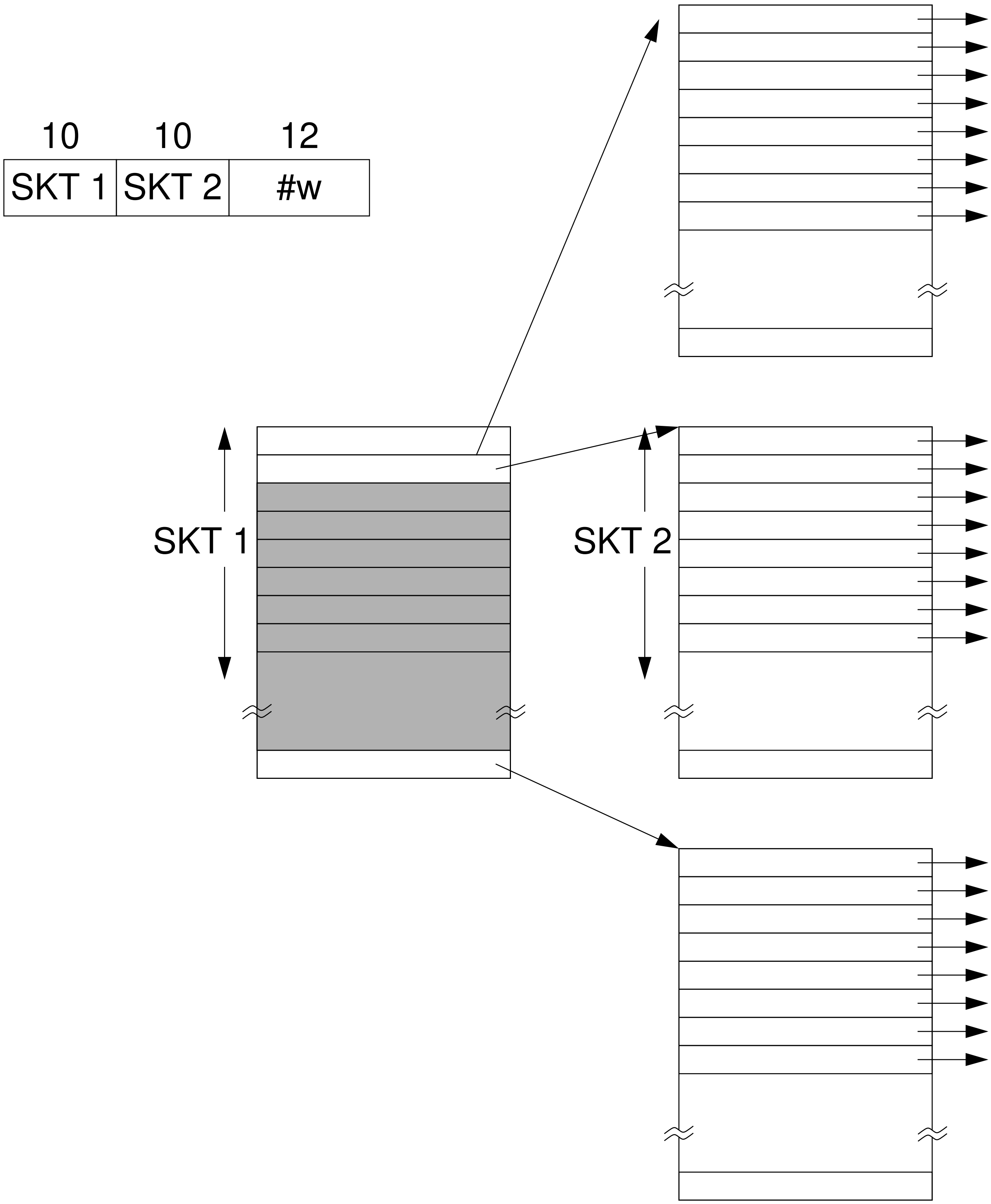
- Unvollständig abgebildete (virtuelle) Adreßräume



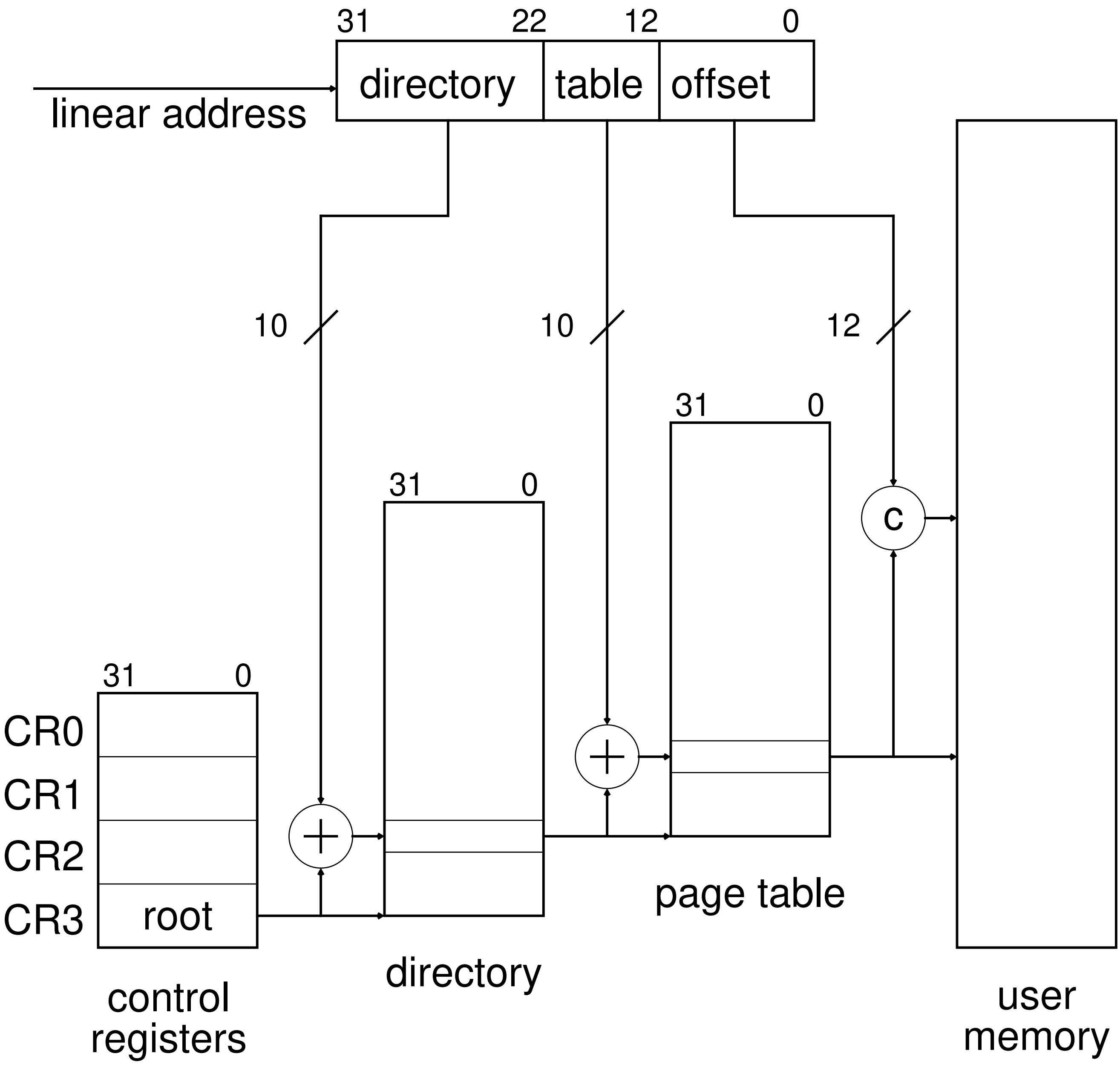
0	010	1
1	001	1
2	110	1
3	000	1
4	100	1
5	011	1
6	xxx	0
7	xxx	0
8	xxx	0
9	xxx	0
10	xxx	0
11	101	1
12	xxx	0
13	111	1
14	xxx	0
15	xxx	0

Seiten-Kachel-Tabelle

- Mehrstufige Seitenadressierung



- Beispiel: Intel 80486

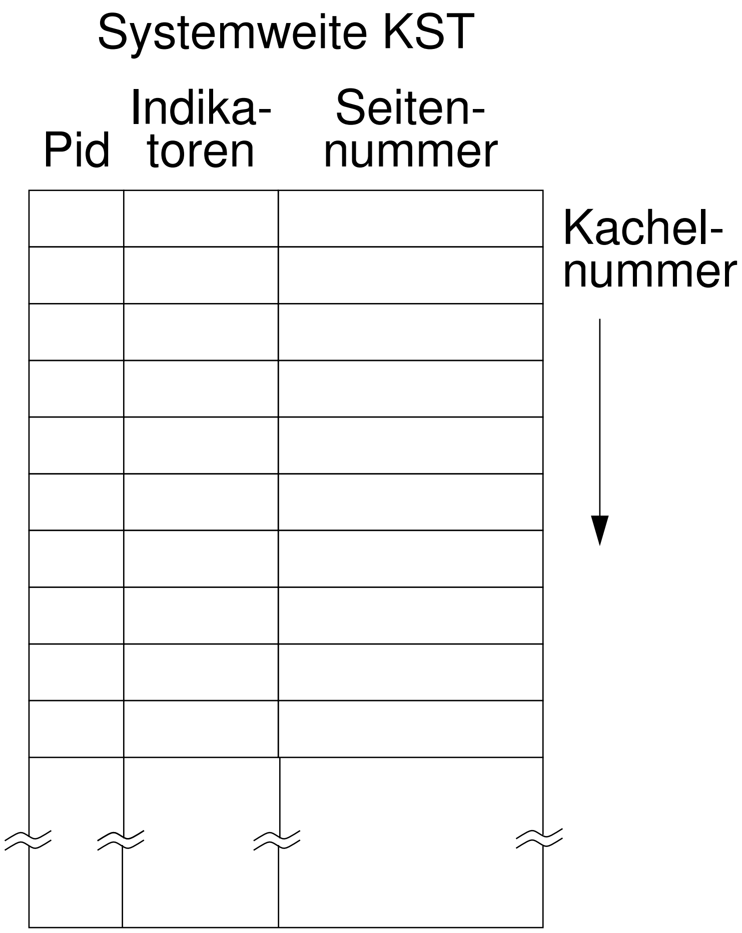


- Invertierte Seiten-Kachel-Tabellen (KST)

  z. B. MIPS 2000

Bits	20	6	6	20	1	1	1	1	8
	Seitennr.	Pid		Kachelnr.	N	D	V	G	

N: 0 Use cache, 1 no cache
D: 0 Entry is clean, 1 entry is dirty
V: 0 Invalid, 1 valid
G: 0 Check pid, 1 do not check pid



Einschlägige Systemaufrufe bei SunOS (Auswahl)

shared memory operations

```
char *shmat(int shmid, char *shmaddr, int shmflg)
int shmdt(char *shmaddr)
```

shared memory control operations

```
int shmctl (int shmid, int cmd, struct shmid_ds *buf)
```

get shared memory segment identifier

```
int shmget(key_t key, int size, int shmflg)
```

- Anlegen eines gemeinsamen Segments

```
// Tilgen eines eventuell unter SHM_KEY
// existierenden gemeinsam nutzbaren Segments
shmid = shmget(SHM_KEY, 1, 0600 | IPC_CREAT);
if ((status = shmctl(shmid, IPC_RMID, 0)) == -1) {
  cout << "***** Fehler beim Tilgen des"
       << " gmeinsamen Sgementes: "
       << sys_errlist[errno] << "\n";
  exit(0);
}

// Deskriptor fuer - eventuell erst neu angelegtes -
// Segment beschaffen
if ((shmid = shmget(SHM_KEY, LAENGE_SHM,
                    0600 | IPC_CREAT)) < 0) {
  cerr << "\n\nError in shmget for shared memory: "
       << sys_errlist[errno]
       << "\n" << flush;
  exit(0);
}

// Segment in den eigenen Adressraum einblenden
if (local_shm = shmat(shmid, (void *)0, 0)) == -1) {
  cerr << "\n\nError: " << sys_errlist[errno]
       << "\n" << flush;
  exit(0);
}
```

- Initialisierung gemeinsamer Informationen

```
pv empty0(1), full0(0), empty1(1), full1(0);

shared_char buffer0[BUFFERLENGTH],
            buffer1[BUFFERLENGTH];
shared_int buffercount(BUFFERCOUNT),
           bufferlength(BUFFERLENGTH);
```

sp1/96/folien/3: 18.12.1996

- Puffer füllen

```
void fuellen()
{ int i, k, l, length;
  float x = 1, y = 3;

  length = buffercount * bufferlength;

  for (i = 1; i <= length; i) {
    empty0.P();
    l = 0;
    do {
      x = 1;
      for (k = 0; k < load; k++)
        x = x / y; // Rechenlast erzeugen
      buffer0[l++] = i++ & 255;
    } while (l < bufferlength);
    full0.V();

    empty1.P();
    l = 0;
    do {
      x = 1;
      for (k = 0; k < load; k++)
        x = x / y; // Rechenlast erzeugen
      buffer1[l++] = i++ & 255;
    } while (l < bufferlength);
    full1.V();
  }
}
```

- ❑ Puffer in Datei schreiben

```
void sichern()
{ int i, file, result;
  float x = 0, y = 3;

  file = open("hilfsdatei",
              O_WRONLY | O_CREAT | O_TRUNC | O_SYNC);

  for (i = 0; i < buffercount; i +=2) {
    full0.P();
    result = write(file, buffer0, bufferlength);
    empty0.V();

    full1.P();
    result = write(file, buffer1, bufferlength);
    empty1.V();
  }
}
```