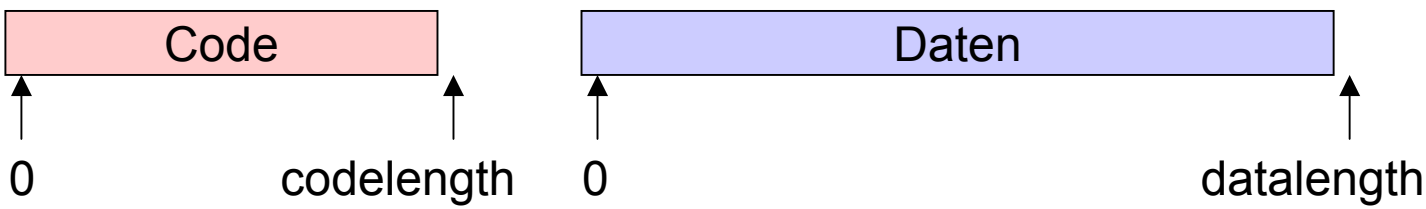
5.4 Segmentierung

Adressraum besteht aus mehreren **Segmenten** *(segments)*, die unabhängig voneinander manipulierbar sind. Segmentierungsstruktur ist festgelegt durch die *Hardware* – den Adressumsetzer.

Einfachstes Beispiel: 1 Code-Segment + 1 Datensegment



Vorteile:

❶ *Unterschiedliche Zugriffsrechte* – durch MMU überwacht – erhöhen die Programmsicherheit; z.B.

Code-Segment: schreibgeschützt *(read/execute-only)*
Daten-Segment: Lese- und Schreibzugriff *(read/write).*

❷ Adressräume verschiedener Prozesse können in sinnvoller Weise *überlappen*; z.B. gemeinsames Code-Segment *(code sharing).*

❸ Segmente können *unabhängig* voneinander umgelagert werden; z.B. kann beim Auslagern eines Prozesses ein schreibgeschütztes Code-Segment ignoriert werden.

Standard-Segmentierungen

2 Segmente:

Code – feste Länge, normalerweise schreibgeschützt, von mehreren Prozessen gemeinsam benutzbar *(shared code)*

Daten – z.B. automatisch expandiert bei Kellerüberlauf

(Beispiel: DEC System 10, *1970*)

3 Segmente:

Code – s.o.

Daten statisch und Halde – Länge explizit änderbar

Daten dynamisch – Keller s.o.

(Beispiel: Original-Unix, DEC PDP-11, *1971*)

3 Segmente:

Code – s.o.

Daten gemeinsam benutzt *(shared data)*

Daten privat

4 Segmente:

Code – Quellcode

Code – Standard-Bibliothek eine Programmiersprache
(shared libraries, dynamic link libraries (DLL))

Daten statisch

Daten dynamisch

5.4.1 Hardware-Voraussetzungen (MMU)

Elementarer Fall – Code- und Daten-Segment:

4 Register:

code base

code length

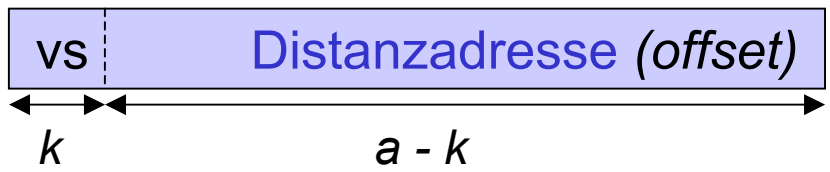
data base

data length

Identifizierung des jeweiligen Registerpaars durch das Steuerwerk: *Befehlsadressen* vs. *Datenadressen*

Allgemeiner Fall – *n* gleichberechtigte Segmente, $n = 2^k$, Adressbreite *a*

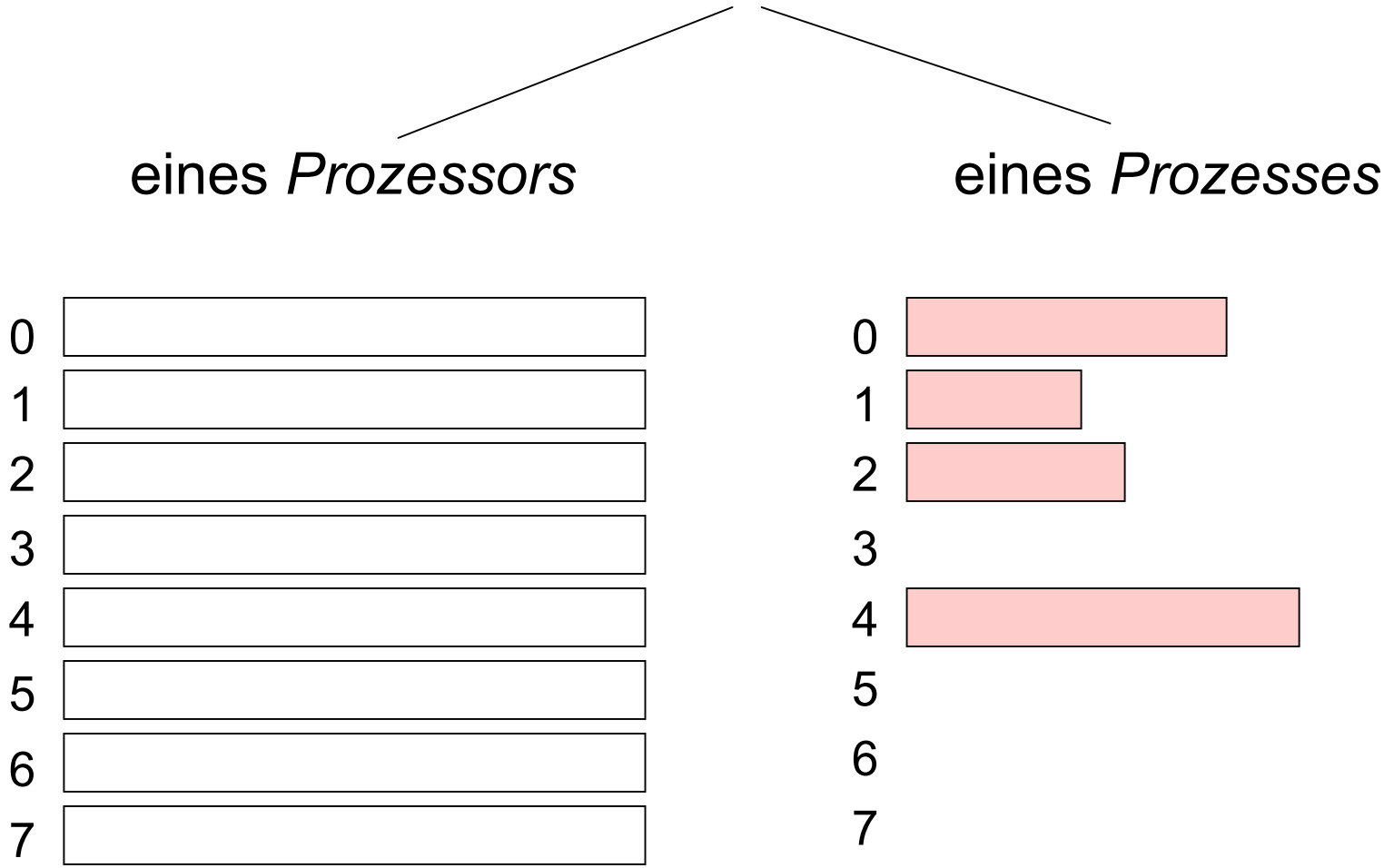
virtuelle Adresse:



vs = *(virtuelle)* Segmentnummer

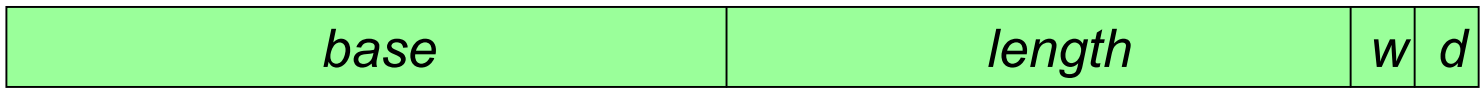
➔ 2^k Segmente mit jeweils maximaler Länge 2^{a-k}

Z.B. für *k=3:*


Adressraum
eines *Prozessors*
eines *Prozesses*
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7

MMU: je Segment 1 **Deskriptor-Register** für

Segmentdeskriptor *(segment descriptor),* z.B. so:



w = writable bit:

legt fest, ob in das Segment geschrieben werden darf.

d = dirty bit:

wird auf 0 gesetzt, wenn

der Deskriptor in das Register geladen wird;

wird auf 1 gesetzt beim ersten Schreibzugriff.

➔ MMU sieht so aus:

Achtung: evtl. verschiedene Registersätze für verschiedene Prozessor-Modi (5.1.1 ←)

Adressumsetzung:

(*write* = Schreibversuch)

if *desc[vs].base == 0* ***then*** *address fault* ***end****;*
(non-existent segment)

if *desc[vs].length <= offset* ***then*** *address fault* ***end****;*
(off limits)

if *not desc[vs].w* ***and*** *write* ***then*** *access fault* ***end****;*

location = desc[vs].base + offset;

desc[vs].d = desc[vs].d ***or*** *write .*

(vereinfachende Vorausetzung: *base/length* in Bytes gemessen)

5.4.2 Segmentverwaltung

Segmentdeskriptoren, die für die *Verwaltung* der Segmente benötigt werden, enthalten mehr Information als die Deskriptor-Register:

base	*length*	*d*	*backupBase*	*refcnt*	*c*

w-Bit fehlt ! (s.u.)

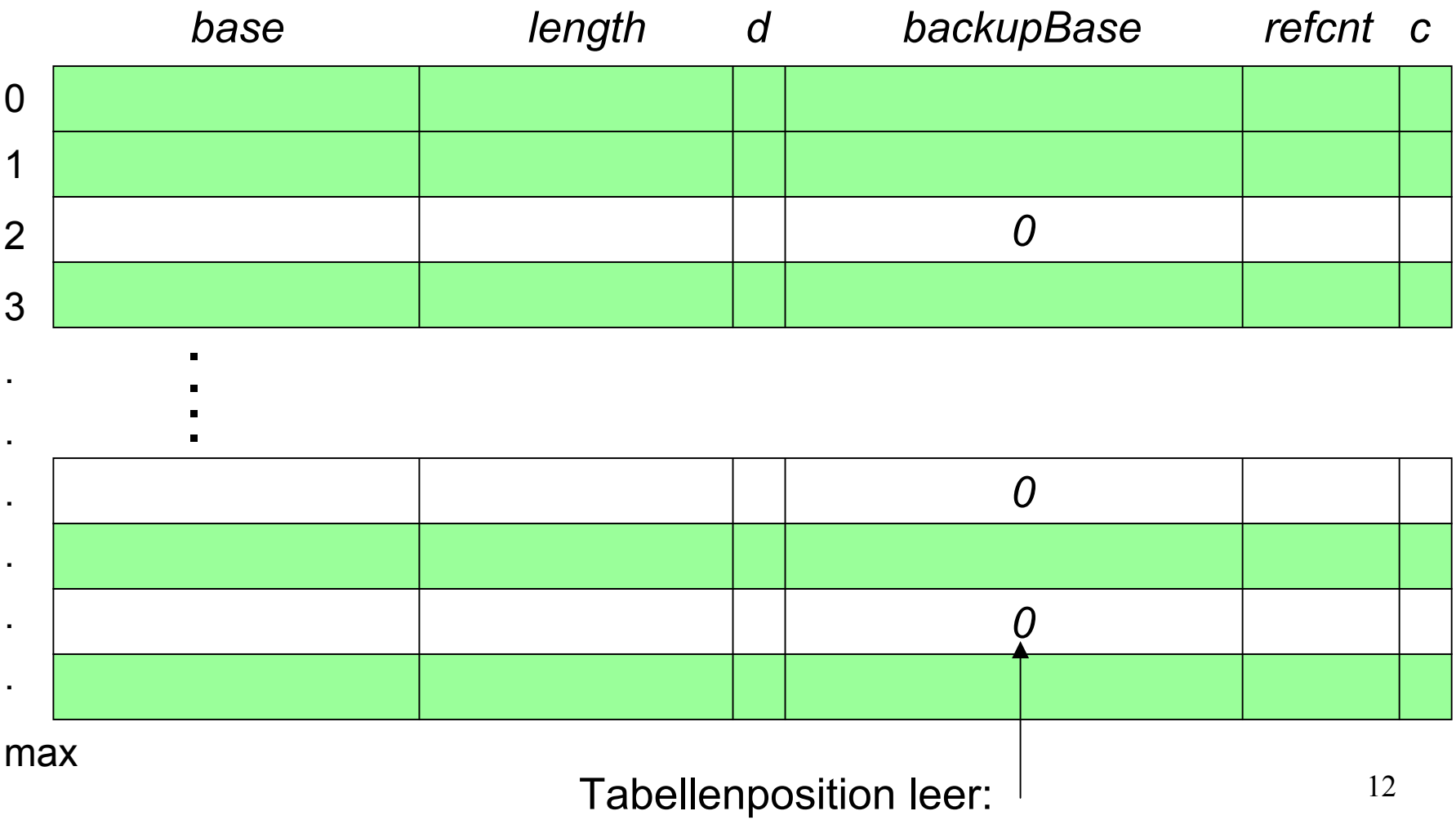
backupBase = Auslagerungsadresse (5.3.2←)

refcnt = Verweiszähler = Anzahl der Prozesse, die das Segment gemeinsam benutzen

c = *„coming“-Bit* = Einlagerungsvorgang läuft

Segmenttabelle *(segment table)* (Größe *max)*

enthält die Deskriptoren aller Segmente im System:



Segmentliste *(segment map)* eines Prozesses

im Prozessdeskriptor

enthält für jedes Segment des Prozesses:

- *(reale) Segmentnummer rs*
  (= Index in Segmenttabelle)
- *w-Bit*

Segmentliste *(segment map)* eines Prozesses

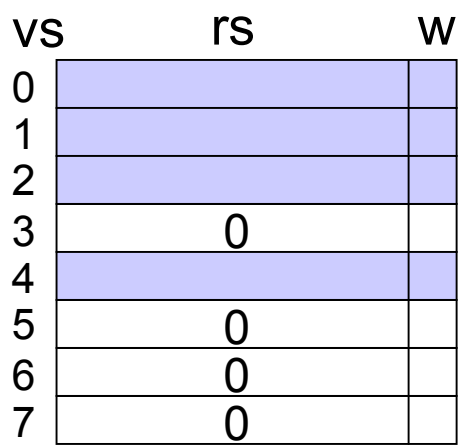
im Prozessdeskriptor

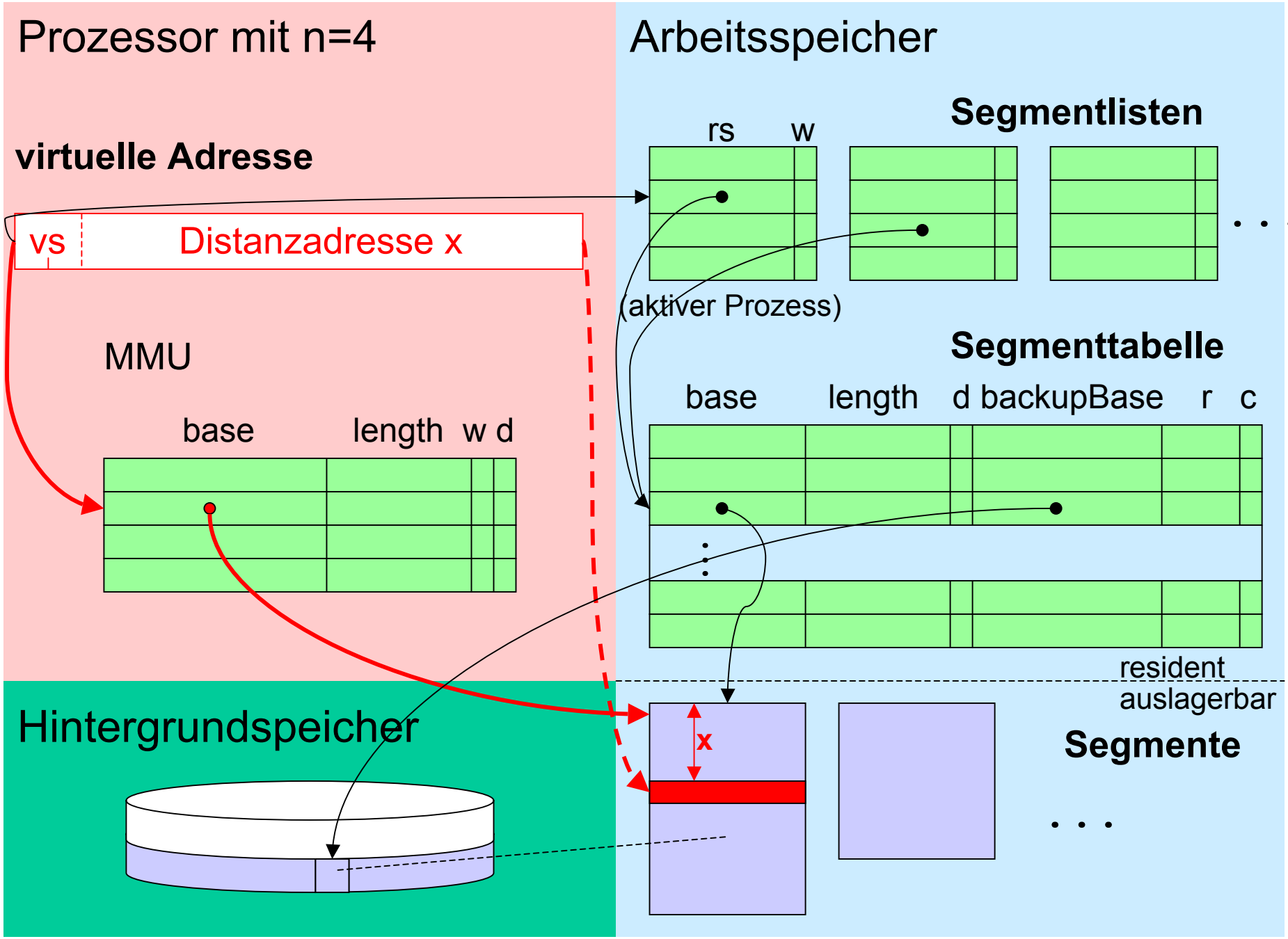
enthält für jedes Segment des Prozesses:

- *(reale) Segmentnummer rs*
  (= Index in Segmenttabelle)
- *w-Bit*

als z.B. bei einer MMU mit *n=8* Segmenten (5.4.1 ←)

(*rs = 0* : Segment existiert nicht)




Prozessor mit n=4
Arbeitsspeicher
virtuelle Adresse
vs
Distanzadresse x
MMU
base
length
w
d
Segmentlisten
rs
w
(aktiver Prozess)
Segmenttabelle
base
length
d
backupBase
r
c
resident
auslagerbar
Segmente
x
Hintergrundspeicher

Adressraumumschaltung bei Prozesswechsel (5.1.2🡐)

① *dirty bits* aus den Deskriptor-Registern der MMU
in die Segmenttabelle *retten* (***or*** !)

② *base/length* des neuen aktiven Prozesses aus der
Segmenttabelle in die Deskriptor-Register laden
– nach Maßgabe der Segmentliste

③ *w-Bits* in den Deskriptor-Registern setzen
– nach Maßgabe der Segmentliste