Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

VI. Segmentadressierung

Wolfgang Schröder-Preikschat / Volkmar Sieh

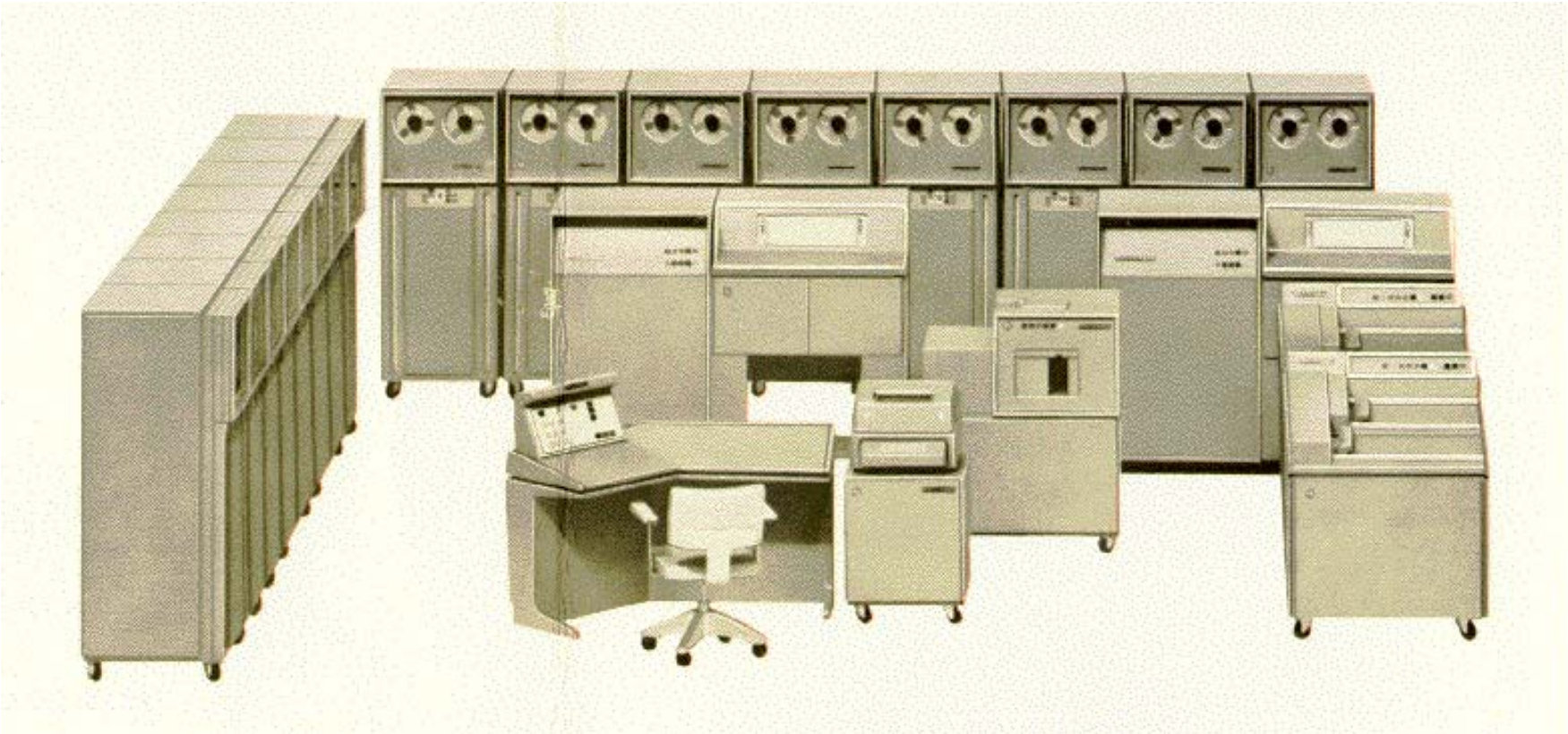
SS 2024

Gliederung

Höhere Maschinensprachen

- Hardware, Betriebssystem und primäre Sprachen eines Rechensystems sollten als Einheit konzipiert werden [4]
  - Systemplementierer/innen produzieren Systeme, die von vielen anderen Benutzer/innen aller Klassen verwendet werden
  - daher ist es vorrangig wichtig, die Maschinenhardware für die Klasse der Systemimplementierer/innen zu optimieren
  - die Hardware in einem entsprechenden Gesamtsystementwurf hätte die folgenden wesentlichen Eigenschaften:
    - von Benutzer/innen (anderer Klassen) verwendete Informationsstrukturen samt Operatoren sind Grundkonstrukte der Maschinensprache
    - Steuerfluss und Datenzugriffsverfahren in der Maschine gleichen denen ihrer Grundsprachen und ihrem Betriebssystem
    - das System führt implizite Funktionen für Benutzer/innen nebeneffektfrei aus und erlaubt aber auch, diese Funktionen außer Kraft zu setzen
- erster Rechner dieser Art ist der B 5000 [1, 2, 3], der eher als **System** funktioniert als nur eine fortgeschrittene Reihe von Hardware

B 5000, 1961



Programmstrukturen und Segmente

- dem Prozessor, genauer: der CPU sind einzelne Strukturelemente der auszuführenden Programme „bewusst“
  - Text- und Datenbestände sind von einem bestimmten Typ
    - Funktion, Prozedur, ...
    - Zeichen, Zahlen, Zeiger, Verbünde, Felder, ...
  - Exemplare davon bilden logisch wie auch physisch **Segmente**
- ein **Segmentdeskriptor** erfasst sodann ein einzelnes Strukturelement oder eine Gruppe solcher Elemente (desselben Typs)
  - also nicht bloß je ein Text-, Daten-, BSS- und Stapelsegment pro Prozess
  - sondern viele Segmente pro Text, Daten, BSS — auch pro Stapel
    - beispielsweise jeden Aktivierungsblock als eigenes Segment repräsentieren

  ↪ des Programm definiert die Segmentanzahl, nicht das Betriebssystem
- Hardware und Betriebssystem, die Segmentadressierung in dieser Art und Weise ermöglichen, sind damit hochsprachenorientiert ausgelegt
  - aber nur „als Einheit“ definieren sie die geeignete **abstrakte Maschine**

Gliederung

Nichtlinearer (zweidimensionaler) Adressraum

Segmentierung steht für eine Strukturierung des Adressraums in Einheiten von möglicherweise verschiedener Größe

- diese Einheiten werden als Segment bezeichnet
  - sie setzen sich zusammen aus gleichgroßen Elementen und
  - bilden eine lineare Folge von „Granulaten“ (Bytes, Seiten):
    - Byte $\mapsto$ unbedingt zusammenhängend auch im realen Adressraum
    - Seite $\mapsto$ bedingt: seitennummerierte Segmentierung (*paged segmentation*)
- die vom Prozess generierte Adresse *la* bildet ein Paar (S, A):
  - S ist Segmentname (auch Segmentnummer) $\rightsquigarrow$ 1. Dimension
    - Wertebereich für $S = [0, 2^M - 1]$; bei IA-32: $M = 13$
  - A ist Adresse, auch Versatz, innerhalb des Segments S $\rightsquigarrow$ 2. Dimension
    - Wertebereich für $A = [0, 2^N - 1]$
- tabellengesteuerte Abbildung von *la* mit S als Segmentindex
  - zur Auswahl des für S gültigen Segmentdeskriptors in der Segmenttabelle

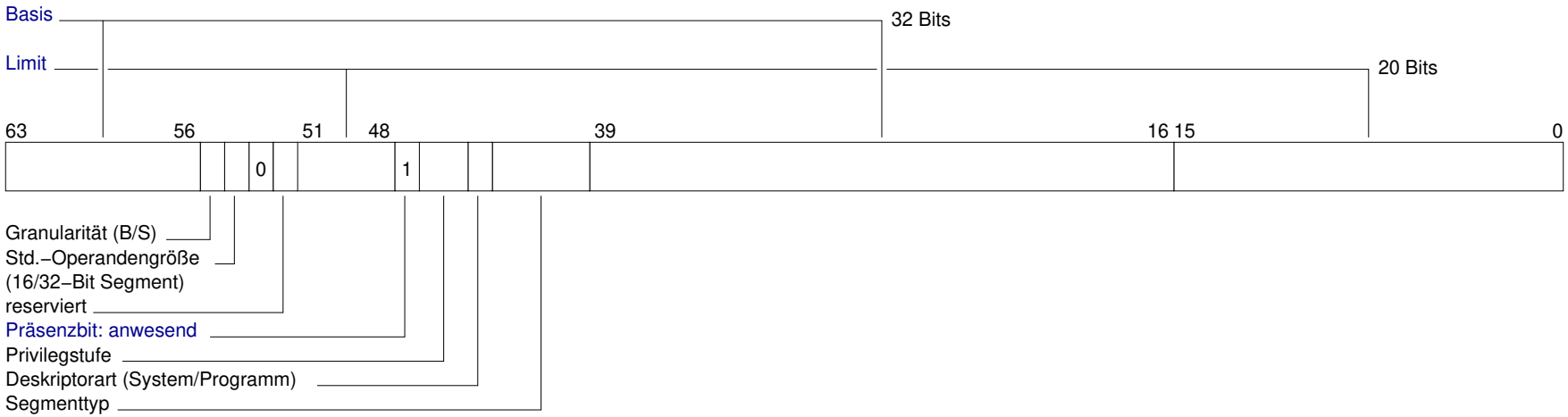
Segmentdeskriptor I

Segmentname/-index identifizieren die die Adressabbildung steuernde und von der Hardware (MMU) vorgegebene Datenstruktur

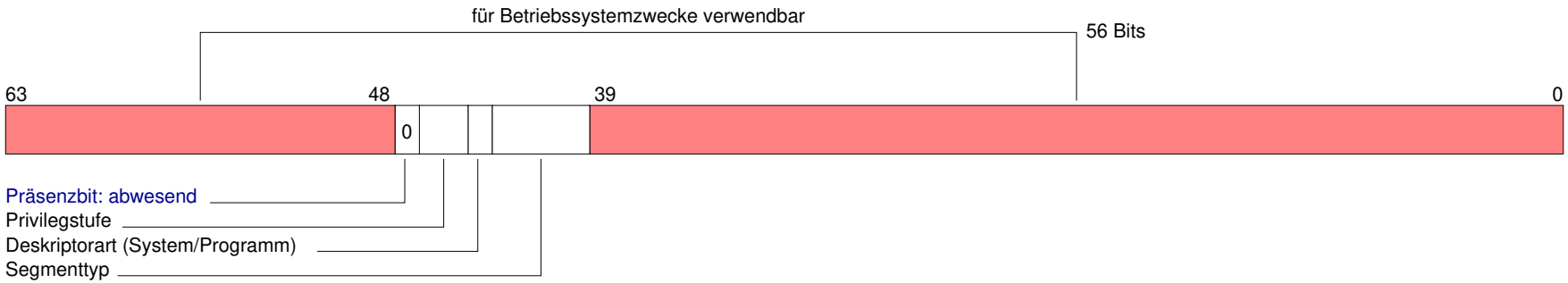
- typischerweise umfassen die darin gebündelten Informationen:
  - Basis
    - Segmentanfangsadresse im Arbeitsspeicher
    - Ausrichtung (*alignment*) entsprechend der Granulatgröße
  - Limit
    - Segmentlänge als Anzahl der Granulate: für gewöhnlich Bytes
    - Zahl der aufeinanderfolgenden Granulatadressen: f. gew. Bytes
  - Attribute
    - Typ (Text, Daten, Stapel)
    - Zugriffsrechte (lesen, schreiben, ausführen)
    - Expansionsrichtung (auf-/abwärts)
    - Präsenzbit
- je nach Hardware und Adressraummodell gibt es weitere Attribute
  - Privilegstufe, Klasse (*interrupt*, *trap*, *task*), Granulatgröße, ...
  - Seiten-Kachel-Tabelle (seitennummerierte Segmentierung)

Segmentdeskriptor II

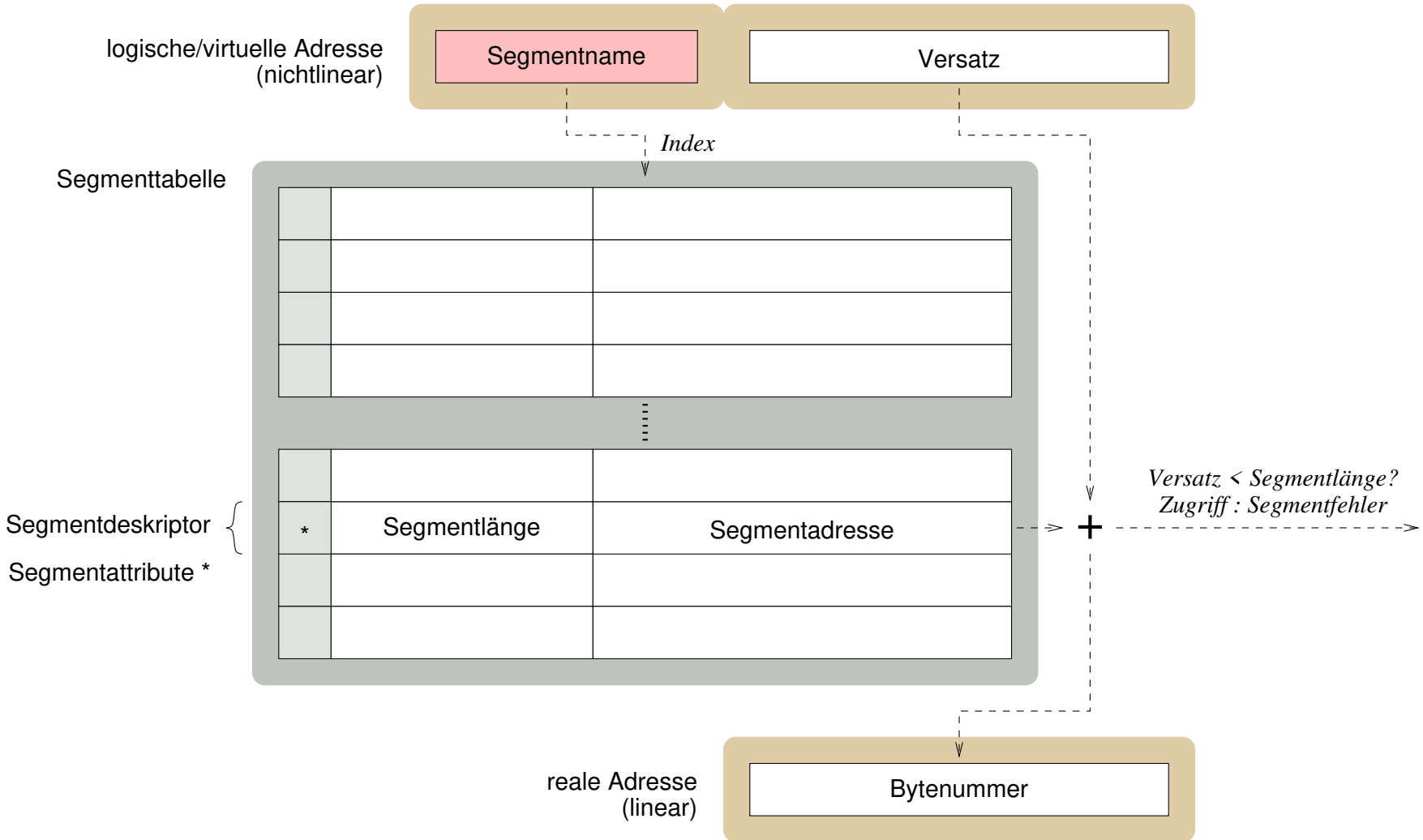
- ein als „anwesend“ markiertes Segment



- ein als „abwesend“ markiertes Segment



Segmentbasierte Adressierung

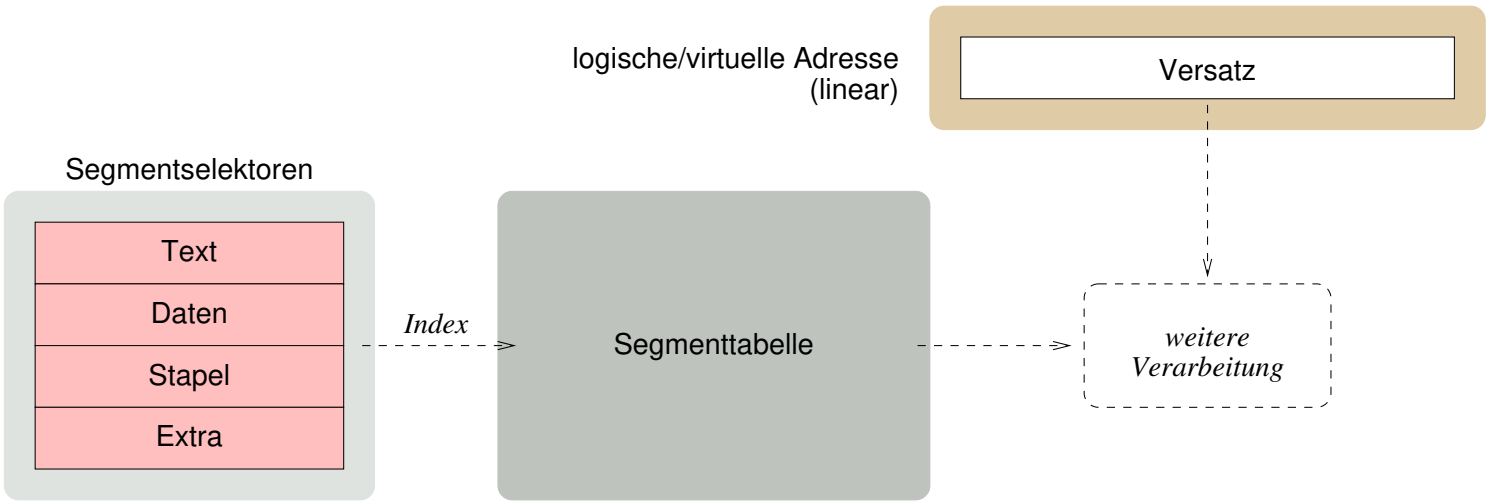



```
1  ra = la < ST[sn].sd_limit ? ST[sn].sd_base + la : segmentation violation ⇝ trap
```

Segmentbasierte Adressierung: implizit

Segmentregister bzw. Segmentselektor (*segment selector*)

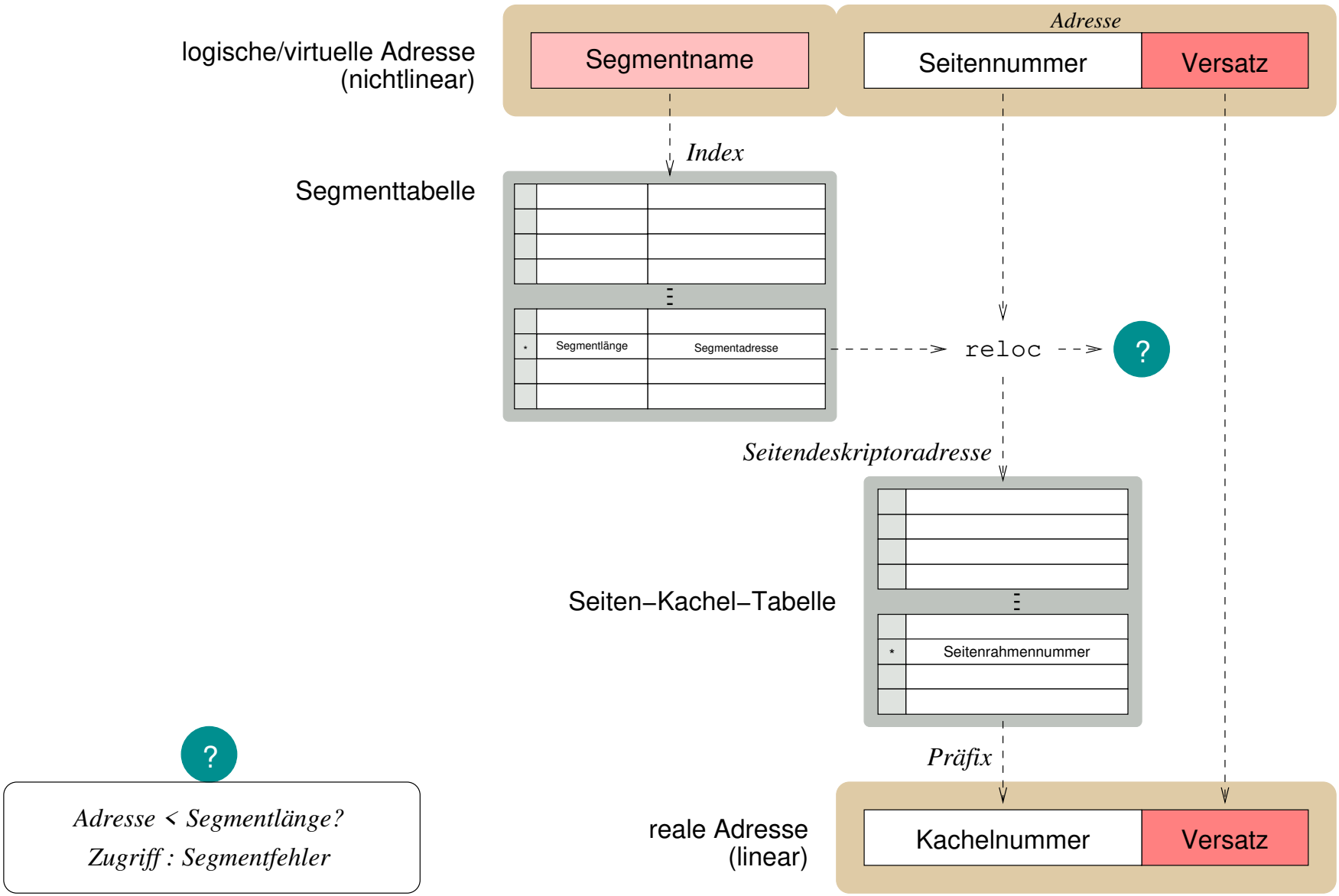
- je nach Zugriffsart selektiert die MMU implizit das passende Segment



- Befehlsabruf (*instruction fetch*) aus Textsegment
  - Operantionskode ↦ Segmentname „Text“
- Operandenabruf (*operand fetch*) aus Text-, Daten-, Stapelsegment
  - Direktwerte ↦ Segmentname „Text“
  - globale/lokale Daten ↦ Segmentname „Daten“ ≡ „Stapel“
- Programme können weiterhin eindimensionale Adressen verwenden

Gliederung

Segmentierung mit Seitenadressierung: Prinzip

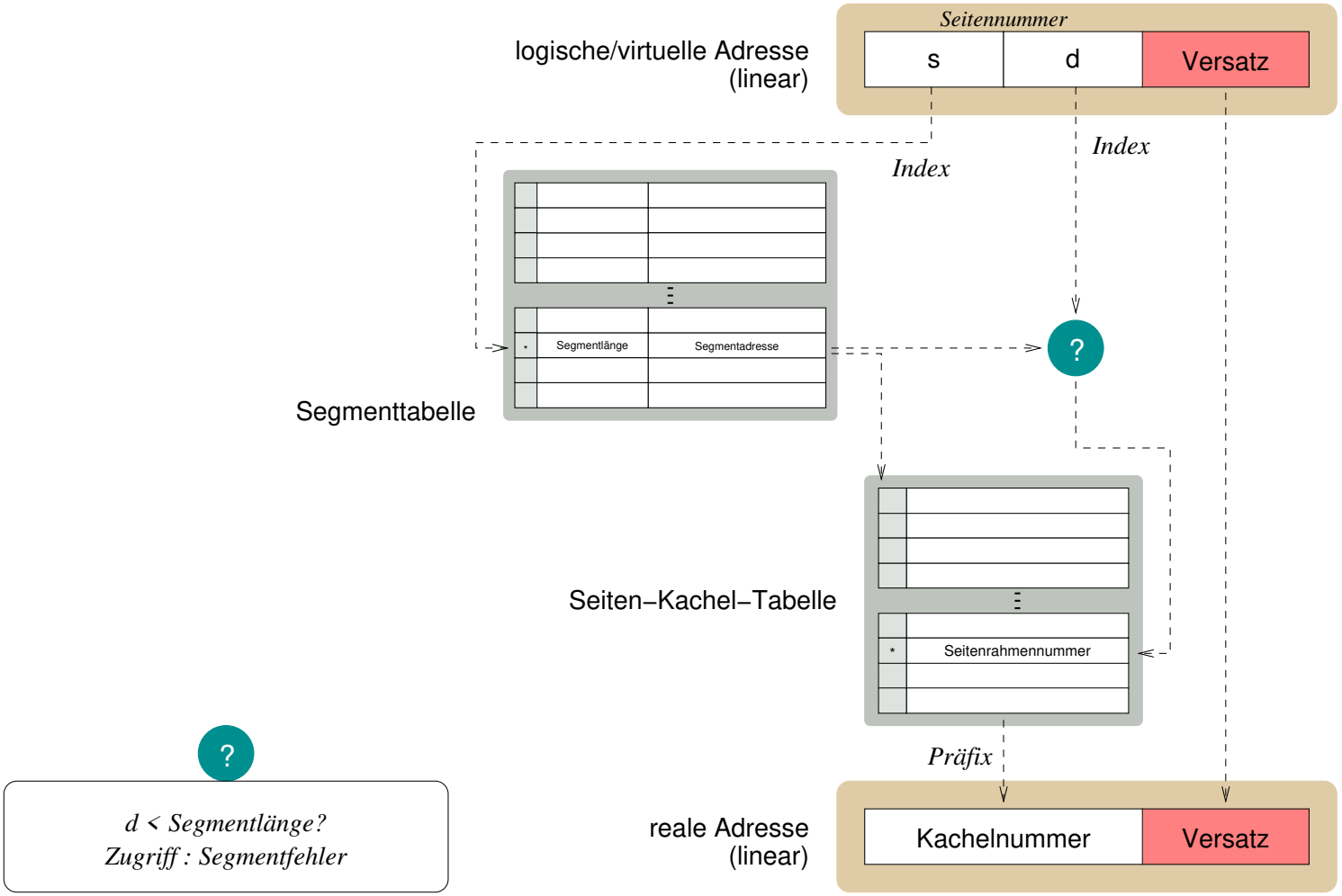


Segmentierung mit Seitenadressierung: Varianten

- in Abhängigkeit davon, wie sich die Seitennummer als Bestandteil der logischen/virtuellen Adresse darstellt:
  - **Paar**
    - $a = (p, o)$, mit $p = (s, d)$ und $s, d, o \in \mathbb{N}$
    - eindimensionaler Adressraum
    - segmentierte Seitenadressierung (*segmented paging*)
  - **sonst**
    - $a = s \wedge (p, o)$, mit $s, p, o \in \mathbb{N}$
    - zweidimensionaler Adressraum
    - seitennummerierte Segmentierung (*paged segmentation*)
      - die Seitentabelle ist ein dynamisches („offenes“) Feld: Prozessgröße
      - die Seitentabelle ist ein statisches Feld: Systemgröße
- zur Dimensionierung/Lokalisierung einer verschiebbaren (*relocatable*) Segmenttabelle dient für gewöhnlich ein Segmentregister
  - **Basis**
    - Anfangsadresse der Tabelle im Hauptspeicher
  - **Limit**
    - Größe der Tabelle (# von Bytes oder Segmentdeskriptoren)

Segmentierte Seitenadressierung

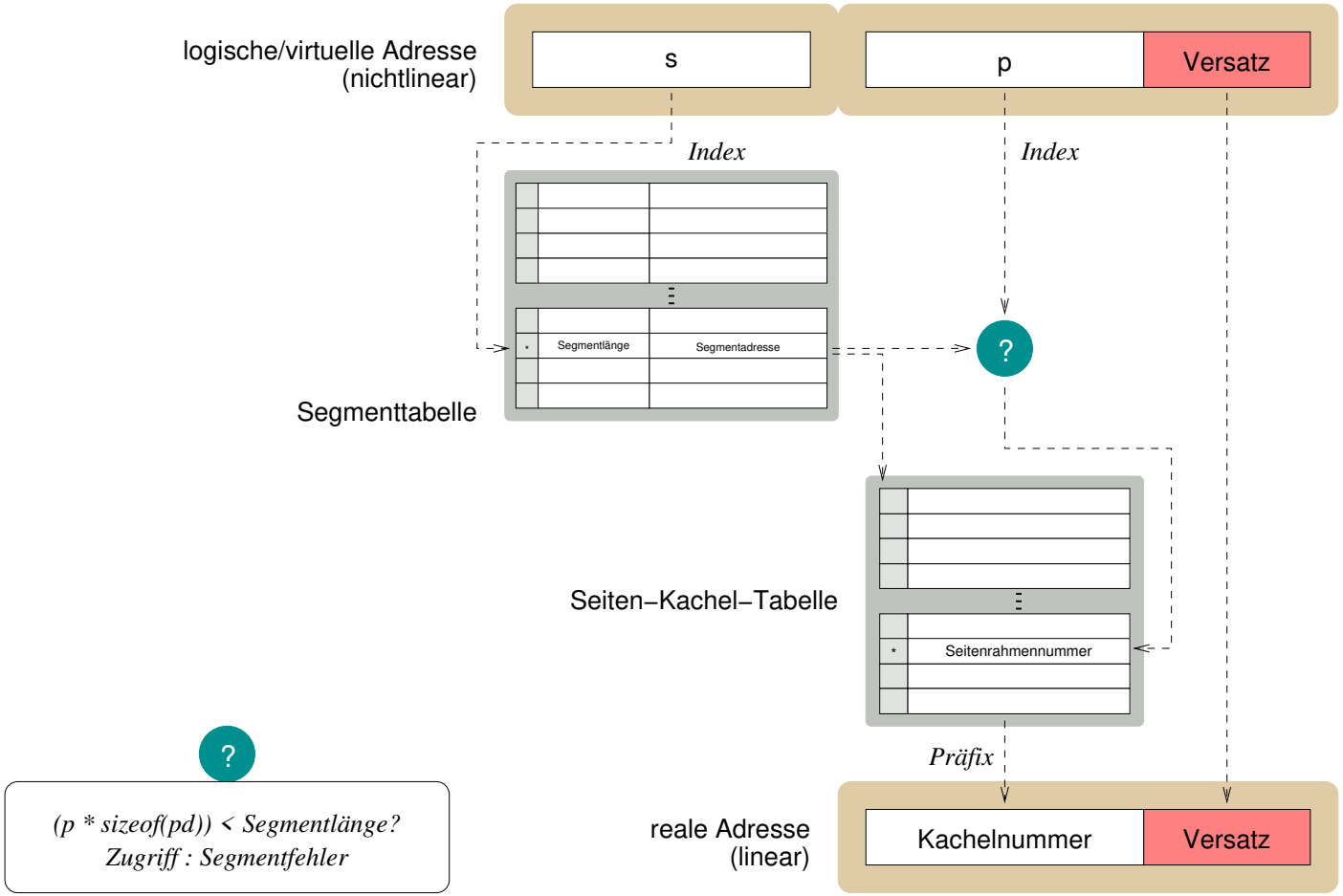
segmented paging




```
1  SKT = la.d < ST[la.s].sd_limit ? ST[la.s].sd_base : segmentation violation ⇝ trap
2  ra  = (SKT[la.d].pd_frame * PSIZE) | (la % PSIZE)
```

Seitennummerierte Segmentierung I

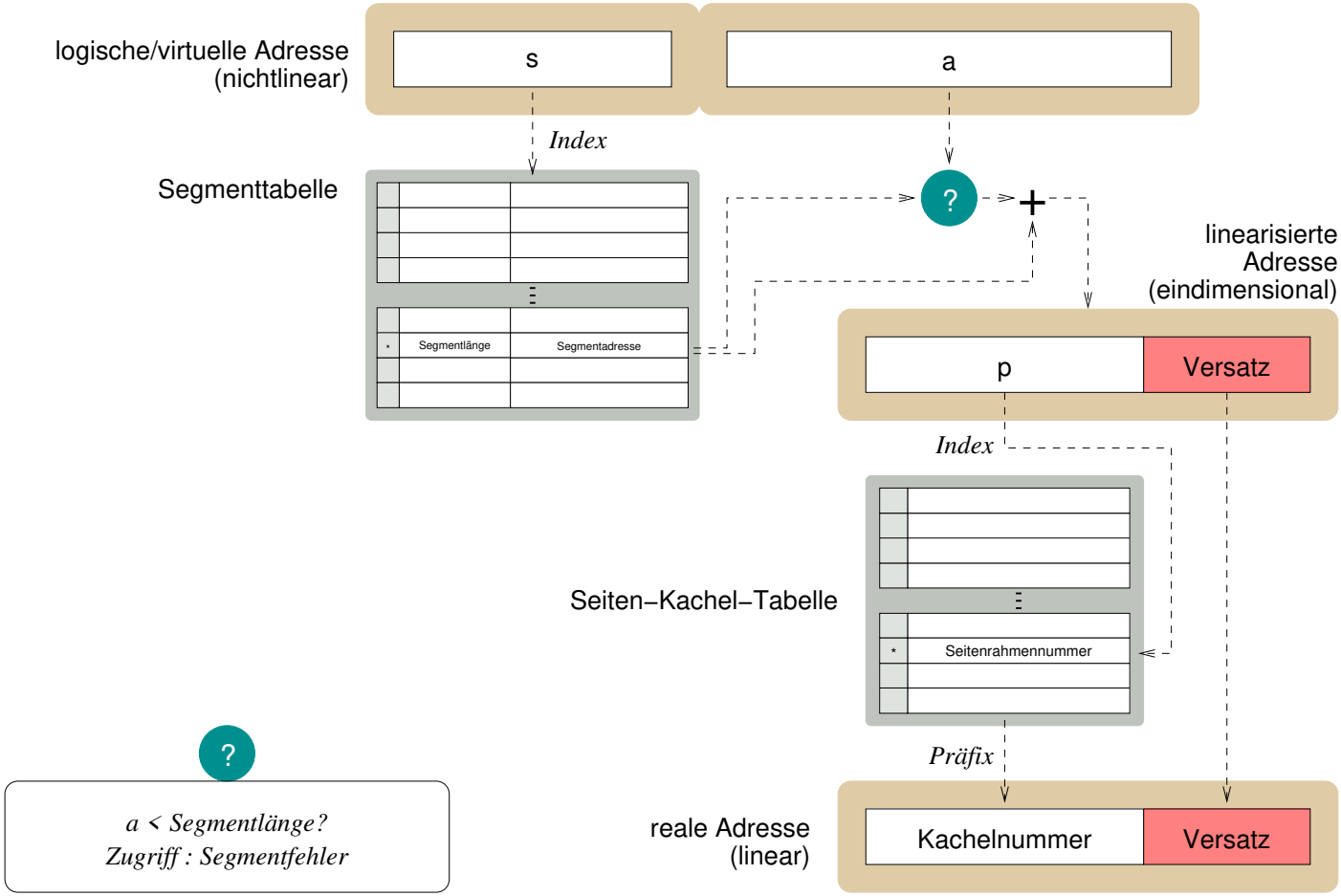
paged segmentation




```
1  len = la.p * sizeof(PDESCRIPTOR)
2  SKT = len < ST[la.s].sd_limit ? ST[la.s].sd_base : segmentation violation ⇝ trap
3  ra  = (SKT[la.p].pd_frame * PSIZE) | (la % PSIZE)
```

Seitennummerierte Segmentierung II

paged segmentation




```
1  ea[1] = la.a < ST[la.s].sd_limit ? ST[la.s].sd_base + la.a: segmentation violation ⇝ trap
2  ra = (SKT[ea.p].pd_frame * PSIZE) | (ea % PSIZE)
```

[1] eindimensionale Adresse ea

Seitennummerierte Segmentierung

- Version I (à la GE645/Multics)
  - der Segmentdeskriptor listet **Seitendeskriptoren**
  - er adressiert damit indirekt ein dynamisches Seitenfeld
  - alle Seiten darin sind gültig für den betreffenden Prozess
  - folglich auch alle Bytes eines jeweiligen Seitenrahmens
  - Fußbereiche von Seitenrahmen können jedoch brach liegen

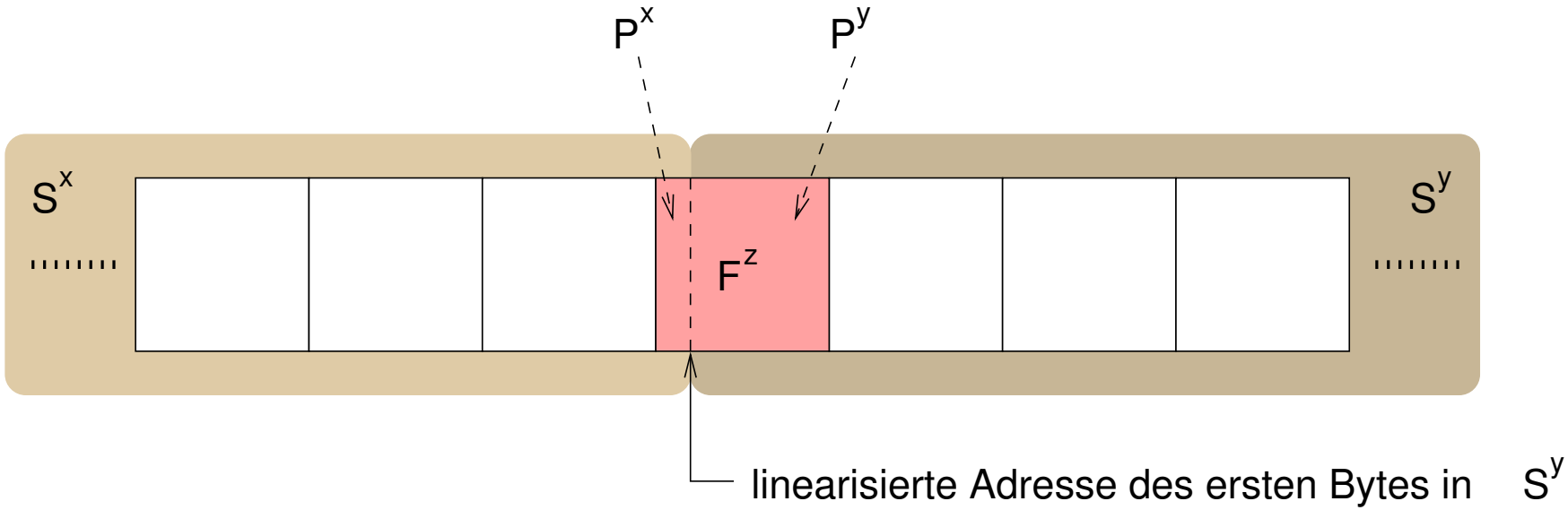
  ↪ Seitenverschnitt unvermeidbar

- Version II (à la IA-32)
  - der Segmentdeskriptor listet **Speicherworte**
  - er adressiert damit direkt ein dynamisches Bytefeld
  - alle Bytes darin sind gültig für den betreffenden Prozess
  - dies unabhängig davon, welche Seitenrahmen sie aufnehmen
  - folglich können Seitenrahmen partiell mitbenutzt werden

  ↪ Verschnitt darin vermeidbar

- Segmenttabellen sind Prozessgrößen (dynamische Felder) und können — je nach Prozess/Programm — sehr groß werden
  - Ausweg ist, Segmenttabellen im virtuellen Speicher zu halten
  - Seitennummerierung sowohl der Segmente als auch Segmenttabellen

Partielle Mitbenutzung von Seitenrahmen I



- Platzierung von Segmentkopf und -fuß in denselben Seitenrahmen F, wobei Kopf- plus Fußlänge die Seitenrahmenlänge (4 KiB) ergibt

  erste Seite
  - $P^y_{[0,4053]}$, Kopf in Segment S^y, liegt auf $F^z_{[42,4095]}$

  letzte Seite
  - $P^x_{[0,41]}$, Fuß in Segment S^x, liegt auf $F^z_{[0,41]}$
  - dabei können S^x und S^y demselben Adressraum (eines Prozesses) oder verschiedenen Adressräumen (zweier Prozesse) angehören
  - falls derselbe Adressraum, kann sogar $S^x = S^y$ gelten, d.h., Kopf und Fuß desselben Segments liegen im selben Seitenrahmen

Partielle Mitbenutzung von Seitenrahmen II

- zu beachten ist, dass sich die **lineare Adresse** des Segmentkopfes auf einen gekachelten logischen/virtuellen Adressraum bezieht
  - diese Adresse ist die im Segmentdeskriptor stehende Segmentbasis und sie muss überhaupt nicht seitenausgerichtet (*page aligned*) sein
  - innerhalb der ersten Seite in diesem Adressraum kann die Segmentbasis um einen Wert $v \in [0, \mathit{sizeof}(P) - 1]$ verschoben sein
  - dieser Wert v entspräche dann der Größe eines zugeteilten Speicherstücks (Segmentfuß) am Anfang eines Seitenrahmens
  - der Rest von $\mathit{sizeof}(P) - v$ Bytes in dem Seitenrahmen entspräche einem Speicherstück, dass einem Segmentkopf zugeteilt werden könnte
- Verschnitt im Seitenrahmen vermeiden, fördert **Interferenz**, bedingt zusätzlich noch die Verwaltung unbenutzter „Seitenschnippsel“ ☹
  - zwei Seiten ggf. zweier Segmente haben Anteile desselben Seitenrahmens
  - Ersetzung des Inhalts dieses Seitenrahmens kann zwei Prozesse „stören“
    - unerwarteterweise bei lokaler Seitenersetzungsstrategie ☹
    - nicht schlimmer als bei globaler Seitenersetzungsstrategie

Segmentierung mit Seitenadressierung

. . . hat Mehrkosten

- **Synergie** der vorteilhaften Merkmale beider Adressumsetzungsarten
  - einfache Platzierungsstrategie, da die Speicherzuteilung kachelorientiert und damit immer in Einheiten gleicher Größe geschieht
  - mehrstufige Seitentabellen fallen weg, da alle Tabellen Segmente und so jeweils in ihrer wirklichen Seitenanzahl beschränkt sind
  - bessere Trennung von Belangen, da Segmente und Seiten bzw. Kacheln verschiedenen Zielen dienen

    Segment – Abbildung und Erfassung von **Programmstrukturen**

    Seite – Optimierung von **Systemfunktionen** der Speicherverwaltung
- Segmentierung unterstützt insbesondere **dynamisches Binden**
  - die „Bindlinge“ sind symbolisch bezeichnete, **physische Segmente**
  - d.h., Programmstrukturen, Adressräume (Seitentabellen), . . . , Dateien
- ein Segment dagegen als „Seitenfeld“ zu begreifen, ist etwas anderes
  - also Seiten zu Text-, Daten- oder Stapelsegmenten zusammenstellen[2]
  - Programmstrukturen lassen sich damit im System nicht wirklich abbilden
    - vom Verwaltungsaufwand mehrstufiger Seitentabellen einmal abgesehen

[2] So, wie es von UNIX-ähnlichen Betriebssystemen (inkl. Linux) bekannt und überhaupt nach Multics [5] eben nur noch gang und gäbe ist.

Gliederung

Resümee

- Segmentierung in Reinform
  - nichtlinearer (zweidimensionaler) Adressraum
  - Segmentdeskriptoren und -tabellen
  - segmentbasierte Adressierung
  - implizite Selektion von Segmentdeskriptoren, je nach Zugriffsart (IA-32)
- Segmentierung kombiniert mit Seitenverfahren
  - segmentierte Seitenadressierung
    - Seitentabelle als Prozessgröße, ein Segment
    - eindimensionale (lineare) logische/virtuelle Adresse
  - seitennummerierte Segmentierung
    - Seitentabelle als Prozessgröße, ein Segment (GE645/Multics)
    - Seitentabelle als Systemgröße (IA-32)

    ↪ Segmenttabellen sind (für gewöhnlich) seitennummeriert organisiert
- doch es ist nicht alles Gold, was glänzt. . .
  - komplexe Hardware
  - komplexe Adressraumverwaltung im Betriebssystem
    - jedoch kaum komplexer als mehr/vielstufige Seitentabellen

Literaturverzeichnis I

[1] Burroughs Corporation (Hrsg.):
The Descriptor — A Definition of the B 5000 Information Processing System.
Detroit 32, Michigan, USA: Burroughs Corporation, Febr. 1961.
(Bulletin 5000-20002-P)

[2] Lonergan, W. ; King, P. :
Design of the B 5000 System.
In: *DATAMATION Magazine* 7 (1961), Mai, Nr. 5, S. 28–32

[3] Mayer, A. J. W.:
The Architecture of the Burroughs B5000: 20 Years Later and Still Ahead of the Times?
In: *ACM SIGARCH Computer Architecture News* 10 (1982), Jun., Nr. 4, S. 3–10

[4] McFarland, C. :
A language-oriented computer design.
In: *Proceedings of the Fall Joint Computer Conference (AFIPS '70)*.
New York, NY, USA : ACM, 1970, S. 629–640

[5] Organick, E. I.:
The Multics System: An Examination of its Structure.
MIT Press, 1972. –
ISBN 0–262–15012–3