Paging in Multics

Ausgewählte Kapitel der Systemsoftwaretechnik

Jonas Rabenstein

7. Dezember 2015

Warum mit Paging befassen?

- Nach wie vor stark verbreitet
- (automatisches) Multiplexen begrenzter Speicherressourcen
- Vermeidung von Verschnitt
- . . . integraler Bestandteil von Multics

Inhalt

Warum Speicherverwaltung?

- Stetig wachsende Nutzerzahlen
  - Komplette Programme im Speicher
  - Kollision von Adressen
  - Time-Sharing zur besseren Auslastung
- Kapazität vs. Kosten vs. Geschwindigkeit
  - viel schneller Speicher nicht wirtschaftlich
  - technische Grenzen
- Komplexität bei Speicherhierarchie

$\Rightarrow$ Verlagerung von Anwender zu Systementwickler

$\Rightarrow$ Automatisierte Lösungen gesucht

Segmentierung

Idee

- Programm in logische Blöcke/Segmente teilen
- Segment hat Basisadresse und Länge
- Zeiger als Tupel $\langle s|i \rangle$

  s Segmentnummer

  i Wortnummer in Segment

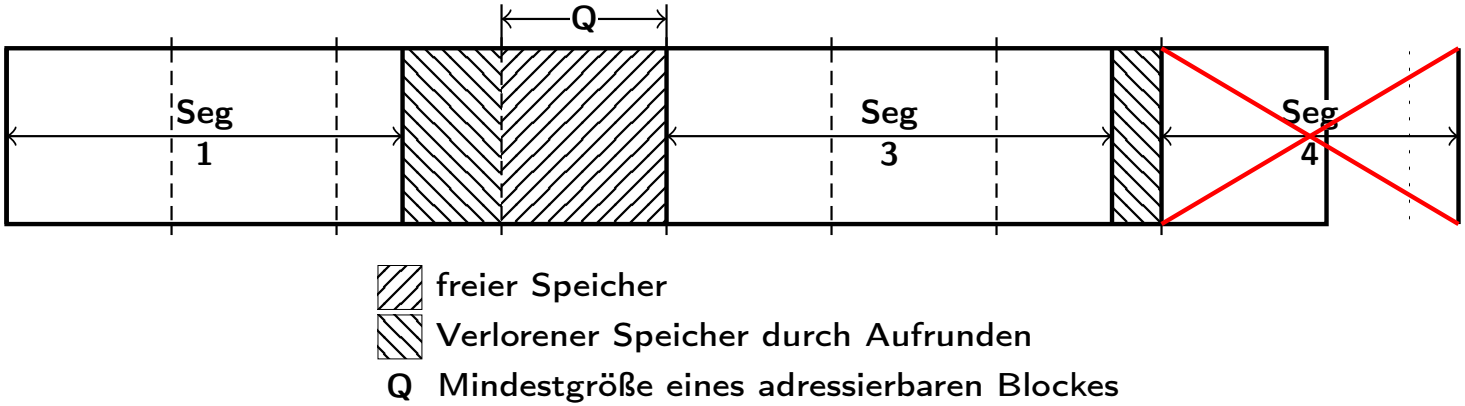
Vorteil

- tatsächliche Position im Speicher irrelevant
- Auslagern unbenötigter Segmente
- Verschieben von Segmenten

Segmentierung

Problem: Segmente haben verschiedene Länge

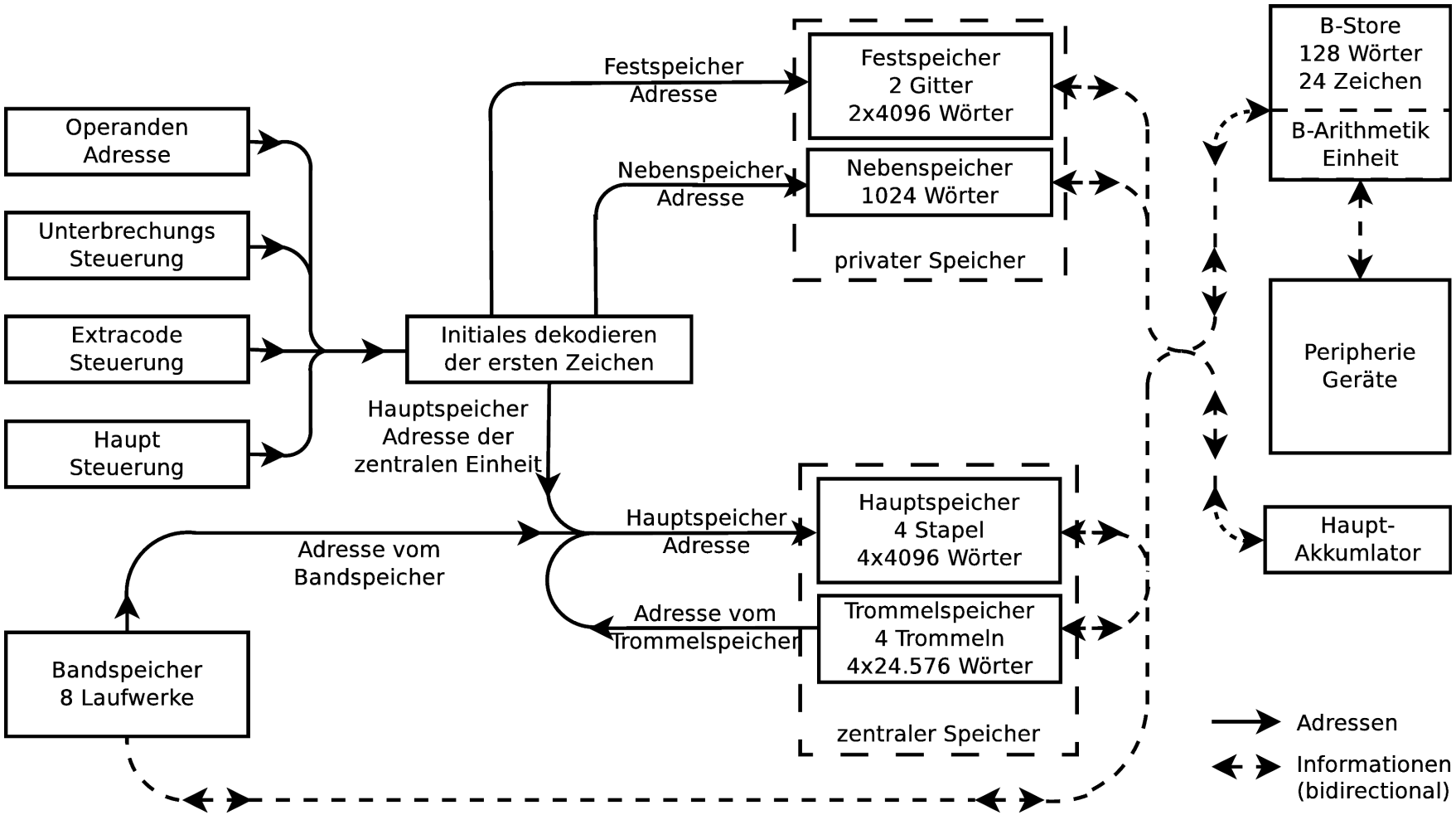
- Komplexität der Verwaltung
- Speicherblöcke nicht komplett genutzt
- Fragmentierung des Speichers



Mögliche Lösung: Paging

- 1962 durch den Atlas Computer
- Transparente Verknüpfung des „zentralen Speichers“
  - Hauptspeicher: 4×4096 Wörter
  - Trommelspeicher: 4×24.576 Wörter
  - Blockgröße: 512 Worte
- „drum transfer learning program“

Aufbau des Atlas Computer

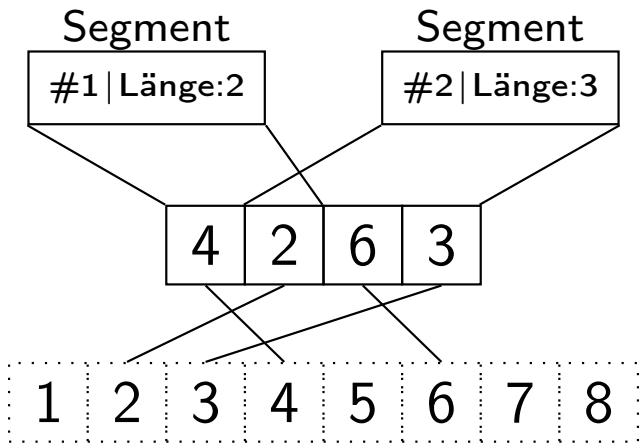


Ziele

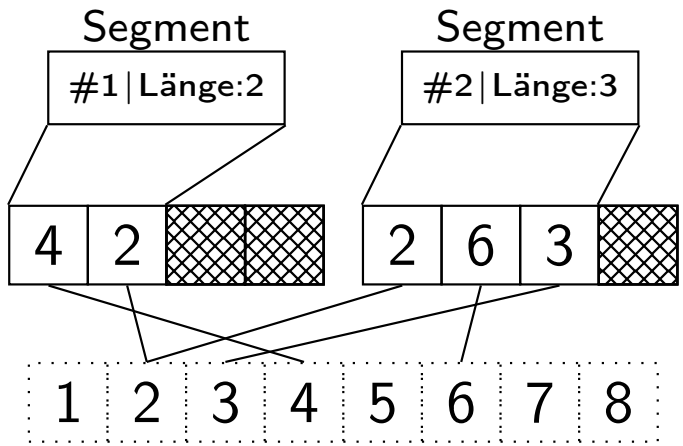
- Verstecken der Transferzeit
  - Vorhalten eines freien Blocks
  - Auslagerung parallel zu Programmfluß
  - Auswahl des zu verdrängenden Blockes während Einlagerung
- Minimierung der Transfers
  - Im Experiment besser als Durchschnittsnutzer [3]
  - Anlehnung an LRU

Verbindung von Beidem

- Zwei Möglichkeiten
  - Segmente über Seiten



  - Seitentabelle für jedes Segment

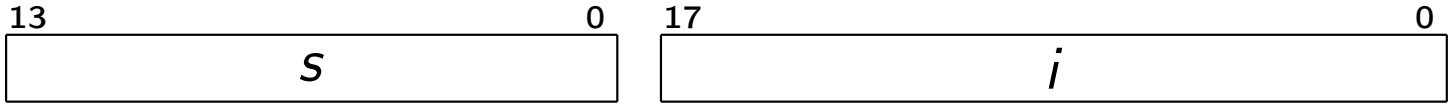


Multics

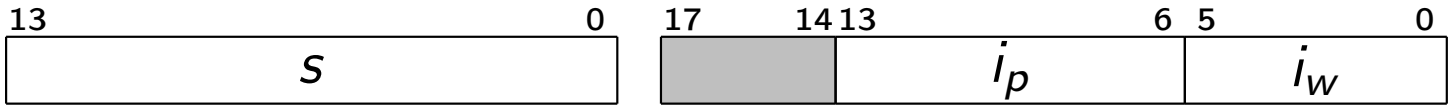
Inhalt

Adressformat

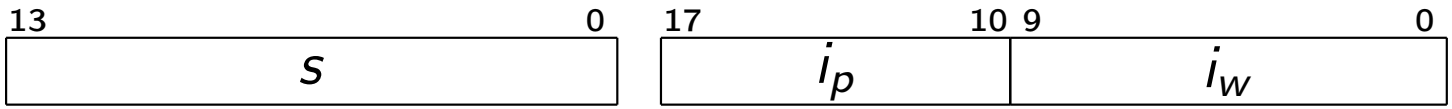
- Ursprünglich 3 Interpretationen eines Zeigers $\langle s|i \rangle$:
  - Nur Segmentierung



  - Segmentierung mit Paging (64 Wort Seiten)



  - Segmentierung mit Paging (1024 Wort Seiten)



- Regelfall: 1024-Wort-Paging

Segment Deskriptor Wort (SDW)

- Beschreibt ein Segment für die Hardware
  - Basisadresse
  - Zugriffsrechte
  - Seitenverwaltung aktiv
  - Länge
  - *invalid bit*
  - Seitengröße (1024 oder 64)
- Pro Prozess in „descriptor segment“ *(DS)* gesammelt
- Wortindex in diesem Segment entspricht Segmentnummer

$\Rightarrow$ Deskriptor Segment als Adressraum des Prozesses

Tabelle bekannter Segmente (KST)

- Segment Informationen für das Betriebssystem
- Verknüpft Dateisystemeintrag mit SDW
- Eintrag mit Index s gehört zu Segment s

Tabelle aktiver Segmente (AST)

- Informationen über aktuell eingelagerte Segmente

  $\Rightarrow$ globale Struktur

- Dupliziert Informationen für zentralen Zugriff
  - Segmentlänge
  - Referenz auf Dateisystemeintrag
  - Segment / File Map [1, 5]

- Liste zugehöriger Segment Deskriptor Worte

  $\Rightarrow$ Prozesse können Segmente teilen

Seitentabelle (PT)

- fest gekoppelt an Eintrag in AST
- zwei Größen verfügbar: 64 oder 256 Einträge
- Anzahl der Tabellen bei Systemkonfiguration festgelegt
- Verwaltet über *used/free list*

Seitentabellen Eintrag (PTE)

- Abbildung Seite $\rightarrow$ Seitenrahmen
- Hardwaregegebene Bits:

Adresse	Adresse des Rahmens (18 Bit)
missing	Eintrag ist ungültig
used	Seite wurde verwendet
modified	Schreibender Zugriff auf Seite

- Zusätzlich von Multics verwendet:

type	*core*, *disk*, *pd* oder *null*
wired	Seite darf nicht ausgelagert werden
first	gerade eingelagert
k Bit	Schieberegister für Ersetzungsstrategie

File / Segment Map

- Geordnete Liste der Adressen ausgelagerter Seiten
- Verschiedene Angaben über Speicherort
  - ahängig von Segmentstatus [1]

    aktiv Eintrag in AST

    inaktiv zugehöriger Zweig im Dateisystem
  - immer in Zweig des Dateisystems [4]

Core map [1]

- speichert Belegung der Seitenrahmen

  $\Rightarrow$ Inverse Abbildung der Seitentabellen

- Verwaltet über vier Listen

  free block Seitenrahmen ist ungenutzt

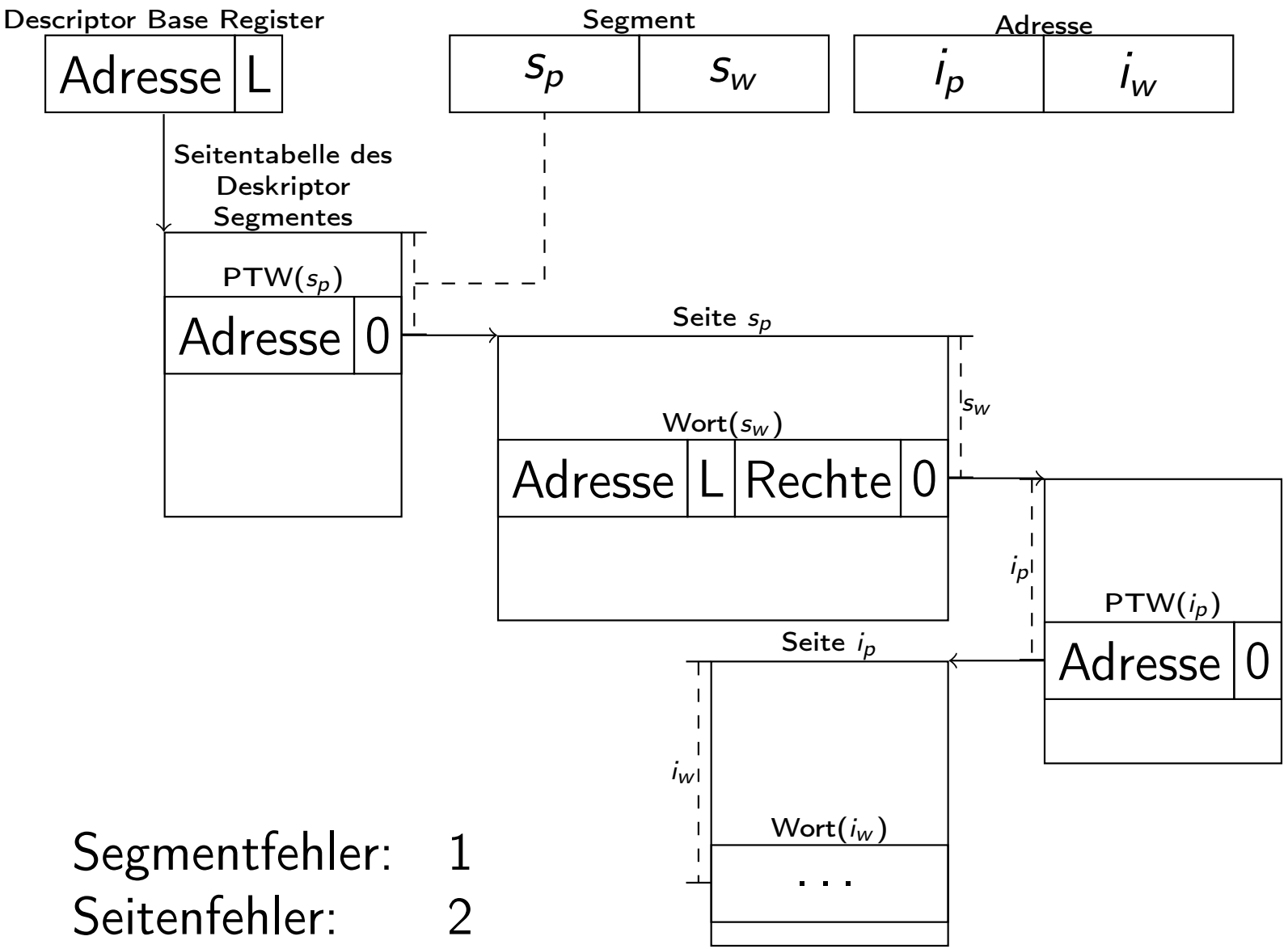
  page pull Transfer zum Einlagern

  page push Transfer zum Auslagern

  replaceable Seitenrahmen ist benutzt

Inhalt

Adressauflösung



Segmentfehler: 1
Seitenfehler: 2

Weitere Fehlerquellen

- Deskriptor Segment bei (lang) inaktiven Prozessen ausgelagert
- Bis 1968: System Segment Table mit 64-Wort Seiten verwaltet

  $\Rightarrow$ ausgelagerte Seitentabellen
- ITS und ITB Zeiger sorgen für weitere Adressauflösungen

$\Rightarrow$ Effektive Behandlung nötig

Behandlung von Seitenfehlern [2] I

- Seite in Behandlung?
  - Warte auf Fertigstellung
  - Behandlung abgeschlossen
- Kein freier Seitenrahmen?
  - Wähle $n = 3$ Seiten für Verdrängung *(Ersetzungsstrategie)*
  - Stoße Auslagerung an: *replaceable page list* $\rightarrow$ *page push list*
- Initiiere Einlagerungsprozess
  - *free block list* $\rightarrow$ *page pull list*
  - Auslagerungs-Speicherort aus Segment Map

Behandlung von Seitenfehlern [2] II

- Behandle fertiggestellte Einlagerungen
  - *page pull list* $\rightarrow$ *replaceable page list*
  - PTW anpassen: Adresse, *missing page* und *first time bit*
  - Wartende Prozesse benachrichtigen
- Behandle fertiggestellte Auslagerungen
  - *page push list* $\rightarrow$ *free block list*
- Warten auf Abschluss des eigenen Transfers
  - Prozessor von anderen Anwendungen genutzt
  - Fertigstellung von anderem Prozess signalisiert

$\Rightarrow$ Programmausführung kann fortgesetzt werden

Mögliche Ausgänge

- Alle Schritte befolgt

  $\Rightarrow$ Seitenfehler behoben

- Behandlung übersprungen aufgrund ...
  - ... Einlagerung $\Rightarrow$ Seitenfehler behoben
  - ... Auslagerung $\Rightarrow$ Seitenfehler wiederholt

$\Rightarrow$ Kontrolle wird stets an Programm zurückgegeben

Ersetzungsstrategie [2] I

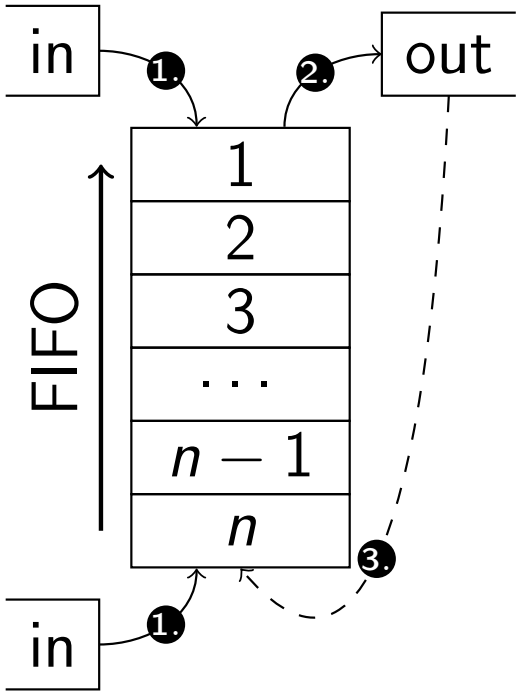
- Entscheidender Teil der Behandlung
  Entscheidungsqualität $\Leftrightarrow$ Übertragungen
  hoher Aufwand $\Leftrightarrow$ weniger Zeitgewinn

- Zwei „extreme“ (bei anzunehmender Lokalität):
  FIFO wenig Aufwand, schlechte Entscheidungen
  LRU hoher Aufwand, mit guten Entscheidungen

$\Rightarrow$ Versuch beides zu verbinden

Ersetzungsstrategie [2] II

FIFO erweitert um



- $k = 1$ Bit Schieberegister pro Element
  - Verwendet bei den letzten k Prüfungen?
  - weitere Chance möglich
- Einfügen an Beginn der Liste und *first time bit*
  - Verdrängen einmalig genutzter Seiten

Ersetzungsstrategie [2] III

1. *first time bit* gesetzt?
   - *first time bit* deaktivieren
   - Weiter bei Schritt 3

2. *used bit* in Schieberegister einfügen
   - falls Schieberegister $= 0$
     $\Rightarrow$ Auswahl getroffen

3. Bearbeite nächstes Element
   - *used bit* zurücksetzen
   - Element an Ende der FIFO einhängen

Inhalt

Experiment zur Effizienz [2]

- Komplexität über k anpassbar

  $k = 0$ entspricht FIFO

  $k = \infty$ entspricht LRU

- Experiment sollte bestes k ermitteln

- Zwei Testfälle

  - Systemstart
  - 10 Standard Anweisungen

Ergebnisse des Experiments [2]

k	Systemstart		10 Standard Anweisungen	
	Seitenfehler	Dauer	Seitenfehler	Dauer
0	8309	184.5*sec*	3628	72.9*sec*
1	4250	107.9*sec*	1659	36.4*sec*
2	4098	106.5*sec*	1635	37.5*sec*
4	4205	112.5*sec*	1598	38.7*sec*
7	4317 ???	120.2*sec*	1725 ???	44.3*sec*

- Erhebliche Verbesserung durch Schieberegister ($\sim 50\%$)
- Beste Ergebnisse für $k = [1, 2]$

$\Rightarrow$ Multics: $k = 1$ aus Gründen der Einfachkeit

Fazit

+ Seitenbasierte Speicherverwaltung heute weit verbreitet
+ Seitengröße an Verwaltungsstrukturen angepasst
+ Verschieden Seitengrößen wurden wieder eingeführt
- Verbindung mit Segmentierung „umgekehrt“

Fragen?

Quellen I

A. Bensoussan, C. T. Clingen, and R. C. Daley.
The multics virtual memory: Concepts and design.
Commun. ACM, 15(5):308–318, May 1972.

F. J. Corbato.
A paging experiment with the multics system.
In *In Honor of Philip M. Morse*, chapter 19, pages 217–228.
M.I.T. Press, 1968.

T. Kilburn, D. Edwards, M. Lanigan, and F. Sumner.
One-level storage system.
Electronic Computers, IRE Transactions on,
EC-11(2):223–235, April 1962.

Quellen II

E. I. Organick.
The Multics System: An Examination of Its Structure.
MIT Press, Cambridge, MA, USA, 1972.

T. H. Van Vlerck.
Multics glossary.
`www.multicians.org/mgloss.html`, 2015.
[Online; abgerufen 22. November 2015].