

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1											
	Kachel 2												
	Kachel 3												
Kontrollzustände (Alter pro Kachel)	Kachel 1	0											
	Kachel 2	>											
	Kachel 3	>											

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1										
	Kachel 2		2										
	Kachel 3												
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1										
	Kachel 2	>	0										
	Kachel 3	>	>										

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1									
	Kachel 2		2	2									
	Kachel 3			3									
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2									
	Kachel 2	>	0	1									
	Kachel 3	>	>	0									

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4								
	Kachel 2		2	2	2								
	Kachel 3			3	3								
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0								
	Kachel 2	>	0	1	2								
	Kachel 3	>	>	0	1								

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4							
	Kachel 2		2	2	2	1							
	Kachel 3			3	3	3							
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1							
	Kachel 2	>	0	1	2	0							
	Kachel 3	>	>	0	1	2							

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4						
	Kachel 2		2	2	2	1	1						
	Kachel 3			3	3	3	2						
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2						
	Kachel 2	>	0	1	2	0	1						
	Kachel 3	>	>	0	1	2	0						

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5					
	Kachel 2		2	2	2	1	1	1					
	Kachel 3			3	3	3	2	2					
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0					
	Kachel 2	>	0	1	2	0	1	2					
	Kachel 3	>	>	0	1	2	0	1					

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5				
	Kachel 2		2	2	2	1	1	1	1				
	Kachel 3			3	3	3	2	2	2				
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1				
	Kachel 2	>	0	1	2	0	1	2	3				
	Kachel 3	>	>	0	1	2	0	1	2				

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5			
	Kachel 2		2	2	2	1	1	1	1	1			
	Kachel 3			3	3	3	2	2	2	2			
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1	2			
	Kachel 2	>	0	1	2	0	1	2	3	4			
	Kachel 3	>	>	0	1	2	0	1	2	3			

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5		
	Kachel 2		2	2	2	1	1	1	1	1	3		
	Kachel 3			3	3	3	2	2	2	2	2		
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1	2	3		
	Kachel 2	>	0	1	2	0	1	2	3	4	0		
	Kachel 3	>	>	0	1	2	0	1	2	3	4		

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5	5	
	Kachel 2		2	2	2	1	1	1	1	1	3	3	
	Kachel 3			3	3	3	2	2	2	2	2	4	
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1	2	3	4	
	Kachel 2	>	0	1	2	0	1	2	3	4	0	1	
	Kachel 3	>	>	0	1	2	0	1	2	3	4	0	

## 7.1 First-In, First-Out

- Älteste Seite wird ersetzt
- Notwendige Zustände:
  - ◆ Alter bzw. Einlagerungszeitpunkt für jede Kachel
- Ablauf der Ersetzungen (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5	5	5
	Kachel 2		2	2	2	1	1	1	1	1	3	3	3
	Kachel 3			3	3	3	2	2	2	2	2	4	4
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	0	1	2	0	1	2	3	4	5
	Kachel 2	>	0	1	2	0	1	2	3	4	0	1	2
	Kachel 3	>	>	0	1	2	0	1	2	3	4	0	1

## 7.1 First-In, First-Out (2)

- Größerer Hauptspeicher mit 4 Kacheln (10 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	5	5	5	5	4	4
	Kachel 2		2	2	2	2	2	2	1	1	1	1	5
	Kachel 3			3	3	3	3	3	3	2	2	2	2
	Kachel 4				4	4	4	4	4	4	3	3	3
Kontrollzustände (Alter pro Kachel)	Kachel 1	0	1	2	3	4	5	0	1	2	3	0	1
	Kachel 2	>	0	1	2	3	4	5	0	1	2	3	0
	Kachel 3	>	>	0	1	2	3	4	5	0	1	2	3
	Kachel 4	>	>	>	0	1	2	3	4	5	0	1	2

- FIFO Anomalie (Belady's Anomalie, 1969)

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1											
	Kachel 2												
	Kachel 3												
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4											
	Kachel 2	>											
	Kachel 3	>											

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1										
	Kachel 2		2										
	Kachel 3												
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3										
	Kachel 2	>	4										
	Kachel 3	>	>										

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1									
	Kachel 2		2	2									
	Kachel 3			3									
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2									
	Kachel 2	>	4	3									
	Kachel 3	>	>	7									

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge): minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1								
	Kachel 2		2	2	2								
	Kachel 3			3	4								
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1								
	Kachel 2	>	4	3	2								
	Kachel 3	>	>	7	7								

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge): minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1							
	Kachel 2		2	2	2	2							
	Kachel 3			3	4	4							
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3							
	Kachel 2	>	4	3	2	1							
	Kachel 3	>	>	7	7	6							

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1						
	Kachel 2		2	2	2	2	2						
	Kachel 3			3	4	4	4						
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2						
	Kachel 2	>	4	3	2	1	3						
	Kachel 3	>	>	7	7	6	5						

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1					
	Kachel 2		2	2	2	2	2	2					
	Kachel 3			3	4	4	4	5					
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1					
	Kachel 2	>	4	3	2	1	3	2					
	Kachel 3	>	>	7	7	6	5	5					

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1				
	Kachel 2		2	2	2	2	2	2	2				
	Kachel 3			3	4	4	4	5	5				
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>				
	Kachel 2	>	4	3	2	1	3	2	1				
	Kachel 3	>	>	7	7	6	5	5	4				

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1			
	Kachel 2		2	2	2	2	2	2	2	2			
	Kachel 3			3	4	4	4	5	5	5			
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>			
	Kachel 2	>	4	3	2	1	3	2	1	>			
	Kachel 3	>	>	7	7	6	5	5	4	3			

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	3		
	Kachel 2		2	2	2	2	2	2	2	2	2		
	Kachel 3			3	4	4	4	5	5	5	5		
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>		
	Kachel 2	>	4	3	2	1	3	2	1	>	>		
	Kachel 3	>	>	7	7	6	5	5	4	3	2		

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	3	4	
	Kachel 2		2	2	2	2	2	2	2	2	2	2	
	Kachel 3			3	4	4	4	5	5	5	5	5	
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>	>	
	Kachel 2	>	4	3	2	1	3	2	1	>	>	>	
	Kachel 3	>	>	7	7	6	5	5	4	3	2	1	

## 7.2 Optimale Ersetzungsstrategie

- Vorwärtsabstand
  - ◆ Zeitdauer bis zum nächsten Zugriff auf die entsprechende Seite
- Strategie  $B_0$  (OPT oder MIN) ist optimal (bei fester Kachelmenge):  
minimale Anzahl von Einlagerungen/Ersetzungen (hier 7)
  - ◆ „Ersetze immer die Seite mit dem größten Vorwärtsabstand!“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	3	4	4
	Kachel 2		2	2	2	2	2	2	2	2	2	2	2
	Kachel 3			3	4	4	4	5	5	5	5	5	5
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>	>	>
	Kachel 2	>	4	3	2	1	3	2	1	>	>	>	>
	Kachel 3	>	>	7	7	6	5	5	4	3	2	1	>

## 7.2 Optimale Ersetzungsstrategie (2)

- Vergrößerung des Hauptspeichers (4 Kacheln): 6 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	1	4	4
	Kachel 2		2	2	2	2	2	2	2	2	2	2	2
	Kachel 3			3	3	3	3	3	3	3	3	3	3
	Kachel 4				4	4	4	5	5	5	5	5	5
Kontrollzustände (Vorwärts- abstand)	Kachel 1	4	3	2	1	3	2	1	>	>	>	>	>
	Kachel 2	>	4	3	2	1	3	2	1	>	>	>	>
	Kachel 3	>	>	7	6	5	4	3	2	1	>	>	>
	Kachel 4	>	>	>	7	6	5	5	4	3	2	1	>

- ◆ keine Anomalie

## 7.2 Optimale Ersetzungsstrategie (3)

- Implementierung von  $B_0$  nahezu unmöglich
  - ◆ Referenzfolge müsste vorher bekannt sein
  - ◆  $B_0$  meist nur zum Vergleich von Strategien brauchbar
- Suche nach Strategien, die möglichst nahe an  $B_0$  kommen
  - ◆ z.B. *Least recently used* (LRU)

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1											
	Kachel 2												
	Kachel 3												
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0											
	Kachel 2	>											
	Kachel 3	>											

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1										
	Kachel 2		2										
	Kachel 3												
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1										
	Kachel 2	>	0										
	Kachel 3	>	>										

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1									
	Kachel 2		2	2									
	Kachel 3			3									
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2									
	Kachel 2	>	0	1									
	Kachel 3	>	>	0									

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4								
	Kachel 2		2	2	2								
	Kachel 3			3	3								
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0								
	Kachel 2	>	0	1	2								
	Kachel 3	>	>	0	1								

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4							
	Kachel 2		2	2	2	1							
	Kachel 3			3	3	3							
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1							
	Kachel 2	>	0	1	2	0							
	Kachel 3	>	>	0	1	2							

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4						
	Kachel 2		2	2	2	1	1						
	Kachel 3			3	3	3	2						
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2						
	Kachel 2	>	0	1	2	0	1						
	Kachel 3	>	>	0	1	2	0						

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5					
	Kachel 2		2	2	2	1	1	1					
	Kachel 3			3	3	3	2	2					
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0					
	Kachel 2	>	0	1	2	0	1	2					
	Kachel 3	>	>	0	1	2	0	1					

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5				
	Kachel 2		2	2	2	1	1	1	1				
	Kachel 3			3	3	3	2	2	2				
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0	1				
	Kachel 2	>	0	1	2	0	1	2	0				
	Kachel 3	>	>	0	1	2	0	1	2				

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5			
	Kachel 2		2	2	2	1	1	1	1	1			
	Kachel 3			3	3	3	2	2	2	2			
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0	1	2			
	Kachel 2	>	0	1	2	0	1	2	0	1			
	Kachel 3	>	>	0	1	2	0	1	2	0			

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	3		
	Kachel 2		2	2	2	1	1	1	1	1	1		
	Kachel 3			3	3	3	2	2	2	2	2		
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0	1	2	0		
	Kachel 2	>	0	1	2	0	1	2	0	1	2		
	Kachel 3	>	>	0	1	2	0	1	2	0	1		

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	3	3	
	Kachel 2		2	2	2	1	1	1	1	1	1	4	
	Kachel 3			3	3	3	2	2	2	2	2	2	
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0	1	2	0	1	
	Kachel 2	>	0	1	2	0	1	2	0	1	2	0	
	Kachel 3	>	>	0	1	2	0	1	2	0	1	2	

## 7.3 Least Recently Used (LRU)

- Rückwärtsabstand
  - ◆ Zeitdauer, seit dem letzten Zugriff auf die Seite
- LRU Strategie (10 Einlagerungen)
  - ◆ „Ersetze die Seite mit dem größten Rückwärtsabstand !“

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	3	3	3
	Kachel 2		2	2	2	1	1	1	1	1	1	4	4
	Kachel 3			3	3	3	2	2	2	2	2	2	5
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	0	1	2	0	1	2	0	1	2
	Kachel 2	>	0	1	2	0	1	2	0	1	2	0	1
	Kachel 3	>	>	0	1	2	0	1	2	0	1	2	0

## 7.3 Least Recently Used (2)

- Vergrößerung des Hauptspeichers (4 Kacheln): 8 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	1	1	1	1	1	5
	Kachel 2		2	2	2	2	2	2	2	2	2	2	2
	Kachel 3			3	3	3	3	5	5	5	5	4	4
	Kachel 4				4	4	4	4	4	4	3	3	3
Kontrollzustände (Rückwärts- abstand)	Kachel 1	0	1	2	3	0	1	2	0	1	2	3	0
	Kachel 2	>	0	1	2	3	0	1	2	0	1	2	3
	Kachel 3	>	>	0	1	2	3	0	1	2	3	0	1
	Kachel 4	>	>	>	0	1	2	3	4	5	0	1	2

## 7.3 Least Recently Used (3)

- Keine Anomalie
  - ◆ Allgemein gilt: Es gibt eine Klasse von Algorithmen (Stack-Algorithmen), bei denen keine Anomalie auftritt:
    - Bei Stack-Algorithmen ist bei  $n$  Kacheln zu jedem Zeitpunkt eine Untermenge der Seiten eingelagert, die bei  $n+1$  Kacheln zum gleichen Zeitpunkt eingelagert wären!
    - LRU: Es sind immer die letzten  $n$  benutzten Seiten eingelagert
    - $B_0$ : Es sind die  $n$  bereits benutzten Seiten eingelagert, die als nächstes zugegriffen werden
- ▲ Problem
  - ◆ Implementierung von LRU nicht ohne Hardwareunterstützung möglich
  - ◆ Es muss jeder Speicherzugriff berücksichtigt werden

## 7.3 Least Recently Used (4)

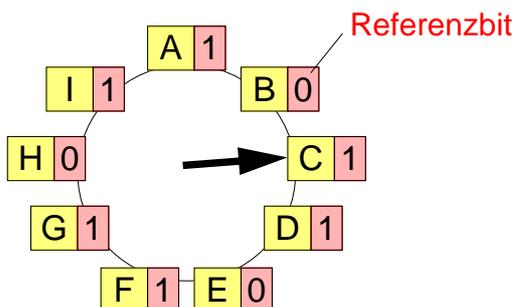
- Hardwareunterstützung durch Zähler
  - ◆ CPU besitzt einen Zähler, der bei jedem Speicherzugriff erhöht wird (inkrementiert wird)
  - ◆ bei jedem Zugriff wird der aktuelle Zählerwert in den jeweiligen Seitendeskriptor geschrieben
  - ◆ Auswahl der Seite mit dem kleinsten Zählerstand
- ▲ Aufwendige Implementierung
  - ◆ viele zusätzliche Speicherzugriffe

## 7.4 Second Chance (Clock)

- Einsatz von Referenzbits
  - ◆ Referenzbit im Seitendeskriptor wird automatisch durch Hardware gesetzt, wenn die Seite zugegriffen wird
    - einfacher zu implementieren
    - weniger zusätzliche Speicherzugriffe
    - moderne Prozessoren bzw. MMUs unterstützen Referenzbits (z.B. Pentium: *Access bit*)
- Ziel: Annäherung von LRU
  - ◆ das Referenzbit wird zunächst auf 0 gesetzt
  - ◆ wird eine Opferseite gesucht, so werden die Kacheln reihum inspiziert
  - ◆ ist das Referenzbit 1, so wird es auf 0 gesetzt (zweite Chance)
  - ◆ ist das Referenzbit 0, so wird die Seite ersetzt

## 7.4 Second Chance (2)

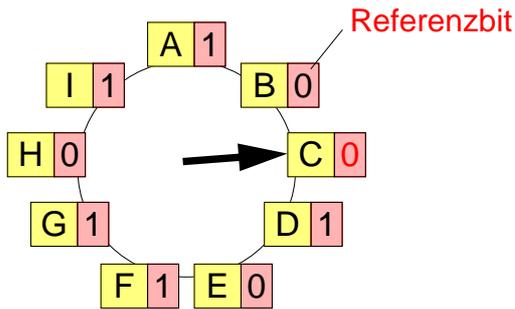
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

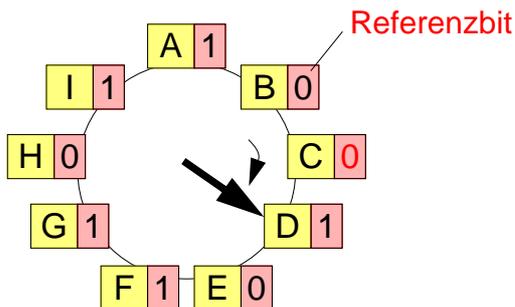
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

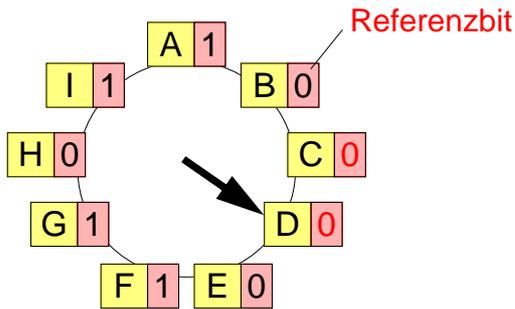
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

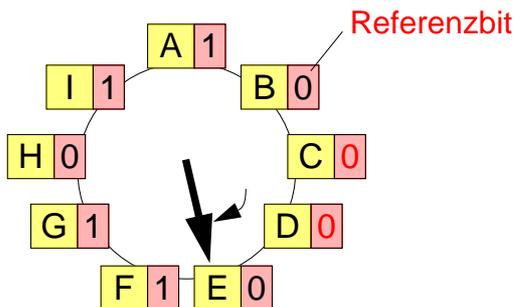
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

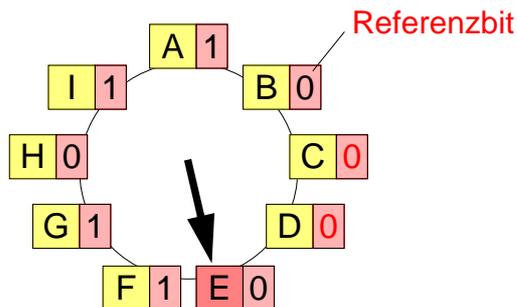
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

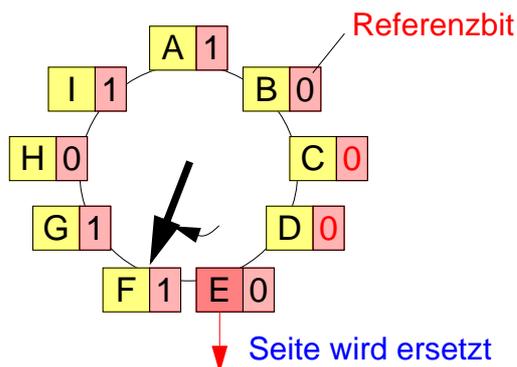
- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (2)

- Implementierung mit umlaufendem Zeiger (*Clock*)



- ◆ an der Zeigerposition wird Referenzbit getestet
  - falls Referenzbit eins, wird Bit gelöscht
  - falls Referenzbit gleich Null, wurde ersetzbare Seite gefunden
  - Zeiger wird weitergestellt; falls keine Seite gefunden: Wiederholung
- ◆ falls alle Referenzbits auf 1 stehen, wird Second chance zu FIFO

## 7.4 Second Chance (3)

### ■ Ablauf bei drei Kacheln (9 Einlagerungen)

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	4	4	4	5	5	5	5	5	5
	Kachel 2		2	2	2	1	1	1	1	1	3	3	3
	Kachel 3			3	3	3	2	2	2	2	2	4	4
Kontroll- zustände (Referenzbits)	Kachel 1	1	1	1	1	1	1	1	1	1	0	0	1
	Kachel 2	0	1	1	0	1	1	0	1	1	1	1	1
	Kachel 3	0	0	1	0	0	1	0	0	1	0	1	1
	Umlaufzeiger	2	3	1	2	3	1	2	2	2	3	1	1

## 7.4 Second Chance (4)

### ■ Vergrößerung des Hauptspeichers (4 Kacheln): 10 Einlagerungen

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
Hauptspeicher	Kachel 1	1	1	1	1	1	1	5	5	5	5	4	4
	Kachel 2		2	2	2	2	2	2	1	1	1	1	5
	Kachel 3			3	3	3	3	3	3	2	2	2	2
	Kachel 4				4	4	4	4	4	4	3	3	3
Kontroll- zustände (Referenzbits)	Kachel 1	1	1	1	1	1	1	1	1	1	1	1	1
	Kachel 2	0	1	1	1	1	1	0	1	1	1	0	1
	Kachel 3	0	0	1	1	1	1	0	0	1	1	0	0
	Kachel 4	0	0	0	1	1	1	0	0	0	1	0	0
	Umlaufzeiger	2	3	4	1	1	1	2	3	4	1	2	3

## 7.4 Second Chance (5)

- Second chance zeigt FIFO Anomalie
  - ◆ Wenn alle Referenzbits gleich 1, wird nach FIFO entschieden
- Erweiterung
  - ◆ Modifikationsbit kann zusätzlich berücksichtigt werden (*Dirty bit*)
  - ◆ drei Klassen: (0,0), (1,0) und (1,1) mit (Referenzbit, Modifikationsbit)
  - ◆ Suche nach der niedrigsten Klasse (Einsatz im MacOS)

## 7.5 Freiseitenpuffer

- Statt eine Seite zu ersetzen wird permanent eine Menge freier Seiten gehalten
  - ◆ Auslagerung geschieht im „voraus“
  - ◆ Effizienter: Ersetzungszeit besteht im Wesentlichen nur aus Einlagerungszeit
- Behalten der Seitenzuordnung auch nach der Auslagerung
  - ◆ Wird die Seite doch noch benutzt bevor sie durch eine andere ersetzt wird, kann sie mit hoher Effizienz wiederverwendet werden.
  - ◆ Seite wird aus Freiseitpuffer ausgetragen und wieder dem entsprechenden Prozess zugeordnet.

## 7.6 Seitenanforderung

- ▲ Problem: Zuordnung der Kacheln zu mehreren Prozessen
- Begrenzungen
  - ◆ Maximale Seitenmenge: begrenzt durch Anzahl der Kacheln
  - ◆ Minimale Seitenmenge: abhängig von der Prozessorarchitektur
    - Mindestens die Anzahl von Seiten nötig, die theoretisch bei einem Maschinenbefehl benötigt werden (z.B. zwei Seiten für den Befehl, vier Seiten für die adressierten Daten)
- Gleiche Zuordnung
  - ◆ Anzahl der Prozesse bestimmt die Kachelmenge, die ein Prozess bekommt
- Größenabhängige Zuordnung
  - ◆ Größe des Programms fließt in die zugeteilte Kachelmenge ein

## 7.6 Seitenanforderung

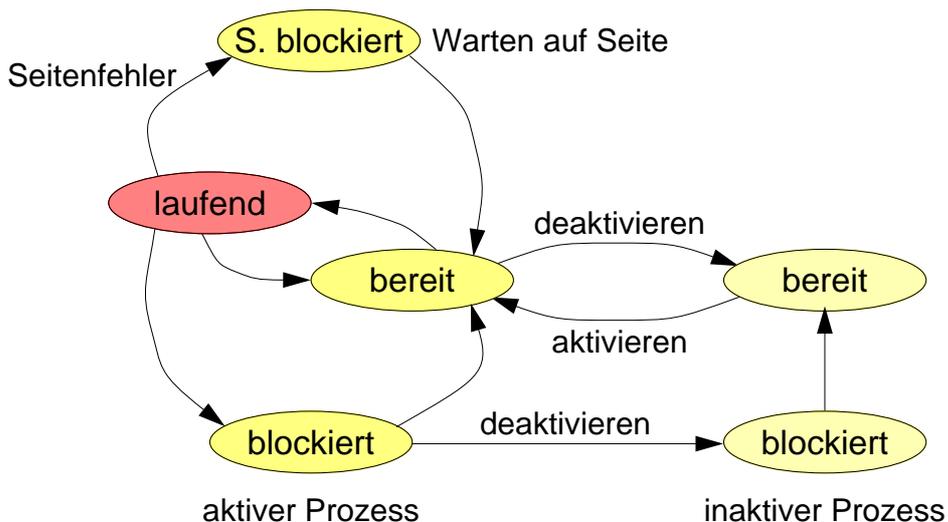
- Globale und lokale Anforderung von Seiten
  - ◆ lokal: Prozess ersetzt nur immer seine eigenen Seiten
    - Seitenfehler-Verhalten liegt nur in der Verantwortung des Prozesses
  - ◆ global: Prozess ersetzt auch Seiten anderer Prozesse
    - bessere Effizienz, da ungenutzte Seiten von anderen Prozessen verwendet werden können

## 8 Seitenflattern (*Thrashing*)

- Ausgelagerte Seite wird gleich wieder angesprochen
  - ◆ Prozess verbringt mehr Zeit mit dem Warten auf das Beheben von Seitenfehler als mit der eigentlichen Ausführung
- Ursachen
  - ◆ Prozess ist nahe am Seitenminimum
  - ◆ zu viele Prozesse gleichzeitig im System
  - ◆ schlechte Ersetzungsstrategie
- ★ Lokale Seitenanforderung behebt Thrashing zwischen Prozessen
- ★ Zuteilung einer genügend großen Zahl von Kacheln behebt Thrashing innerhalb der Prozessseiten
  - ◆ Begrenzung der Prozessanzahl

### 8.1 Deaktivieren von Prozessen

- Einführung von „Superzuständen“



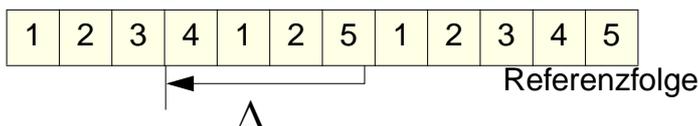
- ◆ inaktiver Prozess benötigt keine Kacheln; Prozess ist vollständig ausgelagert (swapped out)

## 8.1 Deaktivieren von Prozessen (2)

- Sind zuviele Prozesse aktiv, werden welche deaktiviert
  - ◆ Kacheln teilen sich auf weniger Prozesse auf
  - ◆ Verbindung mit dem Scheduling nötig
    - Verhindern von Aushungerung
    - Erzielen kurzer Reaktionszeiten
  - ◆ guter Kandidat: Prozess mit wenigen Seiten im Hauptspeicher
    - geringe Latenz bei Wiedereinlagerung bzw. wenige Seitenfehler bei Aktivierung und Demand paging

## 8.2 Arbeitsmengenmodell

- Menge der Seiten, die ein Prozess wirklich braucht (*Working set*)
  - ◆ kann nur angenähert werden, da üblicherweise nicht vorhersehbar
- Annäherung durch Betrachten der letzten  $\Delta$  Seiten, die angesprochen wurden
  - ◆ geeignete Wahl von  $\Delta$ 
    - zu groß: Überlappung von lokalen Zugriffsmustern zu klein: Arbeitsmenge enthält nicht alle nötigen Seiten



- **Hinweis:**  $\Delta >$  Arbeitsmenge, da Seiten in der Regel mehrfach hintereinander angesprochen werden

## 8.2 Arbeitsmengenmodell (2)

- Beispiel: Arbeitsmengen bei verschiedenen  $\Delta$

Referenzfolge		1	2	3	4	1	2	5	1	2	3	4	5
$\Delta = 3$	Seite 1	x	x	x		x	x	x	x	x	x		
	Seite 2		x	x	x		x	x	x	x	x	x	
	Seite 3			x	x	x					x	x	x
	Seite 4				x	x	x					x	x
	Seite 5							x	x	x			x
$\Delta = 4$	Seite 1	x	x	x	x	x	x	x	x	x	x	x	
	Seite 2		x	x	x	x	x	x	x	x	x	x	x
	Seite 3			x	x	x	x				x	x	x
	Seite 4				x	x	x	x				x	x
	Seite 5							x	x	x	x		x

## 8.2 Arbeitsmengenmodell (3)

- Annäherung der Zugriffe durch die Zeit
  - ◆ bestimmtes Zeitintervall ist ungefähr proportional zu Anzahl von Speicherzugriffen
- ▲ Virtuelle Zeit des Prozesses muss gemessen werden
  - ◆ nur die Zeit relevant, in der der Prozess im Zustand laufend ist
  - ◆ Verwalten virtueller Uhren pro Prozess

## 8.3 Arbeitsmengenbestimmung mit Zeitgeber

- Annäherung der Arbeitsmenge mit
  - ◆ Referenzbit
  - ◆ Altersangabe pro Seite (Zeitintervall ohne Benutzung)
  - ◆ Timer-Interrupt (durch Zeitgeber)
- Algorithmus
  - ◆ durch regelmäßigen Interrupt wird mittels Referenzbit die Altersangabe fortgeschrieben:
    - ist Referenzbit gesetzt (Seite wurde benutzt) wird das Alter auf Null gesetzt;
    - ansonsten wird Altersangabe erhöht.
    - Es werden nur die Seiten des gerade laufenden Prozesses „gealtert“.
  - ◆ Seiten mit Alter  $> \Delta$  sind nicht mehr in der Arbeitsmenge des jeweiligen Prozesses

## 8.3 Arbeitsmengenbestimmung mit Zeitgeber (2)

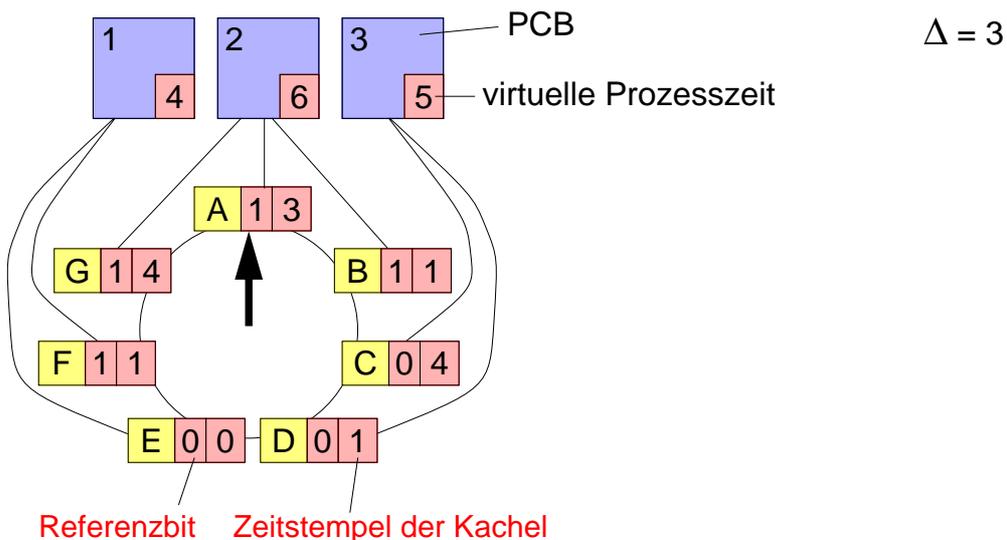
- ▲ Ungenau: System ist aber nicht empfindlich auf diese Ungenauigkeit
  - ◆ Verringerung der Zeitintervalle: höherer Aufwand, genauere Messung
- ▲ Ineffizient
  - ◆ große Menge von Seiten zu betrachten

## 8.4 Arbeitsmengenbestimmung mit WSClock

- Algorithmus WSClock (Working set clock)
  - ◆ arbeitet wie Clock
  - ◆ Seite wird nur dann ersetzt, wenn sie nicht zur Arbeitsmenge ihres Prozesses gehört oder der Prozess deaktiviert ist
  - ◆ Bei Zurücksetzen des Referenzbits wird die virtuelle Zeit des jeweiligen Prozesses eingetragen, die z.B. im PCB gehalten und fortgeschrieben wird
  - ◆ Bestimmung der Arbeitsmenge erfolgt durch Differenzbildung von virtueller Zeit des Prozesses und Zeitstempel in der Kachel

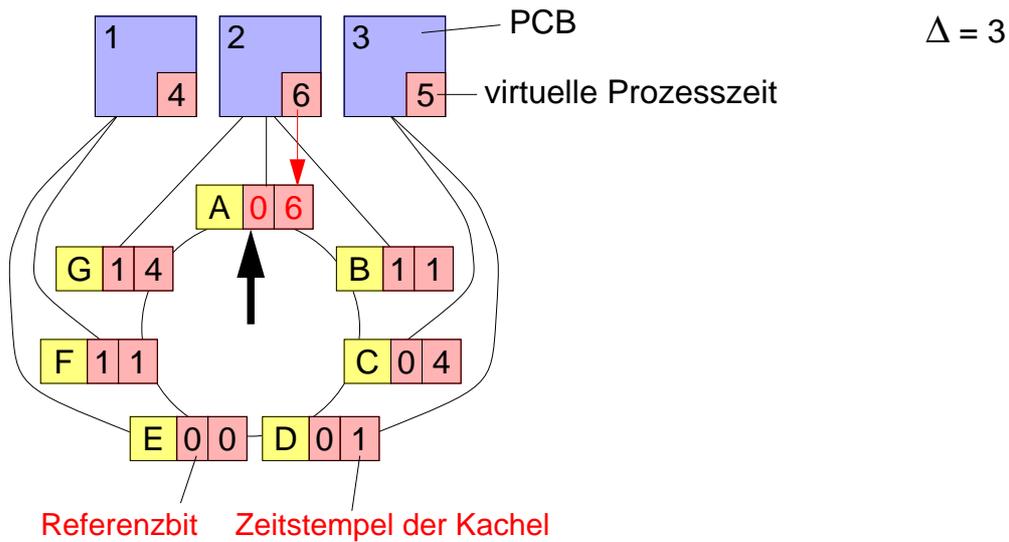
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



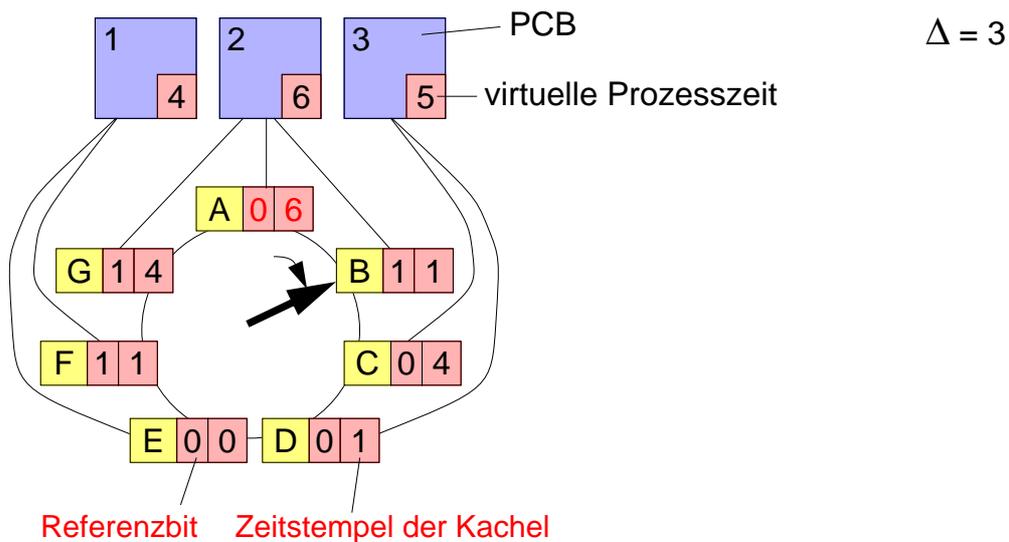
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



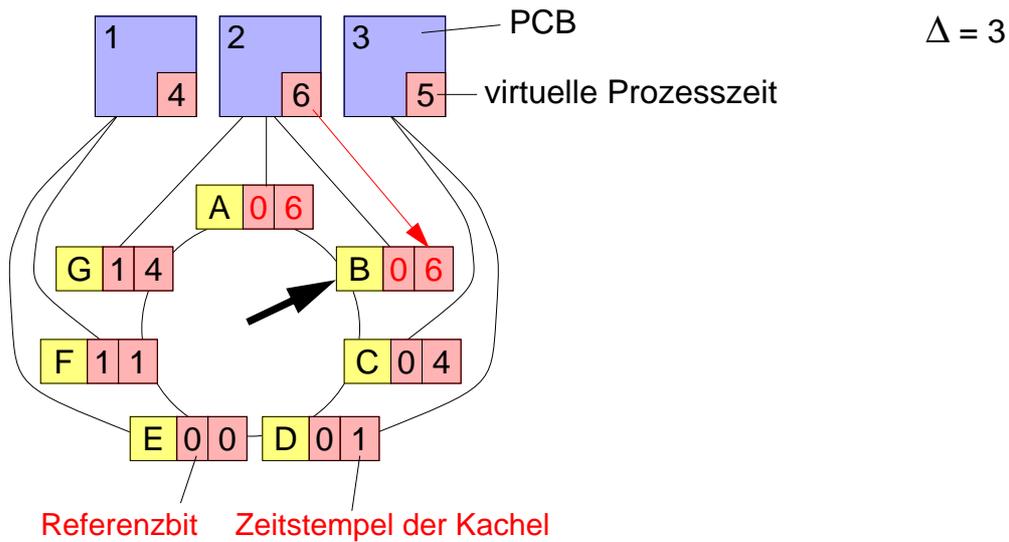
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



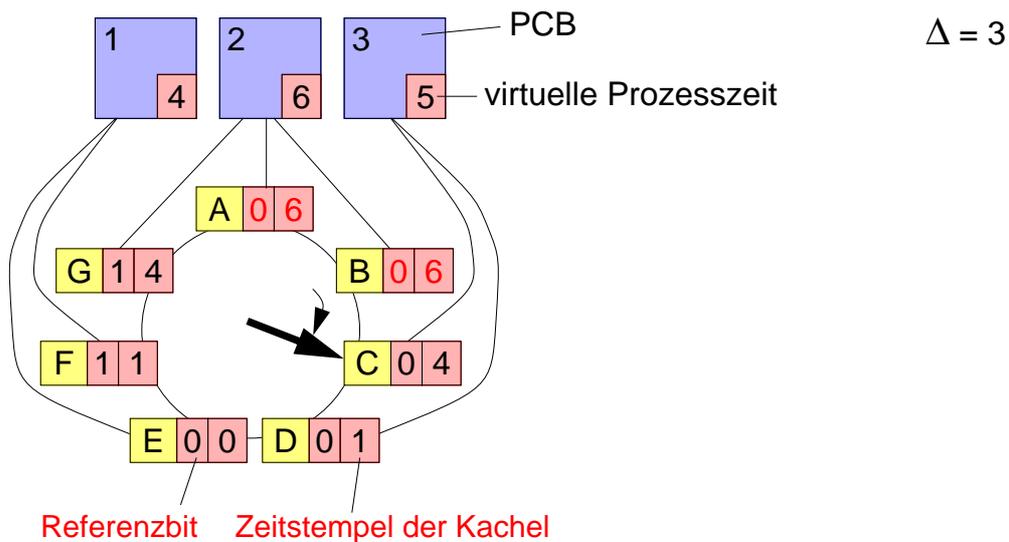
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



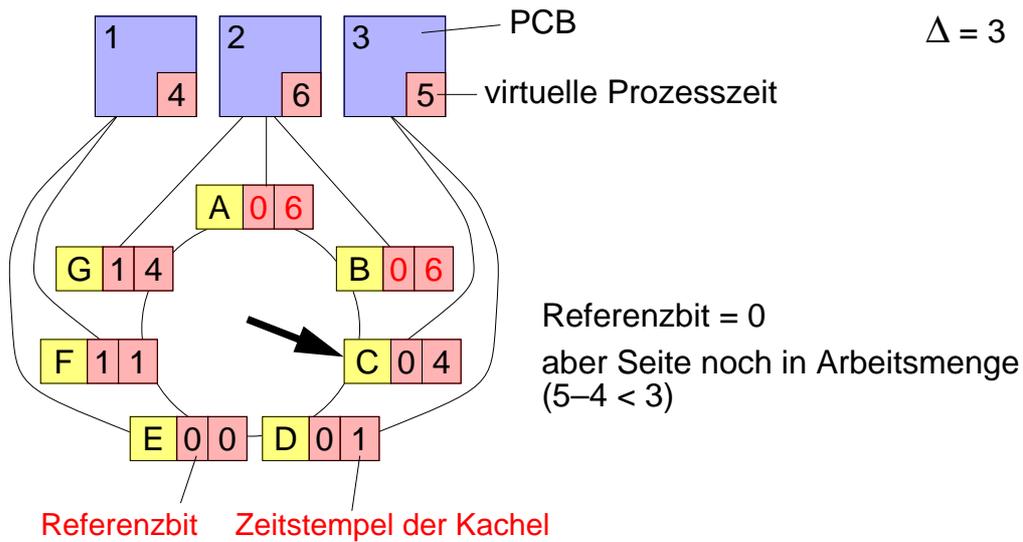
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



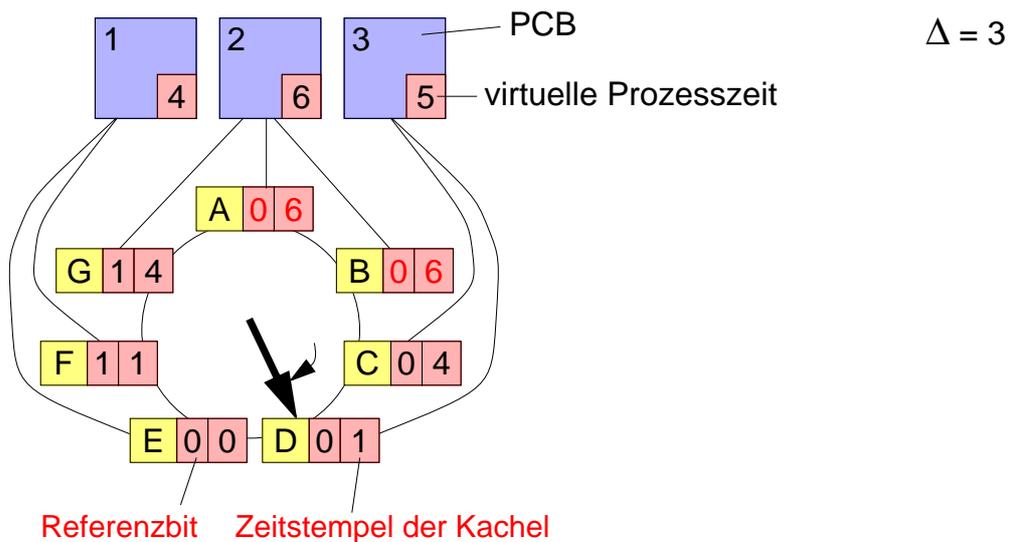
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



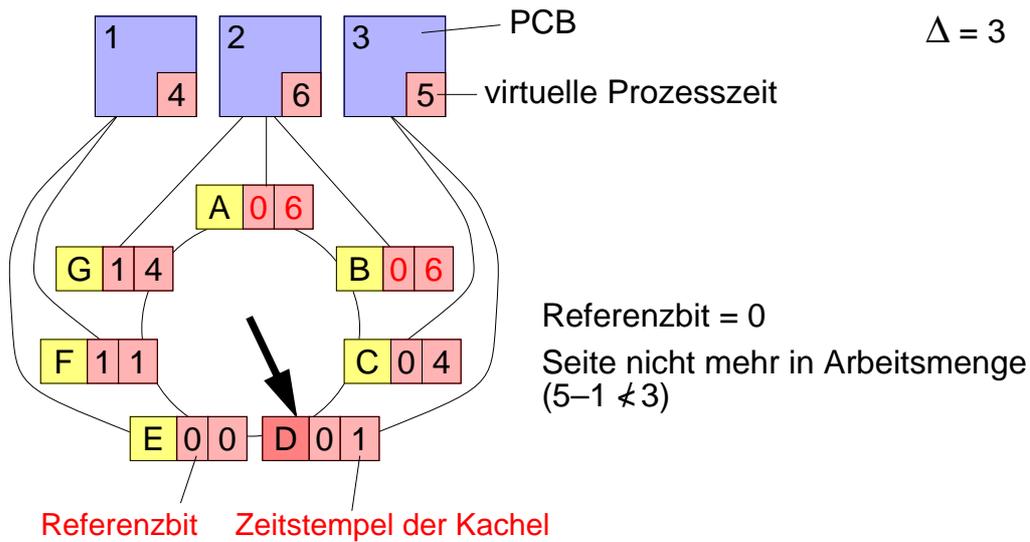
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



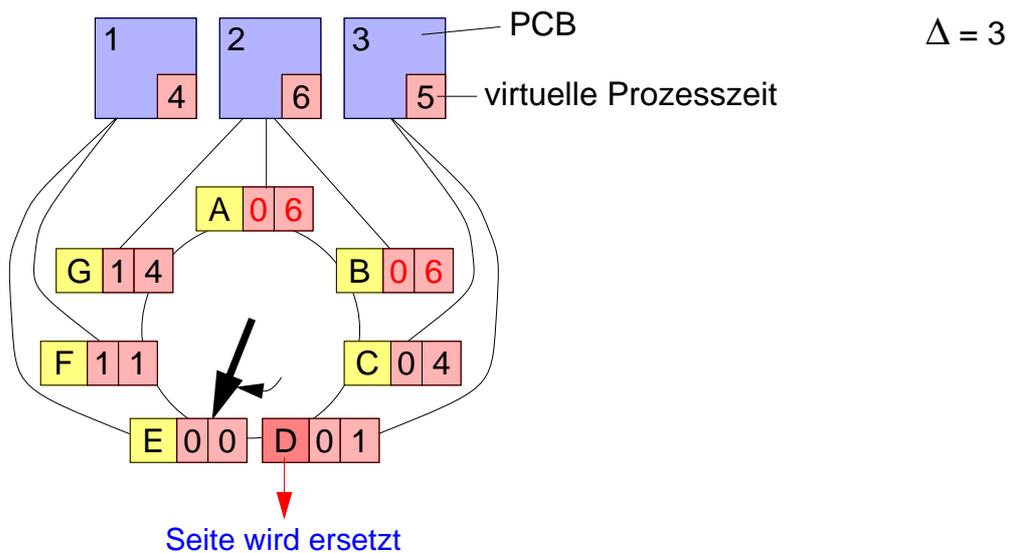
## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus



## 8.4 Arbeitsmengenbestimmung mit WSClock (2)

### ■ WSClock Algorithmus

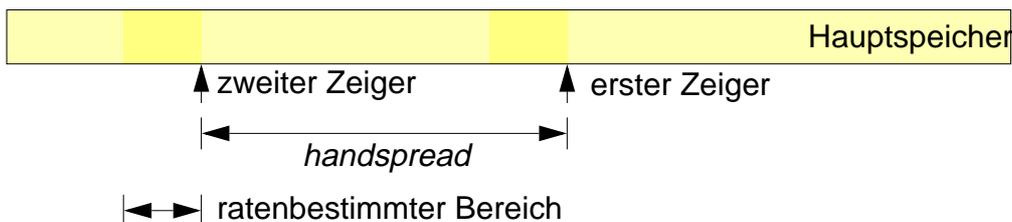


## 8.5 Probleme mit Arbeitsmengen

- ▲ Zuordnung zu einem Prozess nicht immer möglich
  - ◆ gemeinsam genutzte Seiten in modernen Betriebssystemen eher die Regel als die Ausnahme
    - Seiten des Codesegments
    - Shared libraries
    - Gemeinsame Seiten im Datensegment (*Shared memory*)
- ★ moderne System bestimmen meist eine globale Arbeitsmenge von Seiten

## 8.6 Ersetzungsstrategie bei Solaris

- Prozess *pageout* arbeitet Clock-Strategie ab
  - ◆ Prozess läuft mehrmals die Sekunde (4x)
  - ◆ adaptierbare Rate: untersuchte Seiten pro Sekunde
  - ◆ statt ein Zeiger: zwei Zeiger
    - am ersten Zeiger werden Referenzbits zurückgesetzt
    - am zweiten Zeiger werden Seiten mit gelöschtem Ref.-Bit ausgewählt
    - nötig, weil sonst Zeitspanne zwischen Löschen und Auswählen zu lang wird (großer Hauptspeicher; 64 MByte entsprechen 8.192 Seiten)
    - Zeigerabstand einstellbar (*handspread*)

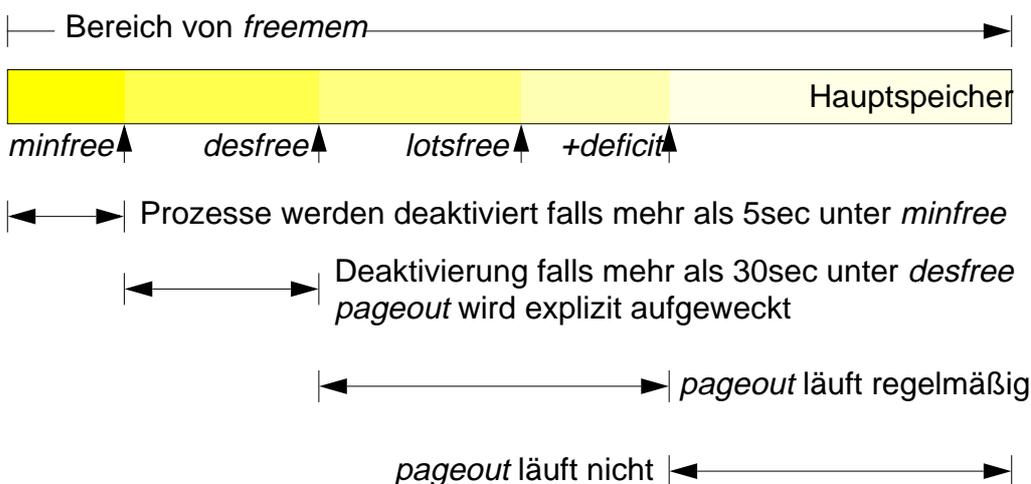


## 8.6 Ersetzungsstrategie bei Solaris (2)

- ◆ ermittelte Seiten werden ausgelagert (falls nötig) und
- ◆ in eine Freiliste eingehängt
- ◆ aus der Freiliste werden Kacheln für Einlagerungen angefordert
- ◆ Seitenfehler können unbenutzte Seiten aus der Freiliste wieder zurückfordern (*Minor page faults*)

## 8.6 Ersetzungsstrategie bei Solaris (3)

- Verhalten von *pageout* orientiert sich an Größe der Freiliste (Menge des freien Speichers)

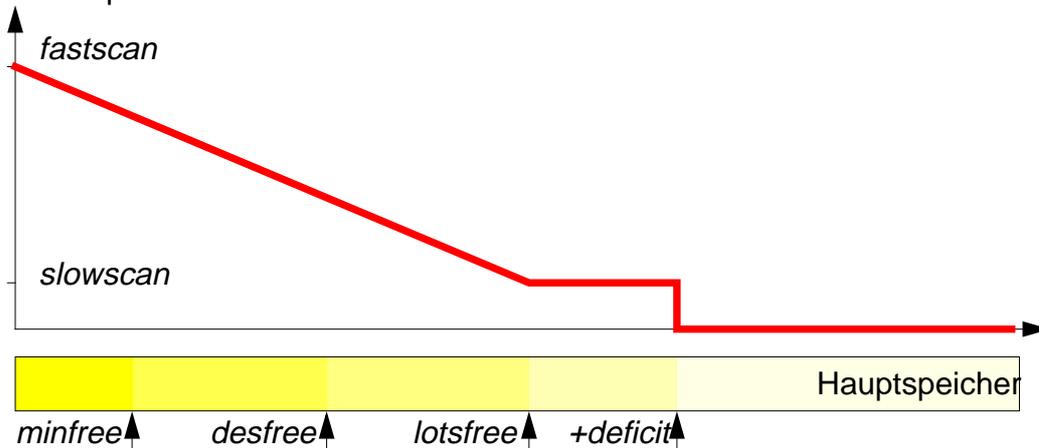


- ◆ *deficit* wird dynamisch ermittelt (0 bis *lotsfree*) und auf *lotsfree* addiert
  - entspricht Vorschau auf künftige große Speicheranforderungen

## 8.6 Ersetzungsstrategie bei Solaris (4)

- Seitenuntersuchungsrate des *pageout* Prozesses

Seiten pro Sekunde



- ◆ je weniger freier Speicher verfügbar ist, desto höher wird die Untersuchungsrate
- ◆ *slowscan* und *fastscan* sind einstellbar

## 8.6 Ersetzungsstrategie bei Solaris (5)

- Weitere Parameter
  - ◆ *maxpgio*: maximale Transferrate bei Auslagerungen (vermeidet Plattensaturierung)
  - ◆ *autoup*: Zeitdauer des regelmäßigen Auslagerns alter Seiten durch den Prozess *flushd* (Default: alle 30 sec)
- Aktivieren und Deaktivieren (*Swap in*, *Swap out*)
  - ◆ Auswahl wird dem Scheduler überlassen
  - ◆ Deaktivierung wird lediglich von Speicherverwaltung angestoßen

## 8.6 Ersetzungsstrategie bei Solaris (6)

- Typische Werte
  - ◆ *minfree*: 1/64 des Hauptspeichers (Solaris 2.2), 25 Seiten (Solaris 2.4)
  - ◆ *desfree*: 1/32 des Hauptspeichers (Solaris 2.2), 50 Seiten (Solaris 2.4)
  - ◆ *lotsfree*: 1/16 des Hauptspeichers (Solaris 2.2), 128 Seiten (Solaris 2.4)
  - ◆ *deficit*: 0 bis *lotsfree*
  - ◆ *fastscan*: min( 1/4 Hauptspeicher, 64 MByte ) pro Sekunde (Solaris 2.4)
  - ◆ *slowscan*: 800 kBytes pro Sekunde (Solaris 2.4)
  - ◆ *handspread*: wie *fastscan* (Solaris 2.4)

## 8.7 Ersetzungsstrategie bei Windows 2000

- Freiliste von freien Kacheln
- Arbeitsmenge pro Prozess
  - ◆ zunächst vorbestimmt
  - ◆ Anpassung der Arbeitsmenge durch Working-Set-Manager
  - ◆ Arbeitszyklus des Working-Set-Managers wird durch Speicherknappheit beschleunigt
  - ◆ Auslagerungsstrategie nach WSClock (nur Monoprocessorvariante) unter Berücksichtigung von Prozessklassen
  - ◆ prozessspezifische Konfiguration der Arbeitsmenge durch Anwender

## 9 Zusammenfassung

---

- Freispeicherverwaltung
  - ◆ Speicherrepräsentation, Zuteilungsverfahren
- Mehrprogrammbetrieb
  - ◆ Relokation, Ein- und Auslagerung
  - ◆ Segmentierung
  - ◆ Seitenadressierung, Seitenadressierung und Segmentierung, TLB
  - ◆ gemeinsamer Speicher
- Virtueller Speicher
  - ◆ Demand paging
  - ◆ Seiteneretzungsstrategien: FIFO, B<sub>0</sub>, LRU, 2nd chance (Clock)
- Seitenflattern
  - ◆ Super-Zustände, Arbeitsmengenmodell