Embedded Systems Architecture

Technische Grundlagen der Informatik 2

Rechnerorganisation

Kapitel 7: Speicherhierarchie
Teil B: Virtueller Speicher

Prof. Dr. Ben Juurlink

Fachgebiet: Architektur eingebetteter Systeme
Institut für Technische Informatik und Mikroelektronik
Fak. IV – Elektrotechnik und Informatik

SS 2014

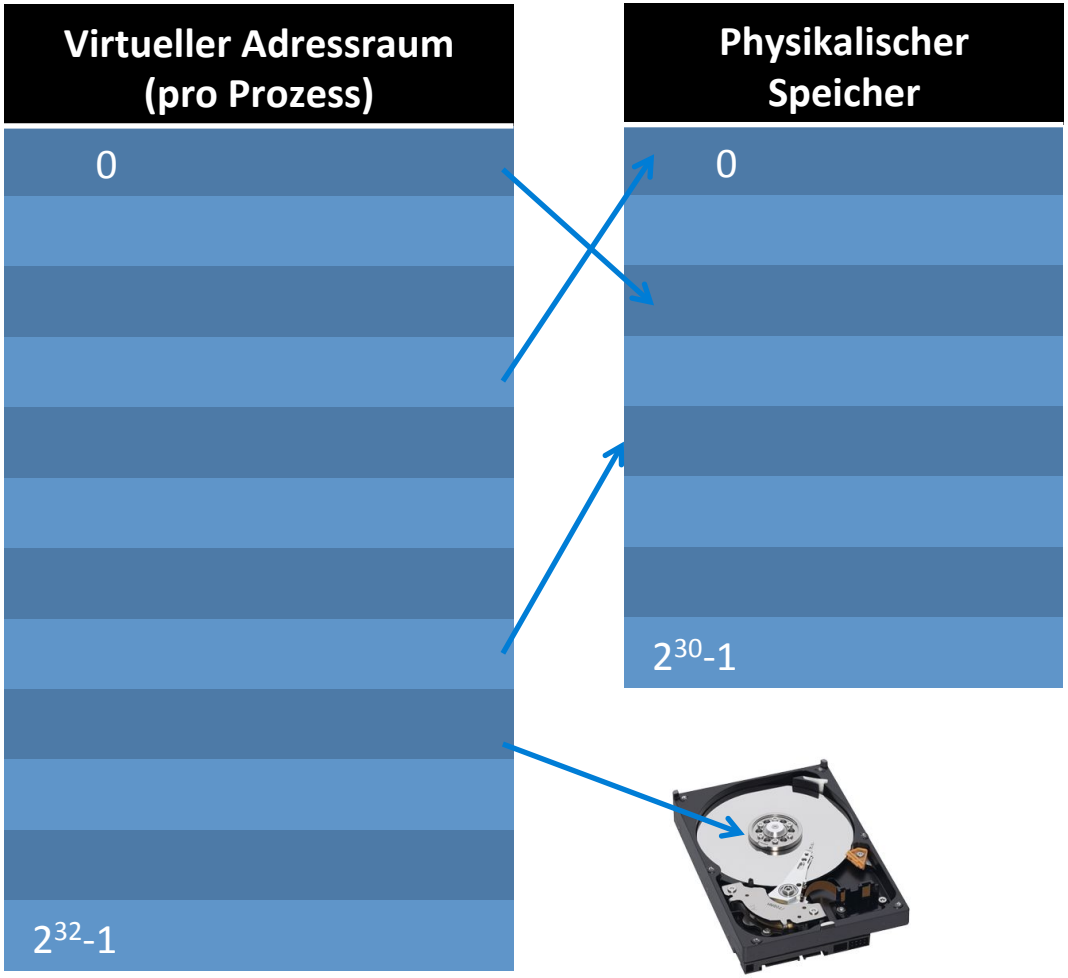
Ziele

Nach dieser Vorlesung sind Sie in der Lage

- Folgende Begriffe zu erklären: Virtueller Speicher, virtuelle/physikalische Adresse, Seitenfehler, Adressübersetzung, Seitentabelle, *Translation-Lookaside Buffer* (Übersetzungspuffer), ...
- virtuelle Adressen in physikalische Adressen zu übersetzen
- zu skizzieren, wie Schutzmechanismen im virtuellen Speichersystem implementiert werden
- Gegeben Eigenschaften eines virtuellen Speichersystems, Größe der Seitentabelle zu berechnen
- anzugeben, wann TLB-Zugriff und Cache-Zugriff parallel stattfinden können
- Zu berechnen wie viele Seitenfehler auftreten

Virtueller Speicher - Motivation

- MIPS32 kann 2^{32} = 4GB-Speicher adressieren.
- Früher hatten Computer nicht solch großen Speicher.
  - Heutzutage wird 4GB-Marke erreicht, aber mehrere Prozesse gleichzeitig aktiv
- 64-Bit Architekturen können 2^{64}-Byte Speicher adressieren
- Idee: verwende Hauptspeicher als „Cache“ für Sekundärspeicher (z. B. Festplatte)

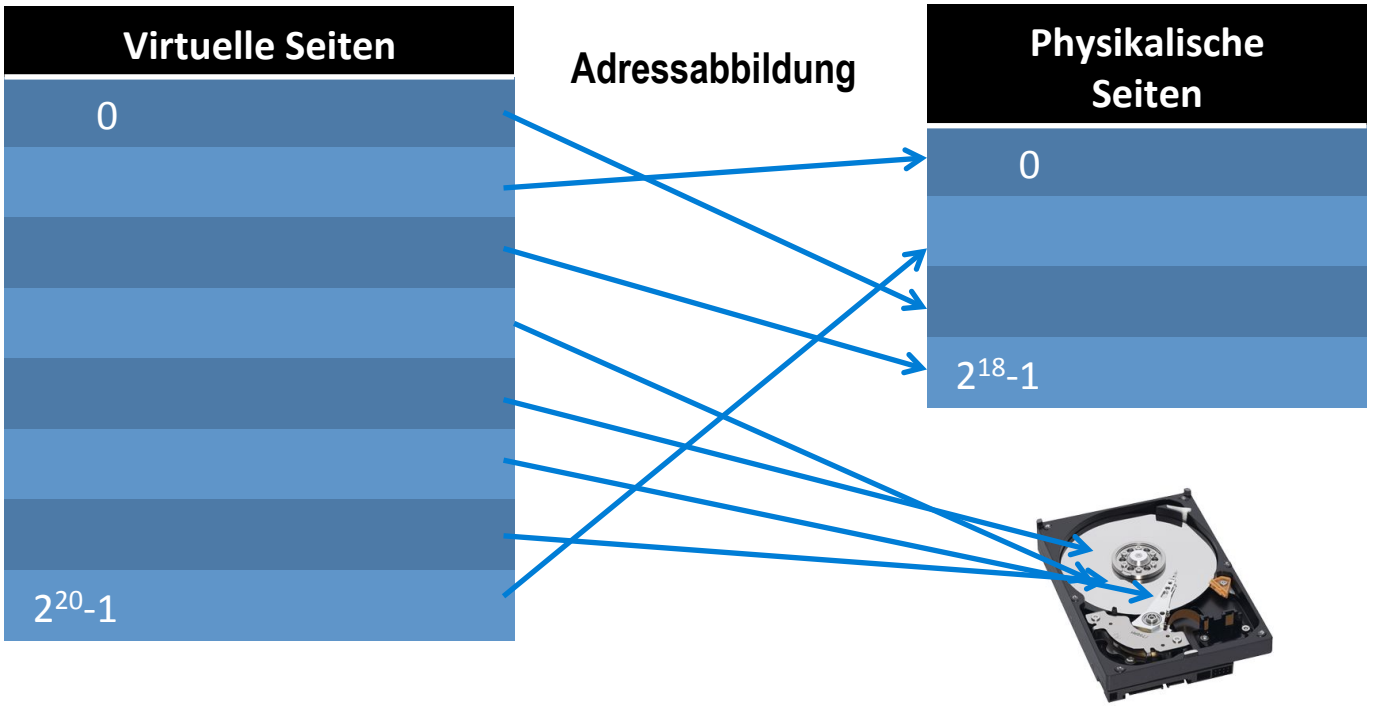


Virtueller Speicher

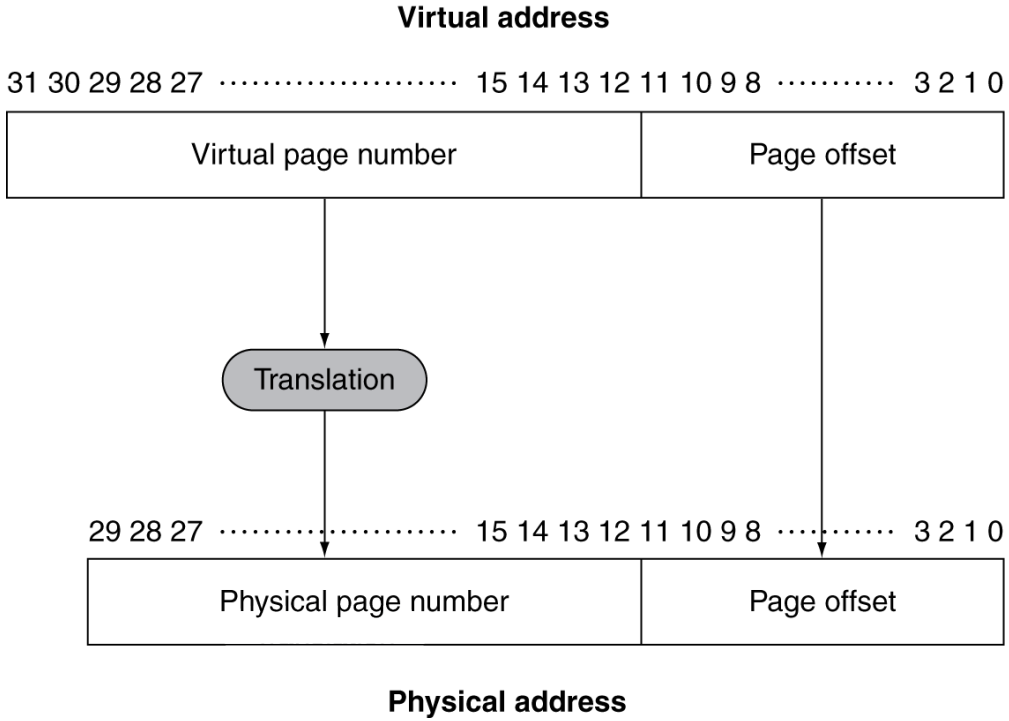
- Virtueller Speicher (*virtual memory*): Technik, die Hauptspeicher als Cache für Sekundärspeicher verwendet
- Vorteile:
  - Programme können größer sein als Hauptspeicher (physikalischer Speicher)
    - früher musste Programm passend machen durch Überlagerungen (*overlays*)
  - Schutzmechanismus (*protection*): sichere gemeinsame Nutzung des Speichers durch mehrere Prozesse
    - Adressübersetzung/-abbildung sorgt dafür, dass Prozess nicht auf Speicherpositionen eines anderen zugreifen kann

Adressübersetzung/-abbildung

- virtueller Adressraum wird in Blöcke genannt Seiten (*pages*) aufgeteilt
- physikalischer Speicher wird ebenfalls in Seiten aufgeteilt
- Seitentabelle (*page table*) bildet virtuelle Seiten auf physikalische Seiten ab



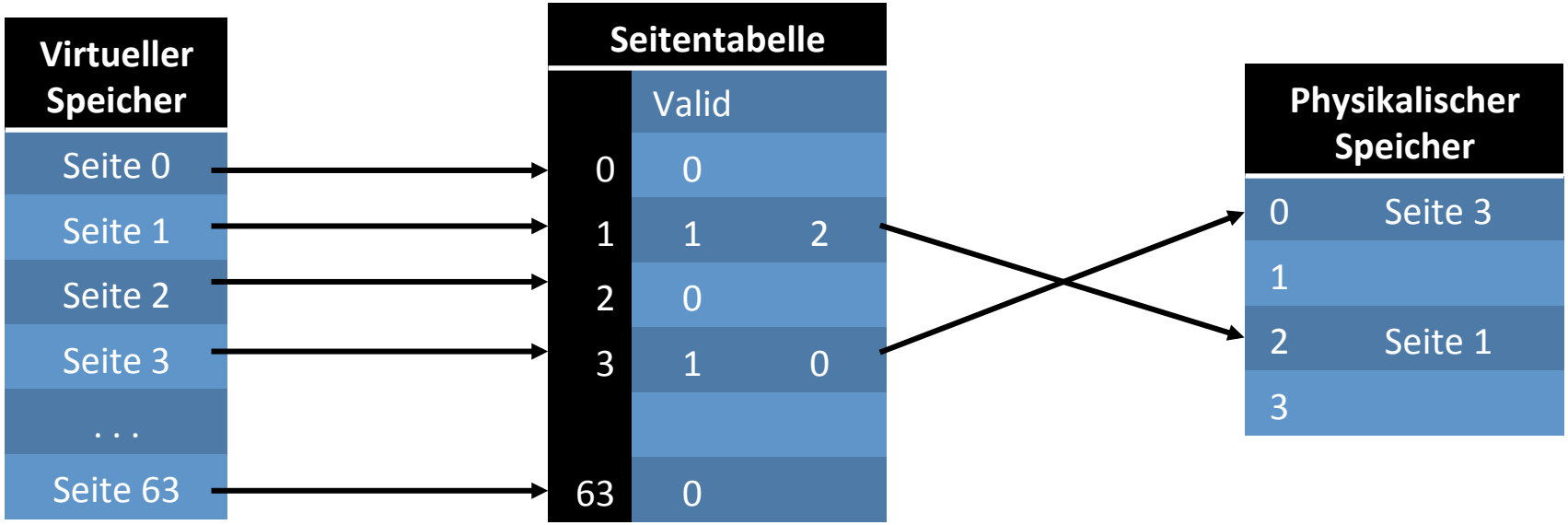
- Virtuelle Adresse: Seiten-Offset ($\log_2$(Seitengröße) unteren Bits) und virtuelle Seitennummer
- Physikalische Adresse: Seiten-Offset und physikalische Seitennummer
- Seitentabelle bildet virtuelle Seitennummer auf physikalische Seitennummer ab

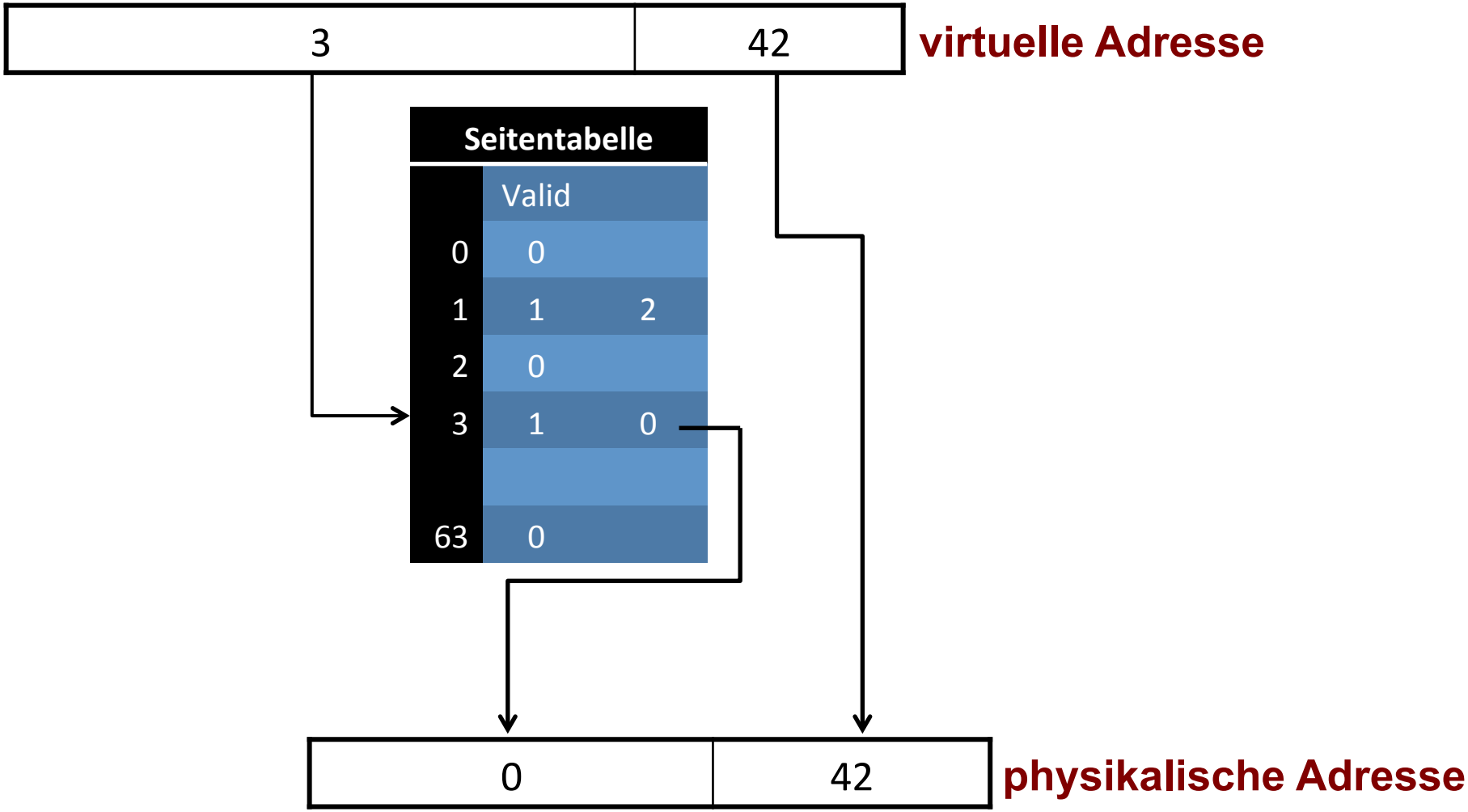


Beispiel:

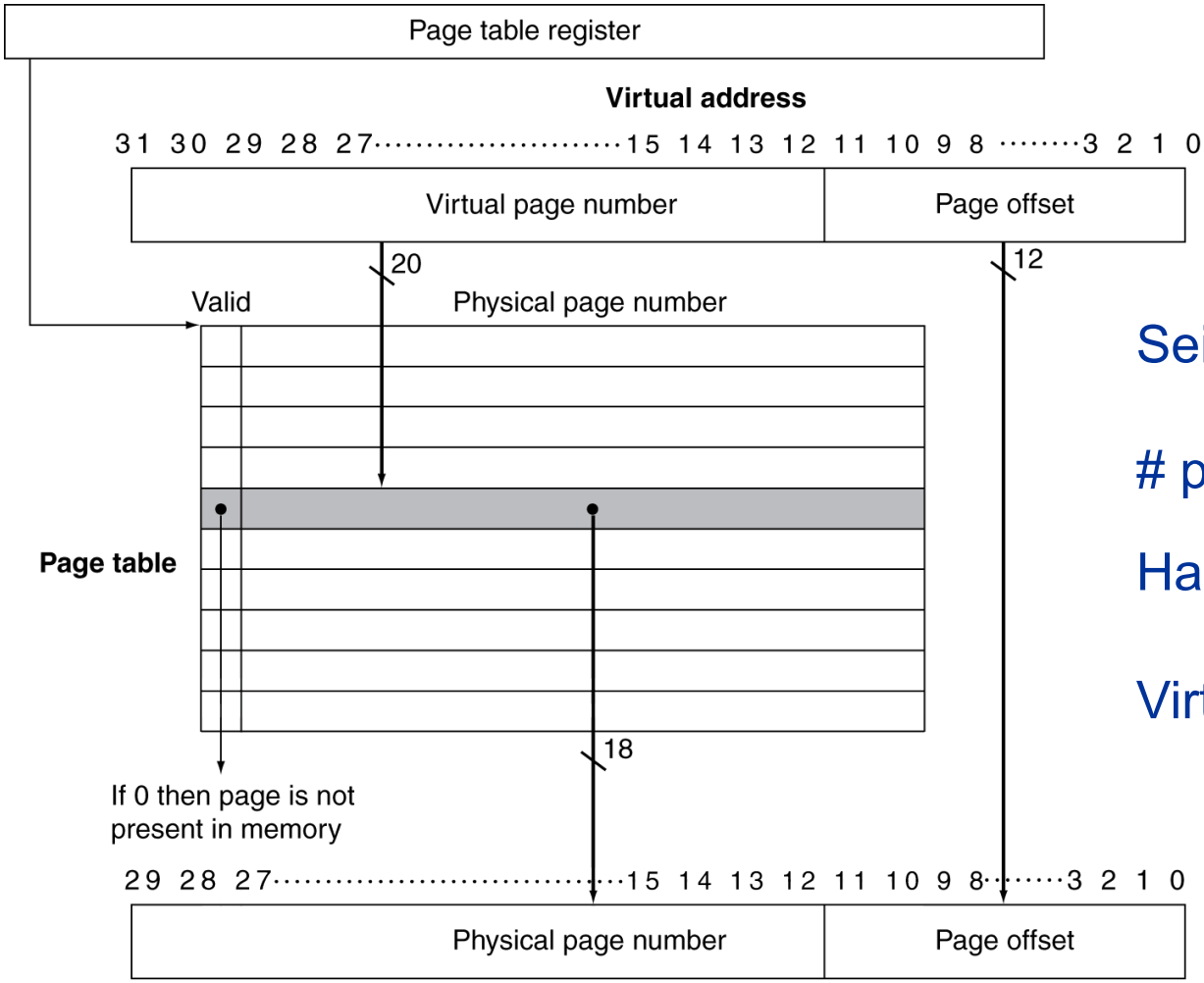
- Seitengröße = 2^{12} = 4KB
- Größe des physikalischen Speichers = 2^{30} = 1GB
- Anzahl virtuelle Seiten = 2^{20}
- Anzahl physikalische Seiten = 2^{18}

- Seitentabelle hat ein Eintrag pro virtuelle Seite. In Eintrag *j* steht u. a.
  - Ob virtuelle Seite *j* im physikalische Speicher steht (Gültigkeits-Bit)
  - wenn ja, wo (physikalische Seitennummer). Wenn nein, wo im Sekundärspeicher
- Jeder Prozess (Programm in Ausführung) hat eigene Tabelle
  - Seitentabellenregister (*page table register*) zeigt auf Tabelle des aktuellen Prozesses




3
42
virtuelle Adresse
Seitentabelle
Valid
0
0
1
1
2
2
0
3
1
0
63
0
0
42
physikalische Adresse

Übersetzung mittels Seitentabelle



Seitengröße = 2^{12} = 4KB

physikalische Seiten = 2^{18}

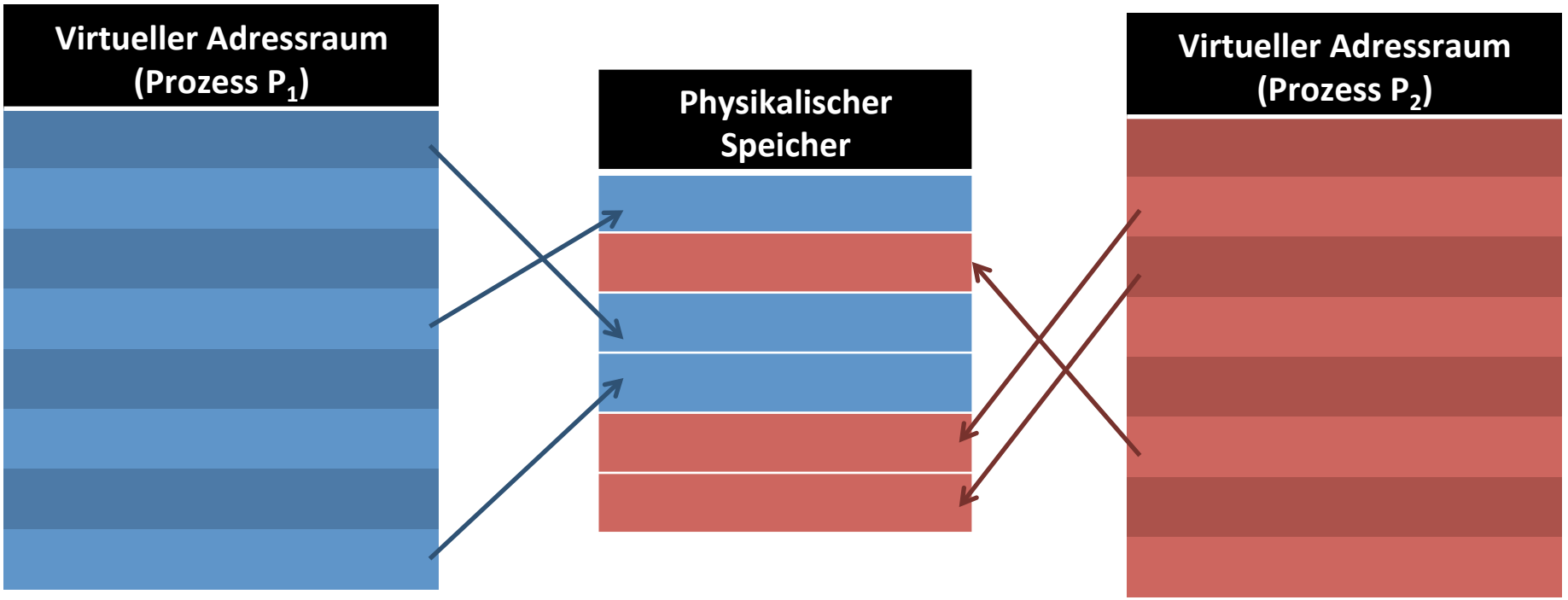
Hauptspeicher $\leq$ 1GB

Virtueller Adressraum = 4GB

- Kosten eines Fehlzugriffs (Seitenfehler, *page fault*) enorm
  - Mehrere Millionen Taktzyklen

- Designentscheidungen:
  - Seiten groß genug um Latenz zu amortisieren
    - Seitengrößen von 4 KB bis 16 KB üblich
  - Seitenfehlerrate reduzieren äußerst wichtig
    - Vollassoziativ (durch Seitentabelle) und LRU Ersetzung
  - Seitenfehler werden in Software (Betriebssystem) abgehandelt
    - Kontextwechsel (*context switch*) zum anderen Prozess
  - Rückschreibetechnik da Durchschreibetechnik zu teuer

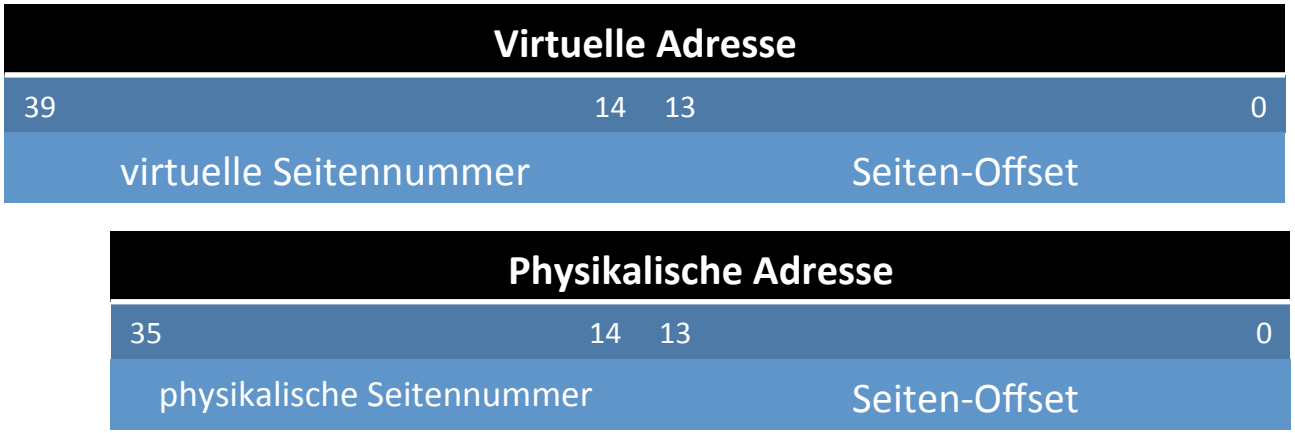
- Da jeder Prozess nur via Seitentabelle auf physikalischen Speicher zugreifen kann, bietet virtueller Speicher gleichzeitig Schutz zwischen Prozessen
- Benutzerprozess darf Seitentabelle lesen aber nicht ändern



- Schutzmechanismen müssen sicherstellen, dass Prozess nicht (un)beabsichtigt in Adressraum anderes Prozesses schreiben kann
- Notwendige Hardwareunterstützung:
  - zwei Modi
    - Benutzer-Modus (*user mode*)
    - Kernel-/Supervisor-Modus (*kernel/supervisor mode*)
  - Benutzerprozess kann Teil des Prozessorstatus lesen aber nicht schreiben:
    - Benutzer/Supervisor-Modus-Bit
    - Seitentabellenregister (*page table register*)
    - können nur durch spezielle Befehle im Kernel-Modus geschrieben werden
  - Mechanismen um zwischen Benutzermodus und Kernel-Modus zu wechseln
    - Systemaufruf-Ausnahme (*system call exception*) übertragt Steuerung an bestimmte Position im Supervisor-Coderaum (`syscall` in MIPS)
    - ERET (*exception return*) Befehl kehrt zurück in Benutzermodus und springt zur Adresse gespeichert im EPC (*exception program counter*)

Größe der Seitentabelle

- Virtuelles Speichersystem mit folgende Eigenschaften
  - 40-Bit virtuelle Byteadressen
  - 16KB Seiten
  - 36-Bit physikalische Byteadressen
- Wie groß ist der Seitentabelle wenn Gültigkeits-, Schutz-, Dirty- und Referenz-Bit insgesamt 4 Bit einnehmen und alle virtuelle Seiten verwendet werden?

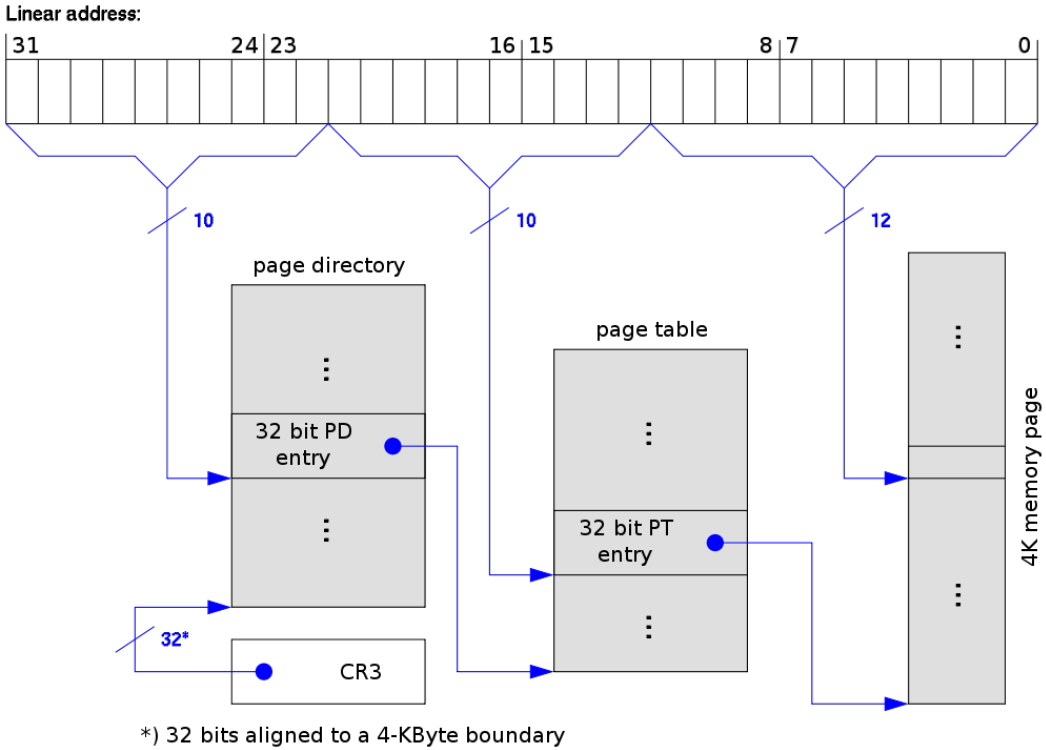


Größe der Seitentabelle

- Virtuelles Speichersystem mit folgende Eigenschaften
  - 40-Bit virtuelle Byteadressen
  - 16KB Seiten
  - 36-Bit physikalische Byteadressen
- Wie groß ist der Seitentabelle wenn Gültigkeits-, Schutz-, Dirty- und Referenz-Bit insgesamt 4 Bit einnehmen und alle virtuelle Seiten verwendet werden?
  - 16KB = 2^{14} Byte Seiten → 14 Bit für Seiten-Offset
  - Virtuelle Seitennummer: 26 Bit
  - Physikalische Seitennummer: 22 Bit
  - Seitentabelle hat 1 Eintrag pro virtuelle Seite → 2^{26} Einträge
  - Jeder Eintrag 4 Bit für G/S/D/R + 22 Bit für physikalische Seitennummer → 26 Bit → 4 Byte
  - 2^{26} Einträge x 4 Byte = 2^{28} Byte = 256 MB (!)

Größe der Seitentabelle

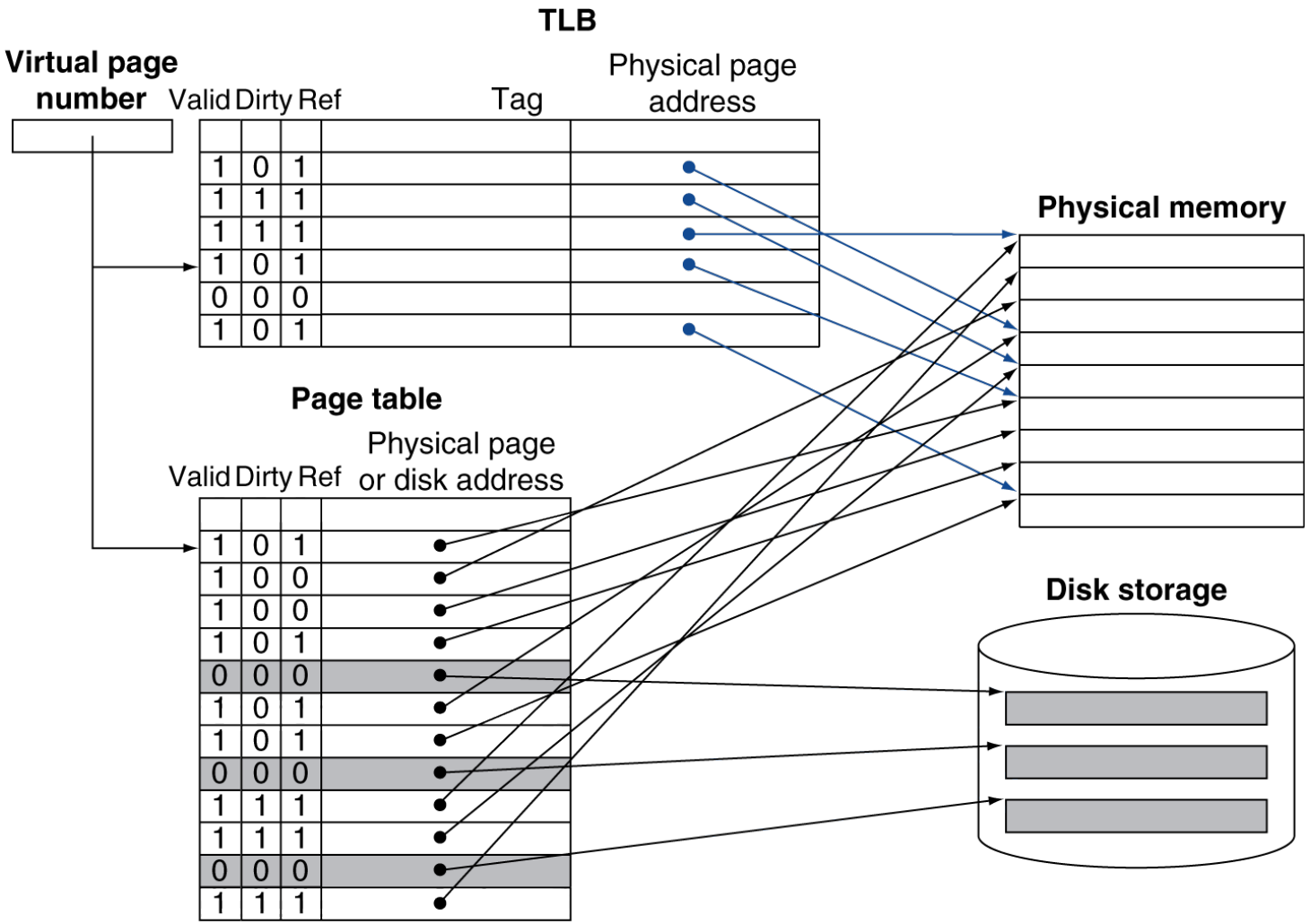
- Lösungsansätze
  - Größe der Seitentabelle einschränken. Wenn Prozess wachst, neue Eintrage hinzufügen
  - Hash-Funktion anwenden, damit man nur Anzahl *physikalischer Seiten* Einträge braucht (*invertierte Seitentabelle*)
  - Seitentabelle selbst in Seiten aufteilen und teilweise auf Festplatte speichern
  - Hierarchische (mehrstufige) Seitentabellen



Adressübersetzung Beschleunigen: TLB

- Seitentabellen im Hauptspeicher untergebracht
- Jeder Speicherzugriff →2 Speicherzugriffe?
  - Zugriff, um physikalische Adresse zu ermitteln
  - Zugriff, um Befehl / Daten zu erhalten
- Nutze Lokalität Zugriffe auf Seitentabelle
  - auf gleiche Seite wird häufig hintereinander zugegriffen
- *Translation-lookaside buffer* (TLB)
  - Spezieller Cache für Adressübersetzung
  - TLB-Eintrag enthält
    - Tag: (Teil der) virtuelle Seitennummer
    - physikalische Seitennummer
    - Gültigkeits- und Referenzbit (wird für LRU-Implementierung verwendet)
    - Bits für Schutz

Schnelle Übersetzung mittels TLB

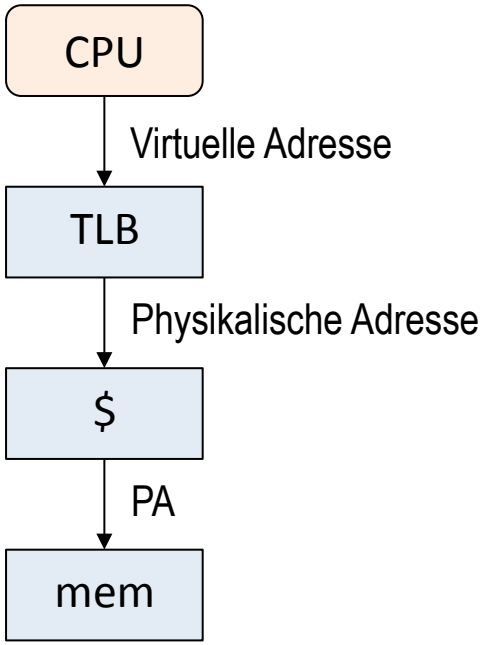


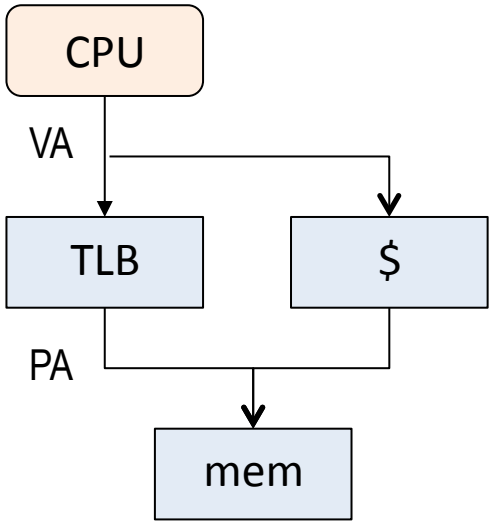
- Typische Werte:
  - 16–512 Einträge
  - hohe Assoziativität
  - Trefferzeit: 0.5–1 Zyklen
  - Fehlzugriffsaufwand: 10–100 Zyklen
  - Fehlerrate: 0.01%–1%

Prozessor	Seitengröße	I-TLB	D-TLB
Pentium	4 KB	32 Einträge (4-fach SA)	64 Einträge (4-fach SA)
Pentium II	4 KB	32 Einträge (4-fach SA)	64 Einträge (4-fach SA)
Pentium III	4 KB	32 Einträge (4-fach SA)	64 Einträge (4-fach SA)
Pentium IV	4 KB	64 Einträge (4-fach SA)	128 Einträge (4-fach SA)
Core Duo	4 KB	64 Einträge (VA)	64 Einträge (VA)

Interaktion vom TLB und Cache

- Wir nehmen an Cache ist physikalisch indiziert (*physically indexed*) und physikalisch getagged (*physically tagged*)
- Muss Cache-Zugriff nach TLB-Zugriff stattfinden oder können parallel?





- Cache- und TLB-Zugriff können parallel stattfinden wenn Seiten-Offset > Block-Offset + Cache-Index
  - Tag-vergleich findet nach Zugriff statt

Seitennummer	Seiten-Offset	
Tag	Index	Block-Offset

  - Leider (gerade) nicht so
    - Typische Seitengröße: 4 oder 8KB → Seiten-Offset 12 oder 13 Bit
    - Core Duo I$: 32 KB, 2-fach SA, 64-Byte Blöcke → #Sätze x Blockgröße = Cache-Größe/Assoziativität = 16KB → Index + Block-Offset 14 Bit
  - Lösungen
    - Virtuell adressierter Cache
      - jede Menge andere Probleme
    - Einschränken welche virtuelle Seiten auf welche physikalische Seiten abgebildet werden können (z. B. letzter Bit unverändert)

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite														
LRU Seite														
Fehler?														

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7													
LRU Seite														
Fehler?	x													

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0												
		7												
LRU Seite														
Fehler?	x	x												

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1											
		7	0											
LRU Seite			7											
Fehler?	x	x	x											

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2										
		7	0	1										
LRU Seite			7	0										
Fehler?	x	x	x	x										

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0									
		7	0	1	2									
LRU Seite			7	0	1									
Fehler?	x	x	x	x										

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3								
		7	0	1	2	0								
LRU Seite			7	0	1	2								
Fehler?	x	x	x	x		x								

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0							
		7	0	1	2	0	3							
LRU Seite			7	0	1	2	2							
Fehler?	x	x	x	x		x								

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4						
		7	0	1	2	0	3	0						
LRU Seite			7	0	1	2	2	3						
Fehler?	x	x	x	x		x		x						

Seitenersetzung (*page replacement*)

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2					
		7	0	1	2	0	3	0	4					
LRU Seite			7	0	1	2	2	3	0					
Fehler?	x	x	x	x		x		x	x					

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2	3				
		7	0	1	2	0	3	0	4	2				
LRU Seite			7	0	1	2	2	3	0	4				
Fehler?	x	x	x	x		x		x	x	x				

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2	3	0			
		7	0	1	2	0	3	0	4	2	3			
LRU Seite			7	0	1	2	2	3	0	4	2			
Fehler?	x	x	x	x		x		x	x	x	x			

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2	3	0	3		
		7	0	1	2	0	3	0	4	2	3	0		
LRU Seite			7	0	1	2	2	3	0	4	2	2		
Fehler?	x	x	x	x		x		x	x	x	x			

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2	3	0	3	2	
		7	0	1	2	0	3	0	4	2	3	0	3	
LRU Seite			7	0	1	2	2	3	0	4	2	2	0	
Fehler?	x	x	x	x		x		x	x	x	x			

- Betriebssystem implementiert Seitenersetzungsalgorithmus
- Beispiel:
  - 3 Seiten passen im Speicher und Prozessor greift auf folgende Seiten zu:

    7 0 1 2 0 3 0 4 2 3 0 3 2 1
  - Wie viele Seitenfehler treten bei LRU-Ersetzung auf?
- Lösen durch „Keller-Algorithmus“ (*stack algorithm*) anzuwenden:

	7	0	1	2	0	3	0	4	2	3	0	3	2	1
MRU Seite	7	0	1	2	0	3	0	4	2	3	0	3	2	1
		7	0	1	2	0	3	0	4	2	3	0	3	2
LRU Seite			7	0	1	2	2	3	0	4	2	2	0	3
Fehler?	x	x	x	x		x		x	x	x	x			x

Fazit Virtueller Speicher

- Durch virtuellen Speicher sind Programme größer als physikalischer Speicher möglich
  - Hauptspeicher als „Cache“ für Sekundärspeicher verwenden
- Seitentabelle bildet virtuelle Adressen auf physikalische Adressen ab
- Außerdem bietet Seitentabelle Schutz (*protection*) zwischen Prozessen
- *Translation-lookaside buffer* (TLB) beschleunigt Adressübersetzung
  - Cache für Seitentabelleneinträge

Fazit Speicherhierarchie

- Caching allgemein nützliches Prinzip
  - Caches für Instruktionen und Daten
  - Hauptspeicher Cache für Sekundärspeicher
  - TLB Cache für Seitentabelleneinträge
  - Auch: Cache im Festplattencontroller, Suchmaschinen, ...
  - Auch im realen Leben anwendbar
- 4 grundlegende Fragen
  - Wo kann ein Block platziert werden?
  - Wie findet man einen Block?
  - Welcher Block wird ersetzt?
  - Was passiert bei Schreiboperationen?

Fazit Speicherhierarchie

- Wo kann ein Block platziert werden?
  - 1 Position (direkt abgebildet)
  - paar Positionen (satzassoziativ)
  - beliebige Position (vollassoziativ)
- Wie findet man einen Block?
  - Tag + Indizierung (direkt abgebildet)
  - Tag + Indizierung Menge + Suche innerhalb Menge (satzassoziativ)
  - Tag + vollständiges Durchsuchen aller Einträge (vollassoziativ)
  - separate Suchtabelle (vollassoziativ, wie Seitentabelle)
- Welcher Block wird ersetzt?
- Was passiert bei Schreiboperationen?

Fazit Speicherhierarchie

- Wo kann ein Block platziert werden?
- Wie findet man einen Block?
- Welcher Block wird ersetzt?
  - Zufällig
  - LRU-Ersetzung (am längsten nicht genutzt)
  - LRU-Approximationen (Pseudo-LRU)
- Was passiert bei Schreiboperationen?
  - Durchschreiben (*write-through*)
  - Rückschreiben (*write-back*)