HAW HAMBURG

Betriebssysteme BS-ITS

4. Speicherverwaltung

1. ***Einführung und Grundlagen***
2. *Swapping*
3. *Virtuelle Speicherverwaltung*
4. *Seitenersetzungsverfahren*
5. *Entwurfsaspekte*
6. *UNIX*

Übersicht Speicherhierachie

Allgemein

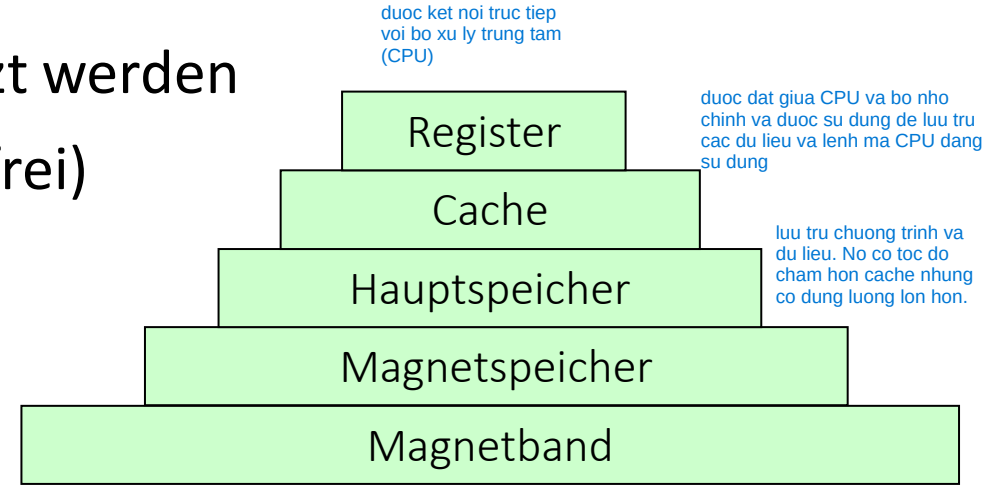
Phan Bo?

- Die Speicherverwaltung des BS organisiert die Vergabe des Hauptspeichers im Rahmen der Speicherhierarchie

  trong khung phan cap cua bo nho

Die Speicherverwaltung

- verfolgt, welche Speicherbereiche gerade benutzt werden
- teilt Prozessen Speicher zu (und gibt ihn wieder frei)
- Adressumrechnung
- Auslagerung von Speicher auf die Festplatte



Anforderungen

chay nhieu chuong trinh cung mot luc, moi chuong trinh co the co dia chi bat dau khac nhau

- Verschiebbarkeit (Relokation) von Programmen (→ Adressberechnung)

cach ly cac tien trinh voi nhau. Dieu nay la can thiet de ngan chan cac tien trinh can thiep vao hoat dong cua nhau.

- Kapselung jedes Prozesses (Schutz und Zugriffskontrolle)

cap phat va giai phong bo nho chinh. Dieu nay la can thiet de he dieu hanh co the quan ly hieu qua bo nho chinh cua he thong.

- Automatische Zuweisung und Freigabe von Hauptspeicher

tu dong chuyen cac tien trinh ra khoi bo nho chinh. Dieu nay la can thiet de he dieu hanh co the giai phong bo nho cho cac tien trinh khac can thiet.

- Automatische Auslagerung von Prozessen (Scheduling!)

ho tro viec chia se tai nguyen giua cac tien trinh. Dieu nay la can thiet de cac tien trinh co the hop tac voi nhau va chia se du lieu va tai nguyen.

- Gemeinsame Nutzung (z.B. Shared Memory, DLLs)

Physikalische und logische Adressen

Physikalische Adressen (reale Adressen)

- Die absoluten Adressen im physikalischen Hauptspeicher

Logische Adressen (virtuelle Adressen)

- Logische Adressen werden in den Prozessen verwendet.
- Referenz auf eine Speicheradresse, ohne dass die reale (absolute) Hauptspeicheradresse bekannt ist.
- Eine „Übersetzung“ muss vom System (Betriebssystem oder Hardware) vorgenommen werden.

Dia chi logic khong phai la dia chi tuyet doi cua mot vi tri trong bo nho, ma la mot dia chi tuong doi.

Frage Warum unterscheidet man zwischen logischen und physikalischen Adressen?

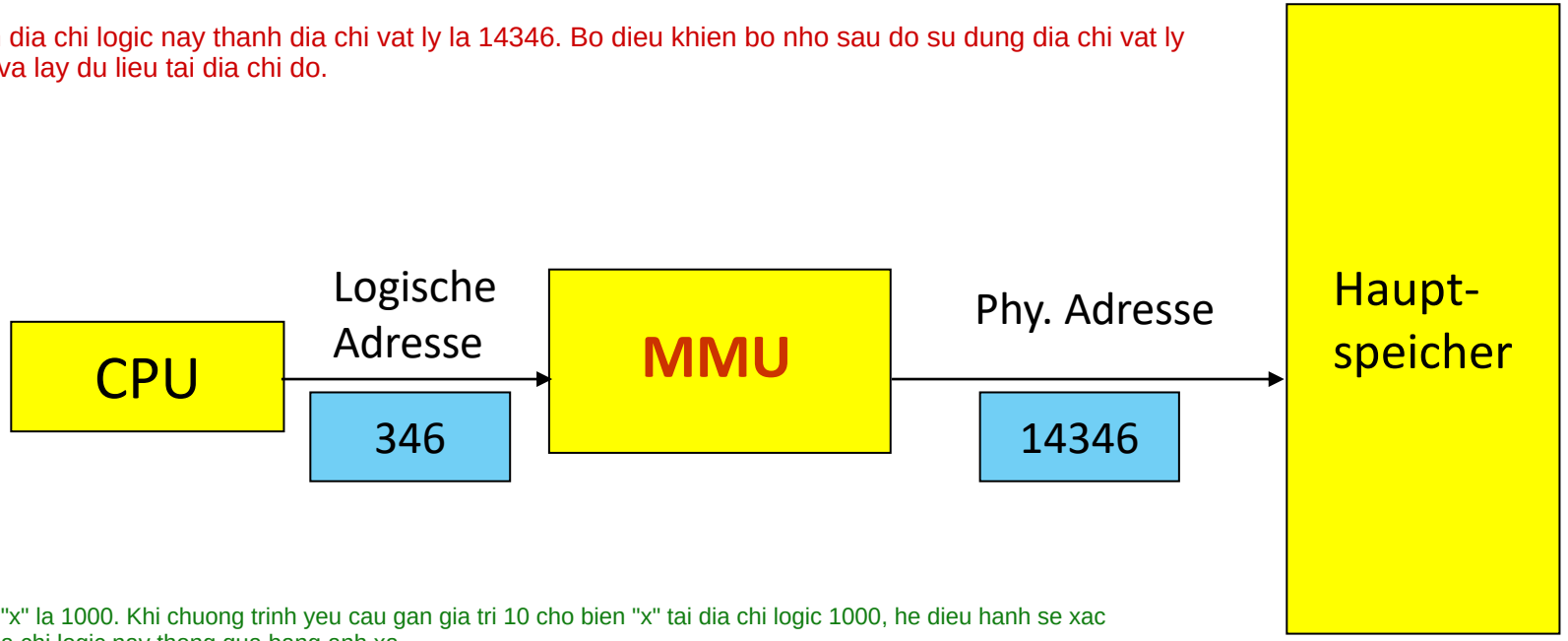
de bao ve bo nho, de ho tro da nhiem, de nang cao hieu suat

Dynamische Adressberechung zur Relokation von Programmen

MMU (Memory Management Unit): Abbildung einer logischen Adresse auf die physikalische Adresse (reale Adresse) durch Hardware (erste Näherung)

Beispiel

dia chi logic la 346. MMU dich dia chi logic nay thanh dia chi vat ly la 14346. Bo dieu khien bo nho sau do su dung dia chi vat ly nay de truy cap bo nho chinh va lay du lieu tai dia chi do.



Vi du, gia su dia chi logic cua bien "x" la 1000. Khi chuong trinh yeu cau gan gia tri 10 cho bien "x" tai dia chi logic 1000, he dieu hanh se xac dinh dia chi vat ly tuong ung cua dia chi logic nay thong qua bang anh xa.

Laden eines Programms in den Hauptspeicher

bien dich/trinh lap rap se gan cac thanh phan cua chuong trinh voi cac dia chi logic

- Compiler/Assembler verknüpfen üblicherweise einzelne Programmelemente mit logischen Adressen. Bezüge zwischen Modulen werden vorerst offen gelassen.

ghep cac mo-dun rieng le lai voi nhau bang cach • di chuyen cac phan vung dia chi rieng le so voi nhau de chung khong bi chong cheo, • giai quyet cac tham chieu cheo giua cac mo-dun

- Linker / Binder fügen die einzelnen Module zusammen, indem sie
  - die einzelnen Teil-Adressräume gegeneinander verschieben, so dass sie sich nicht überdecken,
  - Querbezüge zwischen den Modulen auflösen.

Di chuyen cac phan vung dia chi rieng le so voi nhau de chung khong bi chong cheo

Trinh lien ket se giai quyet cac tham chieu cheo nay bang cach thay the dia chi logic cua vi tri duoc tham chieu bang dia chi logic cua vi tri do trong mo-dun noi no duoc dinh nghia.

- Es entsteht ein einheitlicher Adressraum, dessen Adressen aber noch nicht den physikalischen Adressen entsprechen müssen.

Hinh thanh mot khong gian dia chi thong nhat, cac dia chi cua khong gian dia chi nay van chua nhat thiet phai tuong ung voi dia chi vat ly

- Die Adressen dieses Adressraumes werden entweder statisch durch den Linker/Loader oder dynamisch während der Ausführung des Programms auf die physikalischen Adressen abgebildet.

0xFF: dia chi static / dynamisch <---abbildet--- 0x01: dia chi cua Adresseraume 1 gom 0x02, 0x03, 0x04,.....

- Hierdurch wird eine Unabhängigkeit der Programme von ihrer Platzierung im Hauptspeicher erreicht (**Relokation**)

Einfache Adressumsetzungmethode (Relokation)

Aufgabe

- Abbildung der logischen Adressen eines Prozesses auf physikalische Adressen

Basis-Register („Relocation“-Register)

- Enthält die Startadresse (phy. Adresse) des Prozesses
- Berechnungsverfahren: Physikalische Adresse= Logische Adresse + Wert des Basis-Registers

0x hexzahl

erhalt wert-variable

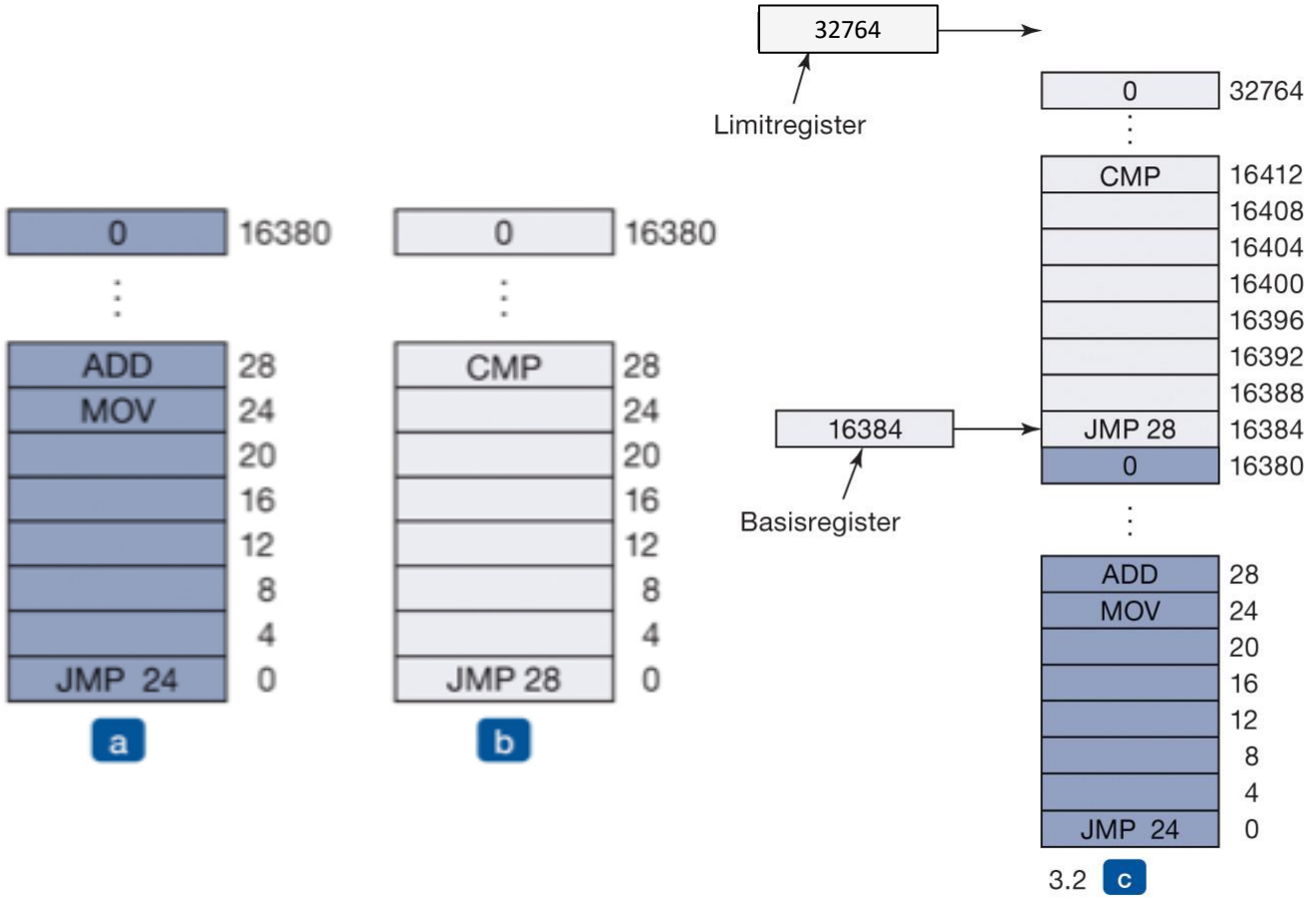
Limit-Register:

- Endadresse (phy. Adresse) des Prozesses
- Wenn Phy. Adresse > Limit-Register ➔ Fehler! (Schutzverletzung)

Register duoc dat la 10000, duoc phep truy cap vao cac dia chi bo nho tu 0 den 10000

- Diese Register werden gesetzt, wenn der Prozess geladen oder verschoben wird!
- Umrechnung bei jedem Speicherzugriff (Sprungadressen, holen von Befehlen, ...)

Beispiel



JMP 28
➔JMP 28 + 16384
= JMP 16412

ADD R1, 20000
➔ADD R1, 36384
Zugriffsverletzung

[AT] **Abbildung 3.3:** Basis- und Limitregister können benutzt werden, um jedem Prozess einen separaten Adressraum zu geben.

HAW HAMBURG

Betriebssysteme BS-ITS

4. Speicherverwaltung

1. *Einführung und Grundlagen*
2. ***Swapping***
3. *Virtuelle Speicherverwaltung*
4. *Seitenersetzungsverfahren*
5. *Entwurfsaspekte*
6. *UNIX*

Swapping vs virtueller Speicher

Qua trinh Swapping thuong duoc thuc hien khi khong gian bo nho RAM(Hauptschpeicher) khong du de chua tat ca cac qua trinh dang hoat dong. He dieu hanh se quyet dinh chuyen doi cac qua trinh khong hoat dong ra khoi RAM de giai phong bo nho cho cac qua trinh khac.

Allgemein

- Ohne Speicherverwaltung: Der Hauptspeicher muss den Speicherbedarf aller (quasi-)parallel laufenden Prozesse abdecken.
- Aber: Oftmals nicht genug Hauptspeicher für alle aktiven Prozesse → einige müssen auf Festplatte ausgelagert und bei Bedarf dynamisch in den Hauptspeicher geladen werden

Khi mot qua trinh can truy cap den mot phan cua bo nho khong nam trong RAM, he dieu hanh se su dung bang anh xa (page table hoac segment table) de chuyen doi dia chi ao cua qua trinh sang dia chi vat ly tuong ung trong RAM hoac dia cung. Bang anh xa nay giu thong tin ve cac trang (pages) hoac doan (segments) cua qua trinh va vi tri cua chung trong bo nho.

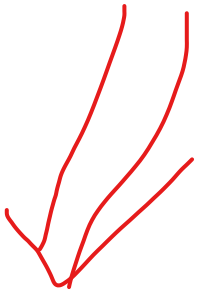
Swapping

- Jeder Prozess wird komplett in einen zusammenhängenden Speicherbereich geladen.
- Wenn ein neuer Prozess geladen werden muss und nicht genug Hauptspeicher (am Stück) zur Verfügung steht, dann werden ein oder mehr Prozesse komplett vom Hauptspeicher auf Platte ausgelagert. hoan doi sang o cung
- Wenn ein ausgelagerter Prozess wieder zur Verarbeitung ansteht, dann wird er wieder komplett geladen.

Frage: Was passiert, wenn der Hauptspeicherbedarf eines Prozesses größer als der Hauptspeicher selbst ist?

Virtueller Speicher

- Prozesse laufen auch dann, wenn sich nur ein Teil von ihnen im Hauptspeicher befindet.
- Dieser Teil muss nicht zuhängenden sein und kann über nicht zusammenhängende Hauptspeicherbereiche verteilt sein.



Beispiel Swapping

Trong swapping, neu bo nho chinh khong du de chua tat ca du lieu cua mot tien trinh, he dieu hanh se loai bo mot hoac nhieu tien trinh khac khoi bo nho chinh de tao cho cho tien trinh do.

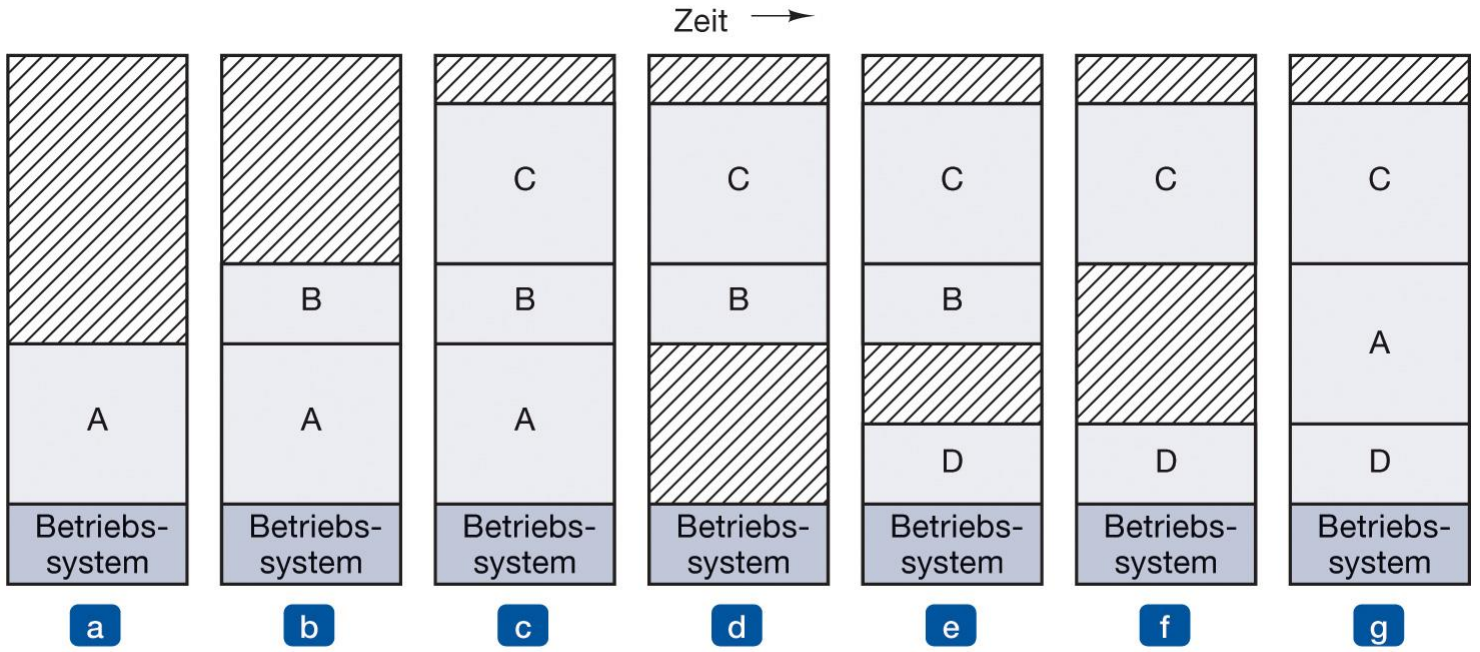


Abbildung 3.4: Mit der Ein- und Auslagerung von Prozessen ändert sich die Speicherbelegung. Die schraffierten Bereiche sind ungenutzt.

Trong virtual memory, neu bo nho chinh khong du de chua tat ca du lieu cua mot tien trinh, he dieu hanh se chi tai vao bo nho chinh mot phan du lieu cua tien trinh do. Phan du lieu con lai cua tien trinh se duoc luu tru tren o cung.

[AT]

+ Einfache Relokation über Basis- und Limit Register
- Fragmentierung
- Für alle Prozesse muss gelten: Hauptspeicher ≥ Speicherbedarf des Prozesses: Ansonsten ist eine aufwendige und fehleranfällige Overlay Technik notwendig
- Working Set nicht genutzt

Phan Vung

Hauptspeicheraufteilung bei Multiprogramming: Feste Partitionierung

Aufteilung in Partitionen fester Größe

- Jeder Prozess, dessen Platzbedarf nicht über der Größe einer freien Partition ist, kann geladen werden
- Das Betriebssystem kann einzelne Prozesse leicht aus-/ einlagern (wenn alle Partitionen belegt sind)

auslagern(roi kho) einlagern(cho vao kho)

Varianten

- Alle Partitionen haben eine einheitliche Größe
- Es gibt unterschiedliche Partitionsgrößen (→ Verringerung des „Verschnitts“)

Nachteile

- Wenn ein Programm zu groß für eine Partition ist, muss der Programmierer das Programm aufteilen: „Overlay“-Technik

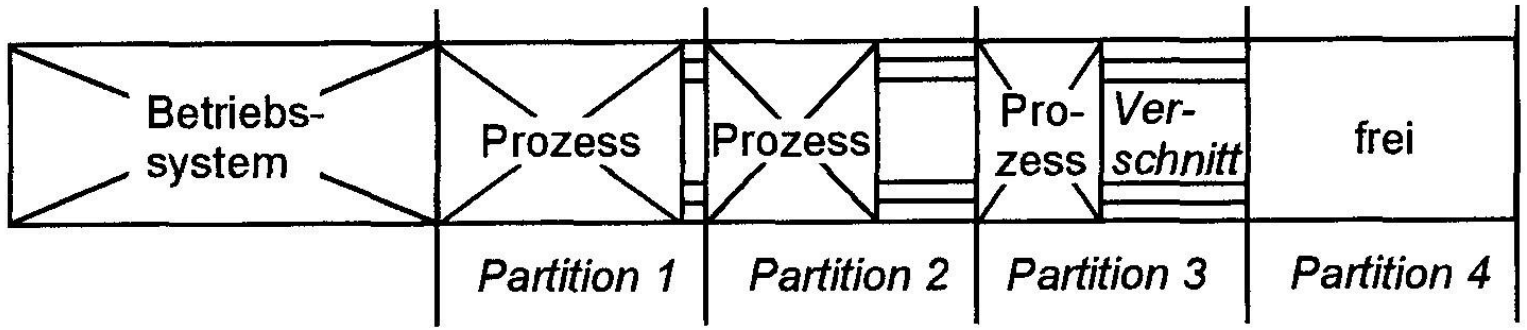
hieu qua

- Der Hauptspeicher wird nicht effizient genutzt, jedes Programm belegt eine komplette Partition → „Verschnitt“ → ungenutzter freier Speicher (Interne Fragmentierung)

Beispiel: Partitionierung

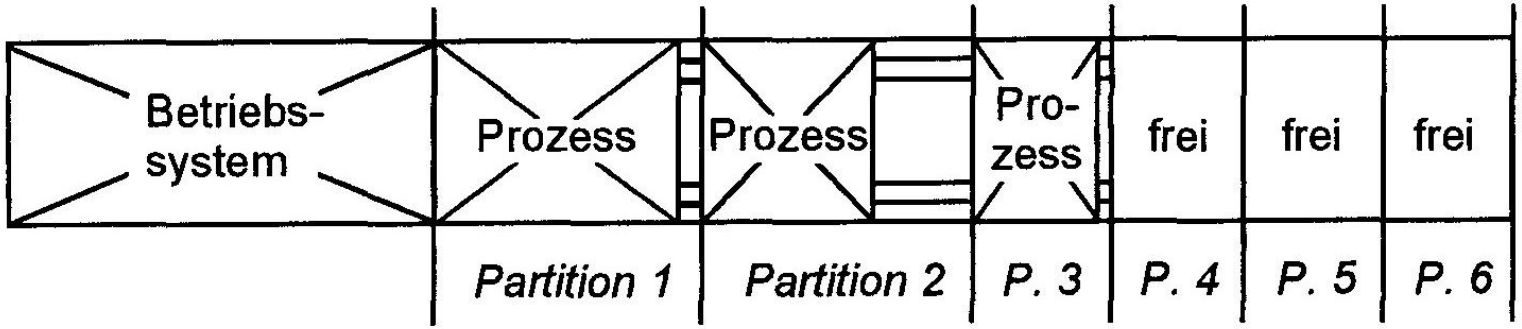
phân vùng dong kich thuoc

b.1.) feste Hauptspeicher-Partitionen einheitlicher Größe



phân vùng khac kich thuoc, tiet kiem bo nho

b.2.) feste Hauptspeicher-Partitionen unterschiedlicher Größen



[CV]

Hauptspeicheraufteilung bei Multiprogramming: Dynamische Partitionierung

dieu phoi phan vung dong

- Variable Anzahl von Partitionen unterschiedlicher Größe
- Die Partitionen werden an die Prozessgröße angepasst

dieu chinh

  - Nach Zuweisung einer Partition zu einem Prozess wird der restliche freie Platz eine neue Partition

phan bo

  - Zusammenfassen von freien aufeinander folgenden Partitionen ist möglich

co the ket hop lien tietp cac Phan vung con dang trong

Nachteile

Neu phan vung nay lon hon nhu cau cua tien trinh, thi se tao ra mot lo trong bo nho. Cac lo nay co the lam giam hieu qua su dung bo nho.

- Der Hauptspeicher wird nicht effizient genutzt: Es entstehen „Löcher“ im Speicher durch kleine Partitionen (**Externe Fragmentierung**)

  Abhilfe: Memory compaction (Speicherverdichtung): Das Betriebssystem könnte die Partitionen umkopieren (ist aber sehr zeitaufwändig)

su nen bo nho

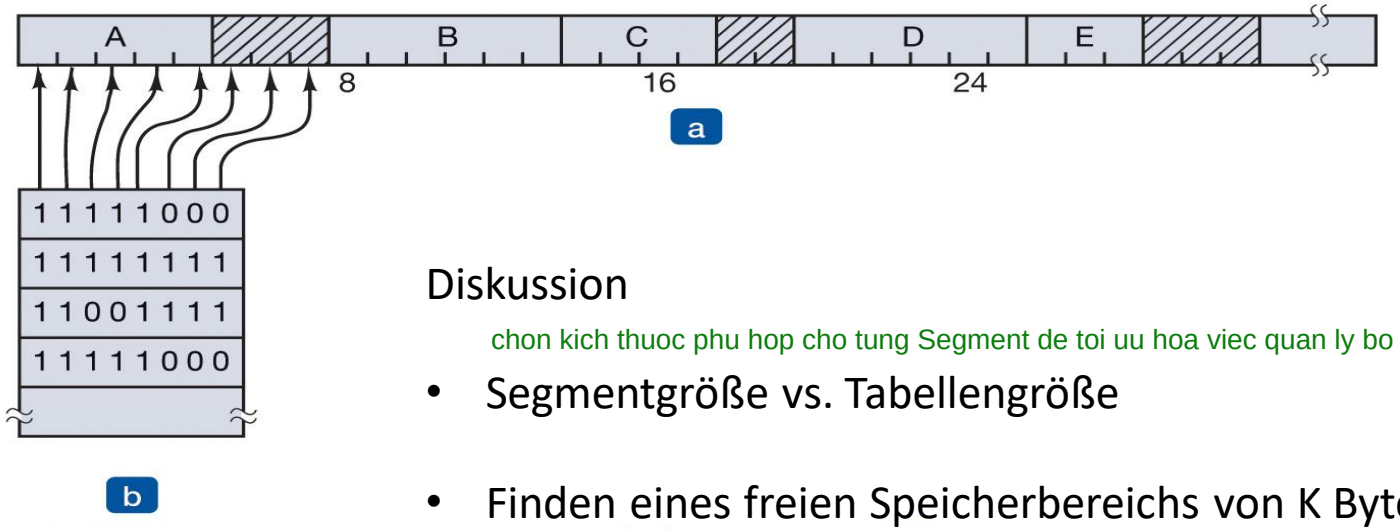
Su nen bo nho la mot qua trinh trong do cac phan vung duoc di chuyen de lap day cac lo trong bo nho

- Wenn ein Programm zu groß für eine Partition ist, muss der Programmierer das Programm aufteilen: „Overlay“-Technik

Verwaltung des freien Speichers

Bitmaps

- Standardverwaltungstechnik: Wird nicht nur für dyn. Partitionen eingesetzt
- Teile den Speicher in Belegungseinheiten (Segmentgröße) fester Größe (einige Kilobyte)
- Eine Tabelle speichert für jedes Segment in einem Bit, ob das Segment belegt (Bit = 1) oder frei (Bit = 0) ist.



[AT]

Diskussion

chon kich thuoc phu hop cho tung Segment de toi uu hoa viec quan ly bo nho va kich thuoc bang bitmap.

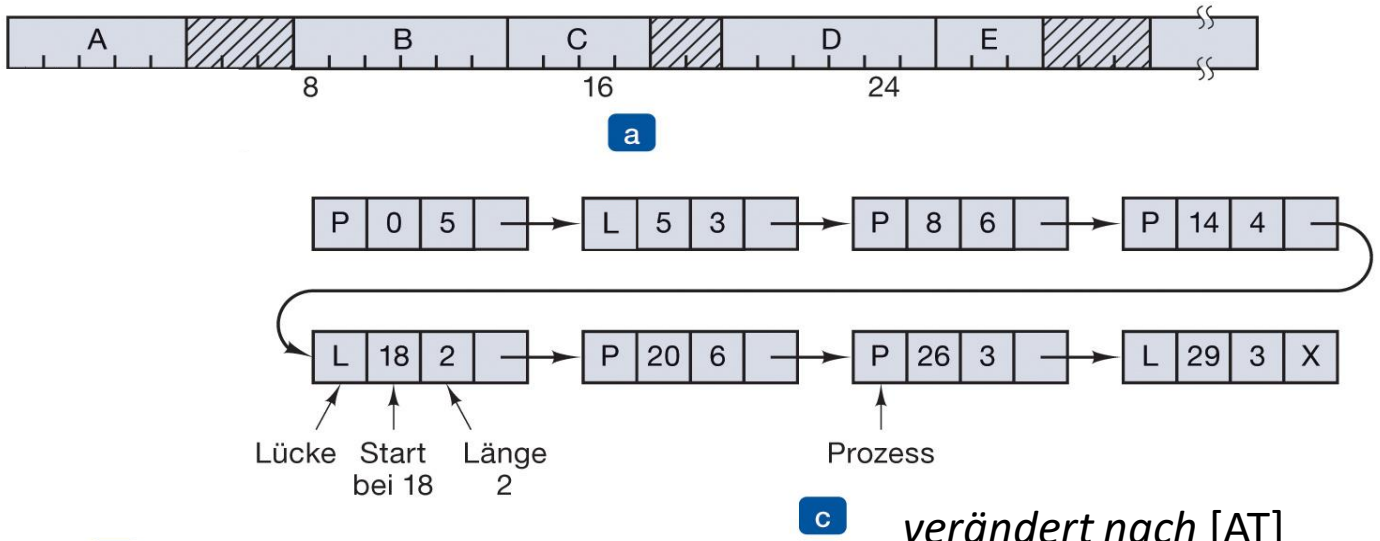
- Segmentgröße vs. Tabellengröße
- Finden eines freien Speicherbereichs von K Byte Größe

he thong phai tim kiem trong bitmap de xac dinh xem co du vung trong lien tuc co kich thuoc K byte khong.

Verwaltung des freien Speichers

Verkettete Liste

- Standardverwaltungstechnik: Wird nicht nur für dyn. Partitionen eingesetzt
- Die Liste enthält folgende Informationen: Belegt (P) / frei (L) ; Startadresse des Segments, Länge des Segments
- Sortierte Liste nach Speicheradressen
- Alternative: Mehrere Listen für unterschiedliche Größen von freiem Speicher



verändert nach [AT]

Dynamische Partitionierung: Platzierungsstrategien

Allgemein

- Das Betriebssystem muss entscheiden., welche freien Partitionen welchem Prozess zugewiesen werden

Algorithmen

bat dau tu dau danh sach cac phan vung trong va tim phan vung dau tien co kich thuoc du lon de chua tien trinh.

- Algorithmus **First-Fit**: Sucht von vorne die nächste freie Partition, die passt

bat dau tu phan vung trong gan nhat voi phan vung ma tien trinh truoc do da duoc cap phat va tim phan vung dau tien co kich thuoc du lon de chua tien trinh.

- Algorithmus **Next-Fit**: Sucht ab der zuletzt belegten Partition die nächste freie Partition, die passt

phan vung trong co kich thuoc nho nhat co the chua tien trinh.

- Algorithmus **Best-Fit**: Auswahl der freien Partition, bei der am wenigsten Platz verschwendet wird

chia danh sach cac phan vung trong thanh cac danh sach con, moi danh sach con chua cac phan vung co kich thuoc tuong tu. He dieu hanh sau do tim phan vung trong danh sach con phu hop nhat voi tien trinh

- Algorithmus **Quick Fit**: Getrennte Listen für Löcher gebräuchlicher Größe

Analyse der Algorithmen auf Basis von charakteristischen Szenarien ist entscheidend

Vergleich der Platzierungsstrategien (quantitativ)

First-Fit

- Schnellstes Verfahren!
- Viele Prozesse im vorderen Speicherbereich, meist hinten noch Platz für große Prozesse

Best-Fit

- Schlechtestes Ergebnis!
- Aufwändigste Suche
- Weil immer kleine Speicherreste bleiben, muss das Betriebssystem häufig umsortieren

Next-Fit

- Belegt Speicher gleichmäßiger als First-Fit, nachteilig für große Prozesse

Quick-Fit

- Sehr schnell

HAW HAMBURG

Betriebssysteme BS-ITS

4. Speicherverwaltung

1. *Einführung und Grundlagen*
2. *Swapping*
3. ***Virtuelle Speicherverwaltung***
4. *Seitenersetzungsverfahren*
5. *Entwurfsaspekte*
6. *UNIX*

Motivation

Probleme Swapping

- Keine Nutzung des Working Sets ko su dung hieu qua cac tap hop lam viec cua Prozess
- Fragmentierung van de phan manh
- Speicherbedarf Prozess > Hauptspeicher: aufwendige Overlay Technik

Anforderungen an einen optimalen Speicher

- Unbeschränkte Größe, so dass jedes beliebig große Prozess ohne zusätzlichen Aufwand geladen und verarbeitet werden kann
- Einheitliches Adressierungsschema für alle Speicherzugriffe (keine Unterscheidung von Speichermedien)
- Direkter Zugriff auf den Speicher (ohne Zwischentransporte)
- Schutz vor fremden Zugriffen in den eigenen Speicherbereich

Lösungsansatz: Virtuelle Speicherverwaltung

Paging

- Ein Prozess hat einen eigenen Adressraum – **virtueller Adressraum**
- Eine vom Programm generierte Adresse ist eine **virtuelle Adresse** aus dem virtuellen Adressraum
- Der virtuelle Adressraum ist in **Seiten (pages)** eingeteilt
- Alle Seiten sind in der Regel auf der Festplatte gespeichert
- Der physikalische Speicher ist in **Seitenrahmen (page frames)** eingeteilt.
- Die MMU/TLB bildet über die **Seitentabelle** Seiten auf Seitenrahmen ab. Eine Seitentabelle pro Prozess.
- Nur die gerade benutzten / relevanten Seiten sind auf Seitenrahmen abgebildet – stehen im Hauptspeicher.
- I.a. sind Seiten und Seitenrahmen gleich groß (zwischen 512 Byte und 64KiB)

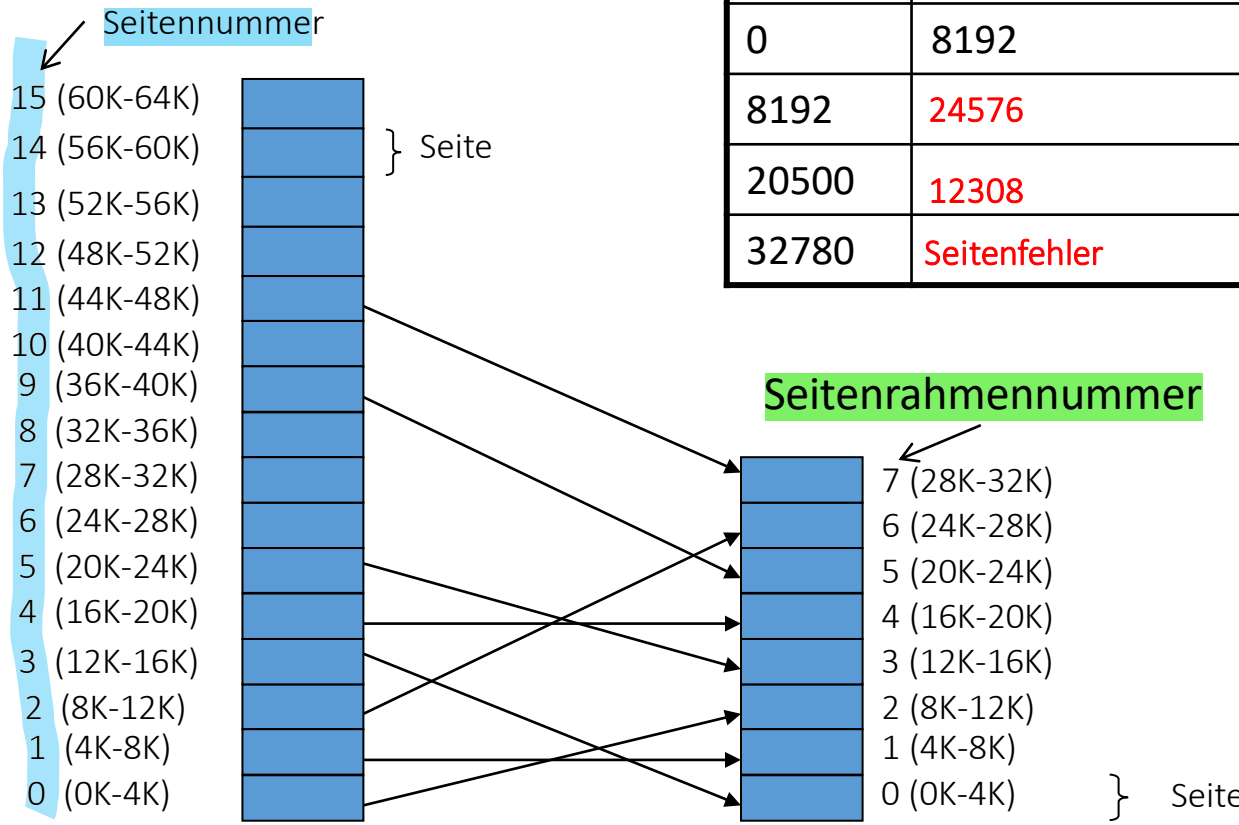
MMU/TLB (Memory Management Unit/Translation Lookaside Buffer): Thuc hien anh xa tu trang(Seite- Page- vitural Adressraum) sang khung trang(seitenrahme-page frames-physikalische Speicher).

Abbildung virtueller Speicher auf den physikalischen Speicher

Platte: Virtueller Adressraum (64KiB)
Hauptspeicher: 32KiB
4096 Byte pro Frame bzw. Page

Beispiele:

Virtuelle Adresse	Physikalische Adresse
0	8192
8192	24576
20500	12308
32780	Seitenfehler



Seitentabelle

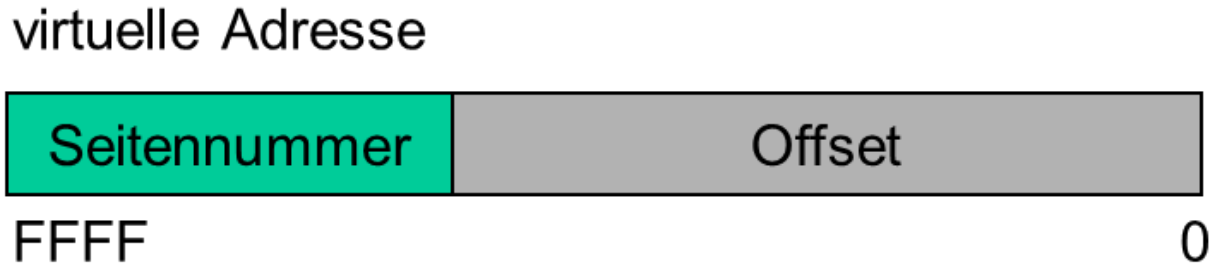
	page frame	present bit
15		0
14		0
13		0
12		0
11	7	1
10		0
9	5	1
8		0
7		0
6		0
5	3	1
4	4	1
3	0	1
2	6	1
1	1	1
0	2	1

HIER K = 1024

Abbildung virtueller Speicher auf den physikalischen Speicher (2)

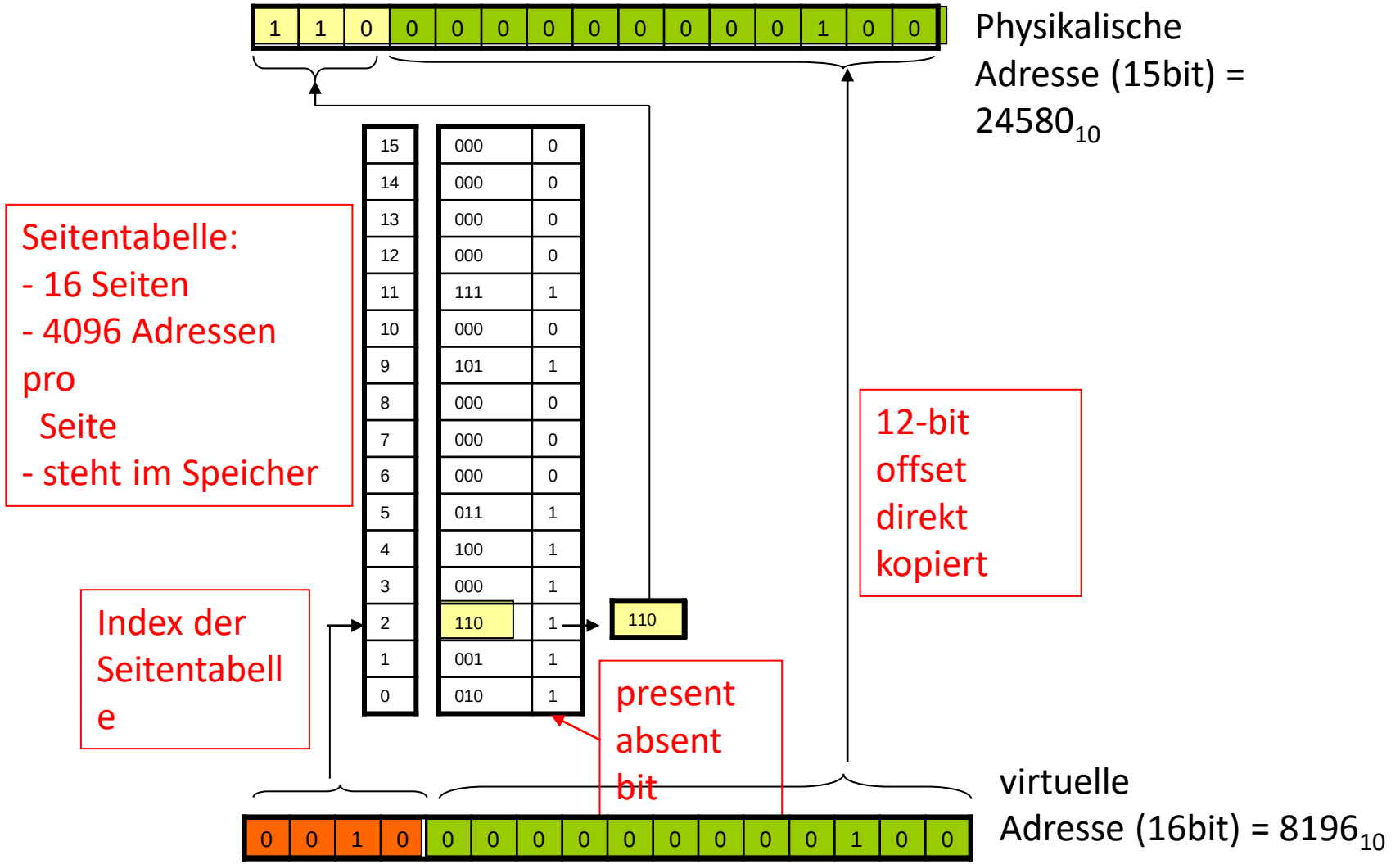
Standardtechnik

- Zweiteilung der virtuellen Adresse
- Seitennummer dient als Index in der Seitentabelle
- Offset: physikalische Adresse im Seitenrahmen

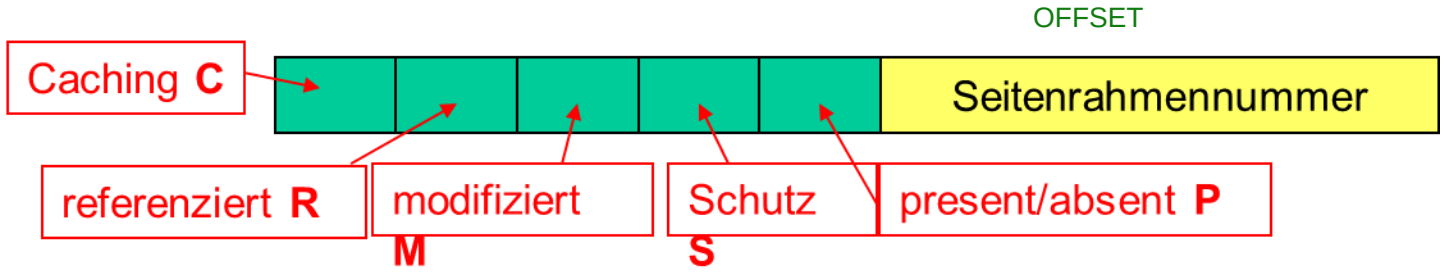


$$adr_{physikalisch} = f_{Seitentabelle}(adr_{logisch})$$

- **Beispiel:** 64KiB virtueller AR, 32KiB physikalischer AR



Seitentabelleneintrag



- P Bit: zeigt an, ob die Seite im Speicher steht

  P = 0 : Seite ist nicht im Hauptspeicher: → Seitenfehler: die Seite muss in den Hauptspeicher geladen werden.

- S Bit: Schreib/Lese Schutz
- M Bit: Die Seitenrahmen hat andere Werte als die Seite auf der Festplatte

  M = 1 : Bein Auslagern muss die Seite auf Festplatte geschrieben werden

- R-Bit: auf die Seite wurde zugegriffen (lesend oder schreibend)
- C Bit: Bei Eingabe per I/O-Gerät

  C = 0 : Seite darf nicht gecached werden

Praktisches Beispiel

Setup

- 32 Bit breite virtuelle Adressen
- 4 KiB Seitengröße

Größe der Seitentabelle

- 232/ 212 = 220 ≈ 1 Million Einträge
- Bei 4 Byte pro Eintrag: Größe der Seitentabelle : 4 MB

Zeit zur Umrechnung einer virtuellen Adr. in eine physikalische Adr.

- Zugriff auf die Seitentabelle darf nicht länger als 1ns dauern, ansonsten werden Tabellenzugriffe zum Engpass

Bedarf

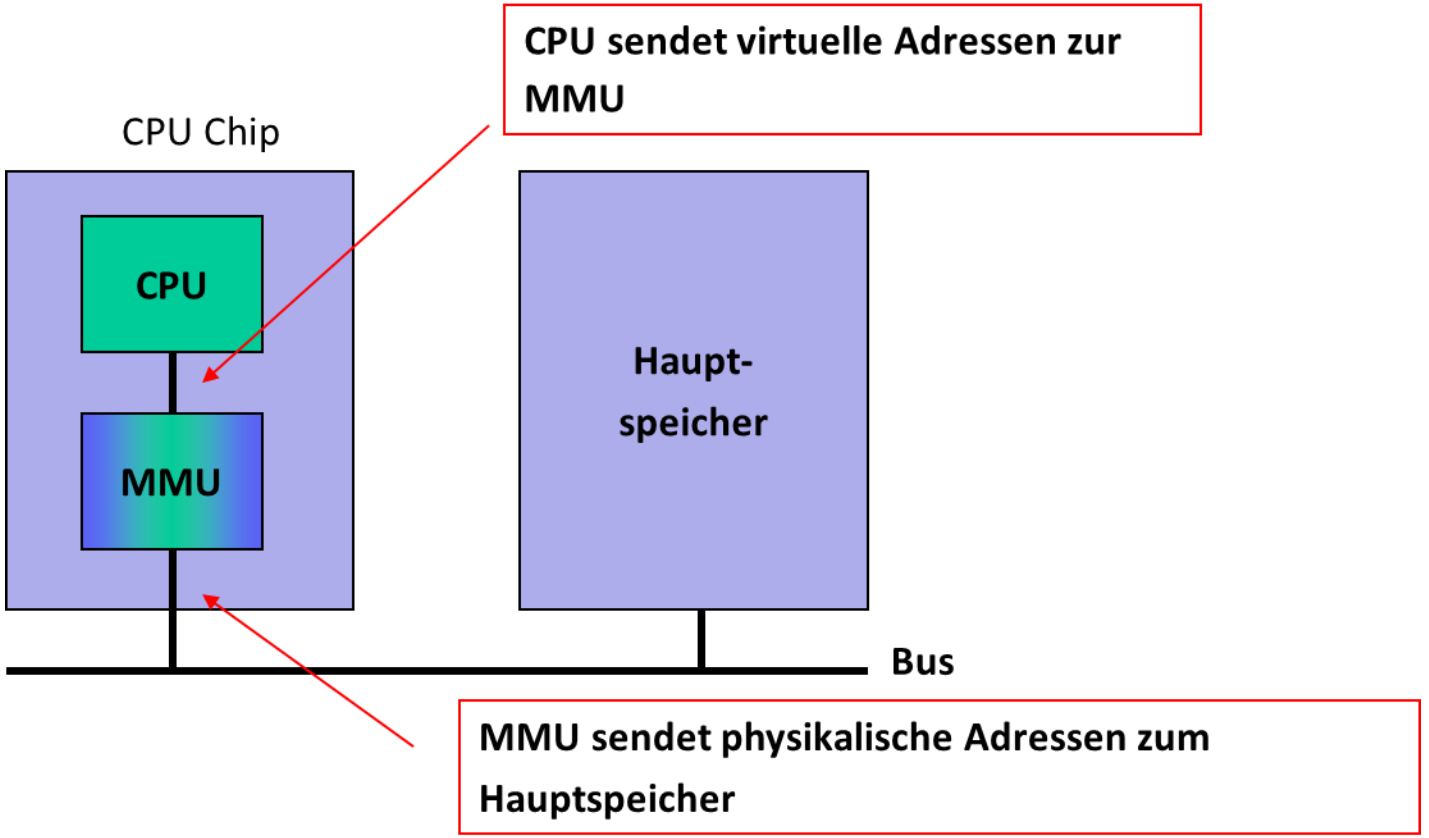
- Schnelles Abbilden - auch für sehr große Seitentabellen

Vorteil

- Nutze Workingset aus

MMU: Memory Management Unit

Die MMU (Memory Management Unit) bildet virtuelle Adressen auf physikalische Adressen ab.



TLB: Translation Lookaside Buffer

- Komponente der MMU
- Ist ein Assoziativspeicher TLB hoat dong nhu mot bo nho lien ket.
- Für eine kleine Zahl (32 bis 1024) häufig genutzter Seiten, bildet der TLB virtuelle Adressen auf physikalische Adressen ab - unter Umgehung der Seitentabelle
- Trefferquoten in der Praxis: 80% - 98%
- Wer behandelt TLB Miss (HW oder BS)?

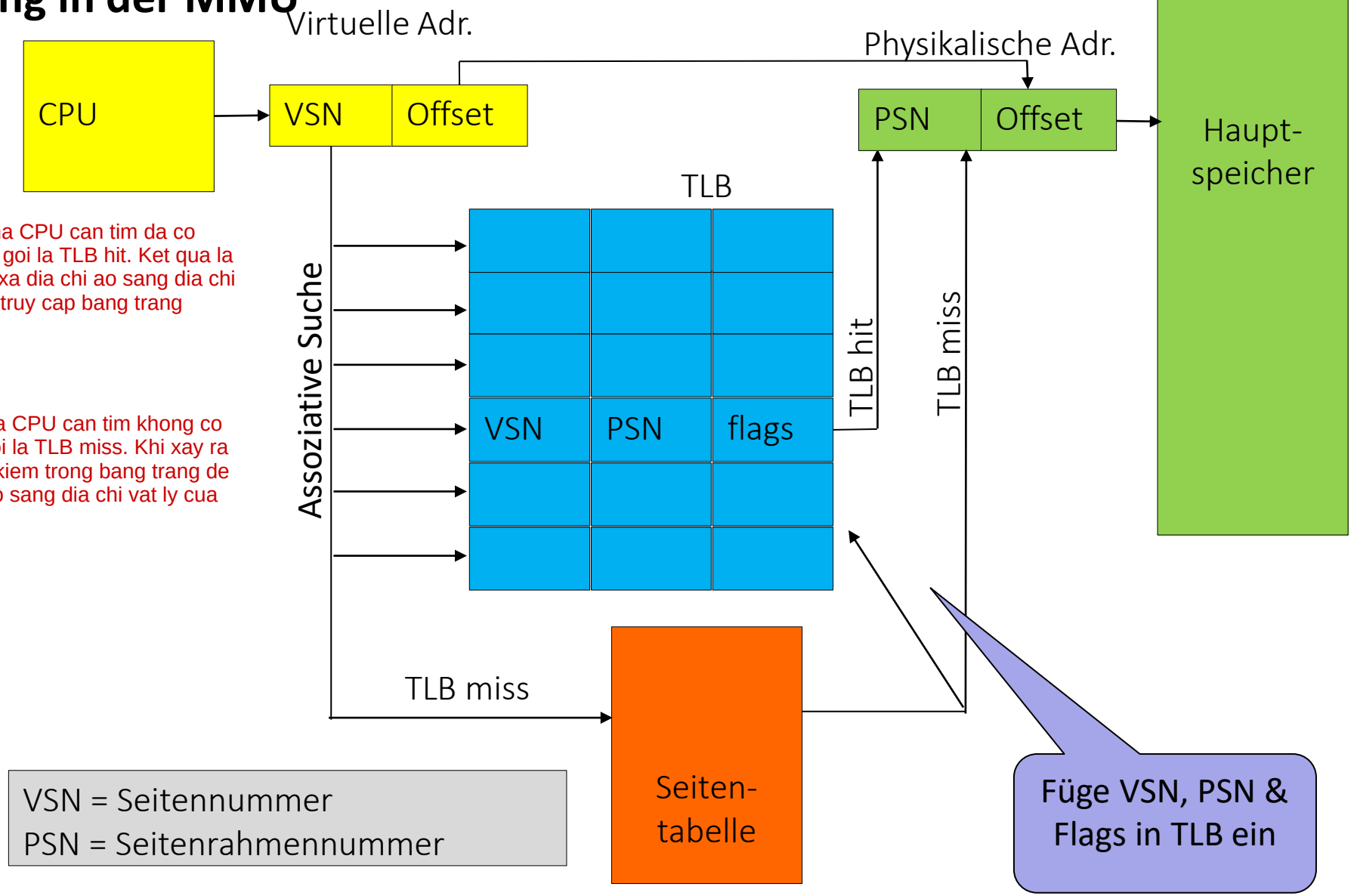
Arbeitsweise

- Falls Seitennummer (virtuell) im TLB (suche parallel)
  → Überprüfe Schreibschutz auf Basis des S-Bits (ggf. löse Schutzverletzung aus) Kiem tra quyen ghi dua tren bit bao ve
  → Falls Schreibschutz o.k., nehme Seitenrahmen-nummer aus dem TLB Neu quyen ghi la hop le, lay so khung trang tu TLB.
- Falls Seitenadresse nicht im TLB Neu dia chi trang khong ton tai trong TLB:
  → MMU nimmt Seitenrahmennummer aus Seitentabelle (wie gewöhnlich) MMU lay so khung trang tu bang trang nhu thong thuong.
  → Überschreibe „ältesten“ Eintrag im TLB mit dieser Seitenadresse + restliche Info, schreibe vorher M-Bit des ältesten Eintrag in die Seitentabelle zurück.

Ghi de thong tin cua "ban ghi cu nhat" trong TLB bang thong tin ve dia chi trang moi + cac thong tin con lai, truoc khi dat lai bit M (modified) cua ban ghi cu nhat trong bang trang.

P Bit	Virtuelle Seite	M Bit	S Bit	Seiten-rahmen
1	140	1	RW	31
1	20	0	R	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R	50
1	21	0	R	45
1	860	1	RW	14
1	861	1	RW	75

TLB Umsetzung in der MMU



TLB Hit: Neu dia chi ao ma CPU can tim da co trong TLB, dieu nay duoc goi la TLB hit. Ket qua la CPU co the de dang anh xa dia chi ao sang dia chi vat ly ma khong can phai truy cap bang trang

TLB Miss: Neu dia chi ao ma CPU can tim khong co trong TLB, dieu nay duoc goi la TLB miss. Khi xay ra TLB miss, CPU se phai tim kiem trong bang trang de xac dinh anh xa tu dia chi ao sang dia chi vat ly cua trang bo nho

VSN = Seitennummer
PSN = Seitenrahmennummer

Mehrstufige Seitentabellen

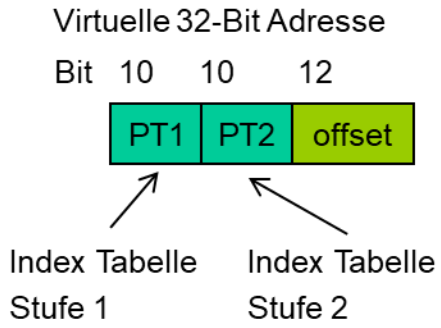
Problem

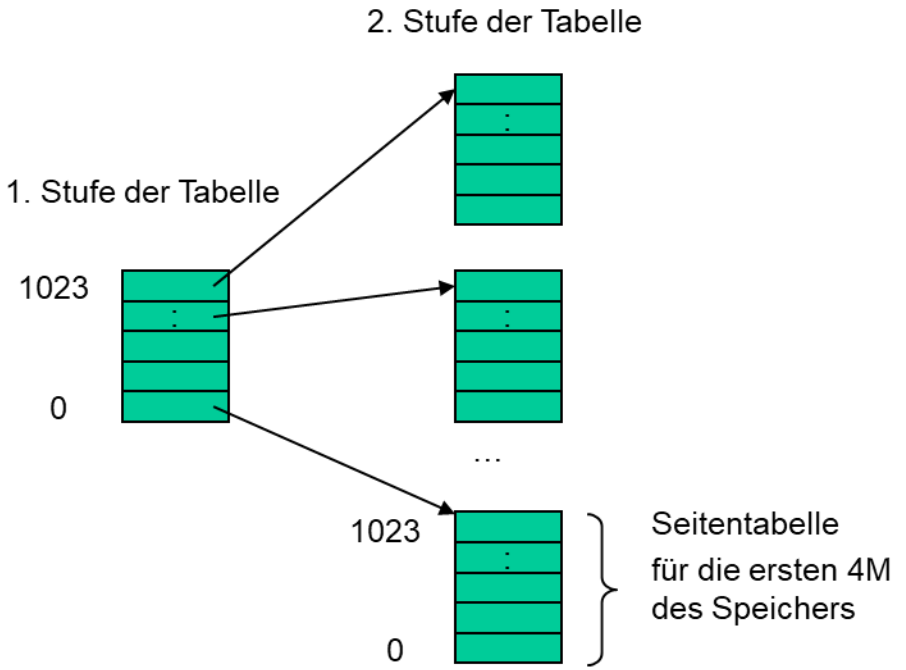
- Große virtuelle Adressräume führen zu sehr großen Seitentabellen.

Idee

- Seitentabelle wird in 2 oder mehr Stufen aufgebaut. Nur Teile der Seiten-tabelle werden zeitgleich im Speicher gehalten (müssen existieren)

Beispiel





Thay vi giu toan bo bang trang trong bo nho moi khi mot tien trinh hoat dong, chi mot phan nho cua bang trang se duoc giu trong bo nho, con phan con lai se duoc luu tru o noi khac, co the la tren dia cung.

Cau truc da cap nay giup giam tai cho viec quan ly bo nho, vi chi can duy tri mot phan nho cua bang trang can thiet trong bo nho tai mot thoi diem. Khi can, he thong se thuc hien viec doi cho cac phan cua bang trang de phu hop voi viec hoat dong cua tien trinh cu the

Invertierte Seitentabellen

Beispiel

- 64 Bit breite virtuelle Adressen, Seitengröße 4 KiB, 256 MiB Speicher.

Ansatz der invertierten Seitentabelle

Thay vi luu tru thong tin ve cac trang, bang trang chi luu thong tin ve cac khung trang (frames)

- Seitentabelle hält Einträge für jeden Seitenrahmen (und nicht für die Seiten selbst)
- Eintrag: Prozess + Seitennummer – also nur eine Tabelle für alle Prozesse.

Vorteil

Phuong phap nay tiet kiem khong gian luu tru rat lon vi chi can luu thong tin ve khung trang, khong can luu thong tin cua tung trang.

- spart enorme Menge an Speicherplatz Beispiel: 256 MiB / 4 KiB = 65536 Einträge

Nachteil

- Abbildungsfunktion aufwändig (langsam)
  → Aber wir haben ja dem TLB
  → praktikabel: Nutze Hashtabelle mit Hashwerten H (ProcessId, virtuelle_Adresse)
  → virtuelle Seiten im Speicher mit gleichen Hashwert sind verkettet

HAW HAMBURG

Betriebssysteme BS-ITS

4. Speicherverwaltung

1. *Einführung und Grundlagen*
2. *Swapping*
3. *Virtuelle Speicherverwaltung*
4. ***Seitenersetzungsverfahren***
5. *Entwurfsaspekte*
6. *UNIX*

Idee

- Bei einem Seitenfehler wählt das Betriebssystem eine Seite aus, welche aus dem Speicher entfernt wird, um einer neuen Seite Platz zu machen

Vorgehensweise

Chon trang can bi loai bo.

- Wähle auszulagernde Seite aus.

Neu trang da bi sua doi, trang tuong ung tren dia cung se duoc cap nhat.

- Wurde die Seite modifiziert, so wird die korrespondierende Seite auf der Platte aktualisiert.
- Ersetze die alte Seite durch eine neue Seite

Thay the trang cu bang trang moi ma CPU can truy cap.

Ziel

- Geringe Anzahl von Seitenfehlern

Ladestrategien

Doc theo yeu cau

Demand Paging liest Seiten erst ein, wenn auf sie zugegriffen wird

- Einlesen der Seite bei Page-Fault — Khi xay ra loi trang (page fault), trang se duoc doc tu dia cung vao bo nho
- Viele Page-Faults bei Prozess-Start — Trong giai doan khoi dau cua mot tien trinh, co the xay ra nhieu loi trang vi da so trang can thiet cho tien trinh nay chua duoc doc vao bo nho.

Doc truoc

Pre-Paging liest neben der angeforderten Seite einige weitere Seiten mit ein

- z.B. die nächsten Seiten des Programmcodes oder beim Prozesswechsel die „zuletzt“ genutzten Seiten des Prozesses. — Doc truoc khong chi trang duoc yeu cau ma con cac trang co the duoc su dung gan day hoac lien quan den trang duoc yeu cau.
- Entspricht der Charakteristik der Platte – d.h. ist effektiv – wenn Seiten auf der Platte physikalisch hintereinander liegen — Hieu qua neu cac trang tren dia cung duoc dat gan nhau vat ly.
- Aber: Werden die Seiten wirklich benötigt?

Tuy doc truoc co the tang toc do nap trang, nhung cung co the dan den viec doc khong can thiet neu cac trang do khong duoc su dung.

1. Optimaler Algorithmus

Vorgehensweise

Moi trang duoc danh dau voi so lenh thuc hien truoc khi trang do duoc truy cap lan tiep theo.

1. Markiere jede Seite mit der Anzahl der Instruktionen, die zur Ausführung gelangen, bevor auf diese Seite das nächste mal referenziert wird.
2. Entferne die Seite mit größter Markierung.

Xoa trang co so lenh lon nhat.

Beurteilung

Thuat toan nay duoc coi la "toi uu" vi no loai bo trang it duoc su dung nhat.

- ”optimal”, da die aktuell am wenigsten genutzte Seite ausgelagert wird.

Thuat toan nay khong thuc hien duoc trong thuc te vi khong the du doan truoc duoc so lan truy cap toi moi trang trong tuong lai.

- nicht implementierbar, da Markierungen nicht ermittelbar sind – Blick in die Zukunft.
- dennoch sinnvoll, da per Simulation und “Zweifachdurchlauf” für konkretes Programm Vergleichsmöglichkeit mit anderen Algorithmen besteht.

Mac du khong thuc hien duoc trong thuc te, nhung viec su dung mo phong va "dong chay kep" cho mot chuong trinh cu the co the so sanh thuat toan nay voi cac thuat toan khac.

2. NRU: Not recently used Algorithmus

Vorgehensweise

He dieu hanh phan loai cac trang vao bon nhom khac nhau.

1. BS ordnet die Seiten in “vier” Kategorien ein

Loai bo mot trang ngau nhien tu nhom co muc do uu tien thap nhat, nhom nay chua it nhat mot trang.

2. Entferne eine zufällige Seite aus der niedrigsten Kategorie, welche zumindest eine Seite enthält

Bit M thuong duoc su dung cung voi bit R (Reference bit) de quan ly viec thay doi va truy cap vao trang. Khi trang duoc truy cap, bit R duoc dat len de chi rang trang da duoc truy cap. Khi trang duoc thay doi, bit M se duoc dat len de chi rang trang da duoc sua doi.

Kategorie	Referenziert (R Bit)	Modifiziert(M Bit)
0	0	0
1	0	1
2	1	0
3	1	1

Wann wird welches Bit gesetzt?

Zyklisches Rücksetzen des R Bits (typisch alle 20 ms)

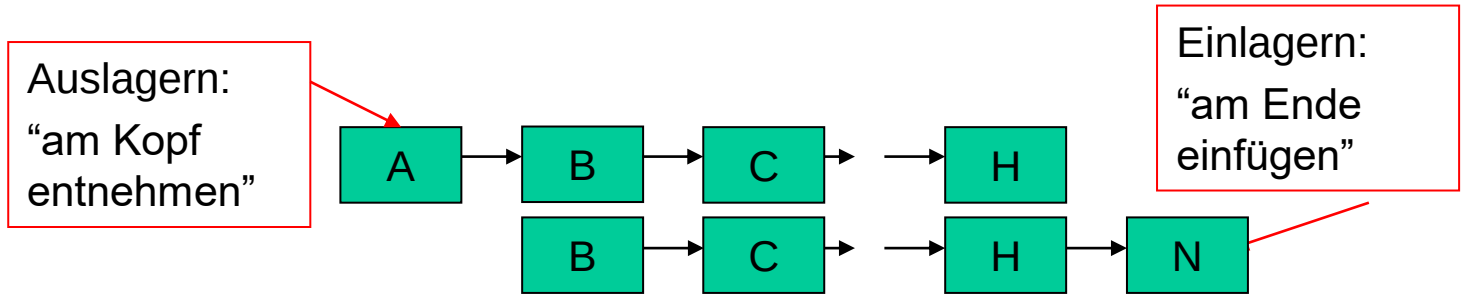
Beurteilung

- Grundlage: Working Set, wobei R-Bit stärker gewichtet ist als M-Bit
- leicht zu verstehen und effizient implementierbar
- bessere Leistung wünschenswert, jedoch oftmals ausreichend

3. FIFO: First In First Out

Vorgehensweise

1. Die Seiten stehen als verkettete Liste im Speicher
2. Bei Seitenfehler wird Seite am Listenkopf entfernt
3. Neue Seite wird an das Ende gesetzt



Beurteilung

- Die älteste Seite wird ausgelagert
- Keine Unterscheidung zwischen intensiv genutzten Seiten und wenig genutzten Seiten
- FIFO ist für den praktischen Einsatz ungeeignet

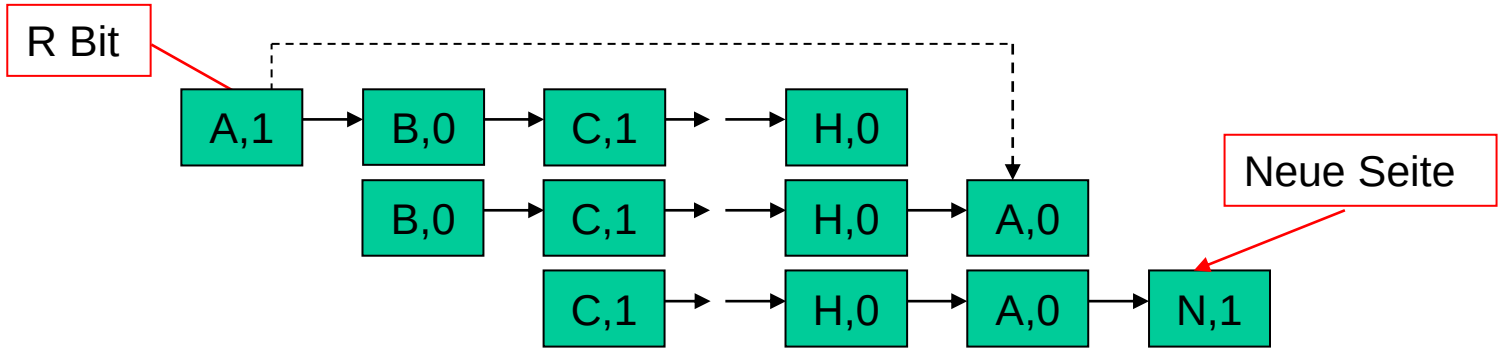
4. 2C-FIFO: Second Chance FIFO

Vorgehensweise

1. Die Seiten stehen als verkettete Liste im Speicher
2. Bei Seitenfehler untersuche Listenkopf:
   If R==0, lösche Kopfseite und füge neue Seite mit R:=1 an das Ende der Liste
   If R==1, verschiebe Kopfseite an das Ende der Liste und setze R:=0
3. Wiederhole Schritt 2, solange bis eine Seite ersetzt wurde

Neu bit R cua trang dau danh sach bang 0, nghia la trang nay khong duoc truy cap gan day, no se bi loai bo va trang moi voi bit R duoc thiet lap thanh 1 se duoc them vao cuoi danh sach.

Neu bit R cua trang dau danh sach bang 1, nghia la trang nay duoc truy cap gan day, no se duoc di chuyen xuong cuoi danh sach va bit R se duoc thiet lap thanh 0.



Beurteilung

- Was passiert, wenn bei allen Seiten das R-Bit gesetzt ist?
- Relativ einfach realisierbar
- Relativ hoher Verwaltungsaufwand → Verschieben von Listenelementen

Neu tat ca cac trang deu co bit R duoc thiet lap, thuat toan se lap lai viec di chuyen cac trang den khi co mot trang co bit R bang 0 hoac mot trang duoc thay the.

5. Clock

Vorgehensweise

Cac trang trong bo nho duoc sap xep duoi dang danh sach lien ket vong

1. Die Seiten stehen als zyklisch verkettete Liste im Speicher

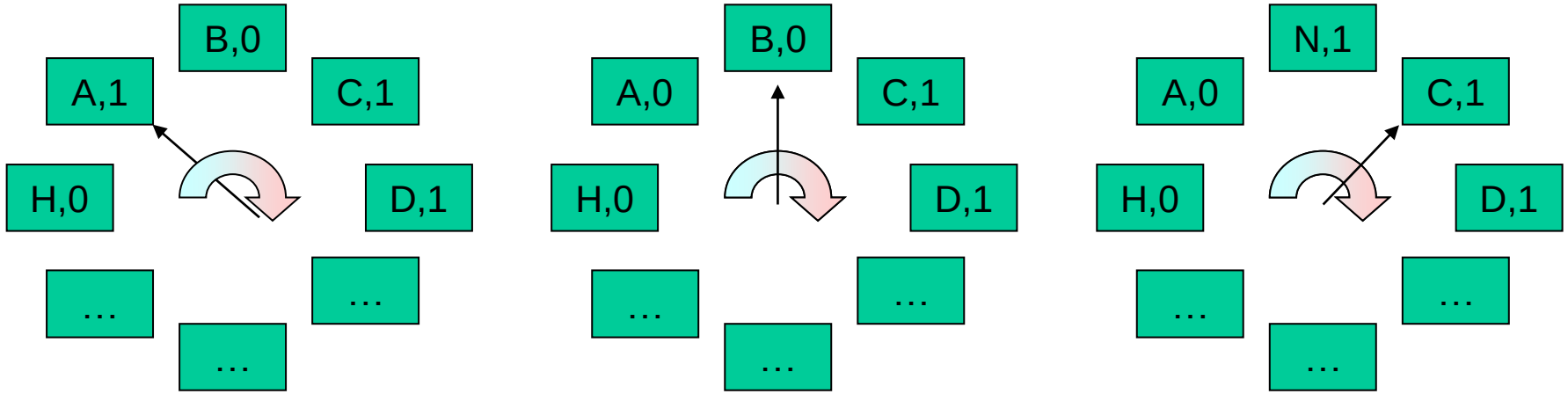
2. Bei Seitenfehler untersuche zyklische Liste:
   If R==0, lösche Seite und setze neue Seite mit R:=1 ein
   else if R==1, setze R:=0 fi
   Gehe zum nächsten Element

   Neu bit R cua trang bang 0, trang se bi xoa va trang moi voi bit R duoc thiet lap thanh 1 se duoc them vao.

   Neu bit R cua trang bang 1, bit R se duoc thiet lap thanh 0

   Di chuyen den phan tu tiep theo trong danh sach.

3. Wiederhole Schritt 2 bis Fall R==0 eingetreten ist



Beurteilung

Gleiches Ersetzungsverhalten wie 2C-FIFO, aber niedrigerer Verwaltungsaufwand als 2C-FIFO

6. LRU: Last Recently Used Algorithmus

Beobachtung

- Seiten, welche für die letzten Befehle oft genutzt wurden, werden mit hoher Wahrscheinlichkeit auch für die kommenden Befehle genutzt
- Bei Seitenfehler: entferne die am längsten ungenutzte Seite

Vorgehensweise 1

1. Alle Seiten stehen in einer verketteten Liste
2. Beim Zugriff auf eine Seite, wird Sie an den Anfang der Liste gesetzt
3. Die Seite am Ende der Liste ist die am längsten ungenutzte Seite

Tat ca cac trang duoc to chuc duoi dang danh sach lien ket.

Khi mot trang duoc truy cap, trang do duoc chuyen len dau danh sach.

Trang o cuoi danh sach duoc coi la trang khong duoc su dung trong thoi gian dai nhat.

Vorgehensweise 2 (in Hardware)

1. 64 Bit Register, das bei jedem CPU Takt erhöht wird
2. Pro Seite eine 64 Bit Eintrag, der bei Zugriff auf die Seite mit den aktuellen Zählerwert belegt wird
3. Die Seite mit dem niedrigsten Eintrag ist die am längsten ungenutzte Seite

Moi trang co mot muc 64 bit, gia tri nay duoc cap nhat khi trang duoc truy cap.

Trang co gia tri nho nhat duoc xem la trang khong duoc su dung trong thoi gian dai nhat.

Beurteilung

- Aufwändig in Hardware
- Kommt dem Optimum sehr nahe

6. LRU: Last Recently Used Algorithmus (2)

Vorgehensweise 3 (in Hardware)

Doi voi n khung trang, can mot ma tran bit co kich thuoc n x n duoc khoi tao voi gia tri 0.

1. Bei n Seitenrahmen wird eine n x n Bitmatrix benötigt, die mit 0 initialisiert ist.

Khi truy cap vao khung trang thu n, hang thu n se duoc thiet lap thanh 1, cot thu n se duoc thiet lap thanh 0.

2. Zugriff auf Seitenrahmen n: n-te Zeile wird auf 1 gesetzt, n-te Spalte wird auf 0 gesetzt.

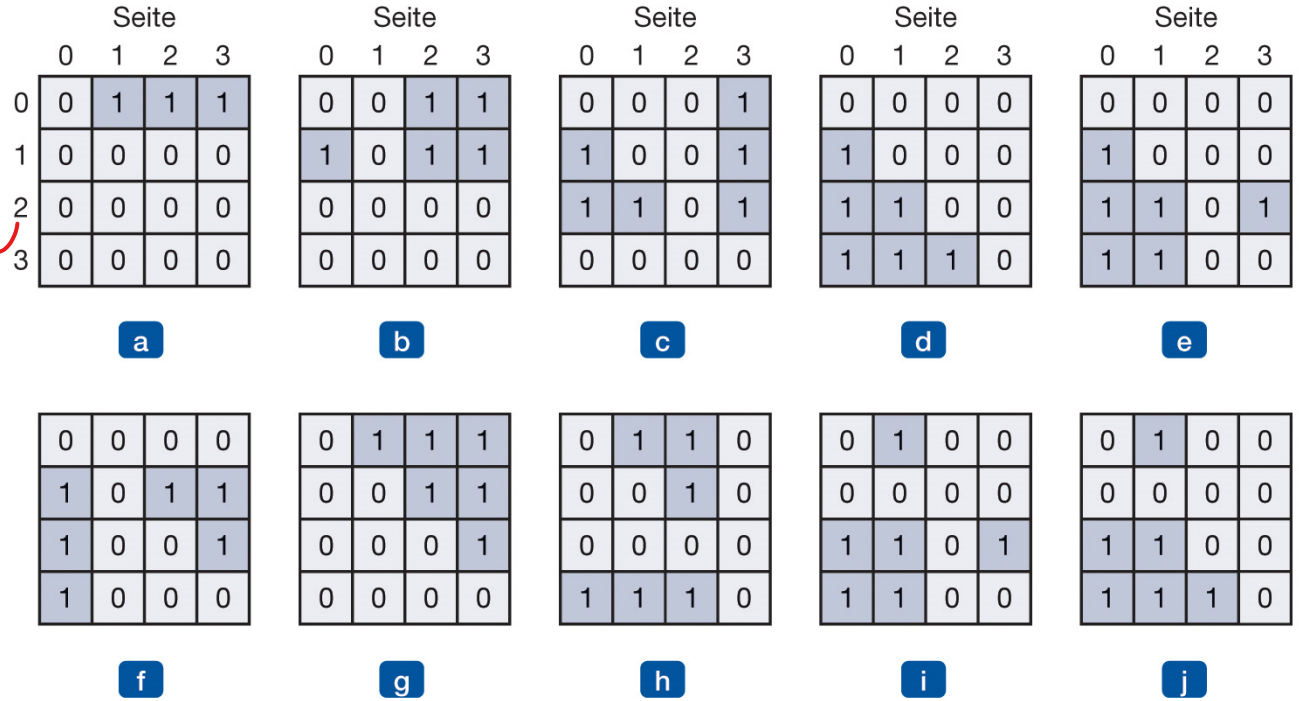
3. Interpretiere die Zeilen als Binärzahlen (unsigned): Die Zeile mit dem niedrigsten Wert wurde am längsten nicht benutzt.

Giai ma cac hang nhu so nhi phan (unsigned): Hang co gia tri thap nhat se bieu thi cho trang khong duoc su dung trong thoi gian dai nhat.

seitenrahmen 0 ,

zeile 0 wird 1 gesetzt

Spalte 0 wird 0 gesetzt



[AT]

Abbildung 3.17: LRU mit einer Matrix. Auf die Seiten wird in der Reihenfolge 0 1 2 3 2 1 0 3 2 3 zugegriffen.

7. NFU: Not Frequently Used

Beobachtung

- LRU in Software sehr aufwändig
- Spezialhardware oftmals nicht vorhanden

=> Näherungsweise Nachbildung von LRU in SW

Vorgehensweise

1. Jede Seite hat einen SW Zähler, der mit 0 initialisiert ist.
2. Zyklisch (z.B.: Timer Intervall alle 20 ms) werden die R-Bit zu den Zählern addiert.

Beurteilung

NFU khong quen cac truy cap cu, dieu nay co the dan den viec cac trang cu va thuong xuyen duoc truy cap van duoc giu lai trong bo nho du chung khong con duoc su dung.

- NFU vergisst keine alten Zugriffe (Problem): Alte, oft referenzierte Seiten bleiben im Speicher „kleben“, auch wenn sie nicht mehr benutzt werden.
- Unschärfe: Es wird nicht gespeichert, wie oft eine Seite in einem Intervall referenziert wird.

NFU khong luu tru so lan mot trang duoc tham chieu trong mot khoang thoi gian cu the, dieu nay co the gay ra su mo ho trong viec xac dinh trang nao can duoc thay the.

8. Aging

Bemerkung

- Lösung des „Nicht-Vergessen-Problems“ des NFU Algorithmus

Vorgehensweise

1. Jede Seite hat einen SW Zähler, der mit 0 initialisiert ist.
2. Zyklisch (z.B.: Timer Intervall alle 20 ms) werden die R-Bit zu dem Zähler wie folgt addiert:
   - Shifte den Zähler im 1 nach rechts (Division durch 2)
   - Setze das R Bit an die linke Position (addiere R-Bit * 2hochwertiges Bit-Position)
   - Setze das R Bit zurück

Dich chuyen bo dem sang phai 1 bit tuong duong voi viec chia bo dem cho 2.

cong gia tri cua (bit R nhan voi 2) tai vi tri bit co gia tri cao nhat cua bo dem (SW Zähler)

Beurteilung

- Unschärfe wie bei NFU
- Da 2-er Potenzen addiert werden, ist die Breite der Zähler „schnell verbraucht“
  Beispiel: n Bit breiter Zähler: Keine Unterscheidung, ob auf eine Seite in den letzten n oder n+x Zyklen nicht referenziert wurde.
- Was passiert in folgenden Fall: Seite wird eingelagert, page fault tritt auf, Aging Time ist in der Zeit nicht abgelaufen.
- Die frisch eingelagerte Seite wird wieder ausgelagert, da age noch 0 ist.

Do viec them cac luy thua cua 2 vao bo dem, bo dem se "nhanh chong" bi tieu hao.

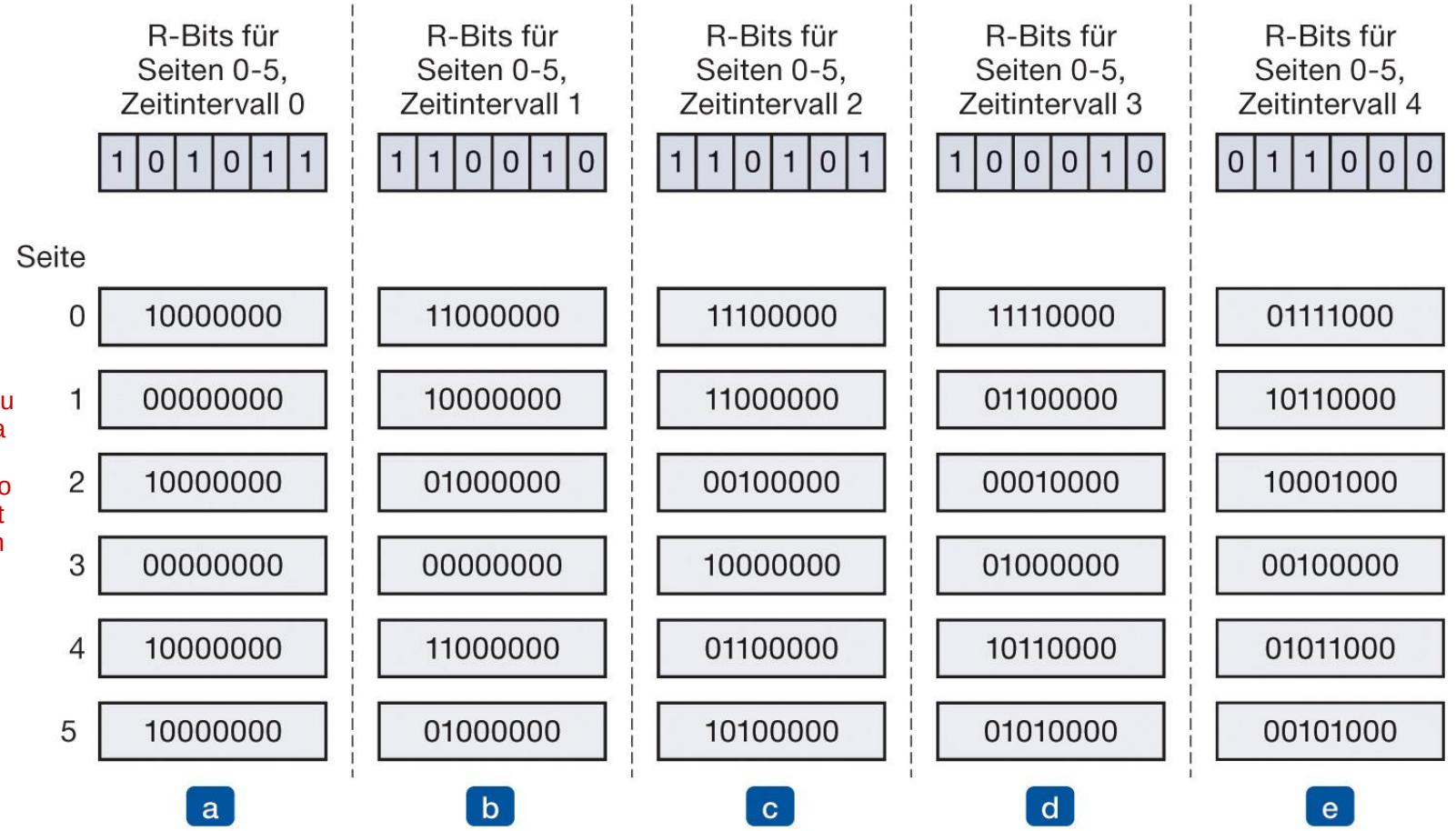
Trang vua duoc luu tru se bi xoa lai vi gia tri age cua no van la 0, do qua trinh Aging chua hoan thanh.

Beispiel Aging

Moi Seite duoc gan voi mot byte (8 bit) trong Age-Table

cac page khong duoc truy cap gan day se co gia tri byte thap hon

letzte-bit cua byte do duoc su dung de luu tru gia tri bit trang thai cua page tuong ung. Cac bit con lai trong byte do se duoc Link-SHift mot vi tri de tao khong gian cho gia tri bit moi nhat cua trang.



[AT]

Abbildung 3.18: Der Aging-Algorithmus ist eine Software-Simulation von LRU. Dargestellt werden die Zähler von sechs Seiten für fünf Intervalle. (a) bis (e) zeigen die Zustände nach den Intervallen 1–5.

9. Working Set Algorithmus

Beobachtung

- Lokalitätsprinzip des Working Sets
- zu jedem Zeitpunkt t gibt es eine Menge von Seiten, die in den letzten k Speicherzugriffen genutzt wurden

Tai moi thoi diem t, co mot tap hop cac trang da duoc su dung trong k lan truy cap gan day, duoc goi la Arbeitsbereich (Working Set).

- diese Menge w(t,k) ist der Arbeitsbereich
- Viele Betriebssysteme merken sich den (Working Set) eines Prozesses wenn sie ihn auslagern
- **Grund**: Prepaging - letzter aktueller Arbeitsbereich wird später wieder geladen bevor Prozess weiter ausgeführt wird
- **Ziel**: Verhindern von Seitenflattern (trashing) durch Seitenfehler (nur einige Nanosekunden zur Befehlsausführung, aber ca. 10ms eine Seite von Platte zu lesen)
- **Näherung**: w(t,k) aufwendig zu berechnen: Statt der letzten k Speicherzugriffe wird die Ausführungszeit des Prozesses verwendet.
- **Multiprozessing**: virtuelle Zeit (current virtual time)
- **Arbeitsbereich** eines Prozesses zum Zeitpunkt t: Die Seiten, im virtuellen Zeitintervall [t-τ,t]

9. Working Set Algorithmus (2)

Verfahren

Khi xay ra loi trang, mot trang khong thuoc Working Set se duoc xoa.

- Lagere bei einem Seitenfehler eine Seite aus, die nicht im Working Set liegt

1. Betrachte nur die eingelagerten Seiten. Info pro Seite: R-Bit, M-Bit, der letzte Zugriff auf die Seite (ungefähr)

   Chi xem xet cac trang da duoc luu tru. Thong tin cho moi trang bao gom bit R, bit M va thoi diem truy cap gan nhat cua trang do (xap xi).

2. R-Bit, M-Bit wird von der HW gesetzt. Zyklisch wird das R-Bit zurückgesetzt
3. Durchlaufe die Tabelle, bis eine Seite ersetzt wurde.

   Dat thoi gian truy cap thanh thoi gian ao hien tai

   (3.1) R-Bit == 1: Setze Zugriffszeit auf aktuelle virtuelle Zeit, R-Bit = 0

   (3.2) R-Bit == 0: if (aktuelle virtuelle Zeit – τ) > letzte Zugriffszeit : Seite liegt nicht im Working Set, wird ausgelagert und durch die neue Seite ersetzt

   Neu (thoi gian ao hien tai –) > thoi diem truy cap gan nhat: Trang khong thuoc Working Set, se duoc xoa va thay the bang mot trang moi.

4. Für die restlichen Seite : 3.1 – Zur Aktualisierung der Zugriffszeiten
5. Falls keine Seite mit 3.2 ausgelagert wurde, lagere die älteste Seite aus.

Beurteilung

- Verfahren pro Prozess
- Aufwendig: alle Seitenrahmen werden bei einen Seitenfehler bearbeitet.

Beispiel Working Set Algorithmus

virtuelle Zeit	τ
2204	200

...	Zugriffszeit	R-Bit
	2084	1
	2005	0
	1980	1
	2000	0
	1903	0
	2020	1
	2032	0
	1620	0

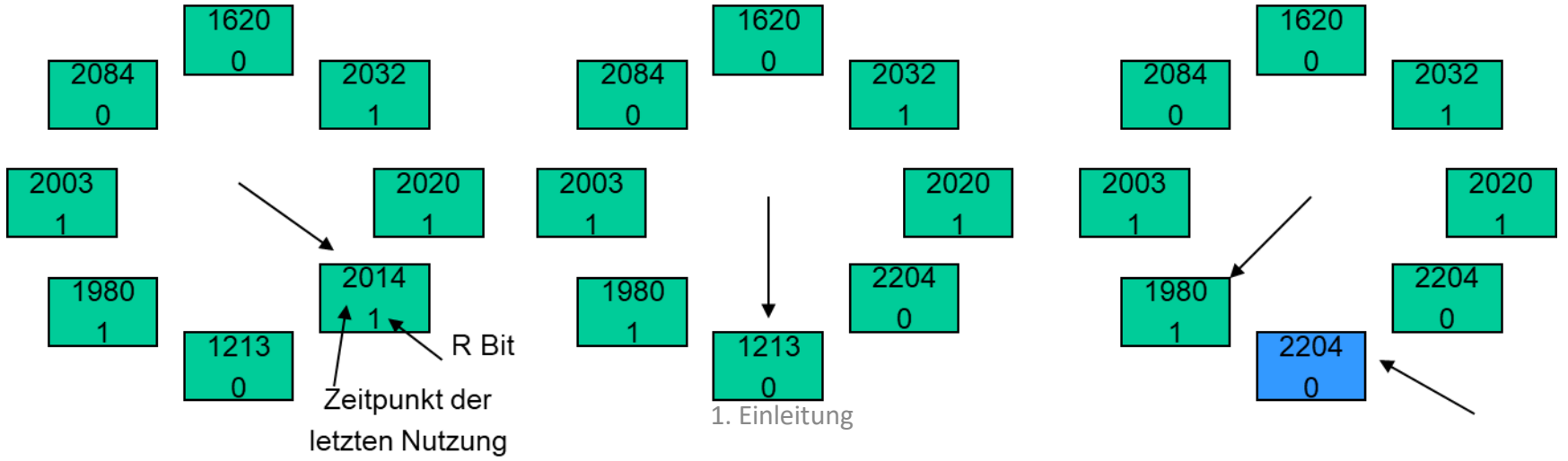
...	Zugriffszeit	R-Bit
	2204	0
	2005	0
	2204	0
Neue Seite	2204	0
	1903	0
	2204	0
	2032	0
	1620	0

10. WSClock: Working Set Clock Algorithmus

Vorgehensweise

Cac trang da duoc luu tru se duoc to chuc thanh danh sach lien ket vong trong bo nho. Moi trang co cac thong tin nhu bit R, bit M va thoi diem truy cap gan nhat (theo thuat toan Working Set).

1. Die eingelagerten Seiten stehen als zyklisch verkettete Liste im Speicher
   Pro Seite: R-Bit, M-Bit, der letzte Zugriff auf die Seite (gemäß WS Algorithmus)
2. R-Bit, M-Bit wird von der HW gesetzt. Zyklisch wird das R-Bit zurückgesetzt
3. Bei Seitenfehler untersuche zunächst Seite auf die der Zeiger zeigt
   if (R-Bit ==1) then
   Neu bit R bang 1, dat bit R ve 0, thoi gian truy cap la thoi gian ao hien tai, dich con tro sang trang ke tiep.
   R-Bit = 0, Zugriffszeit = aktuelle virtuelle Zeit, schiebe den Zeiger eine Seite weiter
   if (R-Bit==0) && (Seitenalter ≥ τ) then
   Neu bit R bang 0 va trang da lon hon hoac bang thoi gian quy dinh (t), xoa trang ra khoi bo nho, thay the bang mot trang moi, va cap nhat thoi gian truy cap thanh thoi gian ao hien tai.
   Seite auslagern, neue Seite einlagern, Zugriffszeit = aktuelle virtuelle Zeit
   Cap nhat thong tin thoi gian truy cap va bit R cho cac trang con lai trong danh sach neu bit R bang 1.
4. Laufe noch den Rest der Liste durch: Update von Zugriffszeit und R, wenn R == 1



10. WSClock: Working Set Clock Algorithmus (2)

Optimierung

...

3. …
   if (R-Bit==0) && (Seitenalter ≥ τ) then
      Seite auslagern, neue Seite einlagern, Zugriffszeit = aktuelle virtuelle Zeit

Ansatz

- Seite wird nur ausgelagert, wenn das M Bit nicht gesetzt ist
- Wenn das M Bit gesetzt ist wird das System mit Auslagerung der Seite beauftrag

Trang chi duoc xoa neu bit M cua trang khong duoc dat.

Beurteilung

- Wird aufgrund der guten Leistung und einfachen Umsetzbarkeit häufig eingesetzt

Zusammenfassung Ersetzungsstrategien

Algorithmus	Kommentar
Optimal	Nicht realisierbar, aber nützlich als Maßstab
NRU (Not Recently Used)	Sehr grobe Annäherung an LRU
FIFO (First In First Out)	Entfernt evtl. auch wichtige Seiten
Second Chance	Enorme Verbesserung gegenüber FIFO
Clock	Realistisch
LRU (Least Recently Used)	Exzellent, aber schwierig zu implementieren
NFU (Not Frequently Used)	Ziemlich grobe Annäherung an LRU
Aging	Effizienter Algorithmus, gute Annäherung an LRU
Working Set	Etwas aufwändig zu implementieren
WSClock	Guter und effizienter Algorithmus

Abbildung 3.22: Die behandelten Seitenersetzungsalgorithmen

[AT]