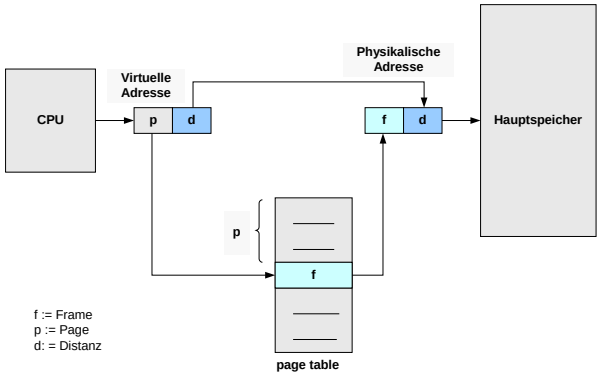
MAS: Betriebssysteme

Speicherverwaltung - Grundlegende Konzepte

T. Pospíšek

Gesamtüberblick

1. Einführung in Computersysteme
2. Entwicklung von Betriebssystemen
3. Architekturansätze
4. Interruptverarbeitung in Betriebssystemen
5. Prozesse und Threads
6. CPU-Scheduling
7. Synchronisation und Kommunikation
8. **Speicherverwaltung**
9. Geräte- und Dateiverwaltung
10. Betriebssystemvirtualisierung



Zielsetzung

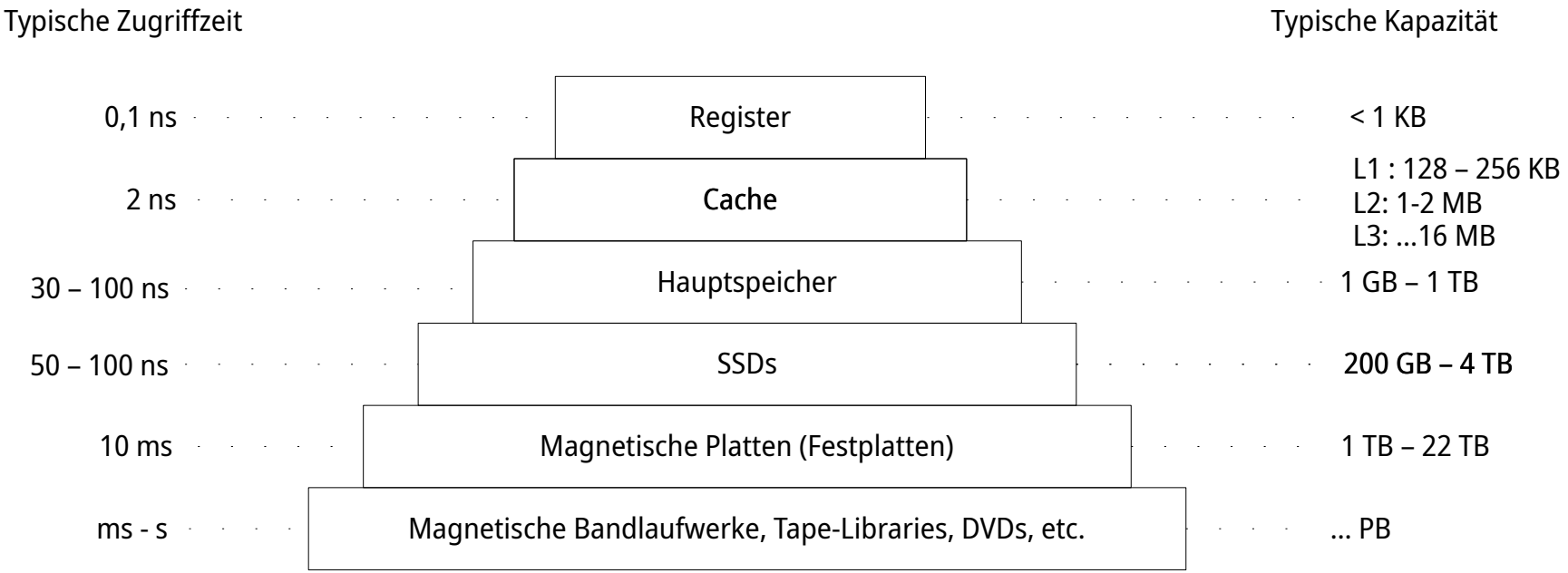
- Die Grundlagen der Speicherverwaltung, insbesondere des Hauptspeichers, kennenlernen und verstehen
- Die virtuelle Speichertechnik sowie einige Optimierungskonzepte für den virtuellen Speicher verstehen

Überblick

1. **Einführung in die Speicherverwaltung**
2. Grundprinzipien des virtuellen Speichers
3. Optimierung der virtuellen Speichertechnik

Speicherhierarchie moderner Rechnersysteme

- Schneller Speicher ist teuer und daher in Rechnersystemen knapp



- siehe auch Animationen von Ben Dickens:
- RAM vs CPU Cache
- RAM vs SSD

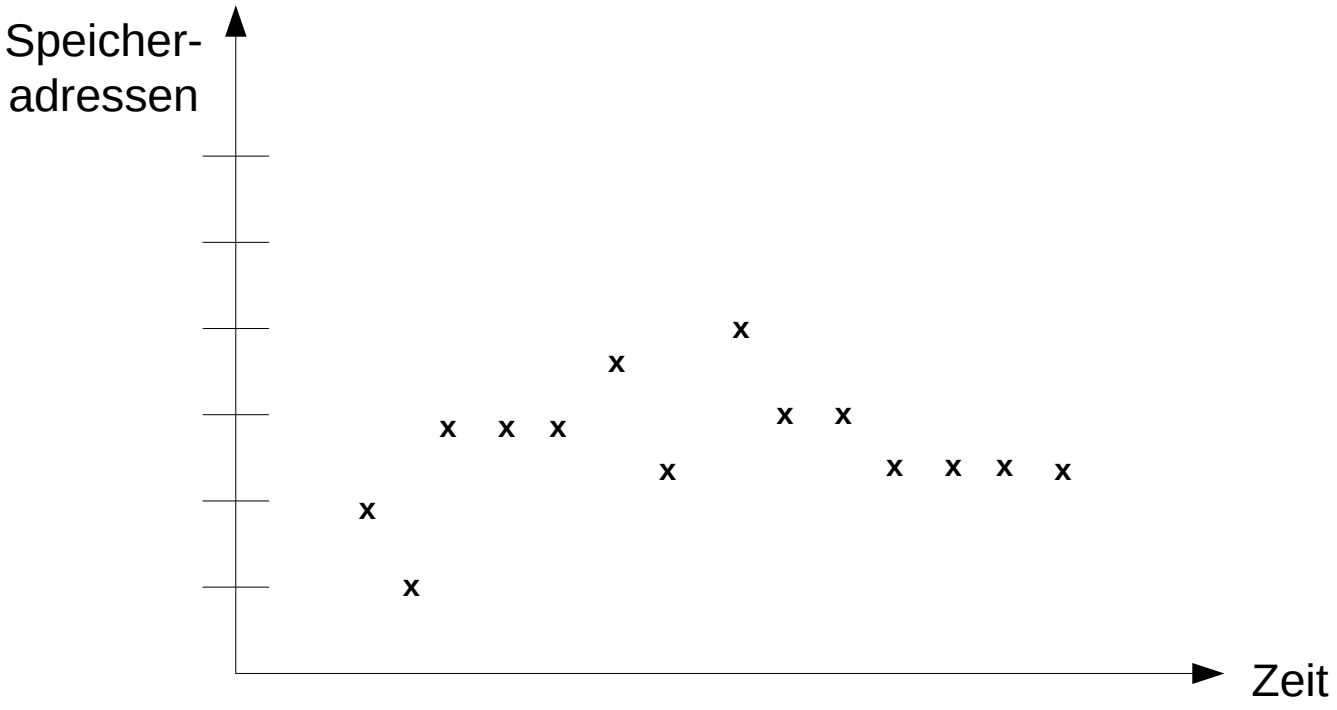
Aufgaben der Speicherverwaltung

- Wir betrachten im Folgenden die **Hauptspeicherverwaltung**
- Aufgabe des Betriebssystems:
  - Versorgung der Prozesse mit dem Betriebsmittel „Arbeitsspeicher“ (Hauptspeicher)
- Verantwortliche Softwarekomponente: **Memory Manager (Speicherverwalter)**
- Der Memory Manager verwaltet die freien und belegten Speicherbereiche

Lokalitätsprinzip

- **Zeitlich**: Daten/Code-Bereiche, die gerade benutzt werden, werden mit hoher Wahrscheinlichkeit gleich wieder benötigt

→ Diese sollten für den nächsten Zugriff bereitgehalten werden

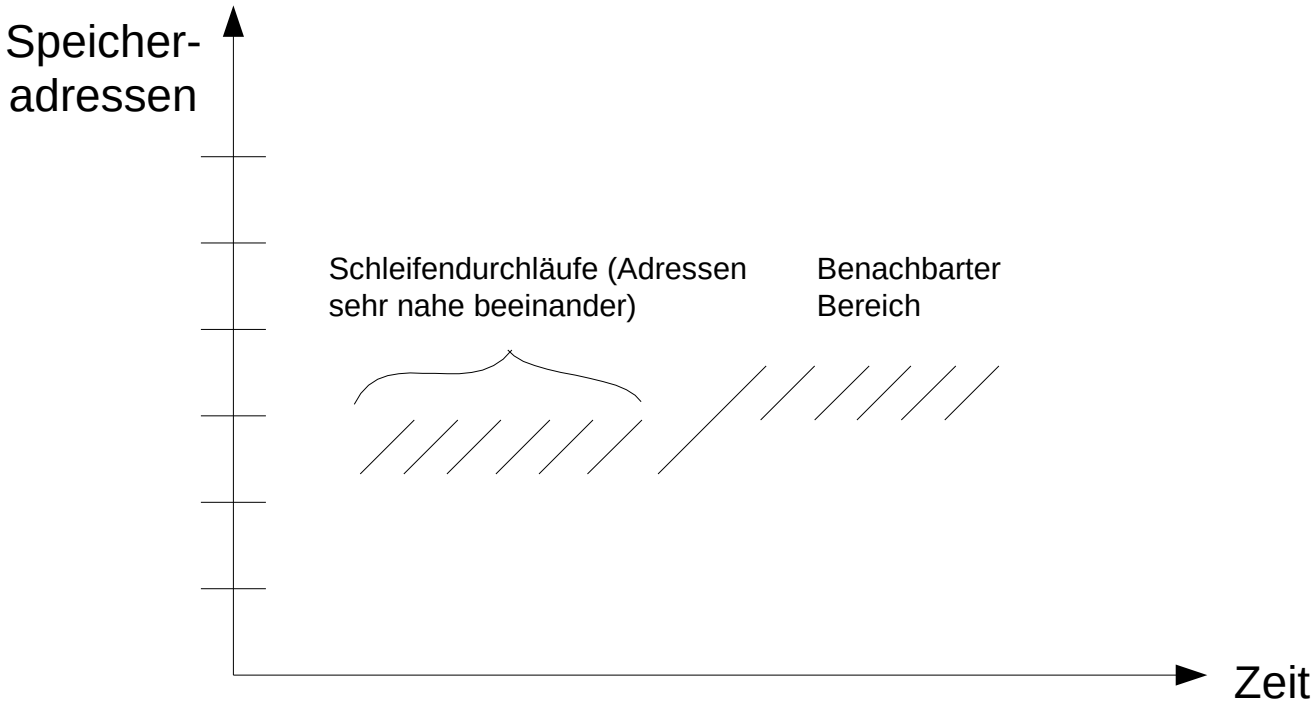


Lokalitätsprinzip

- **Örtlich**: Nächster Daten/Code-Zugriff ist mit hoher Wahrscheinlichkeit in der Nähe der vorherigen Zugriffe

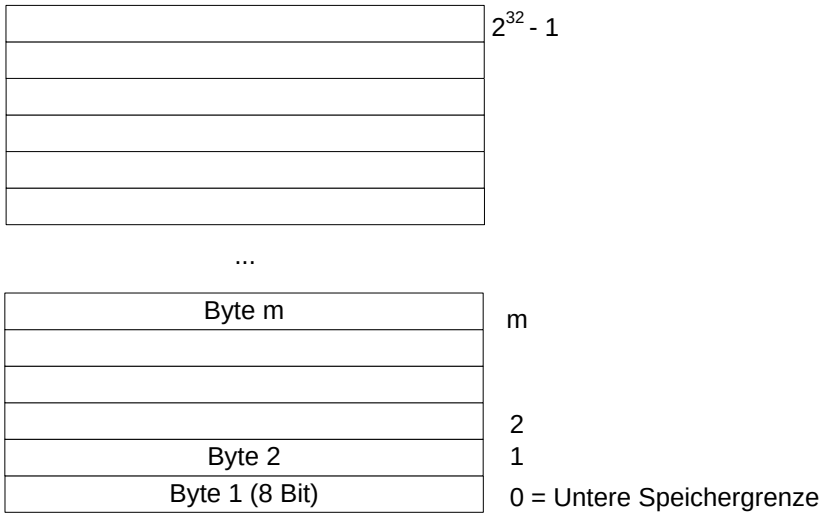
→ Benachbarte Daten beim Zugriff auch gleich in schnelleren Speicher laden

→ „prefetch“



Adressen und Adressräume

- Hauptspeicher ist in logisch adressierbare **Speicherstellen** unterteilt, meist byteweise (8 Bit)
- Ein Byte ist also die kleinste adressierbare Einheit,
  → Alignement
- 32-Bit-Adressen → 2^{32} adressierbare Bytes,
  allerdings → Segmente
- Ein **Adressraum** ist die Menge aller adressierbaren Adressen
  - 32-Bit-Adressen → {0, 1, 2, … , 2^{32}-1}



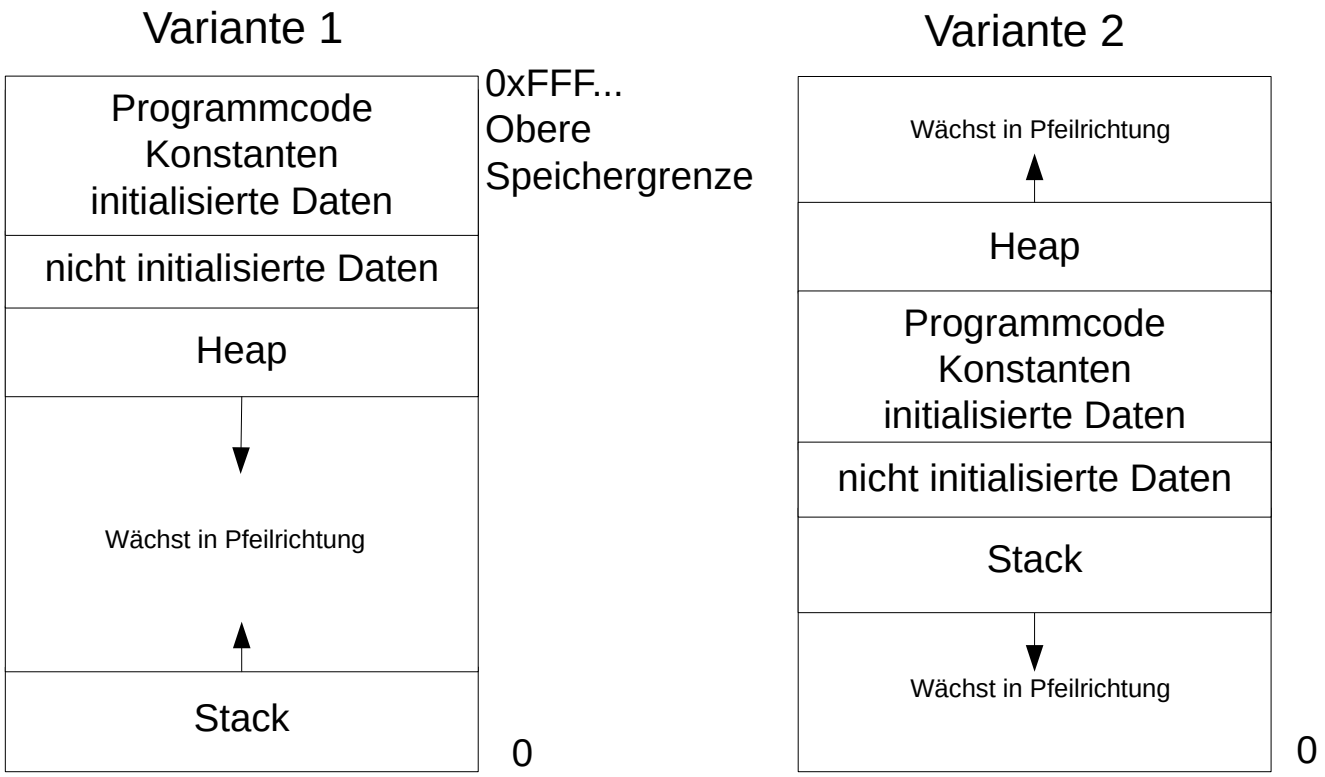
- 2^{32} → 4GB
- 2^{40} → 1PB
  Addressbus Breite!
- 2^{64} → 16 Mio PB

Adressraumbelegung

- Wird durch Adressraumbelegungsplan bestimmt
- Festlegung im Betriebssystem
- Ausrichtung auf Maschinenwörter wichtig wegen optimalem Zugriff
- Bereich für Anwendungsprogramme und Anwendungsdaten organisiert der Compiler/Interpreter bzw. das Laufzeitsystem

Adressraumbelegung für Programme

- Abhängig von der Programmiersprache
- Mehrere Varianten möglich

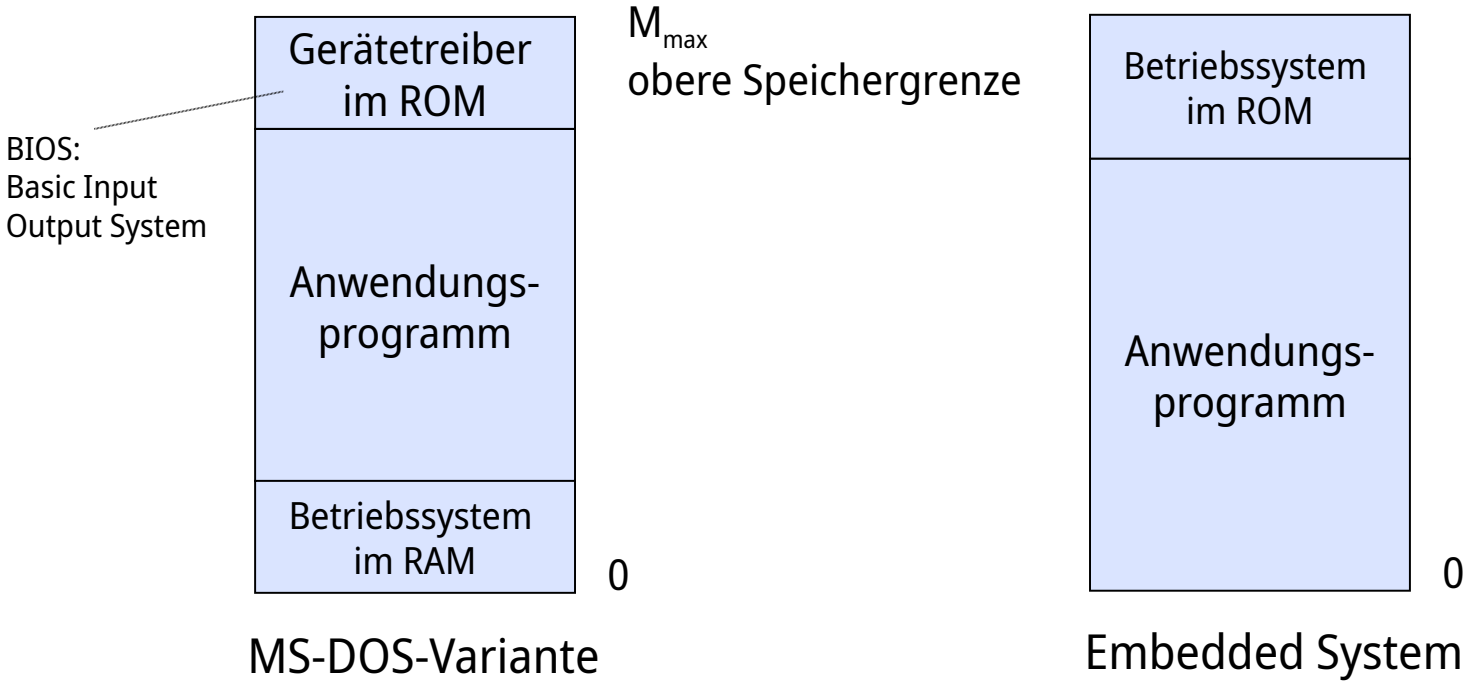


Verschiedene Mechanismen der Speicherverwaltung

- Es gibt verschiedene Mechanismen für die Speicherverwaltung
- Historische Entwicklung:
  - Speicherverwaltung bei Monoprogramming
  - Speicherverwaltung mit festen Partitionen
  - Overlay-Technik
  - Swapping
  - Virtueller Speicher

Speicherverwaltung bei Monoprogramming

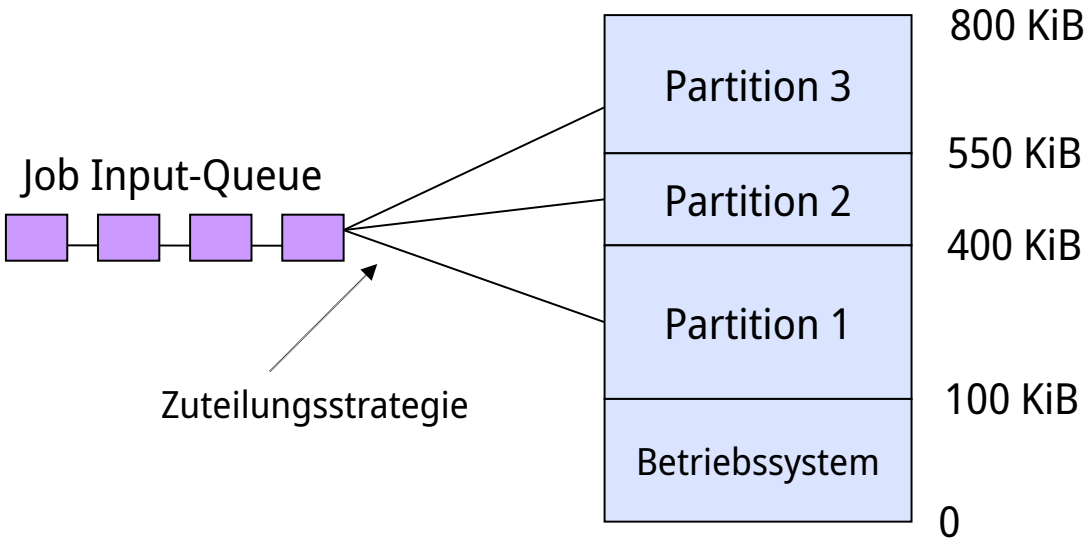
- Einfachste Form der Speicherverwaltung
- Nur ein Programm läuft zu einer Zeit



- **BIOS** = Programm zum Starten eines Rechnersystems, bis das Betriebssystem übernimmt. Es liegt in einem nicht flüchtigen ROM oder in einem Flashspeicher
- Weiterentwicklung von BIOS: **EFI** = Extensible Firmware Interface, unterstützt auch 64-Bit-Systeme

Speicherverwaltung mit festen Partitionen

- Aufteilung des Speichers in **feste Teile** (Partitionen)
- Multiprogramming und Verbesserung der CPU-Auslastung möglich
- Job wird in eine Queue eingetragen
  - Für jede Partition eine Queue oder eine globale Queue



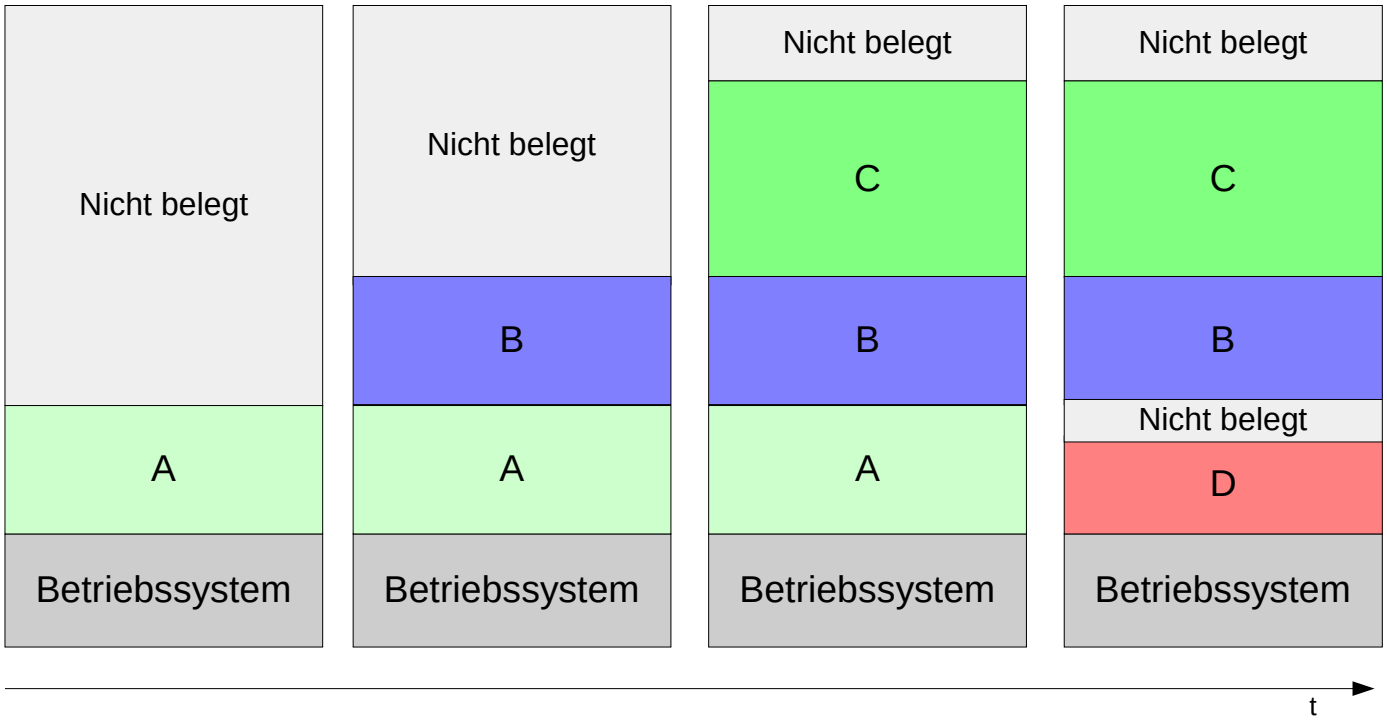
IBM Mainframe OS/360 Variante

Speicherverwaltung bei Multiprogramming mit Swapping

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

- **Grundgedanke:** Timesharing!
  - Es passen nicht immer alle Prozesse in den Hauptspeicher
  - Prozess wird im Gesamten geladen
  - Prozess wird nach einer gewissen Zeit wieder auf einen Sekundärspeicher (Platte) ausgelagert
  - Entstehende Löcher können durch Kombination benachbarter Speicherbereiche eliminiert werden, aber aufwändig!
- **Hauptunterschied** zu festen Partitionen:
  - Anzahl, Speicherplatz und Größe des für einen Prozess verwendeten Speicherbereichs variieren dynamisch
  - Prozess wird immer dahin geladen, wo gerade ausreichend Platz ist

Speicherverwaltung bei Multiprogramming mit Swapping



A - D: Prozesse

vgl. Tanenbaum

Überblick

1. Einführung in die Speicherverwaltung
2. **Grundprinzipien des virtuellen Speichers**
3. Optimierung der virtuellen Speichertechnik

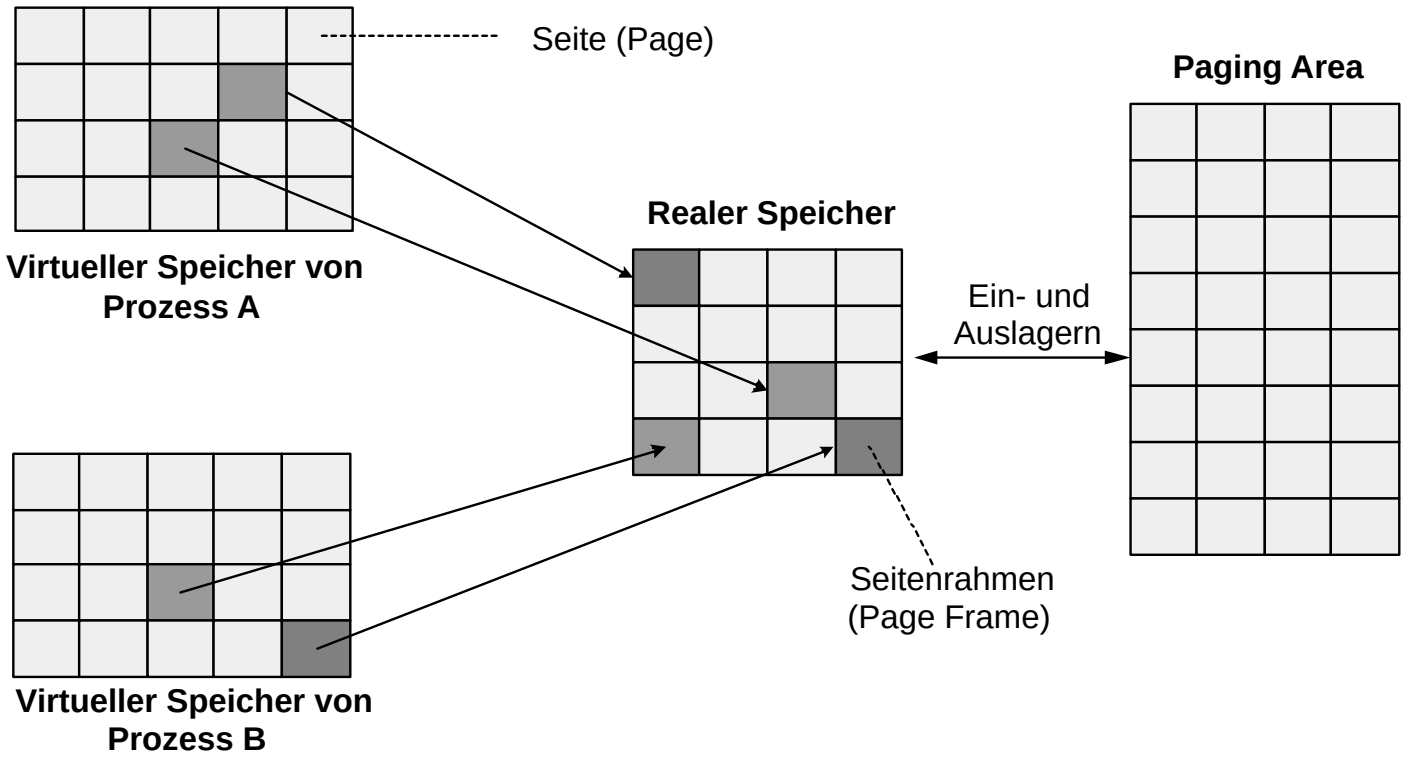
Grundlegende Überlegungen

- Grundlegende Ideen zum virtuellen Speicher
  - Speichergröße eines Programms inkl. Daten und Stack darf den vorhandenen **physikalischen Hauptspeicher überschreiten**
  - Prozess kann auch ablaufen, wenn er **nur teilweise im Hauptspeicher** ist
  - Programmierer soll sich am besten nur mit einem **linearen Adressraum** befassen müssen
- Das Betriebssystem hält die gerade benutzten Teile im Hauptspeicher und den Rest auf einer Festplatte
- Primäre Nutzung in Multiprogramming-Systemen

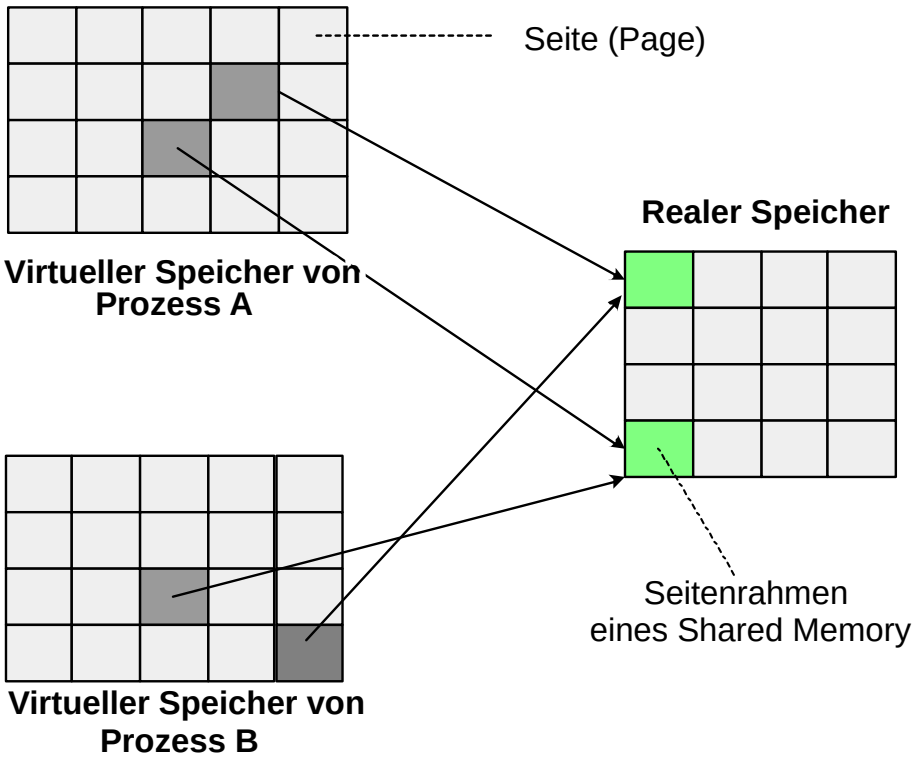
Grundprinzip und Grundbegriffe (1)

- Virtueller Adressraum
- Realer Adressraum
- Seiten (Pages - virtuel)
- Seitenrahmen (Frames - physisch)
- Paging Area, Swap (Schattenspeicher)
  - Für den Hauptspeicher wird ein Schattenspeicher in einem speziellen Plattenbereich reserviert (**Paging Area**)
- Mapping: Page → Frame

Grundprinzip und Grundbegriffe (2)



Einschub: Shared Memory



Grundprinzip und Grundbegriffe (3)

- Wie funktioniert das Mapping von virtueller Adresse auf eine reale Adresse?
- Wie wird der virtuelle Speicher verwaltet?
- Was macht die Hardware, was macht die Software?
- Wie groß sind Pages und Frames?
- Was ist, wenn der Hauptspeicher voll ist, aber ein Prozess noch Speicher anfordert?
  - Seitenersetzung, Verdrängung
- Welche Probleme ergeben sich und wie werden sie gelöst?

Strategien zur Verwaltung von virtuellem Speicher

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

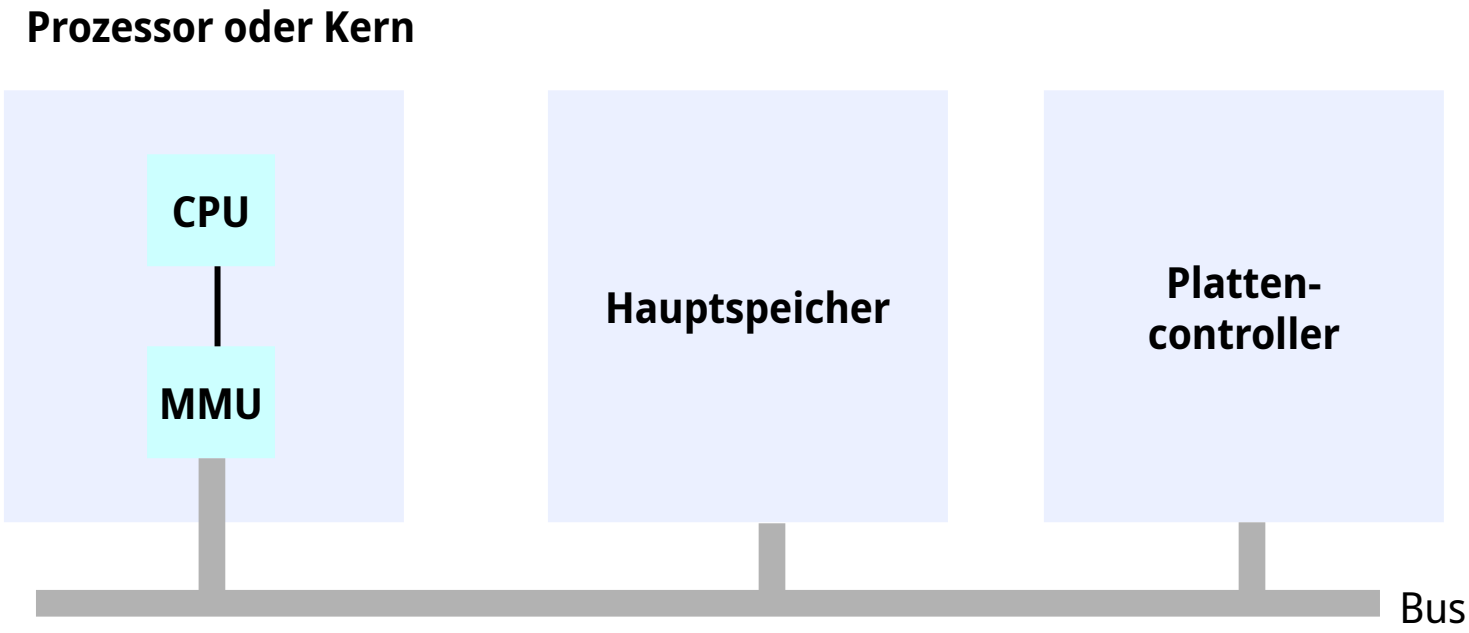
- Abrufstrategie - Fetch Policy
  - Demand Paging oder Prepaging
  - wann Seite holen?
- Speicherzuteilungsstrategie - Placement Policy
  - wohin Seite legen? Gibt's bessere und schlechtere Orte?
- Austauschstrategie - Replacement Policy
  - welche Seite rausschmeissen?
- Aufräumstrategie - Cleaning Policy
  - wann soll eine veränderte Seite geschrieben werden?

Paging

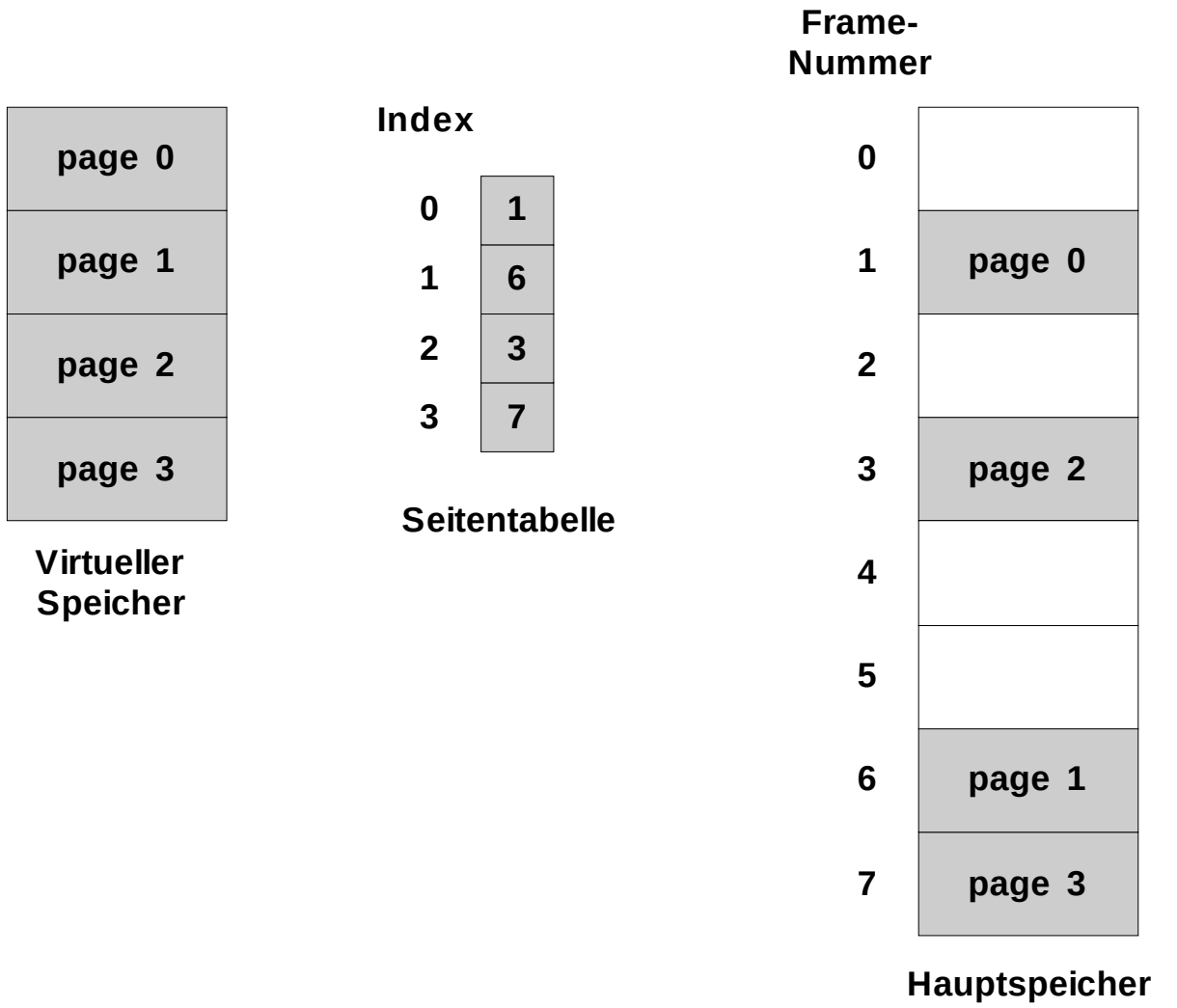
- Die Umlagerung zwischen Hauptspeicher und Platte wird als **Paging** bezeichnet
- Jeder Prozess darf alle Adressen verwenden, die aufgrund der HW-Architektur des Rechners möglich sind
  - unabhängig von der realen Größe des Hauptspeichers
- Bei Systemen mit 32-Bit-Adressen kann jeder Prozess einen Adressraum von 4 GiB verwenden
  - Dies gilt auch, wenn der Hauptspeicher z.B. nur einige MiB realen Speicher hat
  - Dies hat aber seine Grenzen, wenn das System nicht ausschließlich mit **Paging** beschäftigt sein soll!

Hardwareunterstützung durch die MMU

- MMU = Memory Management Unit (Hardware)
- CPU sendet virtuelle Adressen an die MMU
- MMU sendet reale Adressen an den Hauptspeicher

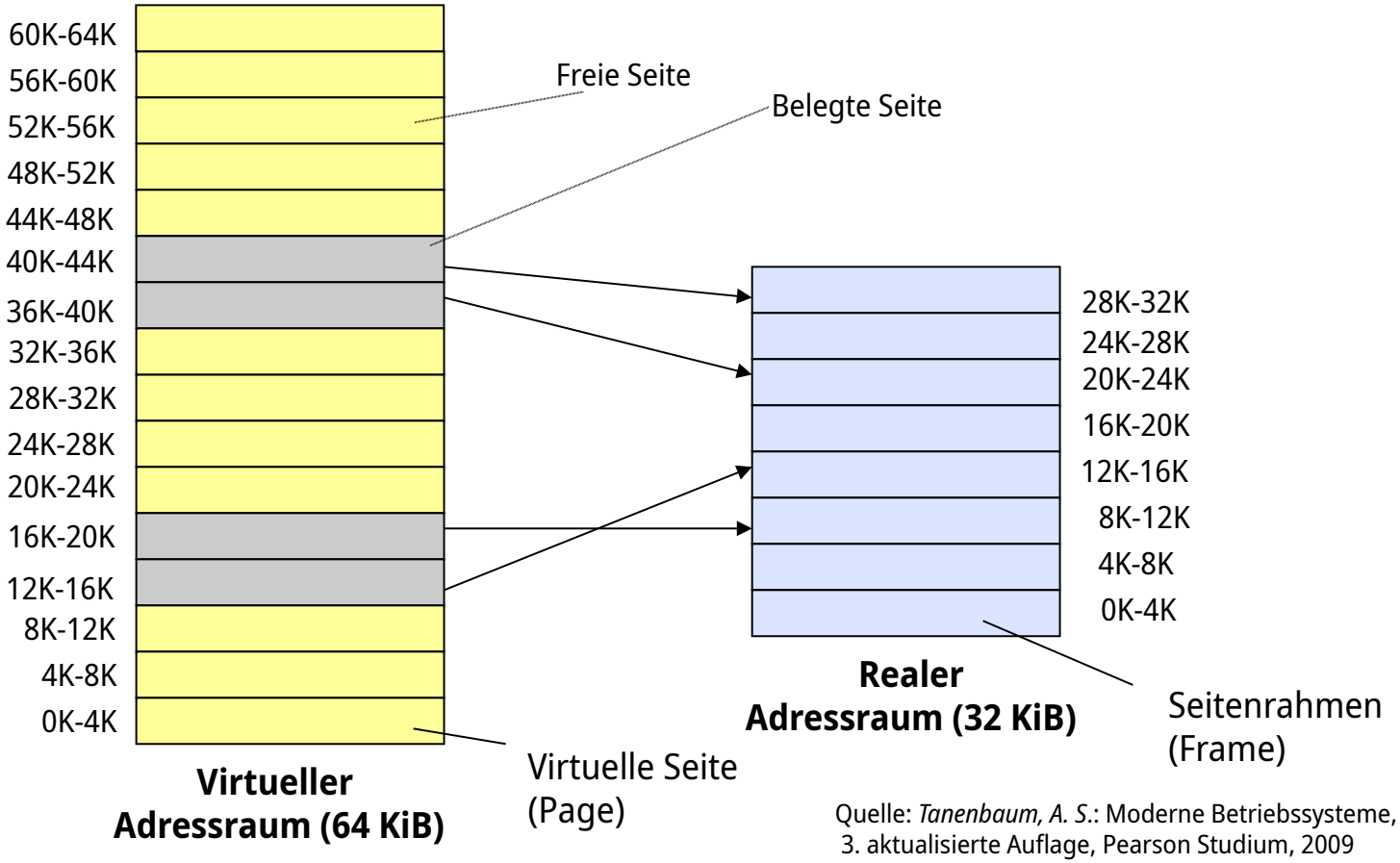


Seitentabellen, vereinfachtes Modell



Mapping

- Virtueller Adressraum → Realer Adressraum
- Hier ein Beispiel eines Adressraums:



Quelle: *Tanenbaum, A. S.*: Moderne Betriebssysteme, 3. aktualisierte Auflage, Pearson Studium, 2009

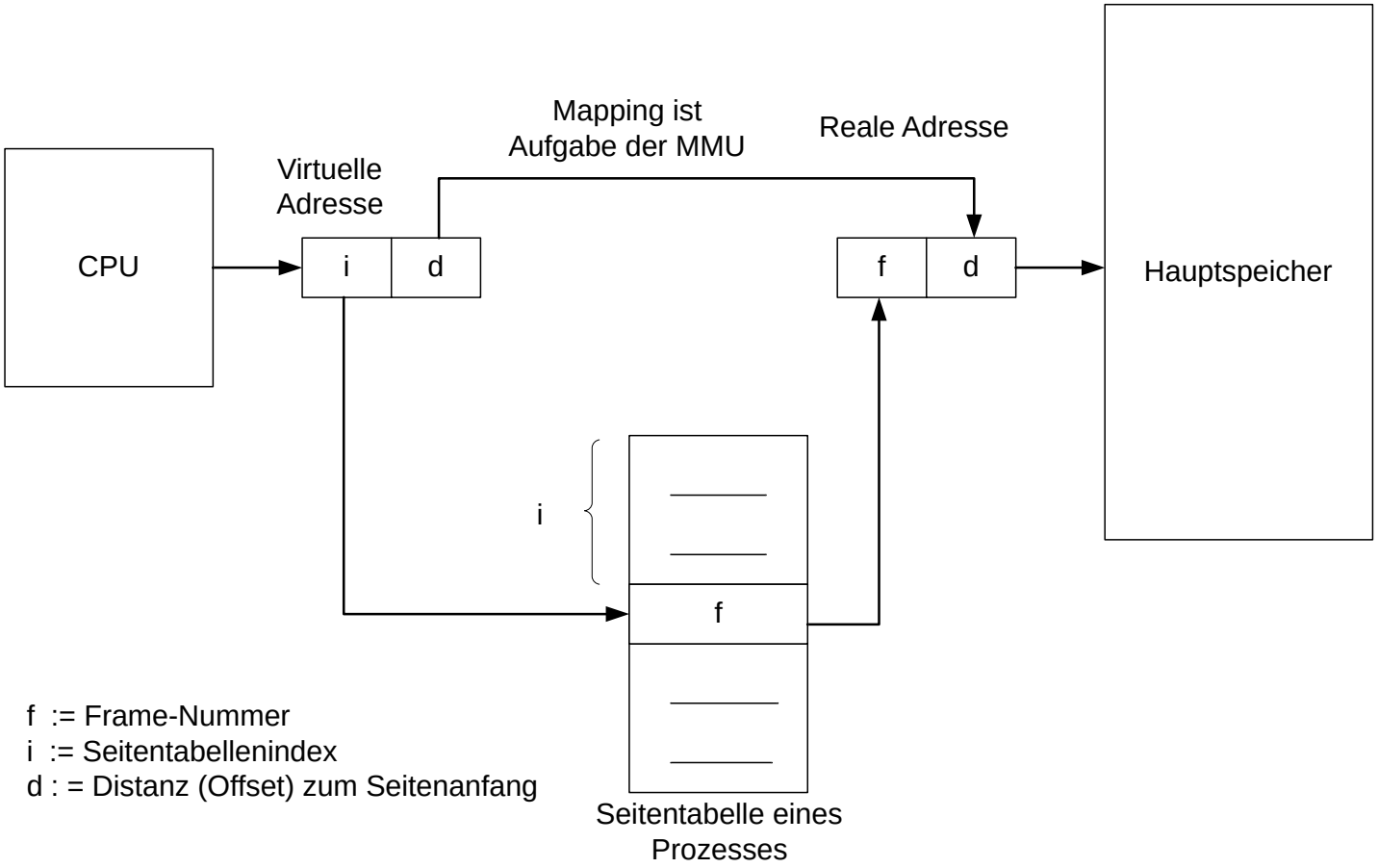
Pages und Frames

- Das Speicherabbild eines Prozesses besteht aus Speicherseiten (Pages)
- Eine Speicherseite ist ein Segment einer vorgegebenen Größe (z.B. 4 KiB)
- Nur die wirklich benötigten Speicherseiten müssen im Arbeitsspeicher geladen sein, während der Prozess läuft
- Die Größe der Seiten ist meist gering, die Anzahl beliebig groß

Adressumsetzung (1)

- Die virtuelle Adresse wird in die virtuelle Seitennummer und einen Offset geteilt
- Die virtuelle Seitennummer ist ein Index auf die Seitentabelle
- Über diesen Index wird der zugehörige Eintrag in der Seitentabelle gefunden
- Im Eintrag steht die Frame-Nummer, falls die Seite einem Frame zugeordnet ist
- Also: f(Seitennummer) → Frame-Nummer
  - Falls Seite im Speicher
  - Sonst: page fault

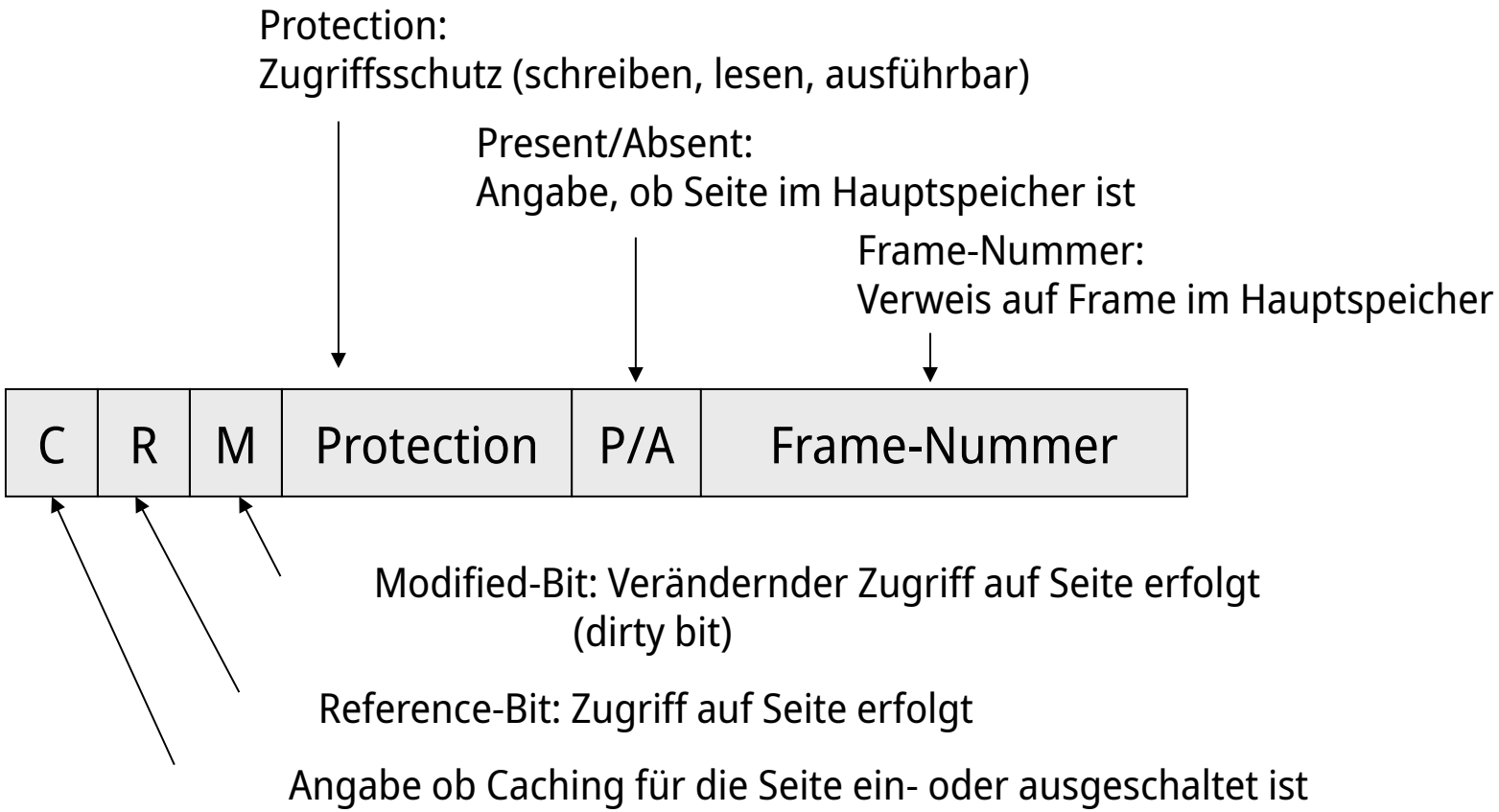
Adressumsetzung (2)



Register CR3 bei Intel enthält Page Directory Adresse

Einschub: Seitentabelleneintrag

- Der Aufbau eines Eintrags in der Seitentabelle hängt stark vom System ab, hier ein Beispiel:



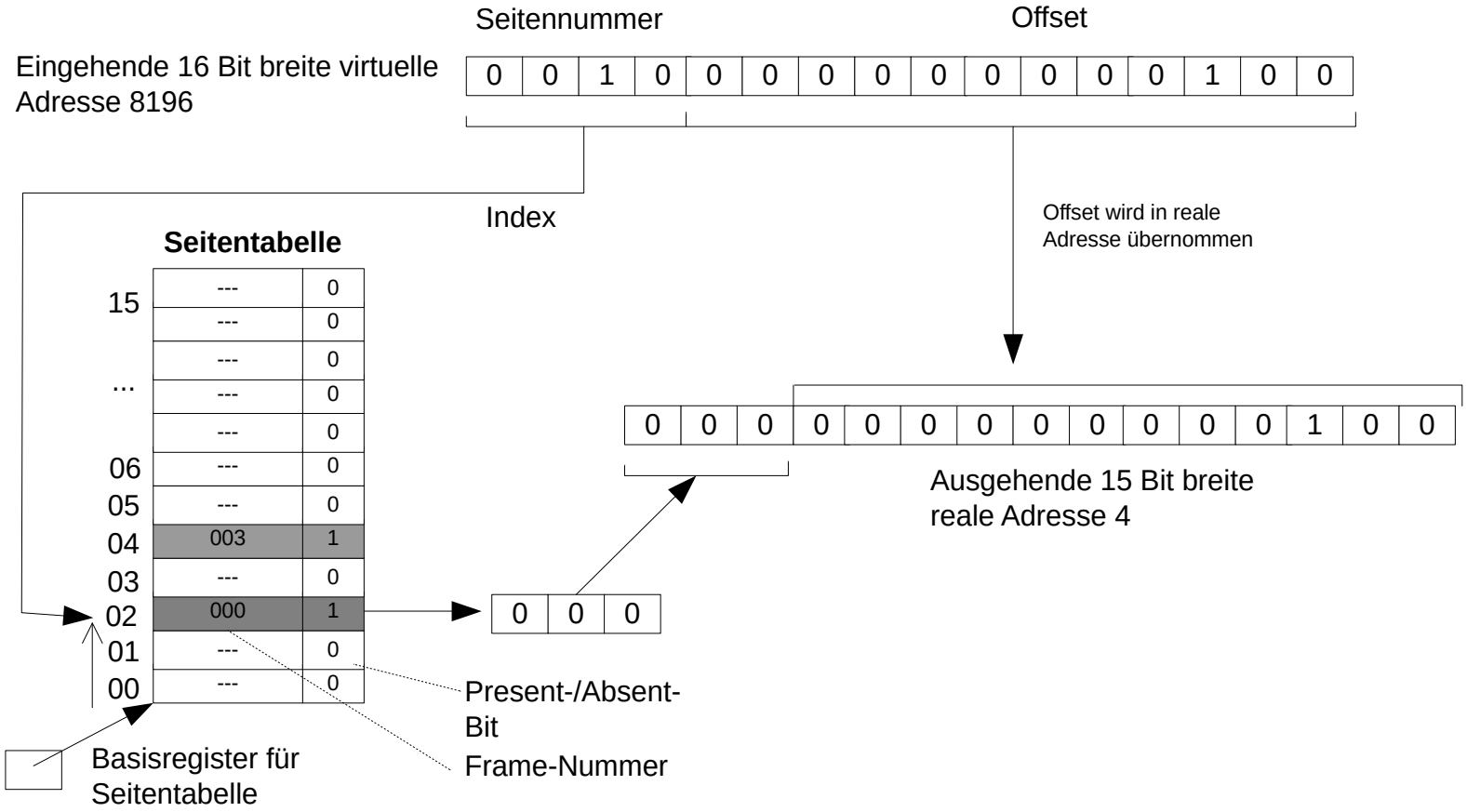
Adressumsetzung (3)

- Beispieladressierung:
  - Befehl: *MOVE R1, 8196* mit R1 = CPU-Register
  - Adresse befindet sich in Page 2 des virtuellen Adressraums, mit Offset 4 (2*4096+4=8196)
  - Adresse wird an die MMU gesendet
  - MMU schaut Page 2 in der Seitentabelle nach
    → Page 2 liegt im Frame 0
  - **MMU transformiert virtuelle Adresse 8196 in die physische Adresse 4** (0*4096 + 4 = 4)
  - Adressiert wird also im Hauptspeicher die Adresse 4 im untersten Frame

Adressumsetzung (3)

- Befindet sich eine angesprochene Adresse nicht im Hauptspeicher, verursacht die MMU bei der CPU einen **Trap** in das Betriebssystem (**page fault** genannt)
  - Seitenersetzungsstrategie notwendig

Adressumsetzung (4)



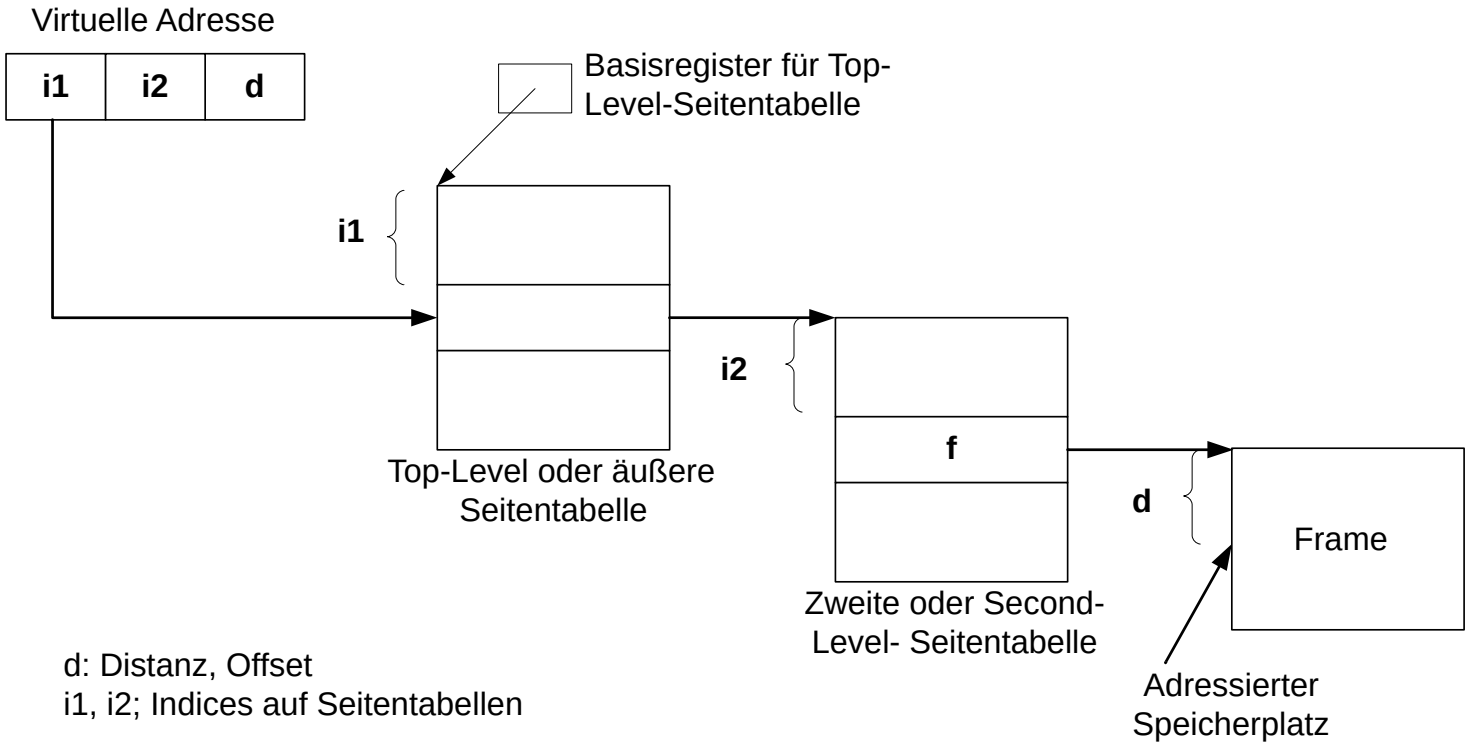
Quelle: *Tanenbaum, A. S.*: Moderne Betriebssysteme, 3. aktualisierte Auflage, Pearson Studium, 2009

Adressumsetzung (5)

- Das **Mapping** muss **schnell** sein
- In großen Adressräumen sind sehr große Seitentabellen möglich
  - Z.B. bei einem 32 Bit Adressraum → 1 Million Einträge in der Seitentabelle bei einer Seitengröße von 4 KiB
  - Bei 4 Byte pro Eintrag → 4 MiB Hauptspeicher notwendig
- Jeder Prozess benötigt seine eigene Seitentabelle
- Speicherersparnis durch **mehrstufige Seitentabellen** (zweistufige, dreistufig,...)
  - Damit wird erreicht, dass nicht immer alle Seitentabellen im Speicher gehalten werden müssen

Mehrstufige Adressumsetzung (1)

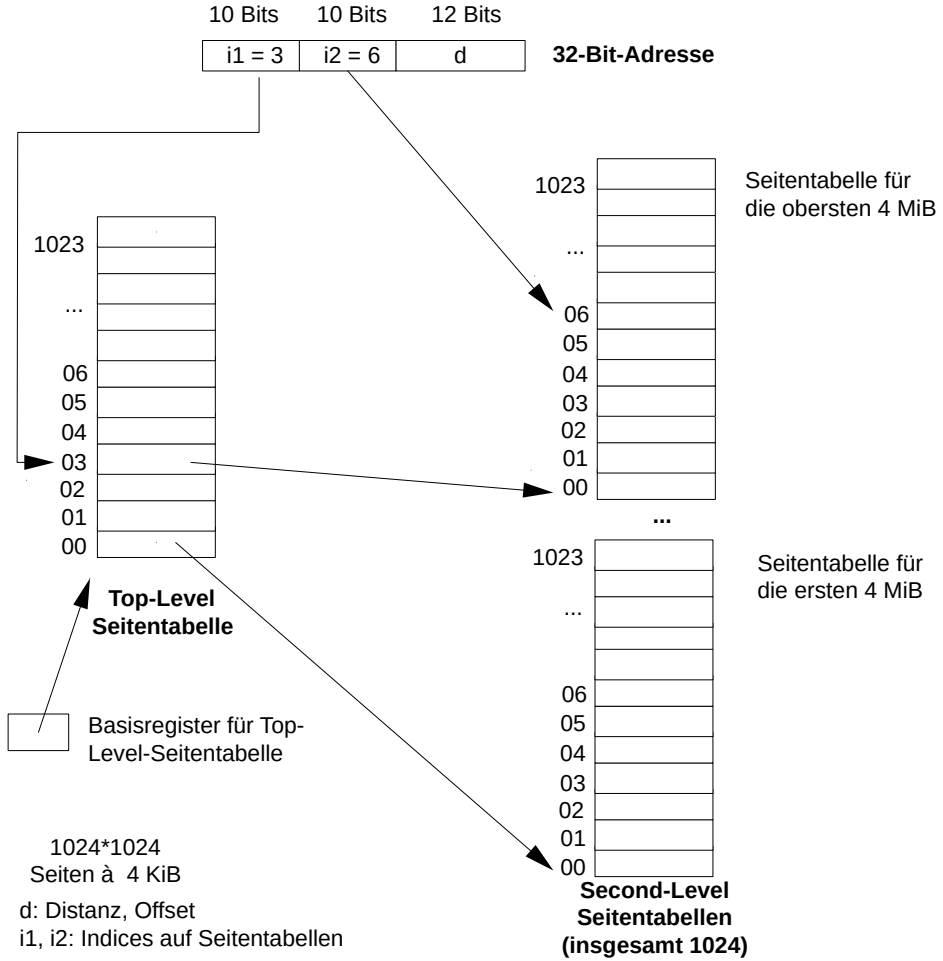
- Zweistufige Seitentabelle für 32-Bit-Adressen
  - Beispiel: i1 = 10 Bit, i2 = 10 Bit, d = 12 Bit



Quelle: *Tanenbaum, A. S.*: Moderne Betriebssysteme, 3. aktualisierte Auflage, Pearson Studium, 2009

Mehrstufige Adressumsetzung (2)

- Zweistufige Seitentabelle für 32-Bit-Adressen



Überblick

1. Einführung in die Speicherverwaltung
2. Grundprinzipien des virtuellen Speichers
3. **Optimierung der virtuellen Speichertechnik**

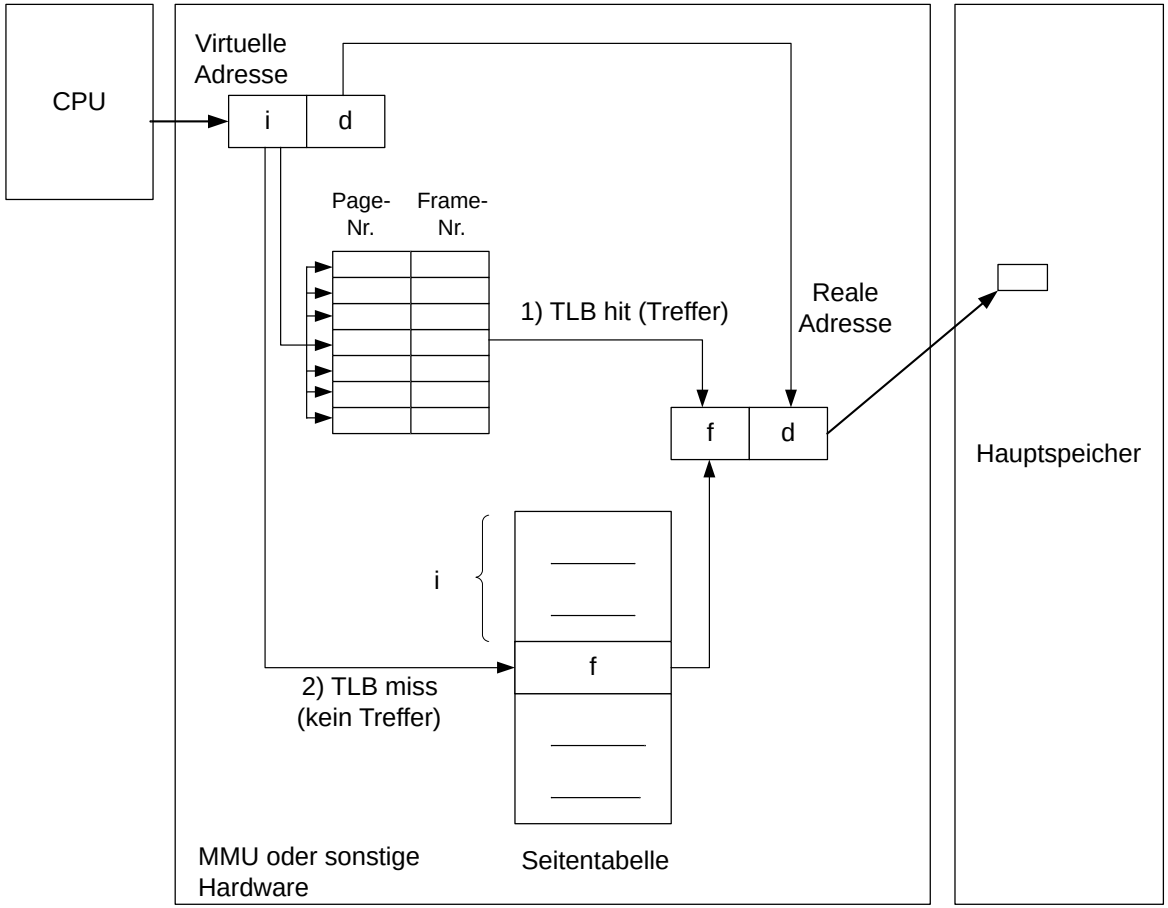
Zwischenbewertung zum virtuellen Speicher

- Die virtuelle Adressierung ist **relativ aufwändig**, da
  - viele umfangreiche Tabellen benötigt werden (Seitentabelle pro Prozess)
  - ein Teil der Festplatte als Paging-Area verwendet wird
  - laufend untersucht werden muss, ob Seiten hauptspeicherresident bleiben oder auf Platte auszulagern sind (Seitenersetzungsalgorithmus)
- Trotz des Overheads: Virtuelle Adressierung ist das heute am meisten verwendete Verfahren
- Optimierungen notwendig
  - Z.B. größere Seiten (Large Pages), bei 64-Bit-Prozessoren sinnvoll
  - Aber es gibt auch noch andere Möglichkeiten

Optimierung durch Adressumsetzpuffer (1)

- Ein Adressumsetzpuffer (Translation Lookaside Buffer, **TLB**) ist ein schneller Speicher
- Zuordnung von virtuellen auf reale Adressen für die aktuell am häufigsten benötigten Adressen
- Bei der Adressumsetzung wird zuerst in den TLB geschaut
- Bei Hit: kein Zugriff auf Seitentabelle notwendig
- Einsparung von Hauptspeicherzugriffen
- Beträchtliche Leistungsoptimierung möglich
- TLB ist Bestandteil der MMU

Optimierung durch Adressumsetzpuffer (2)



Optimierung durch Adressumsetzpuffer (3)

- Ein TLB-Eintrag enthält
  - virtuelle Seitennummer
  - Verweis auf den Seitenrahmen im Hauptspeicher
  - Tag zur Adressraum-Identifikation, z.B. Prozess-Identifikation (PID) → Tagged TLB
    - Grund: Virtuelle Adresse allein ist im Betriebssystem nicht eindeutig
    - evtl. nicht benötigt, wenn TLB bei Kontextwechsel komplett gelöscht wird (TLB flush)

Prozess mit PID 13

Virtuelle Seitennummer 12

Virtuelle Seite 12 des Prozesses mit PID 13 liegt im Hauptspeicher im Seitenrahmen 121!

Tagged TLB (PID = Tag)

Virtuelle Seite 05	PID 10	Seitenrahmen 200
Virtuelle Seite 12	PID 10	Seitenrahmen 098
Virtuelle Seite 43	PID 17	Seitenrahmen 028
Virtuelle Seite 12	PID 13	Seitenrahmen 121
Virtuelle Seite 04	---	ungültig

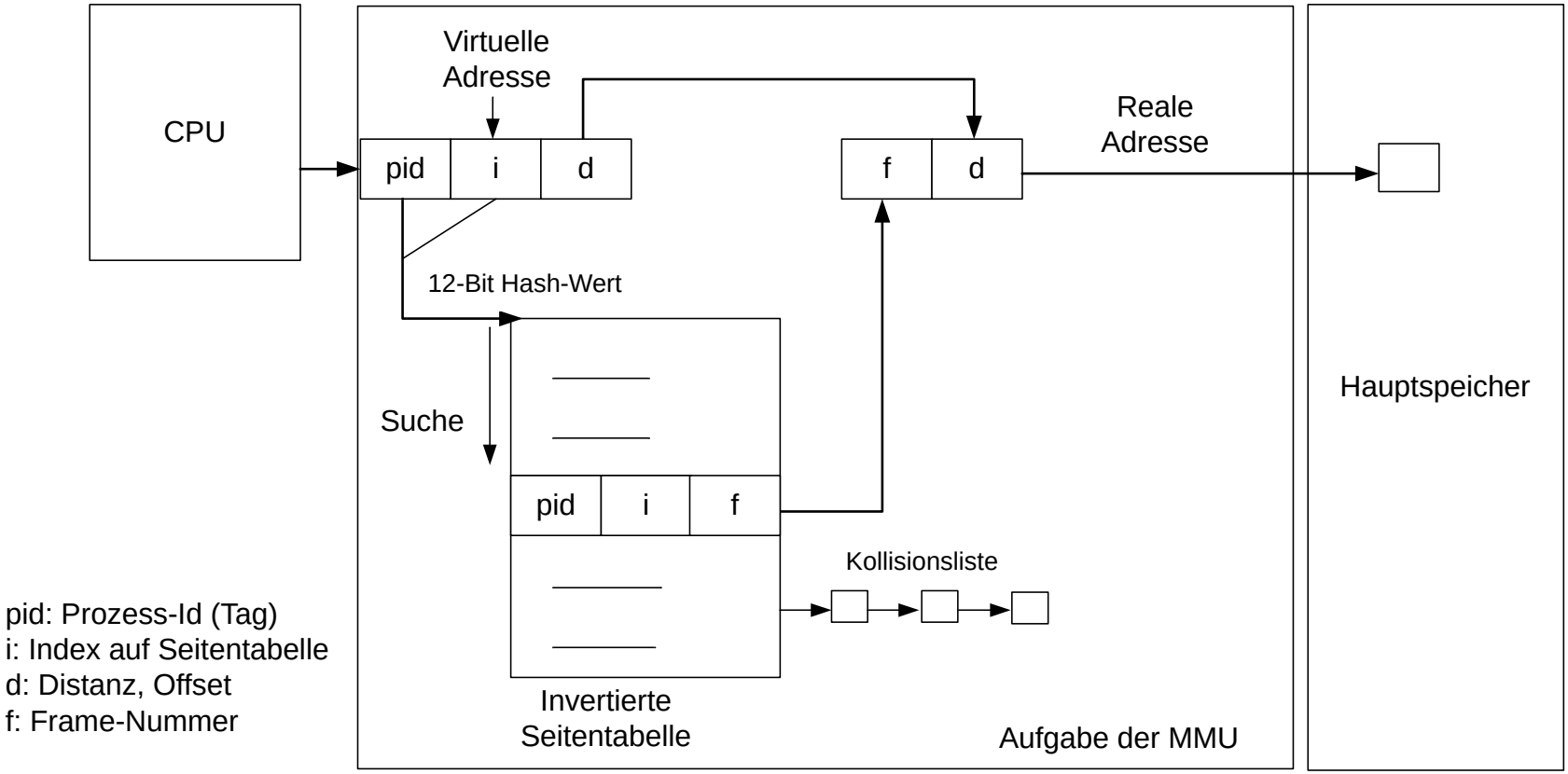
...

Optimierung durch invertierte Seitentabellen (1)

- Bei 64-Bit-Prozessoren ist der virtuelle Speicher viel größer als der reale
- Man bräuchte eher 6 Seitentabellen-Ebenen → Immenser Rechenaufwand
- Idee:
  - Man legt nur eine Tabelle an, in der man reale Adressen auf virtuelle abbildet, also **invertiert** vorgeht --> invertierte Sicht
  - Ein Eintrag pro Frame in einer **invertierten Seitentabelle**
- Vorteil: wesentlich weniger Tabelleneinträge: nur noch so viele wie man Seitenrahmen im Hauptspeicher zur Verfügung hat
- Nachteil: in der Seitentabelle keine Ordnung nach virtuellen Adressen

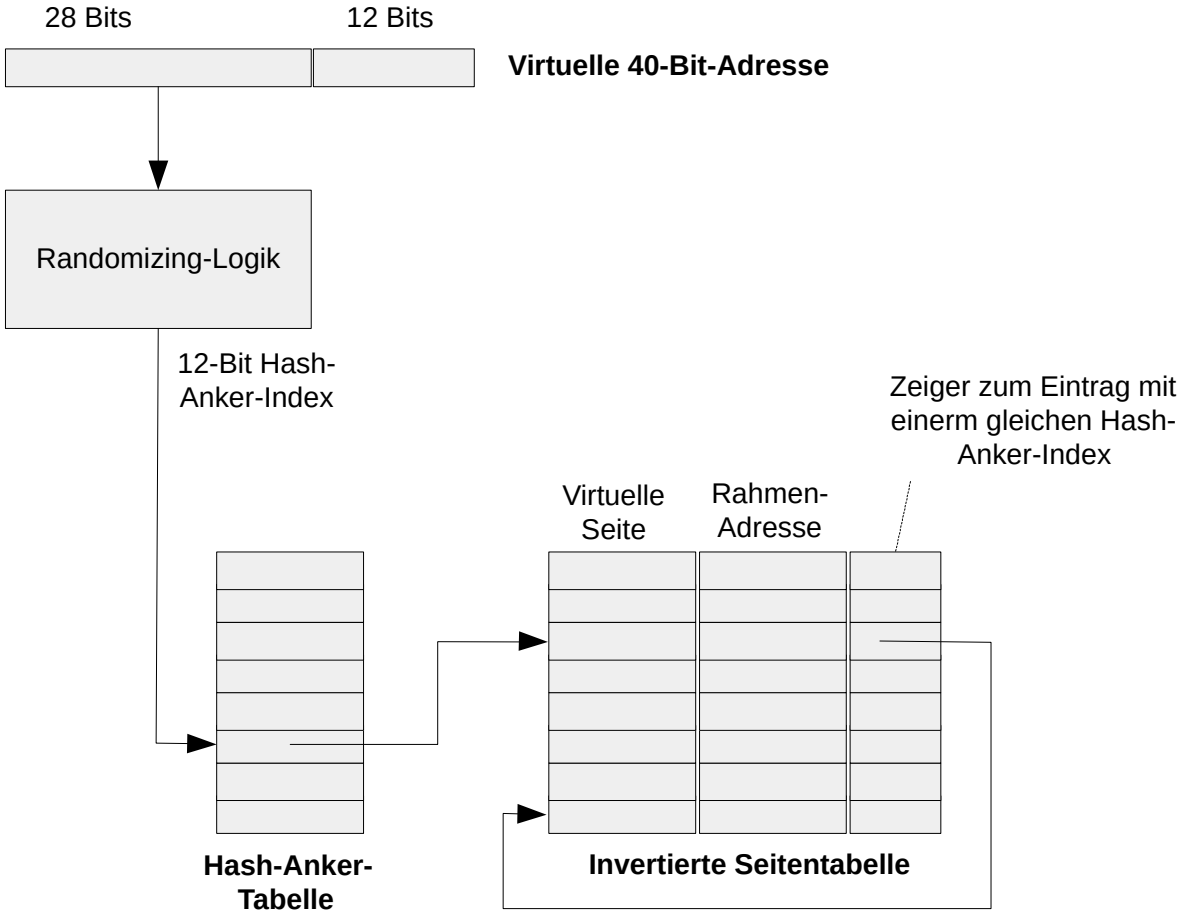
  → Suche etwas aufwändiger, da nicht über Seitentabellenindex positioniert werden kann
- Kombination mit TLB ist üblich

Optimierung durch invertierte Seitentabellen (2)



pid: Prozess-Id (Tag)
i: Index auf Seitentabelle
d: Distanz, Offset
f: Frame-Nummer

Invertierte Seitentabellen am Beispiel RS/6000-Prozessor



vgl. Herrmann (2001) S. 113

- Beispiel 1: **IA64-Architektur (Intel)**
  - Echte 64-Bit-Adressen bei 12 Bit Distanz
    → 2^{52} Seitentabelleneinträge (einstufig)
  - Adressraum: 2^{64} = 16 EiB ($16 * 10^{18}$)
  - Bei Seitentabellen: Bei 4 Byte je Eintrag
    → ca. 18 PiB = $18 * 10^{15}$ für alle Seitentabellen
  - Nutzt daher **gehashte invertierte Seitentabelle**
- Beispiel 2: **x64-Architektur (AMD)** und **Intel 64**
  - Nutzen 48 Bit für die virtuelle Adresse, davon 36 Bit für die Seitentabellen, also 2^{36} Einträge
  - **Nutzen noch 4-stufige Seitentabelle**
  - Adressraum: 2^{48} = 256 TiB
  - Bei 4 Bytes je Eintrag
    → ca. 275 GiB für alle Seitentabellen

Virtuelle Speichertechnik: Vorteile zusammengefasst

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

- Prozesse müssen **nicht komplett speicherresident** sein, um ablaufen zu können
- **Lineare Speicheradressierung**, keine Fragmentierung aus Programmierersicht
- Beim Prozesswechsel **behält** ein Prozess seine hauptspeicherresidenten Seiten. Er verliert sie erst, wenn sie von der Verwaltung des realen Speichers verdrängt werden
- Anwenderprogramme können den **vollen virtuellen Adressraum** nutzen, wenn genügend Festplattenspeicher vorhanden ist
- Der tatsächlich zugewiesene reale Speicherplatz ändert sich dynamisch entspr. Angebot u. Nachfrage
- virtueller Speicher verursacht die gegenseitige Isolation von Speichern von Prozessen → dadurch sind die Speicher von Prozesse automatisch geschützt vor Zugriffen von anderen Prozessen!

Überblick

- ✓ Einführung in die Speicherverwaltung
- ✓ Grundprinzipien des virtuellen Speichers
- ✓ Optimierung der virtuellen Speichertechnik