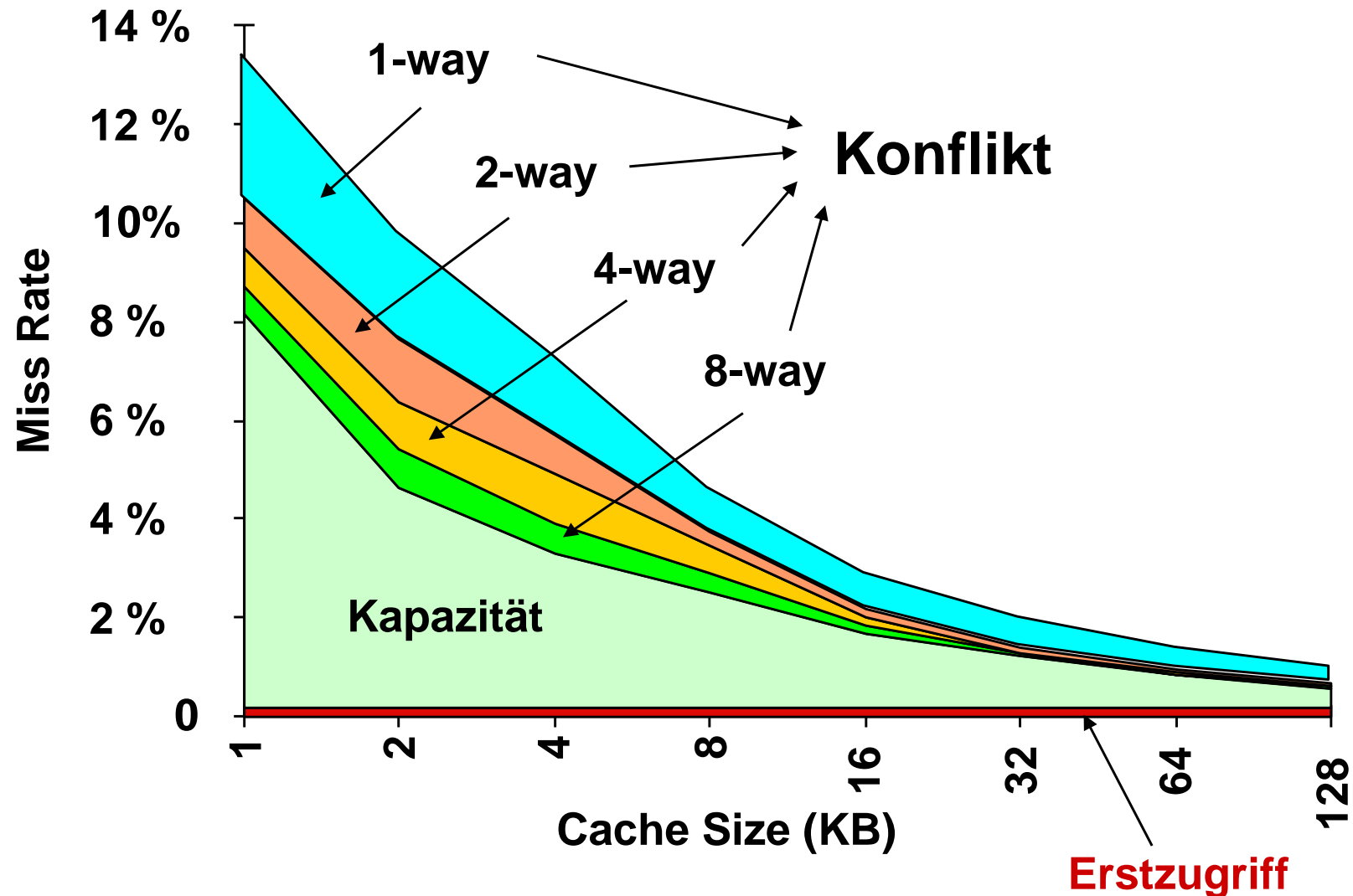
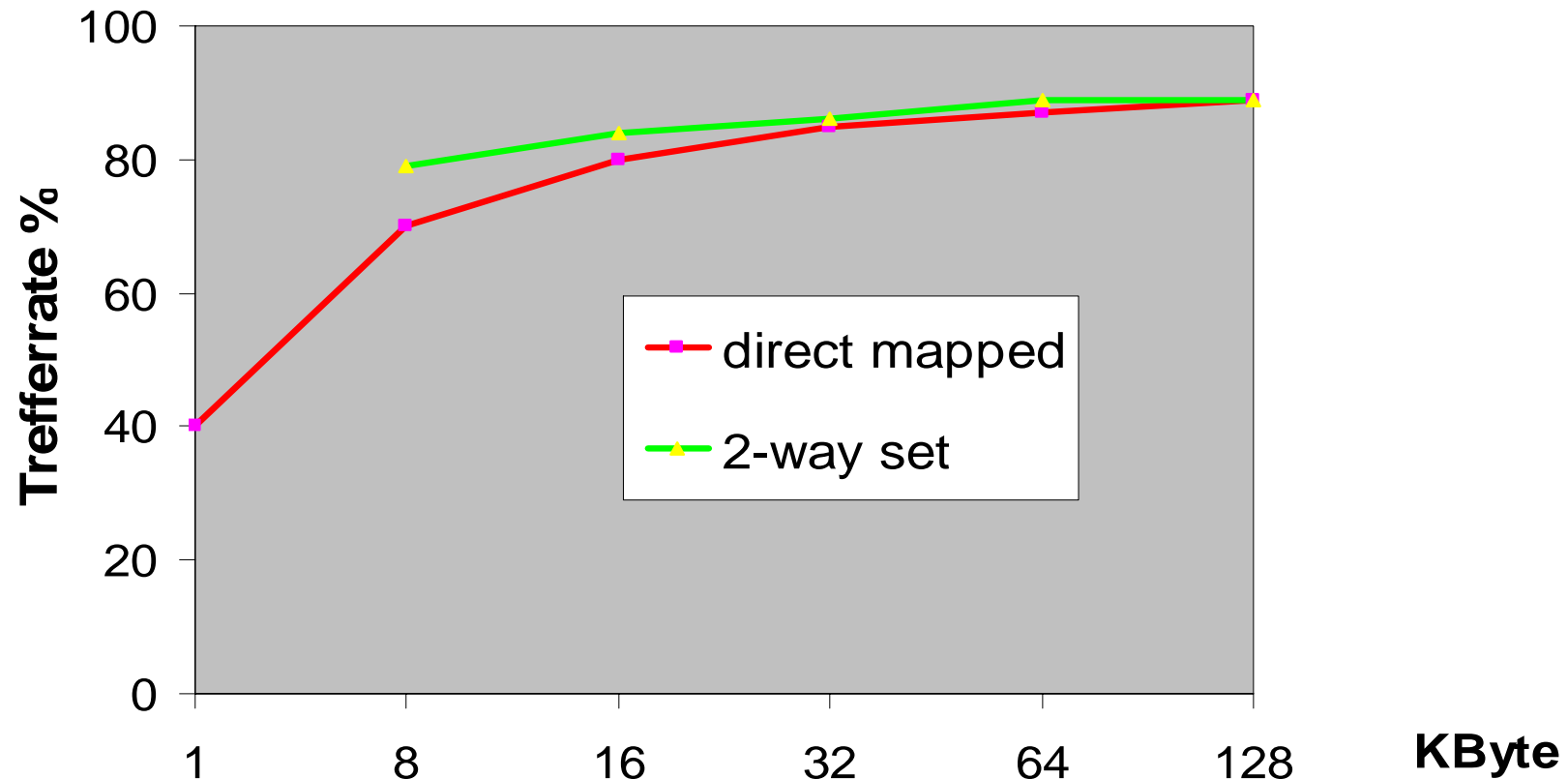


Wdh. Ursachen für die Fehlzugriffe



Wdh. Erzielbare Cache-Trefferquoten



Cache-Größen < 64 kByte: 2-Way Set Associative Cache besser als Direct Mapped Cache

Cache-Größen \geq 64 kByte: kaum noch Unterschiede

Wdh. Erzielbare Cache-Trefferquoten

Nach Untersuchungen von Agarwal, Hennessy und Horowitz:

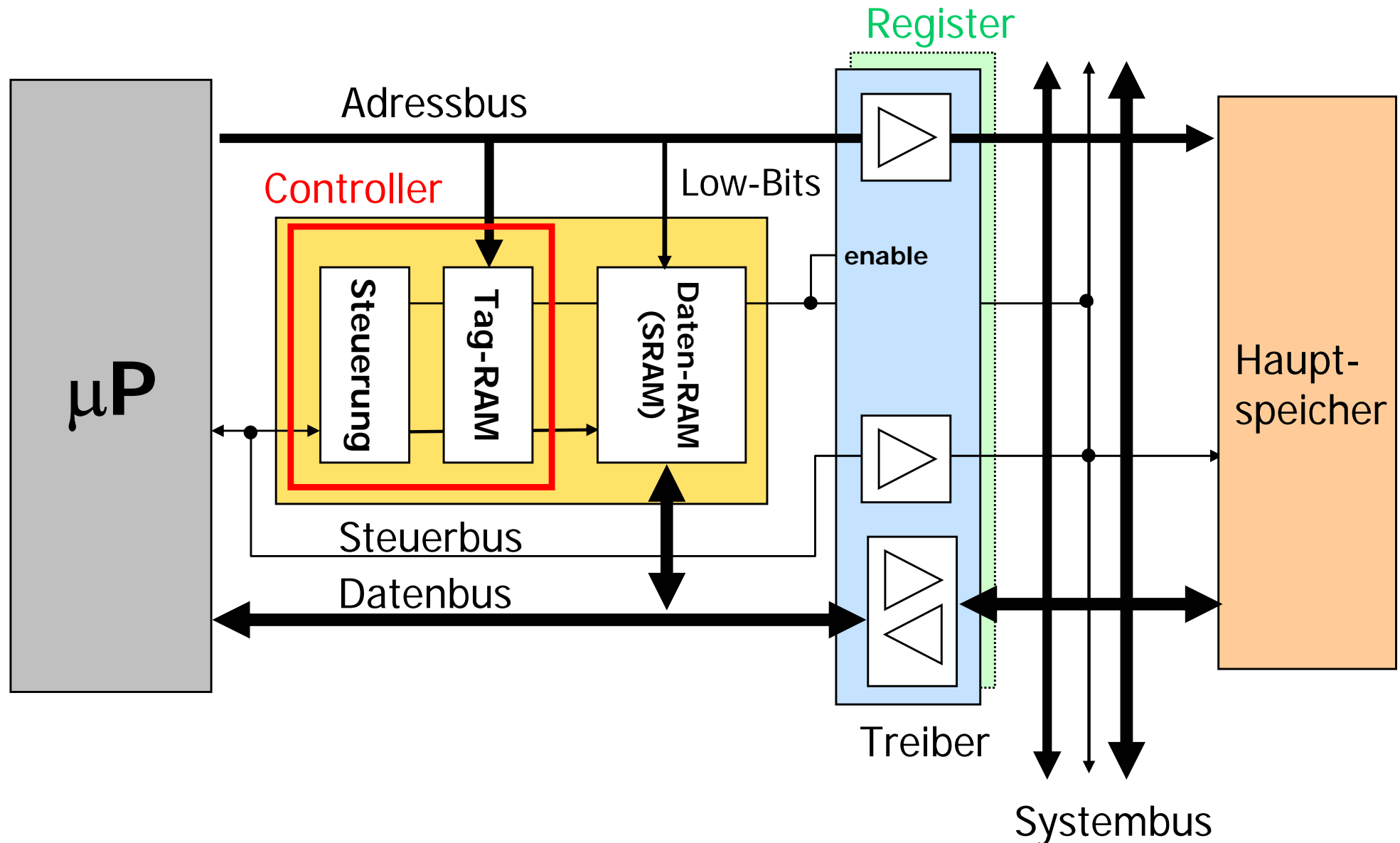
- ❑ Eine Cache-Trefferquote von circa 94% kann bei einem 64 kByte großen Cachespeicher erreicht werden (selbstverständlich gilt: je größer der Cachespeicher, desto größer die Trefferquote)
- ❑ Getrennte Daten- und Befehls-Cachespeicher sind bei sehr kleinen Cachespeichergrößen vorteilhaft, fallen jedoch bei Cachespeichergrößen ab ca. 8 KByte nicht mehr ins Gewicht
- ❑ Bei Cachespeichergrößen ab 64 KByte sind Direct Mapped Cachespeicher mit ihrer Trefferquote nur wenig schlechter als Cachespeicher mit Assoziativität 2 oder 4

Wdh. Erzielbare Cache-Trefferquoten

- ➔ Voll-assoziative Cachespeicher werden heute nur für sehr kleine auf dem Chip integrierte Caches mit 32 bis 128 Einträgen verwendet.

Bei größeren Cachespeichern findet sich zur Zeit ein Trend zur Direct Mapped Organisation oder 2 - 8 fach assoziativer Organisation.

Anbindung des Caches an den Systembus



Anbindung des Caches an den Systembus

- ❑ **Cache-Controller:**

Tag-RAM + Steuerung + Tag-Komparator

Da dieser sehr schnell sein muß ➡ auf einem Chip integriert

- ❑ Cachespeicher selbst ist separat mit SRAM-Bausteinen aufgebaut

- ❑ Cache-Controller übernimmt die Steuerung der Treiber zum Systembus (Systembuszugriff nur bei Cache-Miss, sonst ist der Systembus für andere Komponenten frei), sowie der Systembussignale zur Einfügung von Wartezyklen bei Cache-Miss (READY, HOLD, HOLDA, ...)

Verwendung mehrerer Caches

Oft findet sich eine mehrstufige Cache-Organisation:

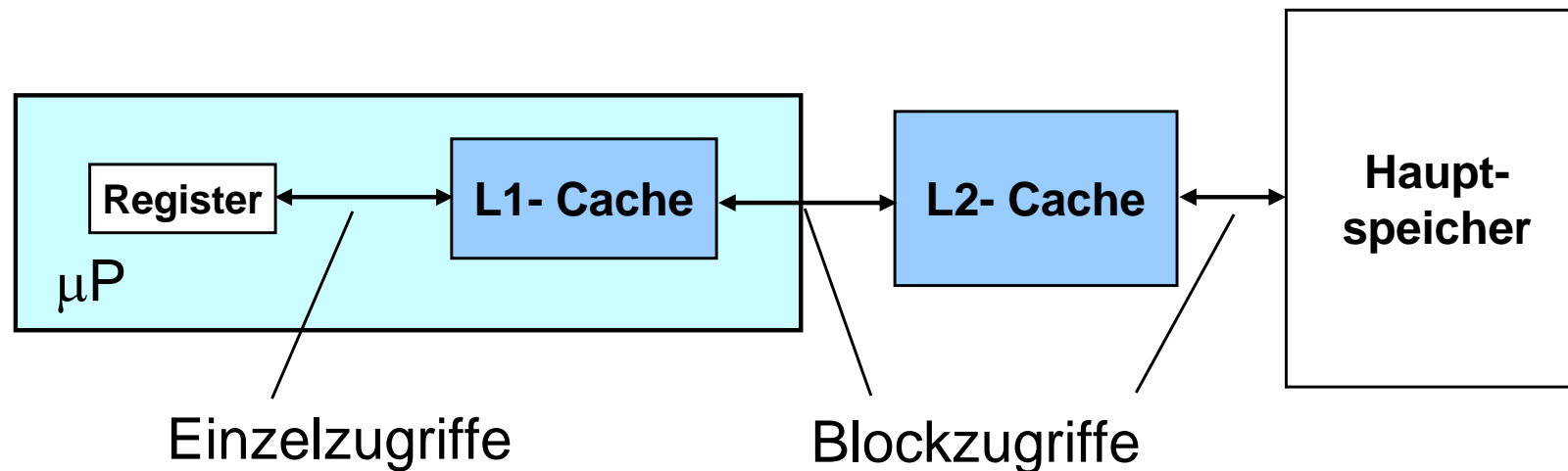
- ❑ Auf dem Prozessor-Chip befindet sich der sogenannte ***First-Level-Cache (On-Chip-Cache)***
- ❑ Häufig getrennte On-Chip-Caches: **Befehlscache** für die Befehle und **Datencache** für die Daten.
 - ➔ paralleler Zugriff auf Programm und Daten, wodurch die hohen Anforderungen bei heutigen Superskalar-Prozessoren an die Speicherbandbreiten erfüllt werden können

Verwendung mehrerer Caches

- ❑ Auf dem Chip wird eine **Harvard-Architektur** realisiert, bei der Programm und Daten in getrennten Speichern liegen
- ❑ Außerhalb des Prozessor-Chips befindet sich häufig ein weiterer, größerer Cache, der sogenannte ***Secondary-Level-Cache*** (***On-Board-Cache***, 64 - 1024 KByte groß)
- ❑ Der Secondary-Level-Cache kann parallel zum Hauptspeicher an den Bus angeschlossen werden (***Look-Aside-Cache***). Er sorgt dafür, dass bei einem First-Level-Cache-Miss die Daten schnell nachgeladen werden können

On-Chip und Off-Chip-Cache

- **On-Chip-Cache:** integriert auf dem Prozessorchip
 - Sehr kurze Zugriffszeiten (wie die der prozessorinternen Register)
 - Aus technologischen Gründen begrenzte Kapazität
- **Off-Chip-Cache:** prozessorextern (SRAM-Speicher)



Cache-Kohärenzproblem (*Cache Coherency Problem*)

- **Cache-Kohärenzproblem** (*Cache Coherency Problem*): Gültigkeitsproblem, das beim Zugriff mehrerer Verarbeitungselemente auf Speicherworte des Hauptspeichers entsteht.
- **Kohärenz** bedeutet das korrekte Voranschreiten des Systemzustands durch ein abgestimmtes Zusammenwirken der Einzelzustände.
- Im Zusammenhang mit dem Cache muss das System dafür sorgen, dass immer die aktuellen Daten und nicht die veralteten Daten gelesen werden.
- Ein System ist **konsistent**, wenn alle Kopien eines Datums im Hauptspeicher und den verschiedenen Cachespeichern identisch sind. Dadurch ist auch die *Kohärenz sichergestellt*, jedoch entsteht ein hoher Aufwand.

Bus-Schnüffeln (*Bus-Snooping*)

- In Mehrprozessorsystemen, bei denen mehrere Prozessoren mit lokalen Cachespeichern an einen gemeinsamen Bus/Hauptspeicher angeschlossen sind, verwendet man das sogenannte **Bus-Schnüffeln**.
- Die **Schnüffel-Logik** jedes Prozessors hört am Bus die Adressen mit, die die anderen Prozessoren auf den Bus legen. Die Adressen auf dem Bus werden mit den Adressen, der im Cache gespeicherten Daten, verglichen.
- Bei Adressübereinstimmung am Bus geschieht folgendes:

Bus-Schnüffeln (*Bus-Snooping*)

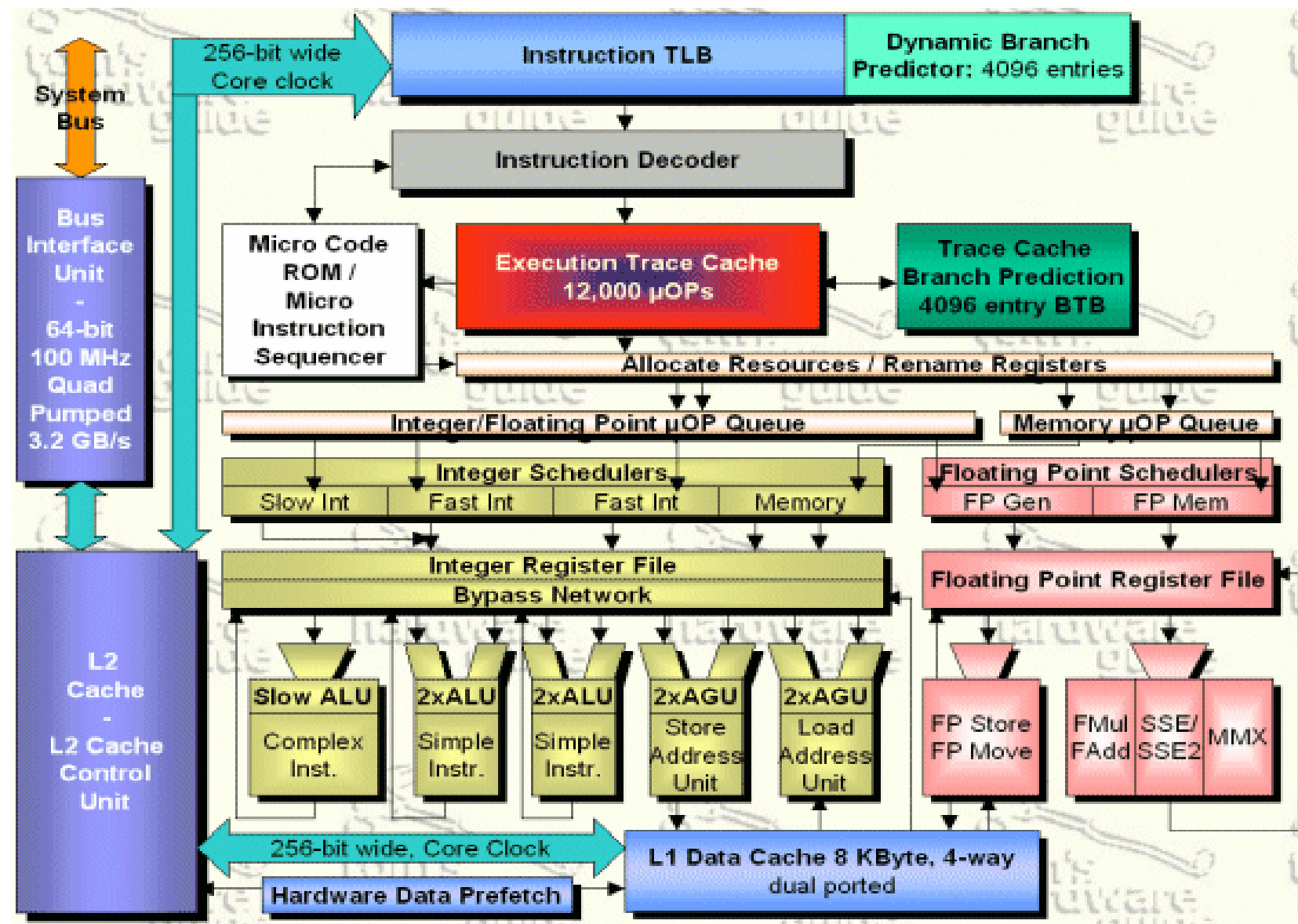
- ❑ Wenn ein **Schreibzugriff** auf dieselbe Adresse vorliegt, dann wird der im Cache gespeicherte Cacheblock für „ungültig“ erklärt (**Write-Invalidate-Verfahren**), oder mit aktualisiert (**Write-Update-Verfahren**).
- ❑ Wenn ein **Lesezugriff** auf dieselbe Adresse mit einer modifizierten Datenkopie im Cachespeicher festgestellt wird, dann legt der Cache-Controller ein Snoop Status Signal (SSTAT) auf den Bus.
 - Der Prozessor, der die Adresse auf den Bus gelegt hat, unterbricht seine Bustransaktion.
 - Der „schnüffelnde“ Cache-Controller übernimmt den Bus und schreibt den betreffenden Cacheblock in den Hauptspeicher.
 - Dann wird die ursprüngliche Bustransaktion erneut durchgeführt.
- ❑ Siehe MESI-Protokoll (bei Multiprozessoren)

Fehlzugriffe (Misses)

Klassifikation von Misses:

- ❑ **Compulsory-Misses:** Beim ersten Zugriff auf einen Block muss der Block in den Cache geladen werden.
(*cold start misses* oder *first reference misses*)
(Tritt auch bei einem unendlich großem Cache)
- ❑ **Capacity-Misses:** Wenn der Cache bei der Ausführung eines Programms nicht alle benötigten Blöcke aufnehmen kann.
- ❑ **Conflict:** Durch die Block-Zuweisungsstrategie (set associative oder direct mapped), auch *collision misses* oder *interference misses* genannt
- ❑ **Coherence–Misses:** durch Cache-Kohärenz.

Cache-Speicher in Pentium 4



Cache-Speicher in Pentium 4

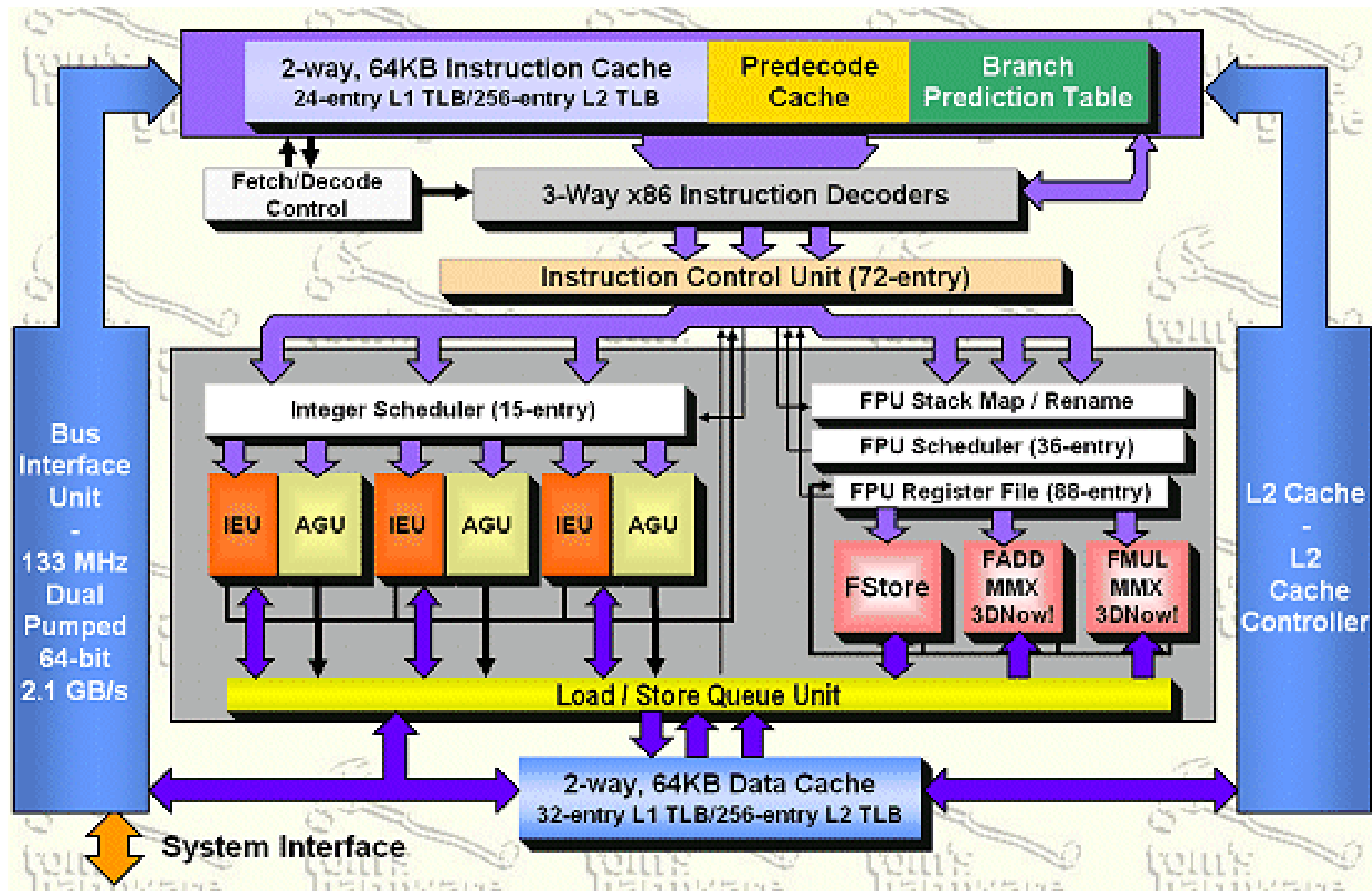
□ L1-Cache:

- 8 Kbyte Daten-Cache (Bei Pentium III: 16 Kbyte; bei Athlon: 64 Kbyte)
- 4-way set associative Daten-Cache
- Cache-lines mit 64 Byte (dual-pot-Architektur)
- Write-Through
- 12K μ OPs „Trace“-Cache

□ L2-Cache:

- 256 Kbyte
- 8-way set associative
- Cache-lines mit 128 Byte
- Write back
- Bandbreite 44,8 Gbyte/s (16 Gbyte/s bei Pentium III)

Cache-Speicher bei Athlon



Fragen, die sich ein Speicherhierarchie-Designer stellen muss:

- ❑ Wohin kann ein Block abgebildet werden?
(Block-Abbildungsstrategie)
 - Vollassoziativ, Satz-Assoziativ, Direct-Mapped
- ❑ Wie kann ein Block gefunden werden?
(Block-Identifikation)
 - Tag/Block
- ❑ Welcher Block soll bei einem Miss ersetzt werden?
(Block-Ersetzungsstrategie)
 - Random, FIFO, LRU
- ❑ Was passiert bei einem Schreibzugriff?
(Schreibe-Strategie)
 - Write back oder Write Through (mit Write Buffer)

Kapitel 8

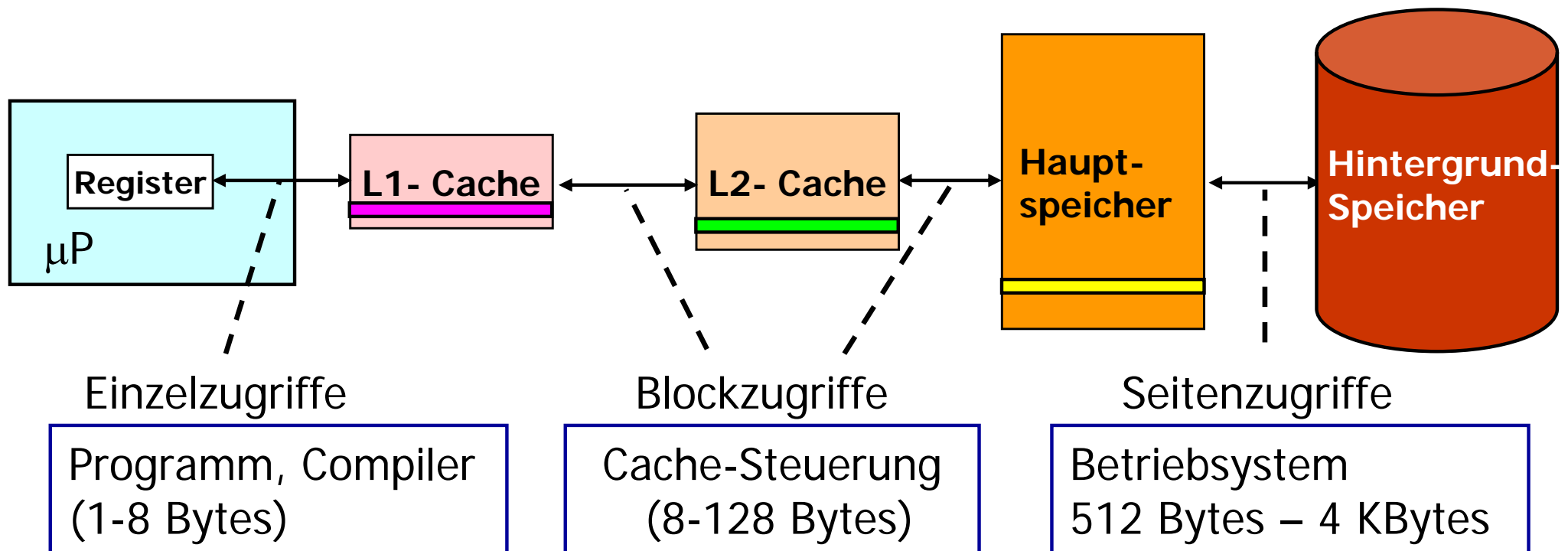
Virtuelle Speicherverwaltung

Software-Prinzipien und Hardware-Unterstützung
der virtuellen Speicherverwaltung



Speicherhierarchie

Daten werden nur zwischen aufeinanderfolgenden Ebenen der Speicherhierarchie kopiert.



Grundprinzip und Zusammenhang mit dem Betriebssystem

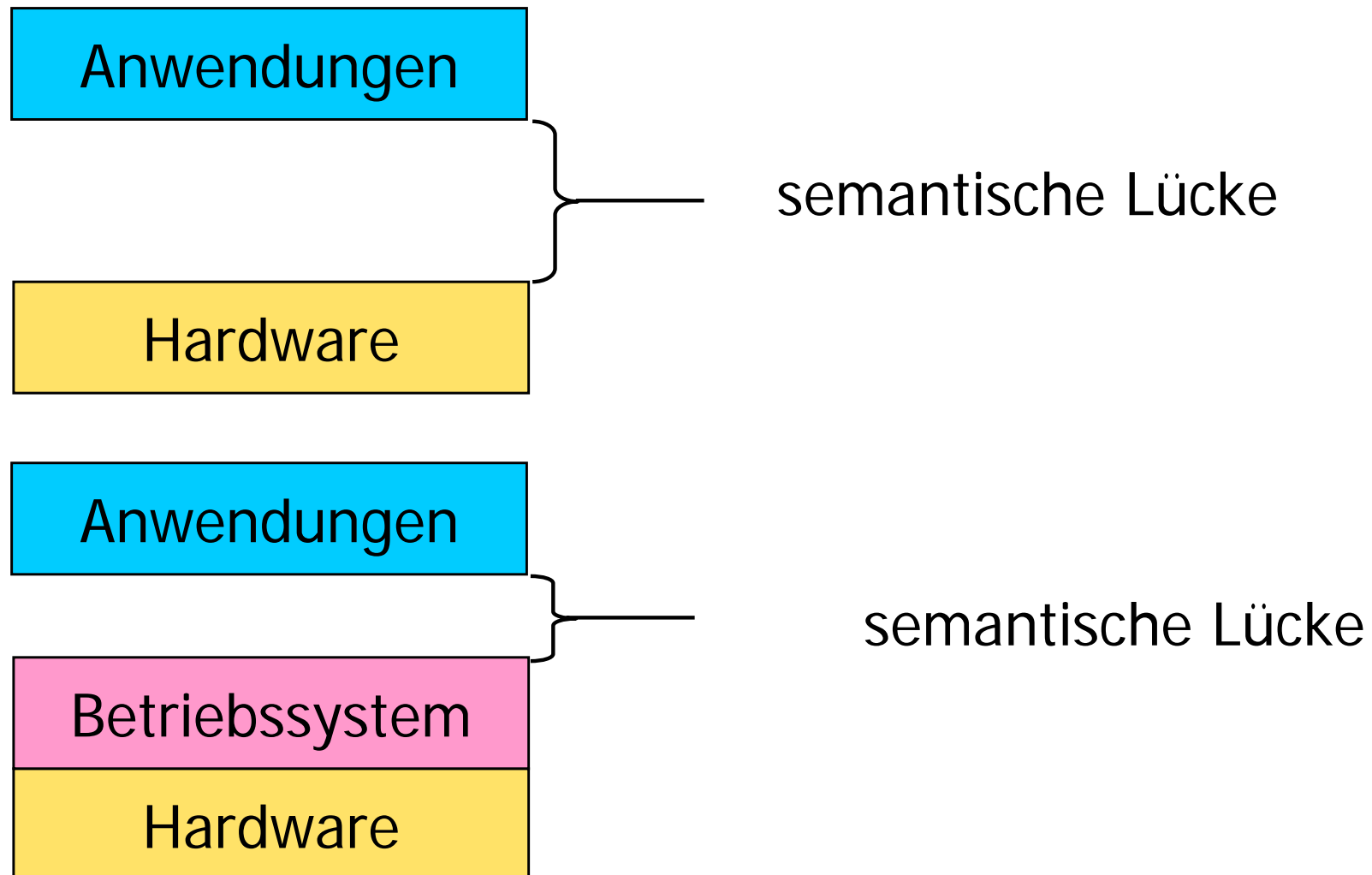
Definition des Begriffs "Betriebssystem":

z. B. nach dem Deutschen Institut für Normung (DIN 44300):

Ein Betriebssystem umfasst die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften der Rechenanlage die Grundlage der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.

- ➔ Ziel eines Betriebssystems ist es, die semantische Lücke zwischen Anwendung (wünscht möglichst hohe Operationen) und Hardware (bietet elementare Operationen) zu verkleinern.

Betriebssystem verkleinert die semantische Lücke



Betriebssystem erweitert die Fähigkeiten der Hardware.

Spezielle Aufgaben von Betriebssystemen

- ❑ Betriebsmittelverwaltung
- ❑ Auftragsverwaltung (Prozeßverwaltung)
- ❑ Speicherverwaltung

- ❑ **Betriebsmittelverwaltung (*resource management*)**

Verwaltet alle peripheren Betriebsmittel des Rechnersystems, wie z. B. Festplatte, Floppy Disk, CDROM, Drucker, ...

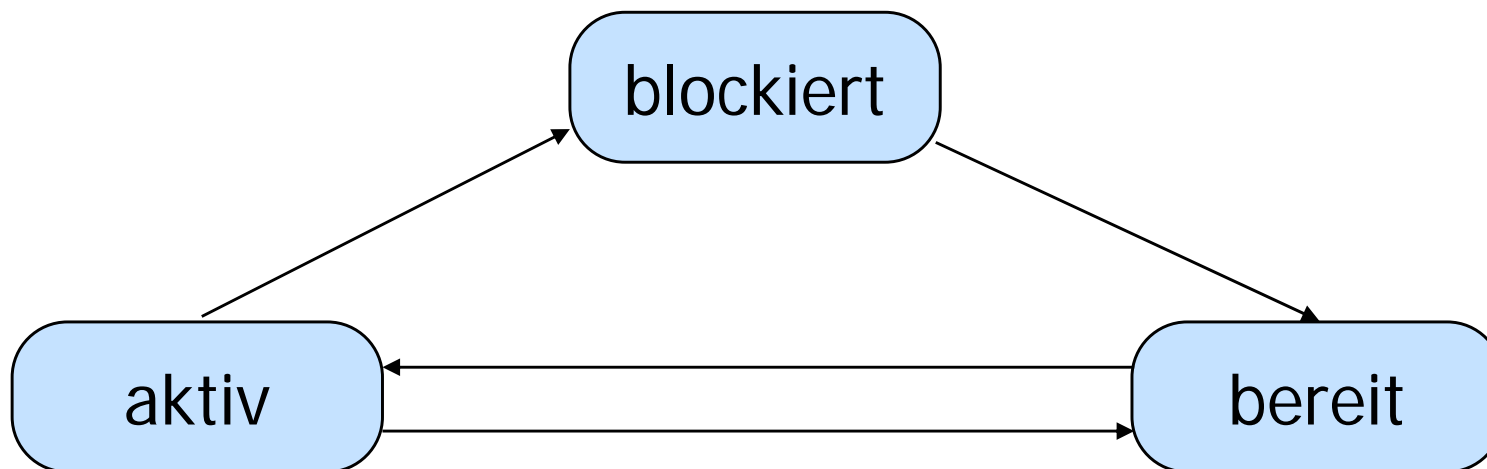
Ziel: einfache, effektive und konfliktfreie Nutzung der Betriebsmittel durch die verschiedenen Prozesse.

Spezielle Aufgaben von Betriebssystemen

□ **Auftragsverwaltung (*task management*)**

Verwaltung der Prozesse (*tasks*)

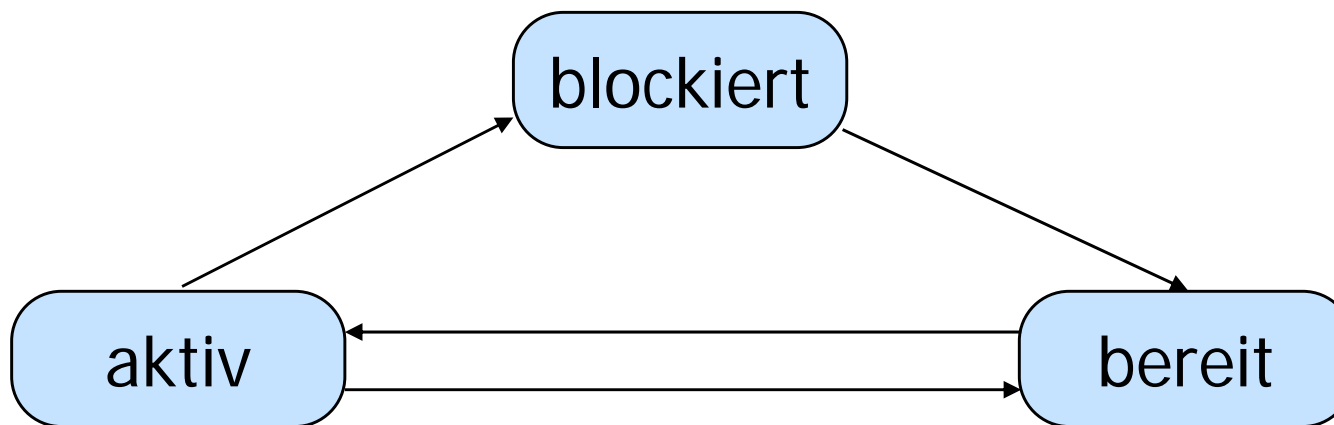
Ein Prozeß kann im Laufe seiner Bearbeitung im wesentlichen folgende unterscheidbaren Zustände annehmen:



Spezielle Aufgaben von Betriebssystemen

Verwaltung dieser Zustände und Übergänge (aufgrund äußerer oder innerer Ereignisse) sind wesentliche Aufgaben der Auftragsverwaltung.

Ziel: möglichst effiziente und scheinbar parallele Bearbeitung mehrerer Aufgaben



Spezielle Aufgaben von Betriebssystemen

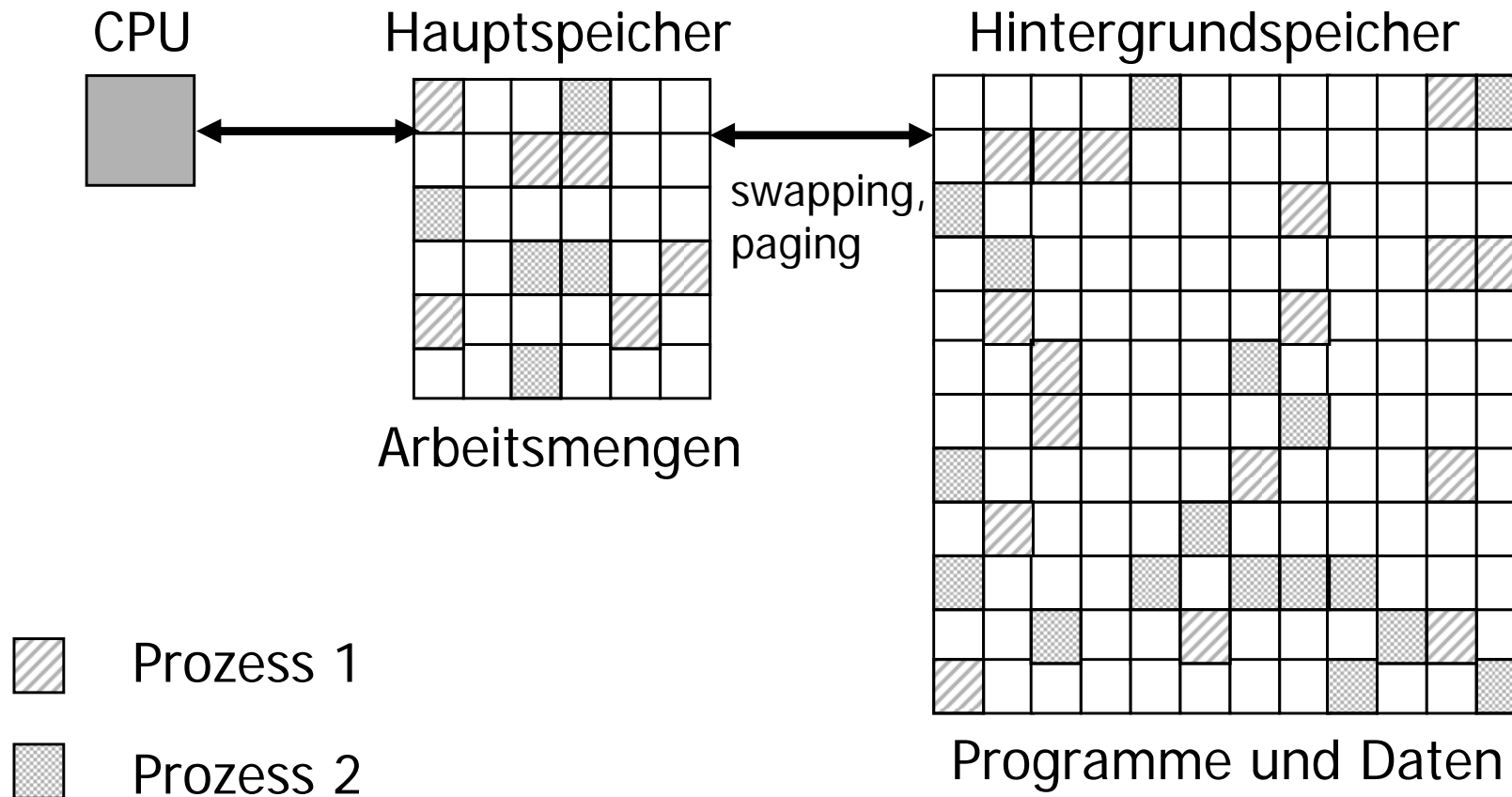
□ **Speicherverwaltung** (*memory management*)

Durch immer größer werdende Programme sowie mehrerer quasi gleichzeitig laufender Programme

➔ der zur Verfügung stehende Arbeitsspeicher wird schnell zu klein

Abhilfe: Nur die gerade benötigten Teile der aktiven Programme werden wirklich im Arbeitsspeicher gehalten, der Rest befindet sich im Hintergrundspeicher und wird bei Bedarf geladen (*swapping, paging*).

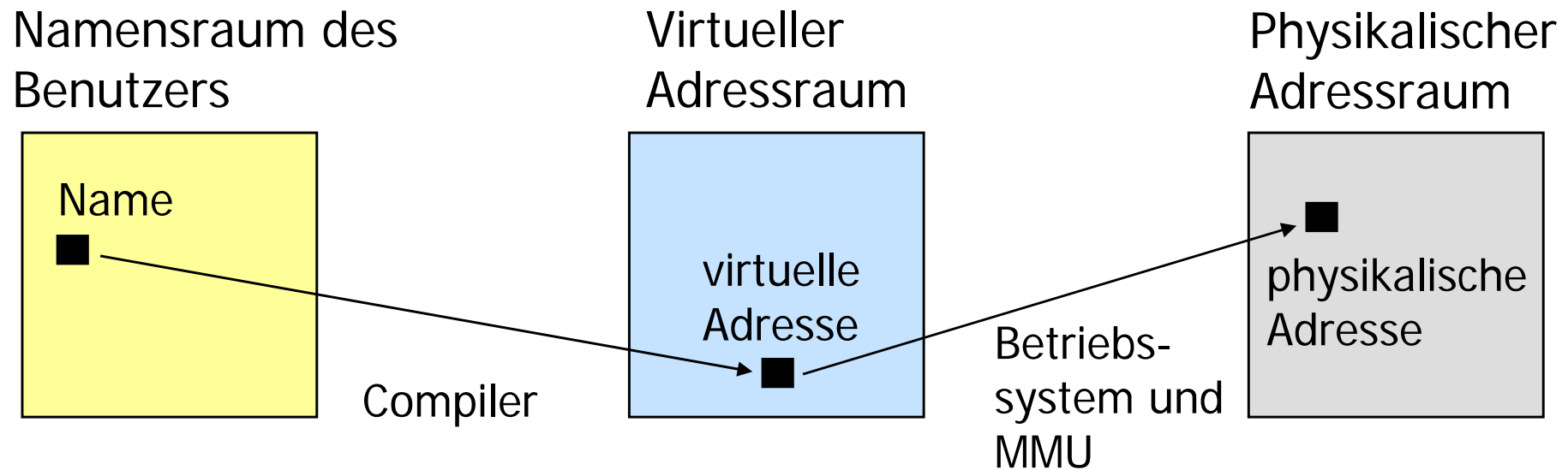
Grundstruktur virtueller Speicherverwaltung



Virtuelle Speicherverwaltung

- ❑ Dieser Vorgang bleibt dem Anwender völlig verborgen, d. h. der Arbeitsspeicher erscheint dem Anwender wesentlich größer, als er in Wahrheit ist.
- ❑ Ein nach diesem Konzept verwalteter Speicher heißt deshalb virtueller Speicher.
- ❑ Von modernen Prozessoren wird die Verwaltung dieses virtuellen Speichers hardwaremäßig durch eine MMU (memory management unit) unterstützt.
- ❑ Hauptaufgabe dieser virtuellen Speicherverwaltung:
Umsetzung virtueller (logischer) Adressen in physikalische Adressen

Virtuelle Speicherverwaltung



Benutzer: kennzeichnet seine Objekte (Programme, Unterprogramme, Variablen, ...) durch Namen

Compiler: übersetzt diese Namen in virtuelle (logische) Adressen.

Virtuelle Speicherverwaltung: wandelt diese Adressen zur Laufzeit je nach gerade gegebener Speicherbelegung in die physikalische Adresse (wirklicher Ort des Objekts im Hauptspeicher) um

Beispiel

Name

JMP WEITER

Compiler

JMP -128

dyn. Adressrechnung
 $((PC) - 128)$

Logische Adresse:

4583

Virtuelle Speicher-
verwaltung (MMU)

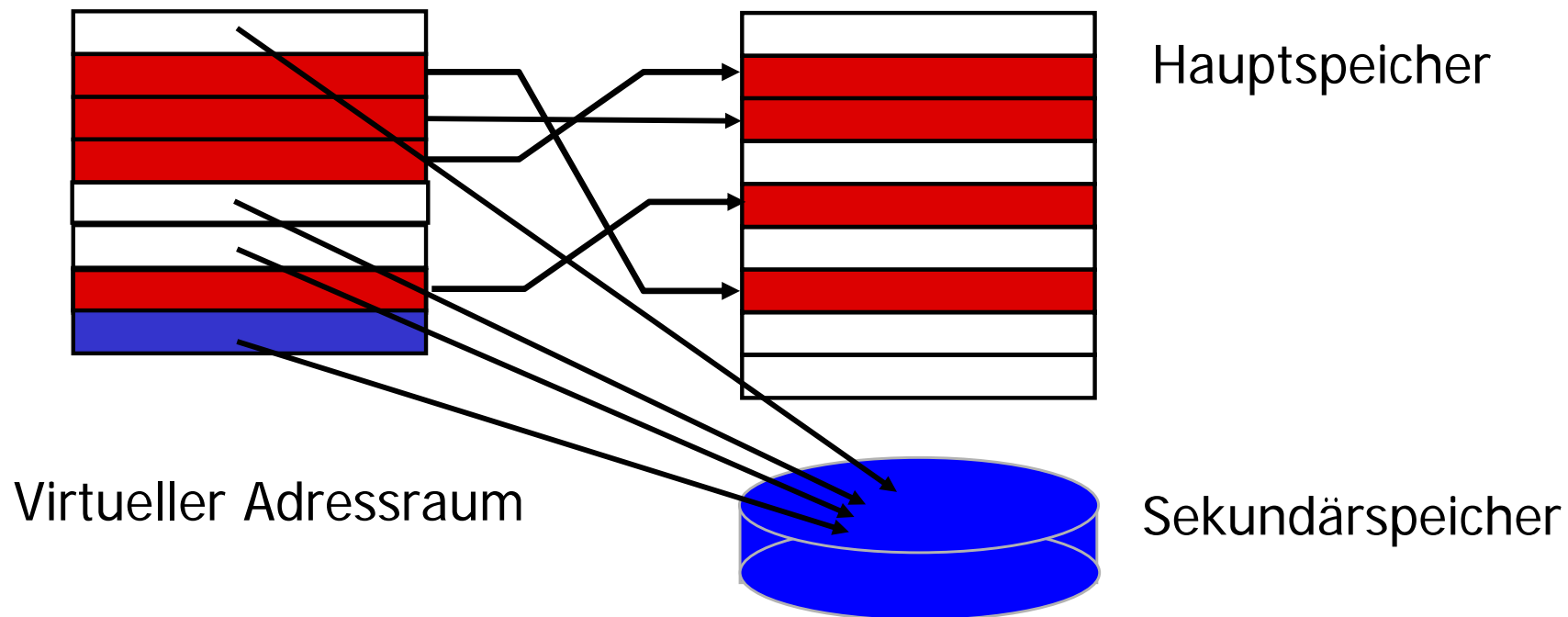
Physikalische Adresse

23112



Speicherverwaltung

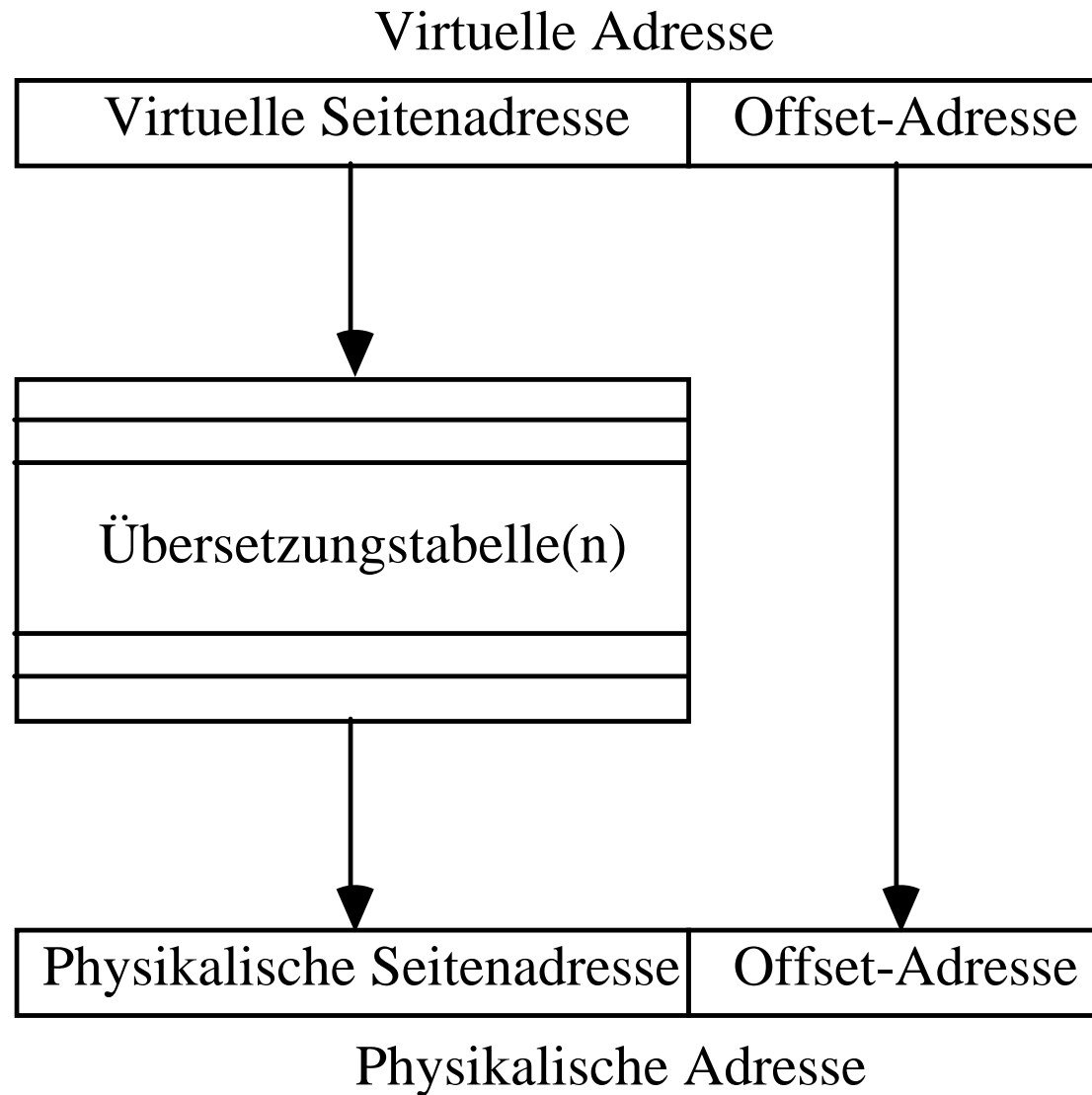
- ❑ Virtuelle Speicherkapazität > effektive Hauptspeicherkapazität
- ❑ Betriebssystem lagert bei Bedarf Speicherbereiche ein und aus
- ❑ Speicherverwaltungseinheiten (*memory management units, MMU*) unterstützt hardwaremäßig eindeutige Adressberechnung
- ❑ Abbildungsinformation in Übersetzungstabellen



Speicherverwaltung

- ❑ Das **Betriebssystem** führt Buch über die freien Speicherbereiche und lagert bei nicht ausreichendem Freiplatz im Hauptspeicher diejenigen Speicherinhalte auf den Hintergrundspeicher aus, die gegenüber ihrem Originalen auf dem Hintergrundspeicher verändert worden sind.
- ❑ Die eindeutige Abbildung des großen virtuellen Speichers auf die effektive Hauptspeicherkapazität wird von der Hardware durch **Speicherverwaltungseinheiten** unterstützt (*memory management units, MMU*)
- ❑ Die erforderliche Abbildungsinformation stellt das Betriebssystem in Form einer oder mehrerer **Übersetzungstabellen** zur Verfügung.

Abbildung virtueller auf physikalische Adressen



Speicherverwaltung

- ❑ Um den Umfang der Übersetzungstabellen gering zu halten, bezieht man die Abbildungsinformation nicht auf einzelne Adressen, sondern auf zusammenhängende Adressbereiche
- ❑ Es gibt zwei Möglichkeiten:
 - **Segmentierung**
 - **Seitenwechsel-Verfahren**

Segmentierungs- und Seitenwechselfverfahren

Es existieren zwei grundlegende Verfahren zur virtuellen Speicherverwaltung (Segmentierung und Seitenwechsel)

Segmentierung

- Hierbei wird der virtuelle Adressraum in Segmente verschiedener Länge zerlegt.
- Jedem Prozess sind ein oder mehrere Segmente z. B. für den Programmcode und die Daten, zugeordnet.
- Die einzelnen Segmente enthalten logisch zusammenhängende Informationen (Programm- und Datenmodule) und können relativ groß sein.

Segmentierungs- und Seitenwechselfverfahren

Aufteilung in Seiten

- Hierbei wird der logische und der physikalische Adressraum in "Segmente fester Länge", die sogenannten Seiten (pages) unterteilt.
- Die Seiten sind relativ klein (256 Byte - 4 kByte)
- Ein Prozess wird auf viele dieser Seiten verteilt (keine logischen Zusammenhänge wie bei der Segmentierung)

Segmentierung

Vorteile:

- Segmentierung spiegelt logische Programmstruktur wieder.
- durch große Segmente relativ seltener Datentransfer.

Nachteile:

- wenn Datentransfer, dann jedoch umfangreich.
- besteht ein Programm nur aus einem Code- und Daten-Segment (wird vom Compiler oder Benutzer festgelegt), so muss es vollständig eingelagert werden.

Seitenaufteilung

□ Vorteile:

- durch kleine Seiten wird nur der wirklich benötigte Teil eines Programms eingelagert.
- geringerer Verwaltungsaufwand als Segmentierung

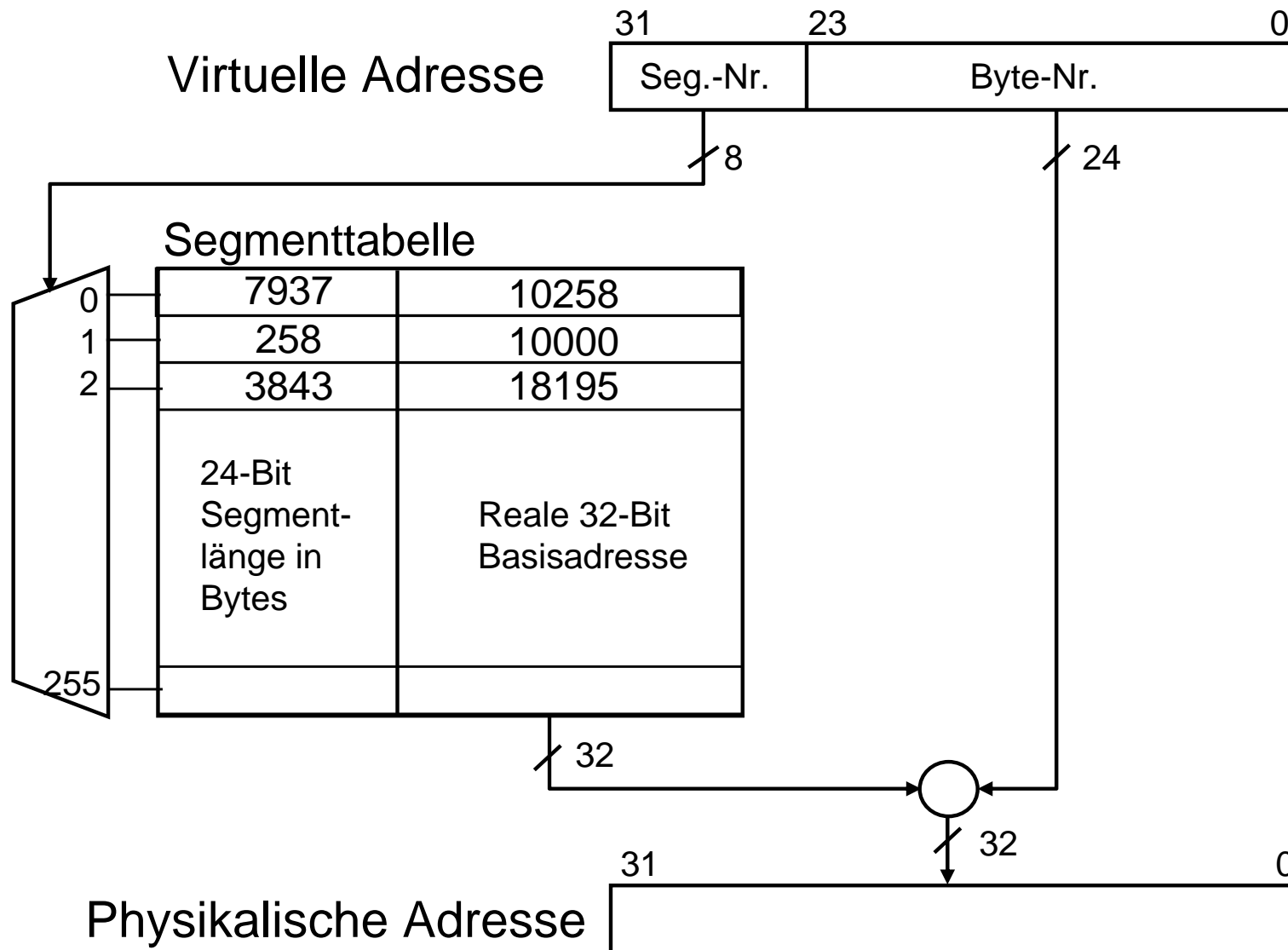
□ Nachteil:

- häufiger Datentransfer

Ältere Prozessoren unterstützten jeweils nur ein Verfahren, z. B. Segmentierung bei 80286, 68010 und Seitenwechsel bei Z8003/4, National 32000

Heutige Mikroprozessoren, z.B. 80386, 80486, Pentium, ... unterstützen beide Verfahren

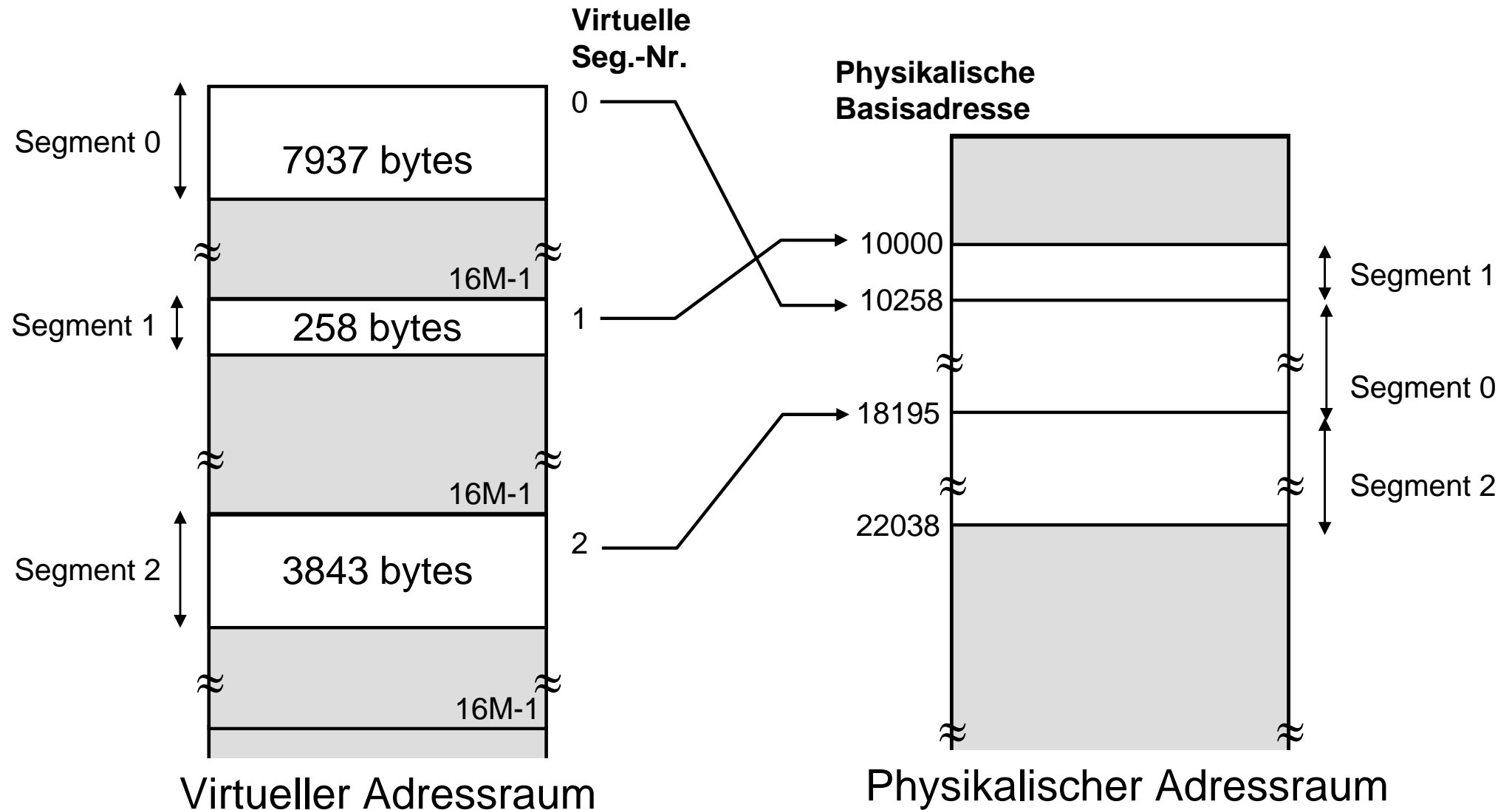
Segmentbasierte Speicherverwaltung



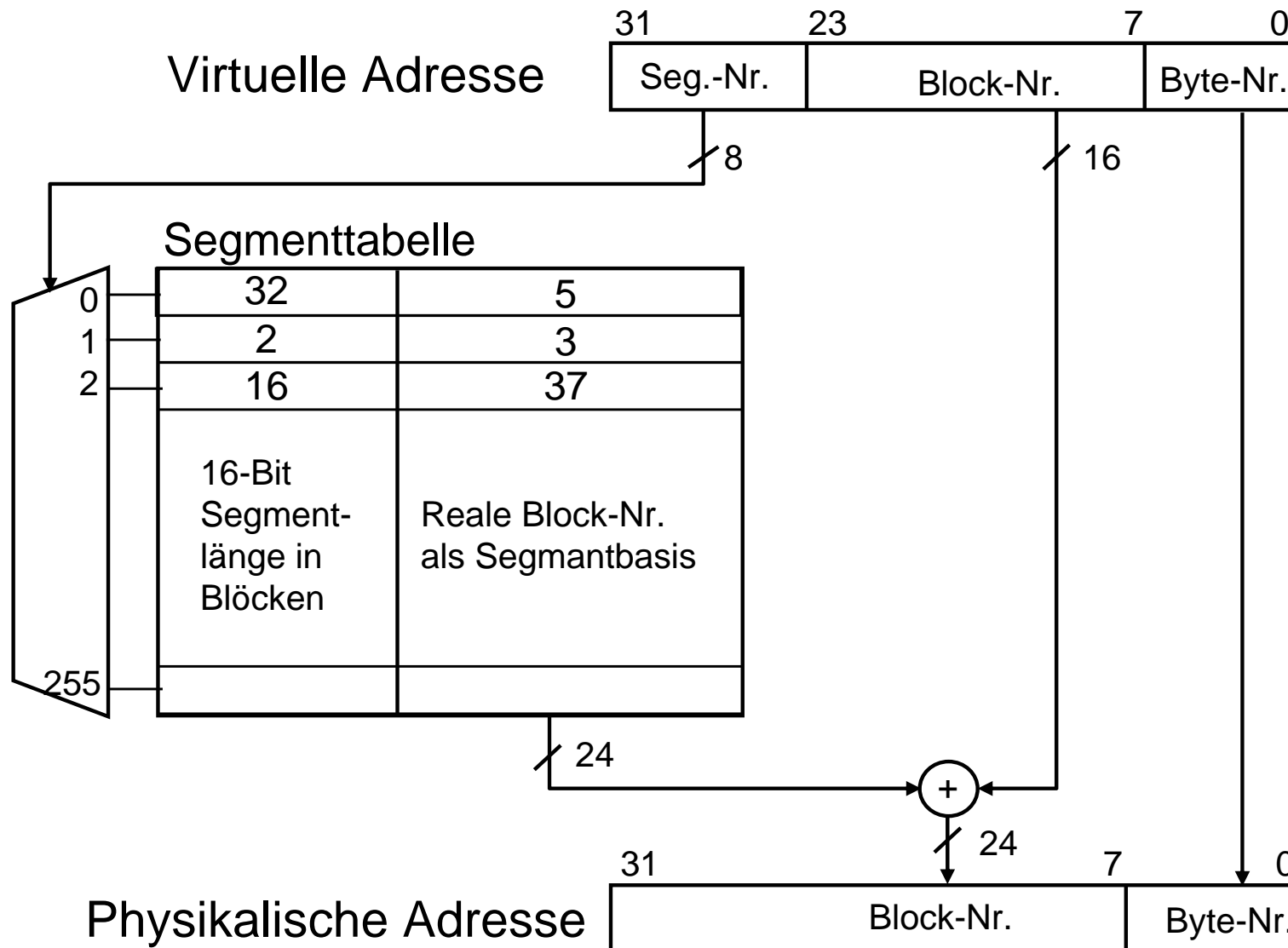
Segmentbasierte Speicherverwaltung

- ❑ Virtuelle Adresse wird in eine Segmentnummer (n höherwertige Bits der virtuellen Adresse) als Kennung eines Segments und in eine Bytenummer (verbleibenden m Bits der Adresse) als Abstand zum Segmentanfang unterteilt.
- ❑ Maximale virtuelle Segmentanzahl = 2^n , Maximale Segmentgröße = 2^m
- ❑ Die Adressabbildung erfolgt über eine Segmenttabelle (im Registerspeicher der MMU).
- ❑ Reale Adresse ergibt sich aus der Segmentbasisadresse, zu der die virtuelle Byte-Nummmmer addiert wird.
- ❑ Segmentlängenangaben in der Segmenttabelle, um segment-überschreitende Zugriffe feststellen und ggf. verhindern zu können.
- ❑ Bei Segmenten mit einer geringeren Größe als 2^m , gilt der ungenutzte Raum als Verschnitt.
- ❑ Gute Ausnutzung des Hauptspeichers, wenn man die Segmentgrenzen an jeder Byteadresse zulässt.

Virtueller und physikalischer Adressraum



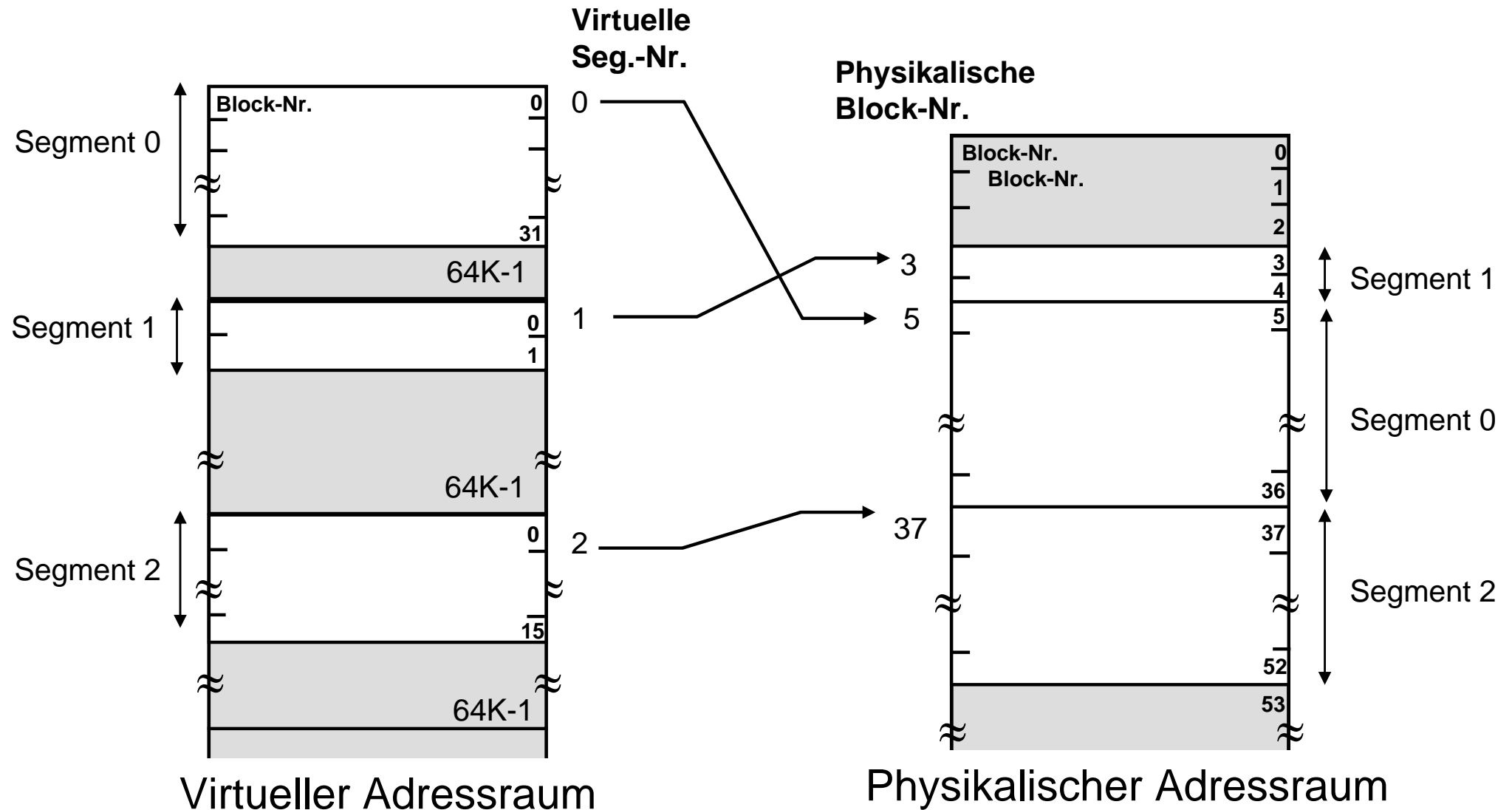
Segmentbasierte Speicherverwaltung



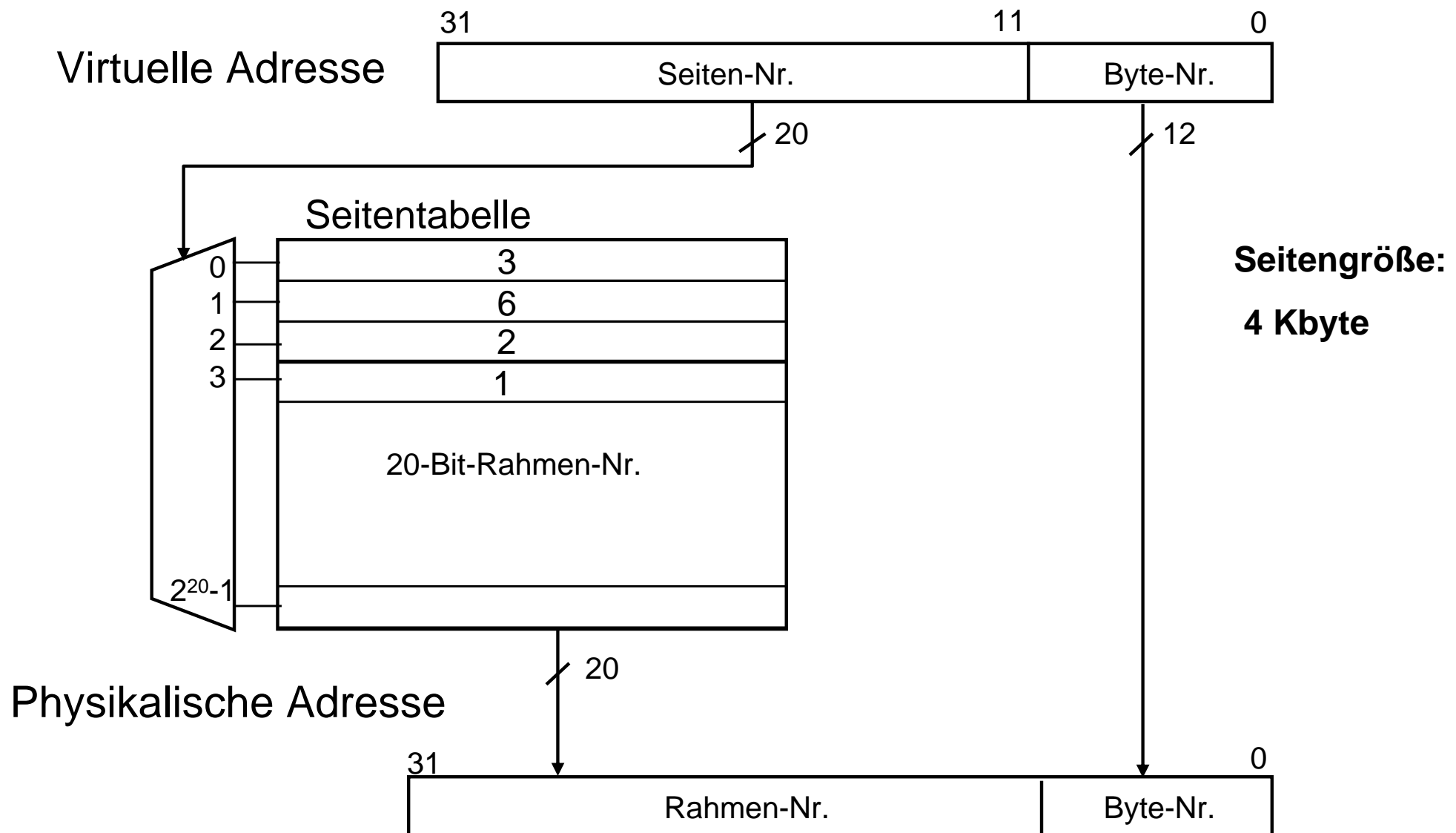
Segmentbasierte Speicherverwaltung

- ❑ Segmentgrenzen nicht an jeder Byteadresse, sondern an Vielfachen von Blöcken (hier von 256 Bytes).
- ❑ Segmente werden im virtuellen physikalischen Adressraum in Blöcke von 256 Bytes unterteilt.
- ❑ D.h. die Bytenummer aus m Bits wird aufgeteilt in eine kürzere Bytenummer für die Byteadressierung im Block und eine Blocknummer.
- ❑ Bei Adressumsetzung wird die virtuelle Segmentnummer auf eine reale 24-Bit-Blocknummer als Segmentbasis abgebildet.
- ❑ Virtuelle Bytenummer für die Adressierung innerhalb des Blocks wird unverändert übernommen.

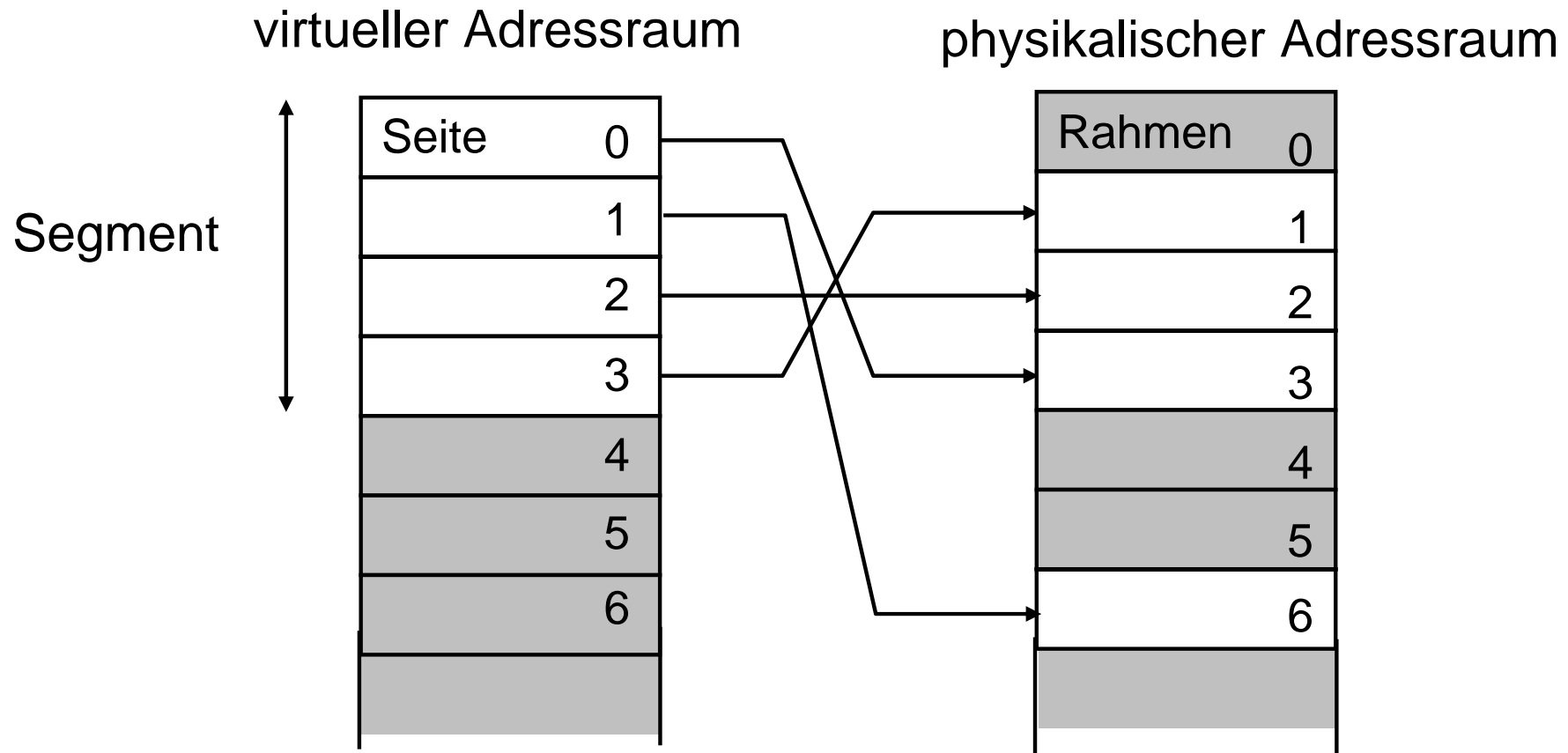
Virtueller und physikalischer Adressraum



Seitenwechsel (Paging)



Virtueller und physikalischer Adressraum



Probleme der virtuellen Speicherverwaltung

Beim Austausch von Daten zwischen Haupt- und Hintergrundspeicher ergeben sich drei Problemkreise:

- ❑ **Einlagerungszeitpunkt:** Wann werden Segmente oder Seiten in den Hauptspeicher eingelagert?
- ❑ **Zuweisungsproblem:** An welche Stelle des Hauptspeichers werden die Seiten oder Segmente eingelagert?
- ❑ **Ersetzungsproblem:** Welche Segmente oder Seiten müssen ausgelagert werden, um Platz für neu benötigte Daten zu schaffen?

Probleme der virtuellen Speicherverwaltung

Beim Austausch von Daten zwischen Haupt- und Hintergrundspeicher ergeben sich drei Problemkreise:

1. Der Einlagerungszeitpunkt

Wann werden Segmente oder Seiten in den Arbeitsspeicher eingelagert ?

Gängiges Verfahren:

Einlagerung auf Anforderung (Demand Paging bei Seitenverfahren)

Probleme der virtuellen Speicherverwaltung

Einlagerung auf Anforderung (*Demand Paging*):

Hierbei werden Daten eingelagert, sobald auf sie zugegriffen wird, sie sich aber nicht im Hauptspeicher befinden.

Der Zugriff auf ein nicht im Hauptspeicher vorhandenes Segment oder Seite heißt **Segment-** oder **Seiten-Fehler** (*segment fault, page fault*).

Probleme der virtuellen Speicherverwaltung

2. Das Zuweisungsproblem

An welche Stelle des Hauptspeichers werden die Seiten oder Segmente eingelagert ?

Bei Segmentierungsverfahren:

Hier muss eine ausreichend große Lücke im Hauptspeicher gefunden werden.

Drei Strategien (Betriebssystem):

- *first-fit*: erste passende Lücke wird genommen
- *best-fit* : kleinste passende Lücke wird genommen
- *worst-fit*: größte passende Lücke wird genommen

Probleme der virtuellen Speicherverwaltung

Problem bei allen drei Verfahren:

Der Speicher zerfällt nach einiger Zeit in belegte und unbelegte Speicherbereiche
➔ externe Fragmentierung.

Die unbelegten Speicherbereiche sind oft zu klein, um weitere Segmente aufnehmen zu können

Hauptspeicher

frei
Segment D
frei
Segment C
frei
Segment B
frei
Segment A

Probleme der virtuellen Speicherverwaltung

Zuweisungsproblem bei Seitenwechself Verfahren

Hier taucht dieses Problem nicht auf, da alle Seiten gleich groß sind und somit immer “passende Lücken” entstehen
➡ keine externe Fragmentierung.

Jedoch: Problem der **internen Fragmentierung**
Diese entsteht bei der Aufteilung eines Programms auf die Seiten.

Einheitliche Seitengröße ➡ auf der letzten Seite jedes Programm-Moduls entsteht mit hoher Wahrscheinlichkeit ein ungenutzter Leerraum.

Probleme der virtuellen Speicherverwaltung

3. Das Ersetzungsproblem

Welche Segmente oder Seiten müssen ausgelagert werden, um Platz für neu benötigte Daten zu schaffen ?

Bei Segmentierungsverfahren:

Meist wird die Anzahl der gleichzeitig von einem Prozeß benutzbaren Segmente limitiert:

➔ bei Einlagerung eines neuen Segments wird ein zuvor für diesen Prozeß benutztes Segment ausgelagert

Es ist jedoch auch eine der im folgenden für Seitenwechsel-Verfahren beschriebenen Methoden möglich:

Probleme der virtuellen Speicherverwaltung

Ersetzungsproblem bei Seitenwechself Verfahren

5 gängigste Strategien zum Ersetzen einer Seite:

- **FIFO (*first-in-first-out*)**: die sich am längsten im Hauptspeicher befindende Seite wird ersetzt
- **LIFO (*last-in-first-out*)**: die zuletzt eingelagerte Seite wird ersetzt
- **LRU (*least recently used*)**: die Seite, auf die am längsten nicht zugegriffen wurde, wird ersetzt

Probleme der virtuellen Speicherverwaltung

- **LFU (*least frequently used*)**: die seit ihrer Einlagerung am seltensten benutzte Seite wird ersetzt
- **LRD (*least reference density*)** : Mischung aus LRU und LFU. Die Seite mit der geringsten Zugriffsdichte (Anzahl Zugriffe / Einlagerungszeitraum) wird ersetzt

Daneben werden bevorzugt solche Seiten ersetzt, die nicht verändert wurden ➡ kein Rückschreiben der geänderten Seite erforderlich.

Zusammenfassung

□ **Segmentierung (Segmente variabler Größe)**

- logische Abbildung einer Programmstruktur
- geringer Datentransfer
- umfangreicher Datentransfer beim Ein-/Auslagern
- externe Fragmentierung

□ **Seitenwechsel-Verfahren (Seiten fester Größe)**

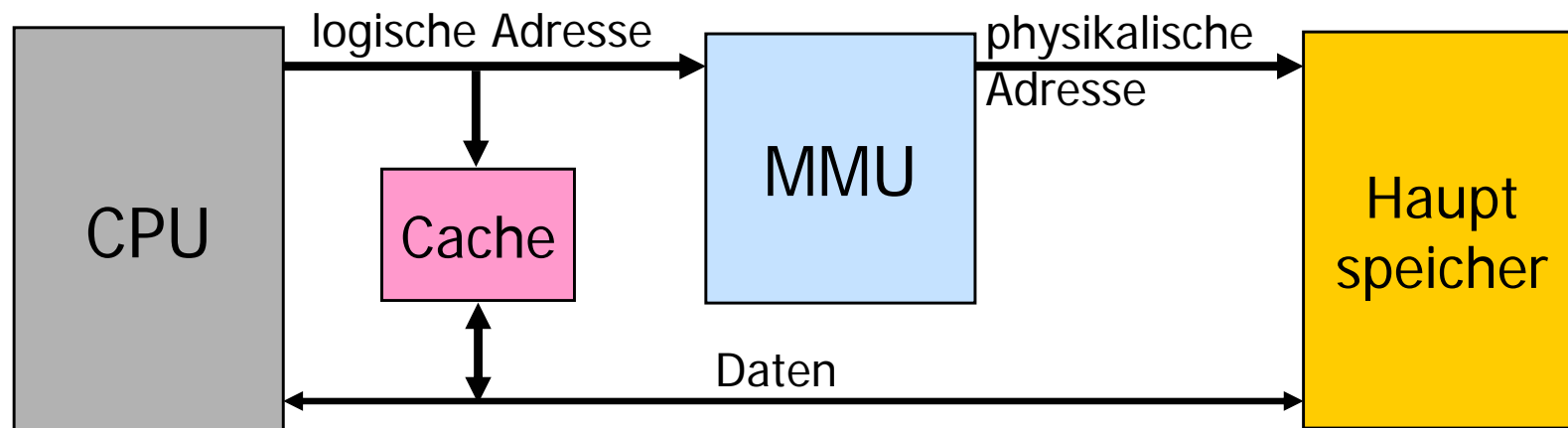
- geringerer Verwaltungsaufwand
- bessere Hauptspeicherauslastung
- häufiger Datentransfer
- interne Fragmentierung

Cache und Speicherverwaltungseinheit

Zwei Möglichkeiten der Cache-Einbindung bei virtueller Speicherverwaltung:

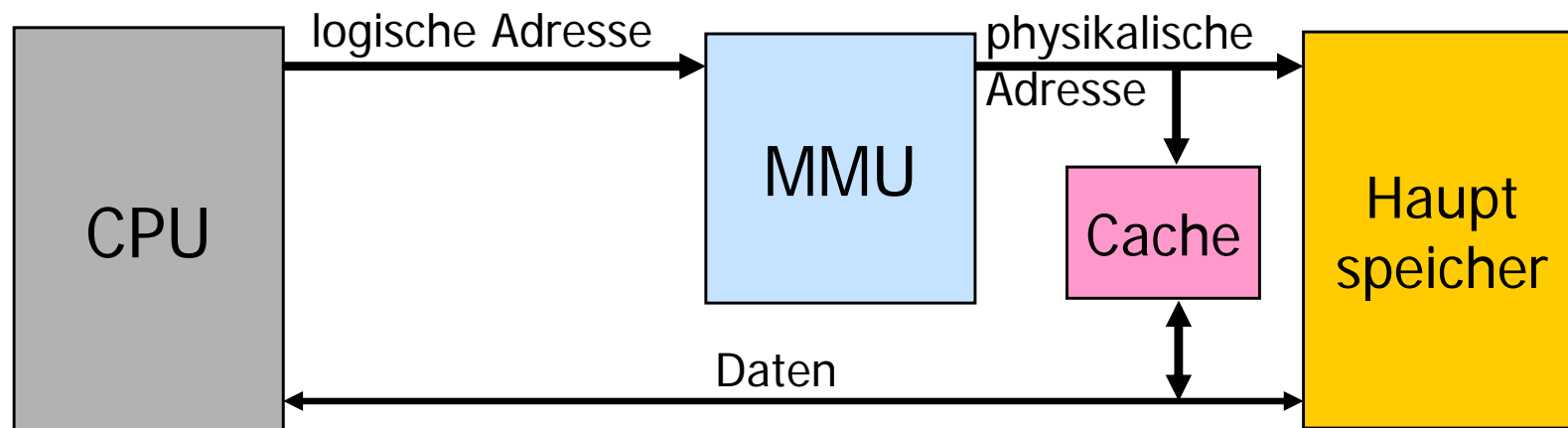
➤ **Virtueller Cache:**

wird zwischen CPU und MMU gelegt. Die höherwertigen Bits der logischen Adressen als Tags abgelegt



Cache und Speicherverwaltungseinheit

- **Physikalischer Cache:**
wird zwischen MMU und Speicher gelegt.
Die höherwertigen Bits der physikalischen Adressen werden als Tags abgelegt



Virtueller und physikalischer Cache

□ Vorteile des virtuellen Caches:

- bei Treffern wird die MMU nicht benötigt → Keine Verzögerung durch die Adressberechnung der MMU

□ Vorteile des physikalischen Caches:

- physikalische Adresse ist i. A. viel kleiner als die logische Adresse → weniger Bits müssen als Tag gespeichert werden.
- befindet sich die MMU auf dem Prozessorchip, so kann nur der physikalische Cache außerhalb erweitert werden.