Angewandte Computer- und Biowissenschaften

HOCHSCHULE MITTWEIDA
University of Applied Sciences

Fraunhofer SIT

Bundeskriminalamt

Betriebssysteme

Betriebssystemarchitektur

Autor: Ronny Bodach

Stand: 09.04.2022

Agenda

1. Grundlegender Aufbau von Rechnern
2. Von Neumann Rechnerarchitektur
3. Harvard-Architektur
4. Computerarchitektur nach Tanenbaum
    1. Transistorebene
    2. Logische Ebene
    3. Microarchitektur
    4. Instruction Set Architecture
    5. Betriebssystem
5. Betriebssystem-klassifikation
6. Interrupts
7. Prozess, Task und Thread
8. Scheduling
9. Parallelität und Nebenläufigkeit
10. **Speicherverwaltung**

Speicherverwaltung

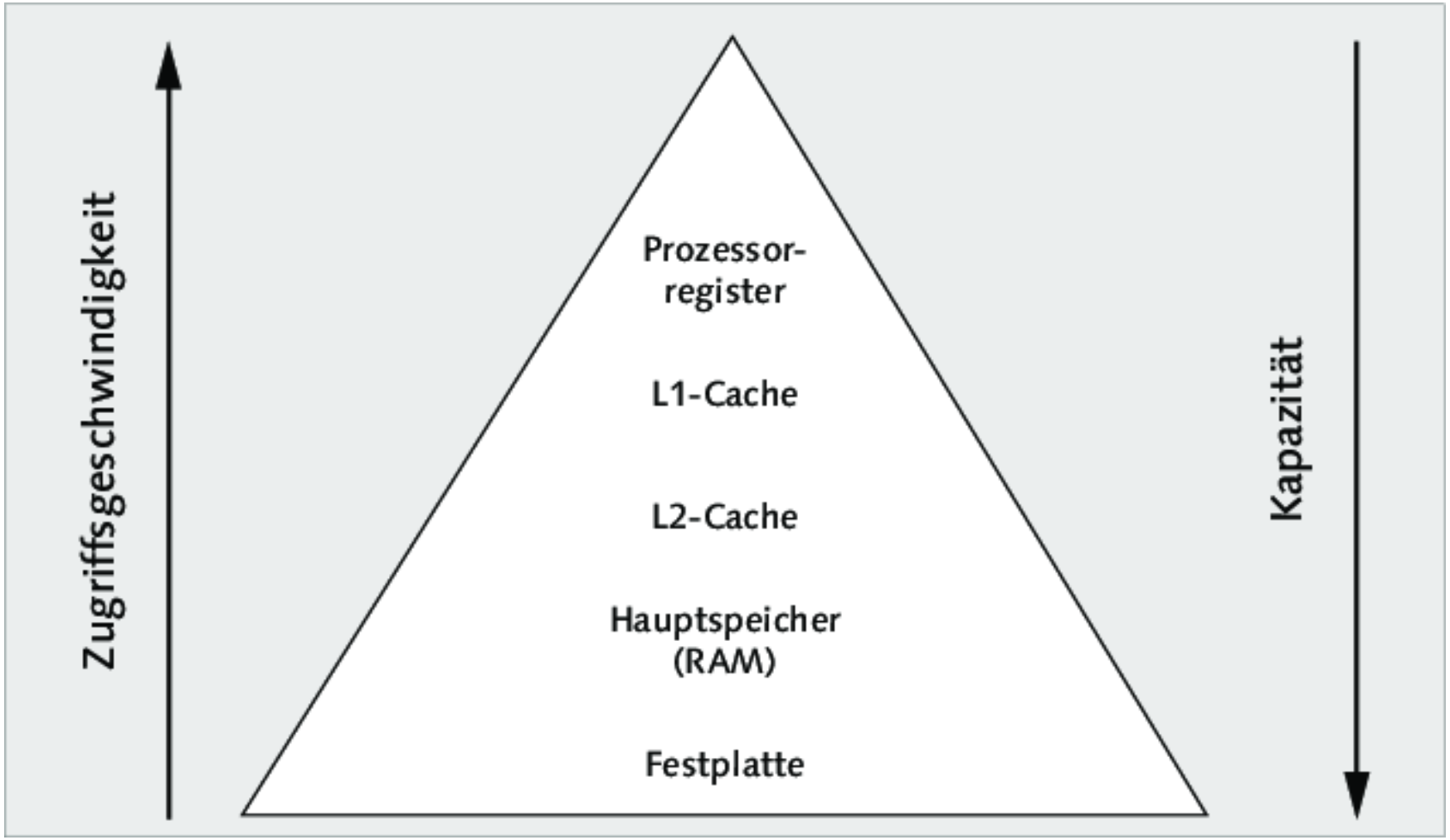
Hauptspeicher
Random Access Memory (RAM)

- Größe des Hauptspeichers auf 32 Bit System 2^{32} Byte was insgesamt 4GiB ausmacht
- Größe des Hauptspeichers auf 64 Bit System 2^{64} Byte was insgesamt 18EiB ausmacht
- verschiedenen Maschinenbefehle können auf diese Adressen zugreifen und die dort gespeicherten Bytes lesen oder schreiben
- während der Prozessabarbeitung referenziert das Befehlsregister den nächsten auszuführenden Befehl, indem es dessen Hauptspeicheradresse speichert

Cachespeicher

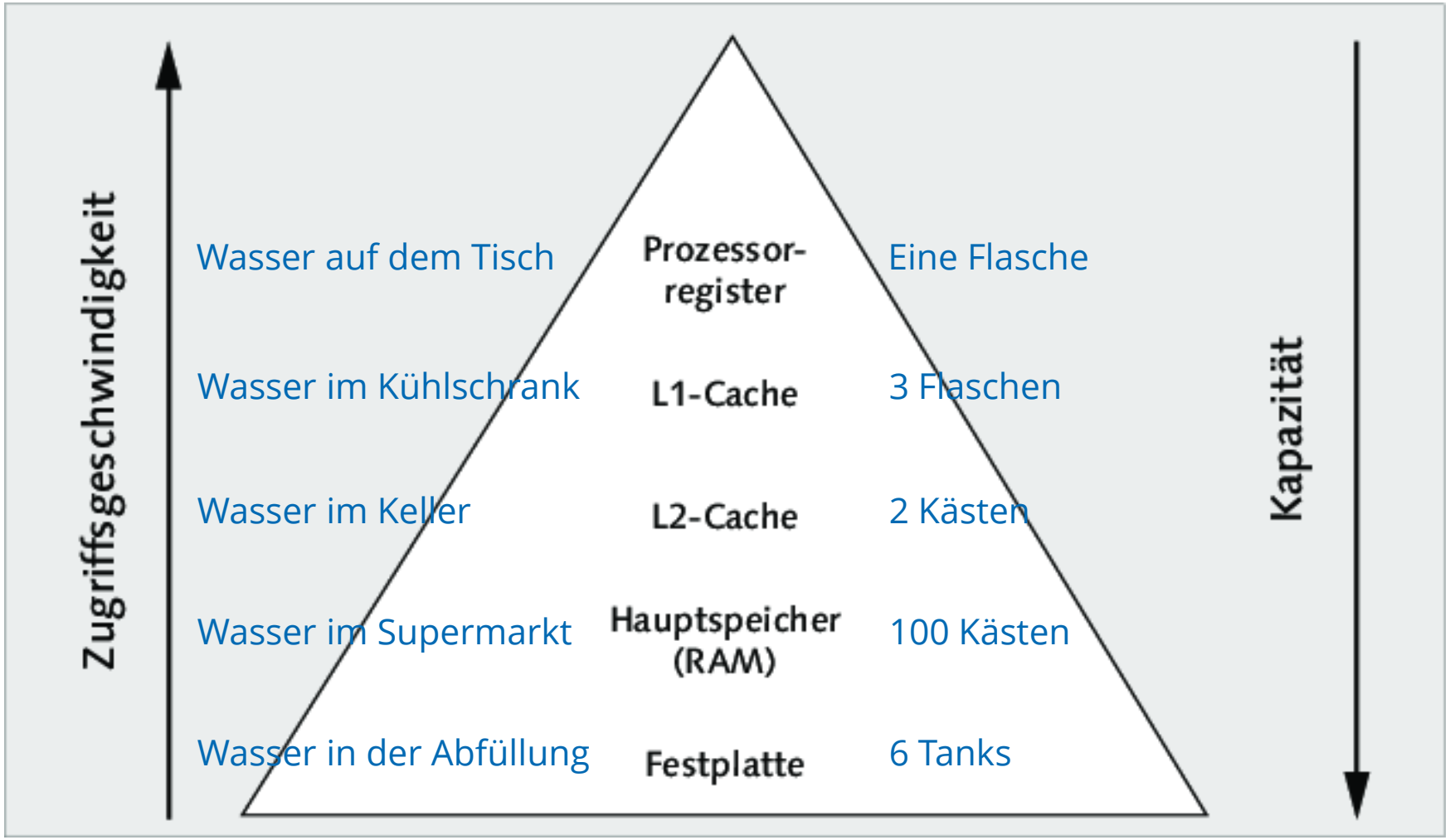
- der Hauptspeicher ist langsamer als die Abarbeitung von Befehlen im Prozessor
- Einführung Pufferspeicher, sogenannte Caches, um Zugriffszeit auf häufig benutzte Datensätze und Variablen des Hauptspeicher zu verkürzen
- diese werden in verschiedenen Level (L1 Cache, L2 Cache und L3 Cache) aufgeteilt
- Caches befinden sich entweder direkt auf dem Prozessor-Chip (L1-Cache) oder „direkt daneben“
- Caches sind zwischen ein paar Kilobytes und wenigen Megabytes groß
- gepufferte Hauptspeicherwerte sind transparent zwischengespeichert, der Prozess greift auf Adressen im Hauptspeicher zu, ohne dass ein Cache für ihn ersichtlich ist

Speicherpyramide



Quelle: Linux, Plötner et.al., Rheinwerk Computing

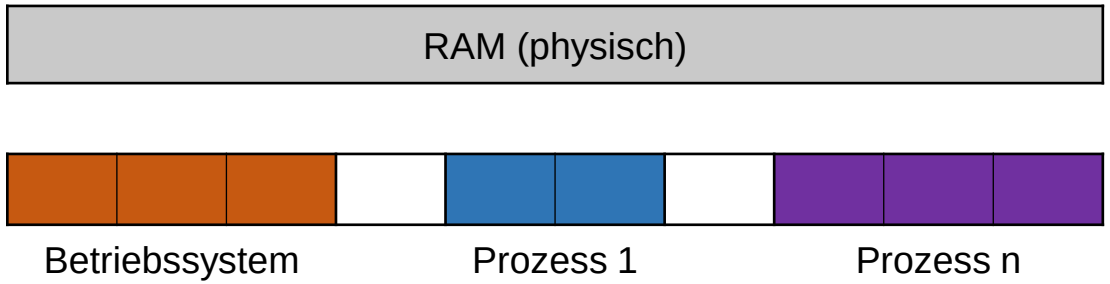
Speicherpyramide (Analogie)



Quelle: Linux, Plötner et.al., Rheinwerk Computing

Hauptspeicherbereiche

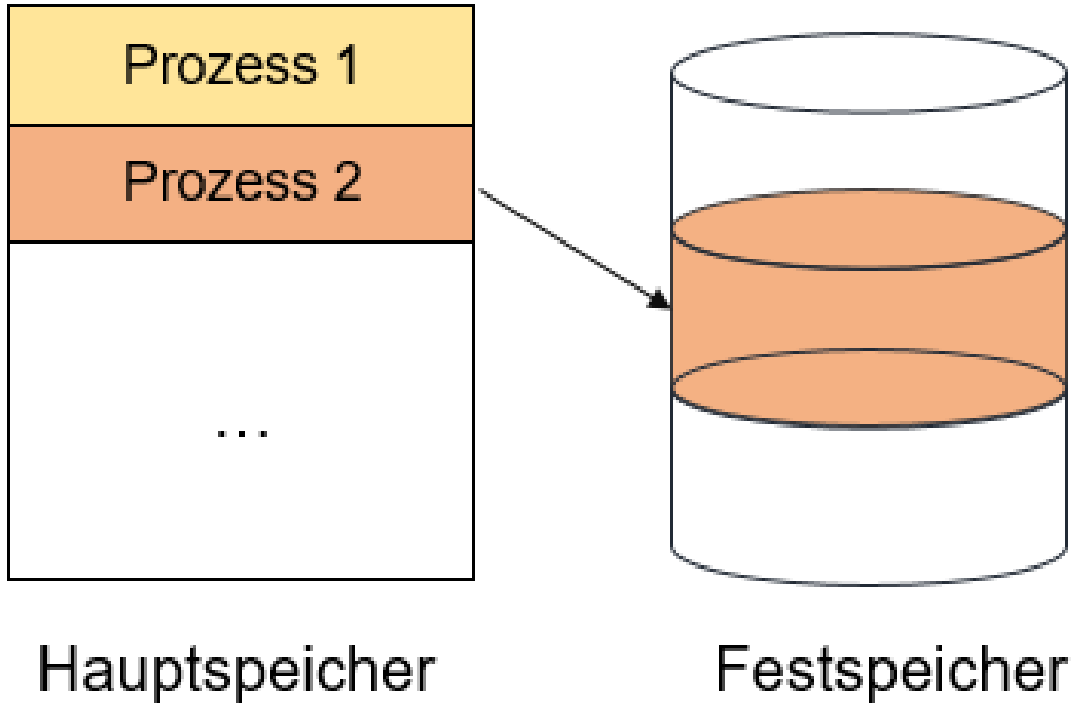
- Der Hauptspeicher ist in mehrere Bereiche aufgeteilt:
  - Speicherbereich für das Betriebs-system (Programm, Puffer, Variablen)
  - Speicherbereich für Prozesse (Programm und Daten)



- Die Speicherverwaltung stellt die statische oder dynamische Aufteilung des Speichers für aktuell gestartete Prozesse bereit:
  - jeder Prozess bekommt eigenen Adressraum (virtueller Speicher)
  - ein Adressraum ist die Abstraktionsebene vom physikalischen Speicher
  - Speicherzellen im Hauptspeicher haben dabei eine eindeutige Speicheradresse

Relokation / Swapping

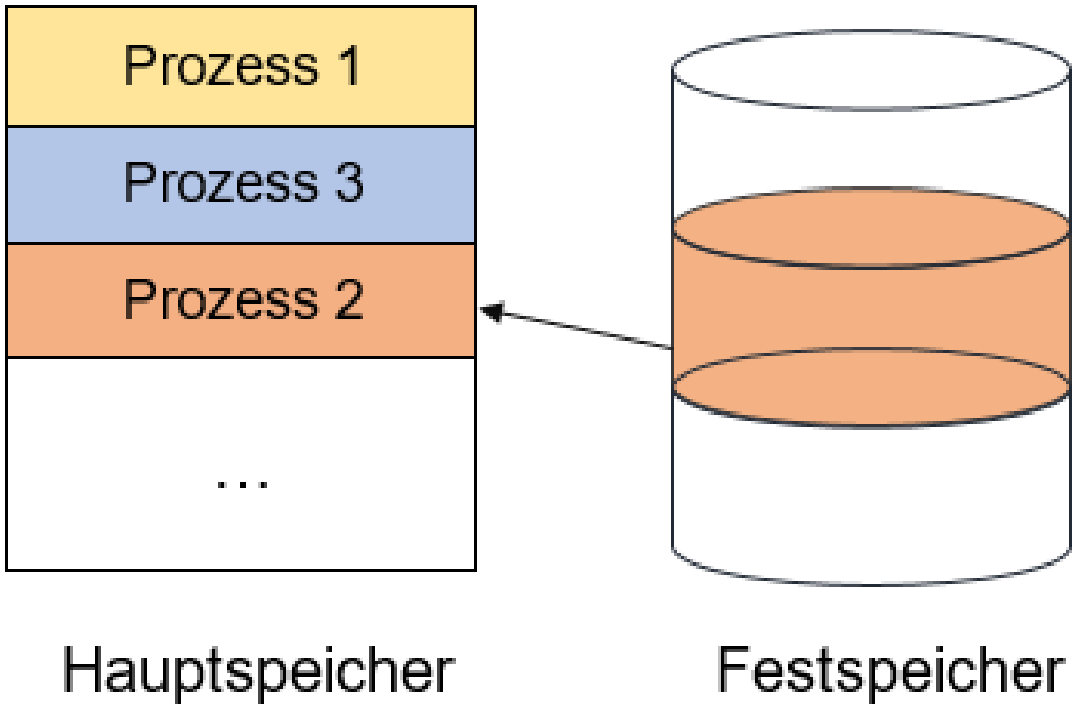
- Relokation = Verlagerung / Swapping = Austauschen
- Zweck ist es mehrere Prozesse gleichzeitig im System zu halten



Auslagern eines Prozesses aus dem Hauptspeicher in den Festspeicher

Relokation / Swapping

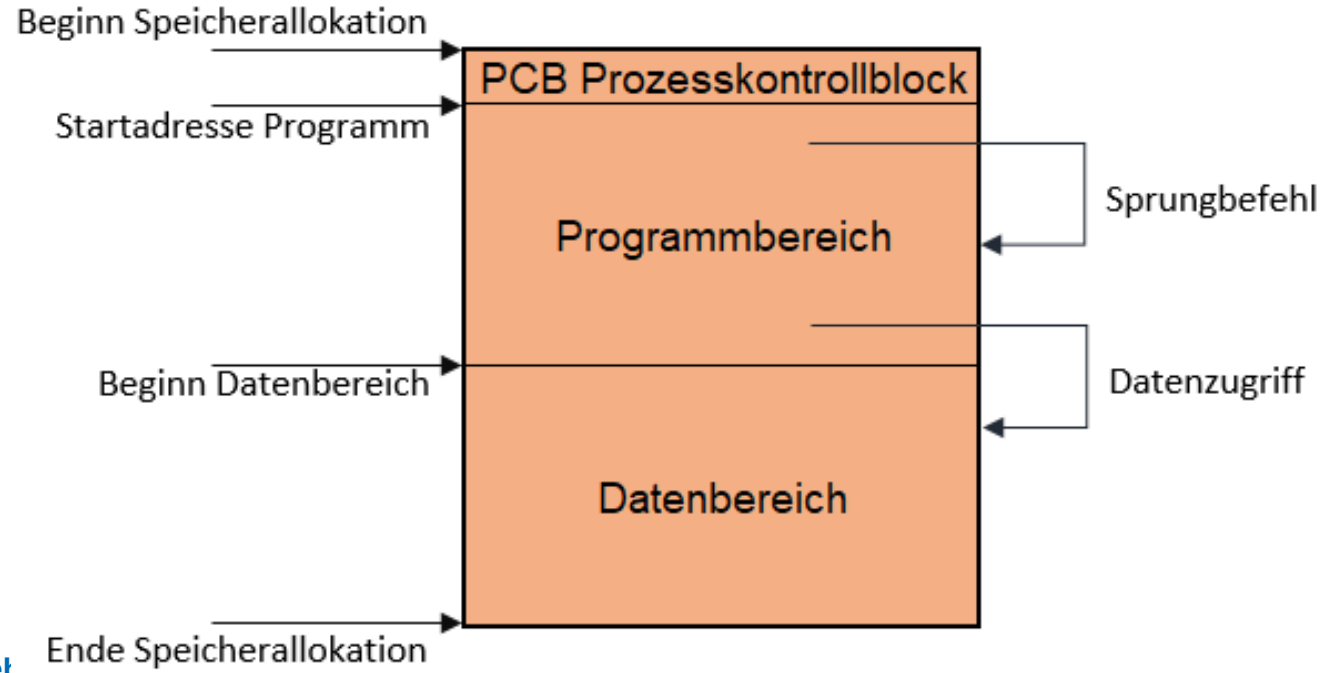
- Relokation = Verlagerung / Swapping = Austauschen
- Zweck ist es mehrere Prozesse gleichzeitig im System zu halten



Wiedereinlagern eines Prozesses aus dem Festspeicher in den Hauptspeicher

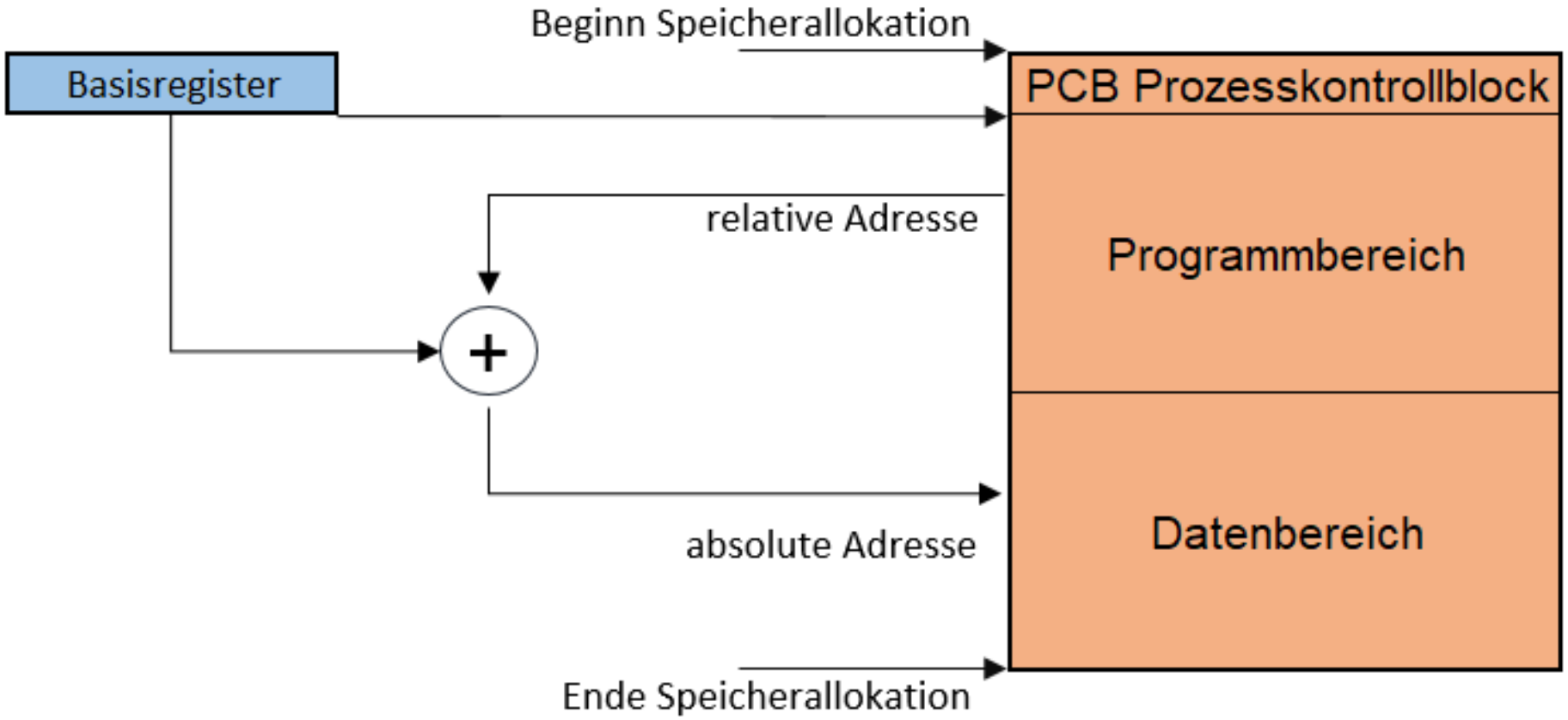
Relokation / Swapping

- Probleme stellen hierbei aber absolute Sprungbefehle in den Prozessorregistern der einzelnen Prozesse dar
- nicht jeder Prozess beginnt an Hauptspeicher Adresse 0x0000 0000
- Konzept zur Verwendung von relativen Speicheradressen notwendig



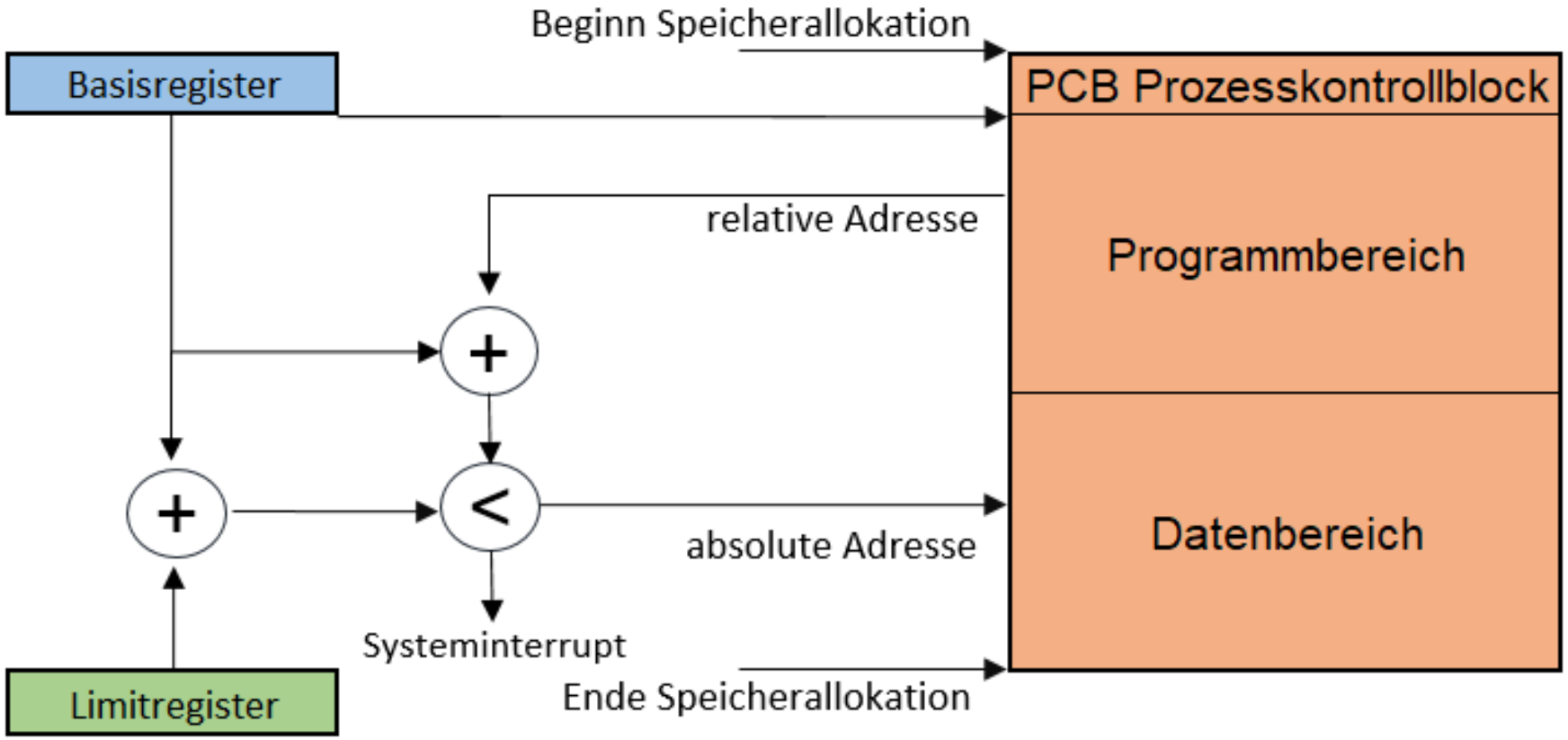
Lokationskonzept

Die absolute Adresse wird über ein Basisregister realisiert:



Schutzkonzept

Mit Hilfe des Limitregister ist ein Schutzmechanismus der einzelnen Prozesse realisierbar:

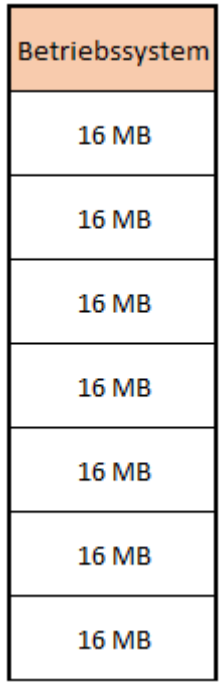


Partitionierung

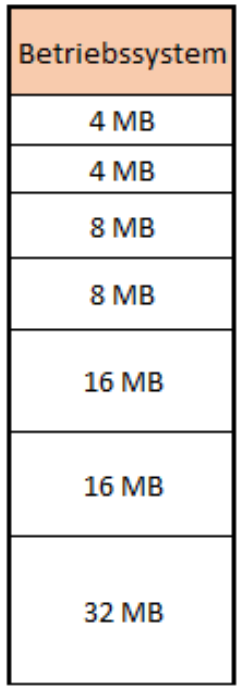
- Die Aufteilung des Speichers in einzelne Bereiche mit definierten Grenzen nennt man Partitionierung
- Partitionierung ist bereits aus dem Bereich der Dateisysteme bekannt
- ein zusammenhängender Teil des Hauptspeichers wird fest für Betriebssystem reserviert
- ein weiterer zusammenhängender Teil des Speichers wird für jeden einzelnen Prozess genutzt
- Realisiert wird dies mit unterschiedlichen Varianten:
  - Statische Partitionierung
  - Dynamische Partitionierung

Statische Partitionierung

Aufteilung des Speichers in feste Anzahl von Fragmenten mit zwei möglichen Varianten:

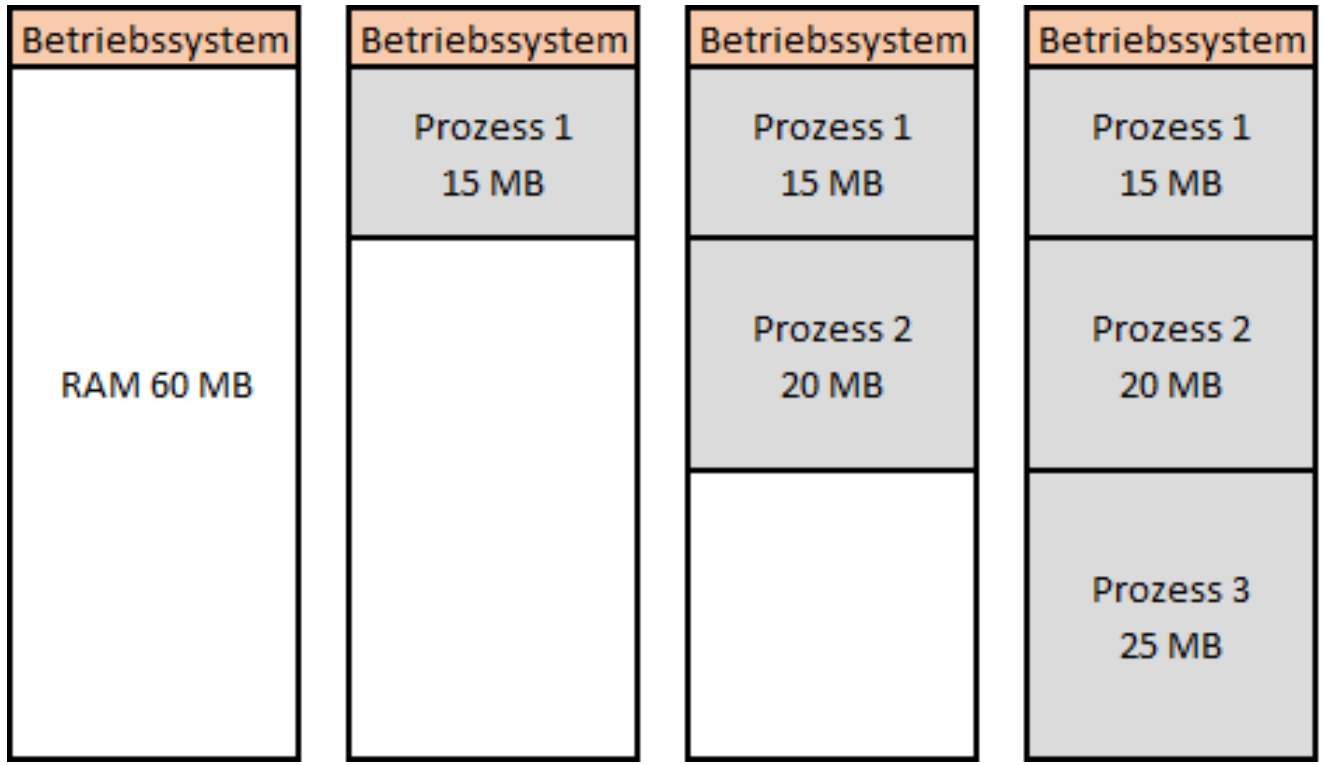


Partitionen mit fester Fragmentgröße

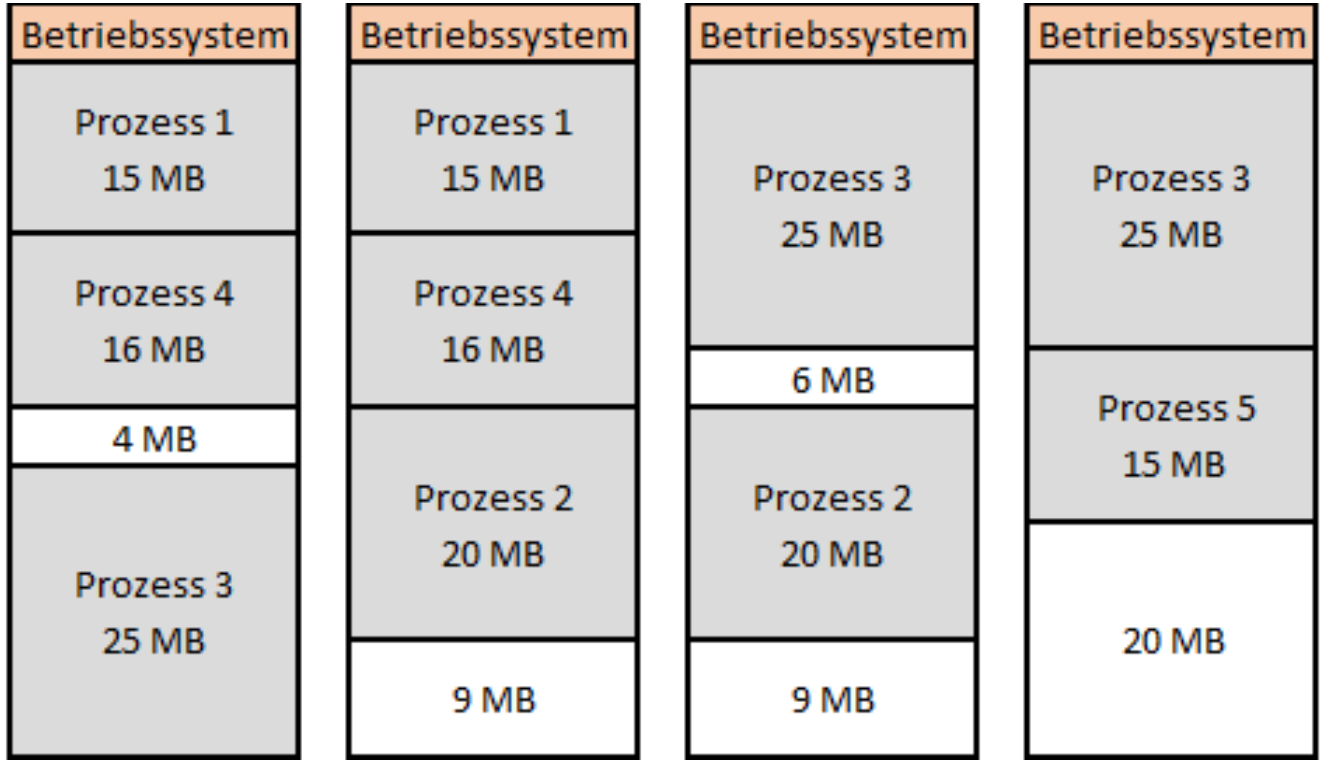


Partitionen mit unterschiedlicher Fragmentgröße

Dynamische Partitionierung



Dynamische Partitionierung



Dynamische Partitionierung

- Strategien für die Auswahl des freien Speicherblocks bei dynamischer Partitionierung:
- Best Fit:
  - Suche nach dem kleinsten Block der den Prozess fasst
- First Fit:
  - Suche nach dem ersten Block der groß genug ist den Prozess aufzunehmen
- Next Fit:
  - Suche nach dem ersten Block der groß genug ist den Prozess aufzunehmen beginnend ab letzter Speicherzuweisung

Das virtuelle Speicherkonzept

- Probleme der bereits vorgestellten Speicherkonzepte:
- die physikalische Grenze des Hauptspeichers gibt Rahmenbedingungen für die Allokation des RAM vor
- Prozesse können nur den Teil des Hauptspeichers beanspruchen, der nach dem Laden des Betriebssystems noch zur Verfügung steht
- Ausführung von Prozessen mit mehr Speicherbedarf nur auf System mit mehr RAM oder besonderen Speicherumgebungen möglich (MS-DOS EMM 386 / HIGHMEM)

Virtuelle Speicherverwaltung pro Prozess

Das virtuelle Speicherkonzept

- Virtuelle Speicherverwaltung pro Prozess bedeutet:
- Aufteilung des zugestandenen Speichers pro Prozess wird virtualisiert
- Management erfolgt durch Betriebssystem mit oder ohne Hardwareunterstützung (MMU)
- die Größe des virtuell zugewiesenen Speichers ist einzig abhängig von der Bitbreite der Betriebssystemarchitektur:

➢ 16 Bit = 64 kiB / 32 Bit = 4 GiB / 64 Bit = 18 EiB

- ein Prozess kann mehr virtuellen Speicher zugewiesen bekommen als real vorhanden ist: Hauptspeicher + Hintergrundspeicher = virtueller Speicher

➢ PROBLEM: Einlagerung ganzer Prozesse nicht möglich

Das virtuelle Speicherkonzept

Mandl 2013 fasst daher das Grundkonzept der virtuellen Speicherverwaltung wie folgt auf:

- Der Speicherbedarf eines Prozesses sollte größer als der physikalisch vorhandene Hauptspeicher sein können.
- Ein Programmierer sollte nur einen kontinuierlichen (linearen) Speicherbereich beginnend mit Adresse 0 sehen und sich nicht um die Fragmentierung des Hauptspeichers kümmern müssen.
- Ein Prozess sollte auch dann noch ablaufen können, wenn er nur teilweise im Hauptspeicher ist. Wichtig ist, dass die Teile des Prozesses (Daten und Code) die gerade benötigt werden im physikalischen Speicher sind.

Das virtuelle Speicherkonzept

Für die Realisierung dieses Grundkonzeptes sind komplexere Methoden der Speicherverwaltung notwendig.

Diese Methoden können wie folgt eingeordnet werden:

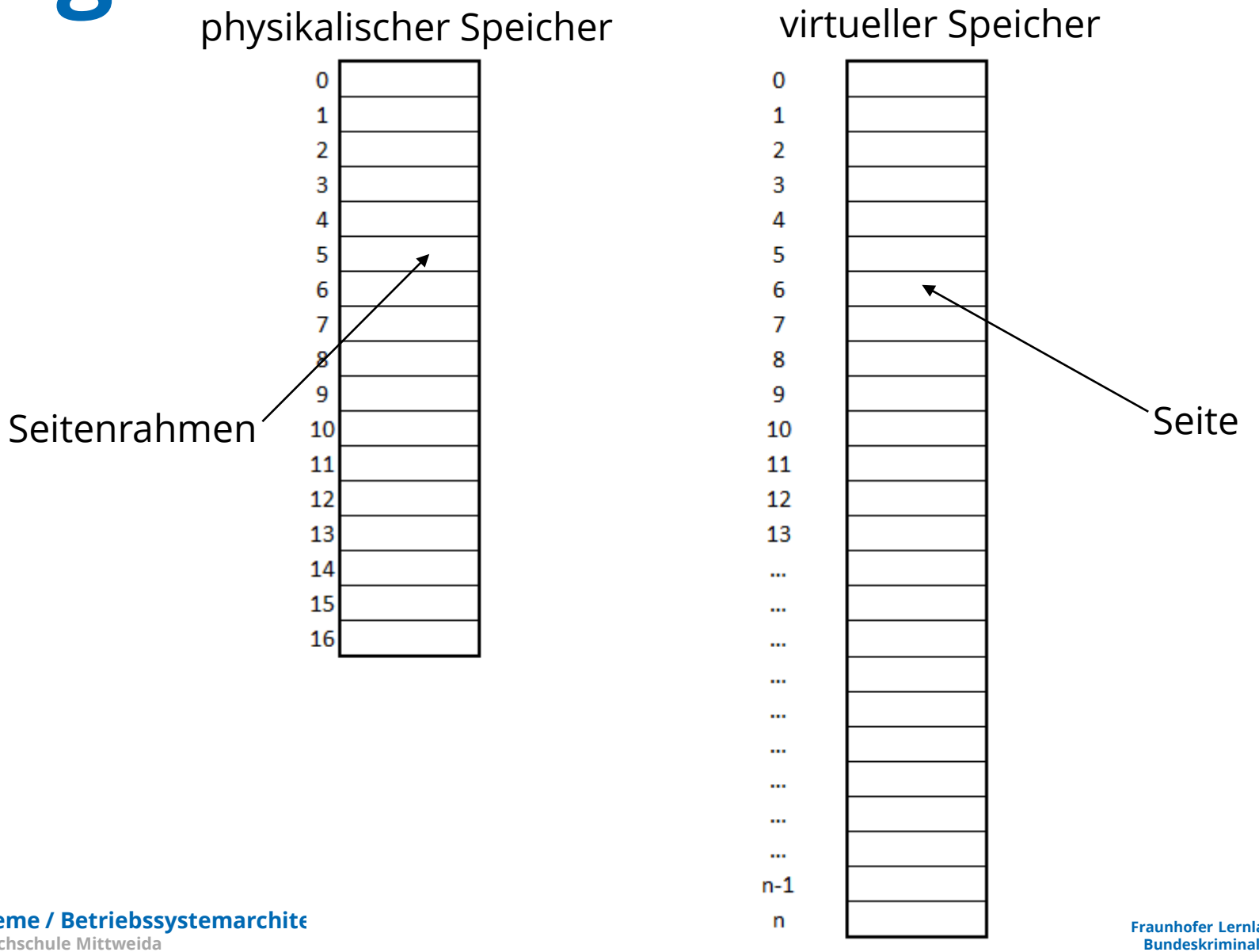
- Paging
- Segmentierung
- Paging mit Segmentierung

Paging

Charakteristika des virtuellen Speicherkonzept mit Paging ist:

- ähnlich dem statischen Partitionieren
- physikalischen Speicher in gleich große Fragmente aufteilen
- Fragmente stellen die sogenannten Seitenrahmen (Page Frames) dar
- ein Prozess muss nicht vollständig in ein solchen Seitenrahmen passen
- ein Prozess erhält einen virtuellen Speicherbereich mit max. 2 Bitbreite Architektur
- die Aufteilung des virtuellen Speichers eines Prozesses ist dynamisch in einzelnen Seiten (Pages) entsprechend der Größe eine Seitenrahmens

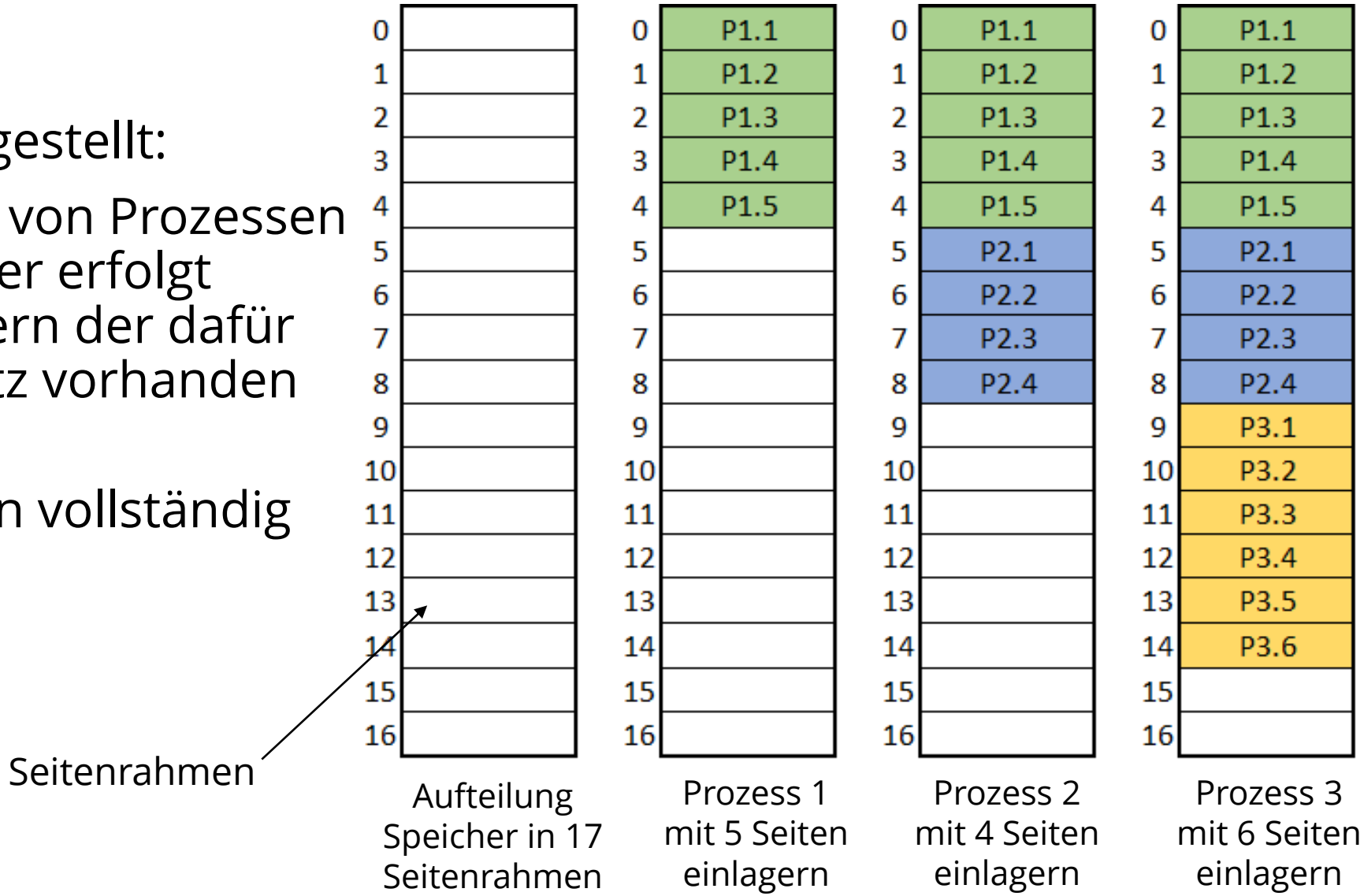
Paging



Paging

Hier im Beispiel dargestellt:

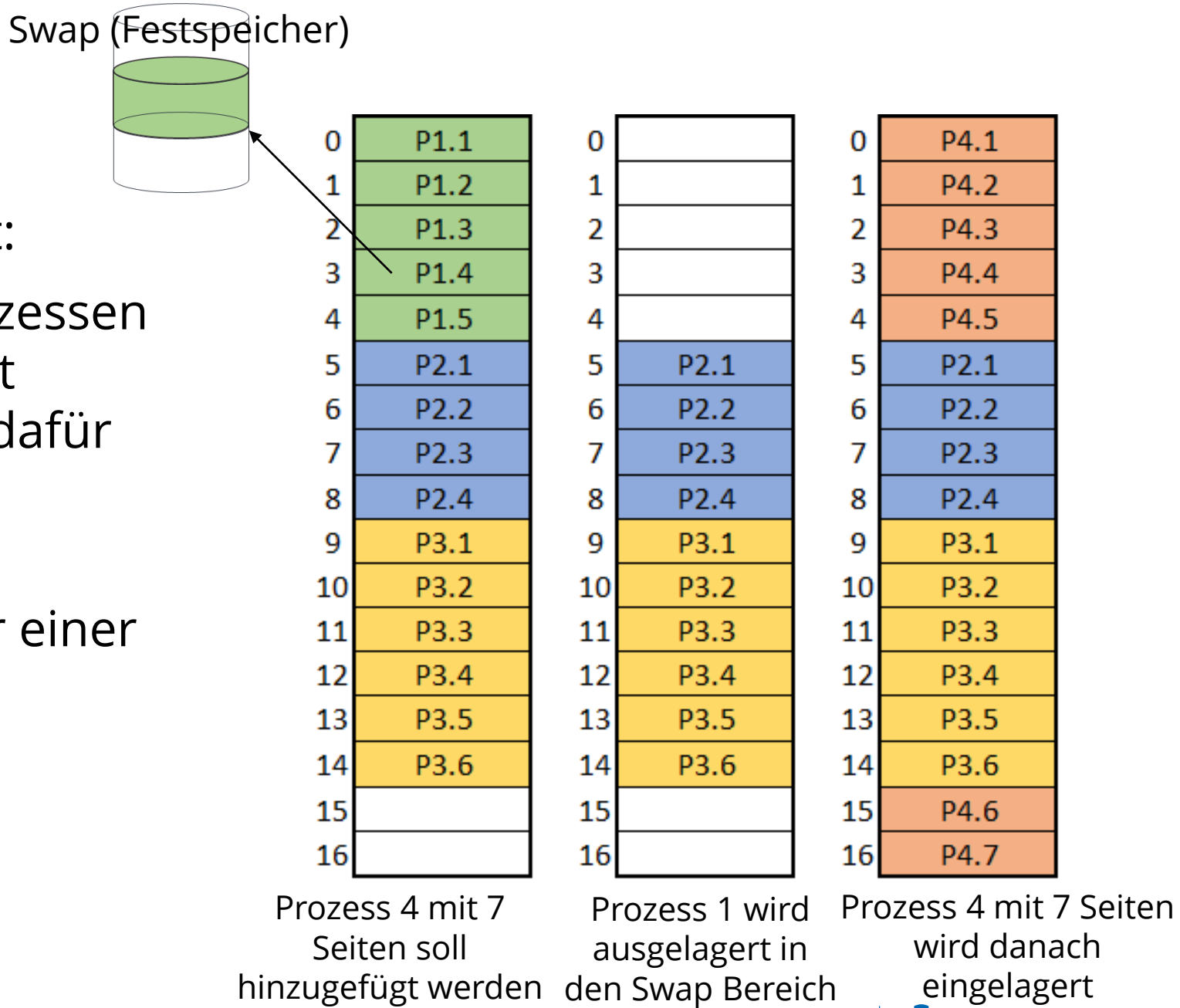
- Die Einlagerung von Prozessen im Hauptspeicher erfolgt sequentiell, sofern der dafür notwendige Platz vorhanden ist.
- Prozesse werden vollständig eingelagert



Paging

Hier im Beispiel dargestellt:

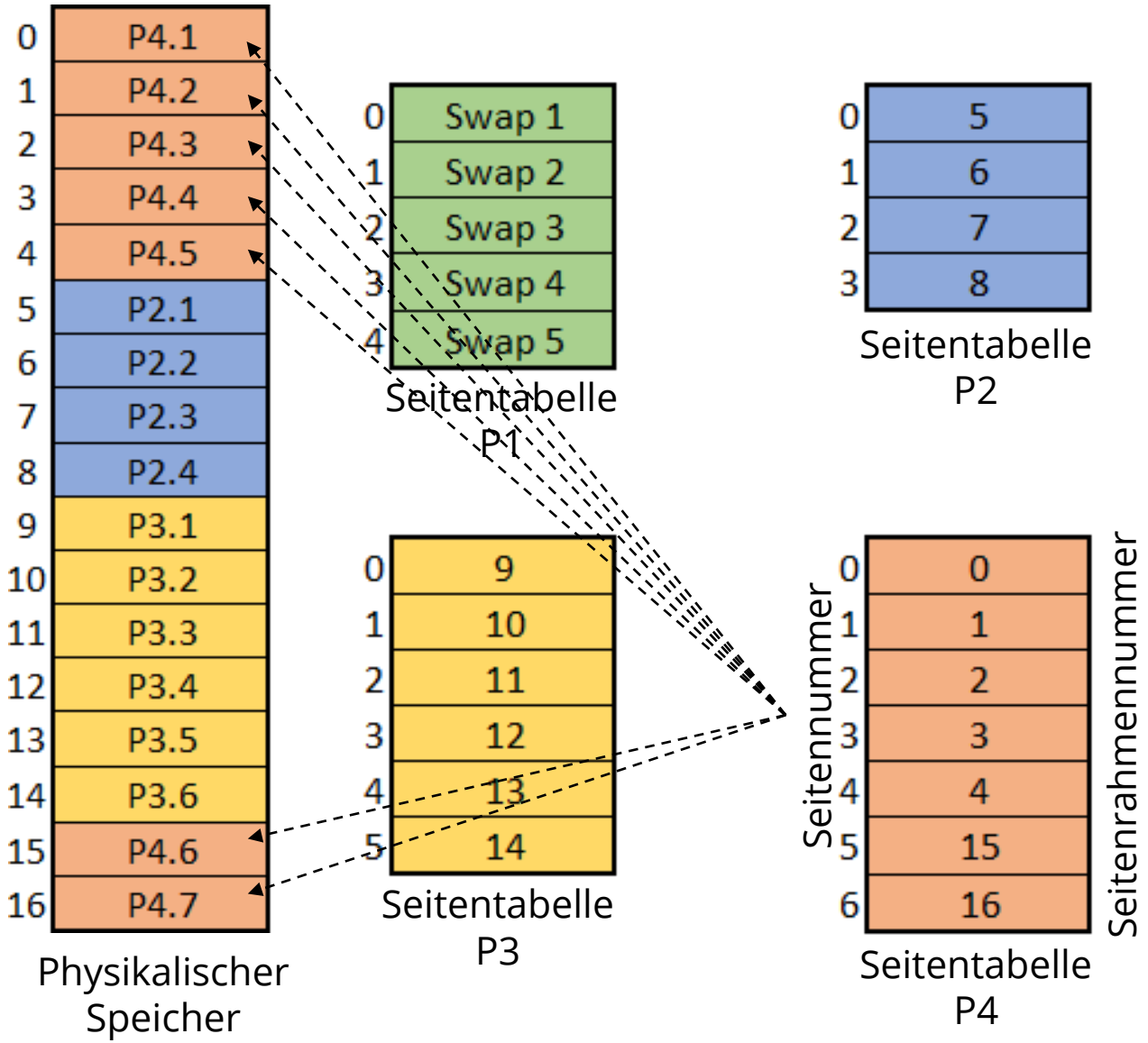
- Die Einlagerung von Prozessen im Hauptspeicher erfolgt fragmentiert, wenn der dafür notwendige Platz nicht vorhanden ist.
- Prozesse müssen vorher einer Relokation / Swapping unterzogen werden.



Paging

Hier im Beispiel dargestellt:

- Die Zuordnung von Prozessfragmenten also den Pages im Hauptspeicher erfolgt mittels Seitentabellen (Page Tables) zum jeweiligen physischen Seitenrahmen
- Jeder Prozess benötigt dazu eine eigene Seitentabelle.



Paging

Die Berechnung von Adressen wird unterschieden zwischen virtueller Adresse und physischer Adresse.

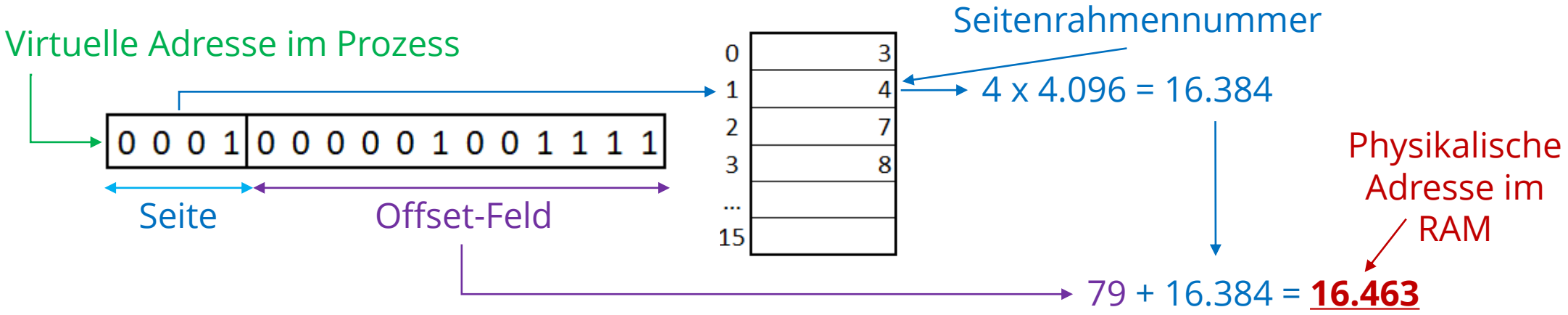
Es gilt:

- die Seitenanzahl und Seitenrahmenanzahl ist eine Zweierpotenz
- die virtuelle Adresse im Programmablauf besteht aus der Seitennummer und dem Offset
- die physikalische Adresse besteht aus der Seitenrahmennummer und dem Offset

Paging

Bei einer Bitbreite von 16 Bit kann die Aufteilung mit einer Seitengröße von 4 KiB = 2^{12} = 4.096 Bytes wie im folgenden Beispiel realisiert werden:

- ein Offset-Feld von 12 Bit wird benötigt, um alle 4.096 Bytes referenzieren zu können
- ein einzelner Prozess kann an Hand der 4 Bit Seitennummer bis zu 2^4 = 16 verschiedene Seiten referenzieren, die über die Seitentabelle des Prozesses auf Seitenrahmen im Hauptspeicher verweisen.



Paging

Die bisherigen Betrachtungen beliefen sich alle auf Prozesse die vollständig in den Hauptspeicher geladen werden können.

Wenn jedoch weniger Hauptspeicher zur Verfügung steht als virtueller Speicher für einen Prozess zugeteilt wird, ist eine Erweiterung des Konzeptes notwendig.

Dabei gilt folgendes als Voraussetzung:

- Einlagerung von Prozessen nur teilweise in einzelnen benötigten Seiten
- Nachladen von Seiten sofern Seite nicht im Speicher eingelagert
- Programmabarbeitung unbeeinträchtigt von der Seitenauslagerung

On Demand Paging

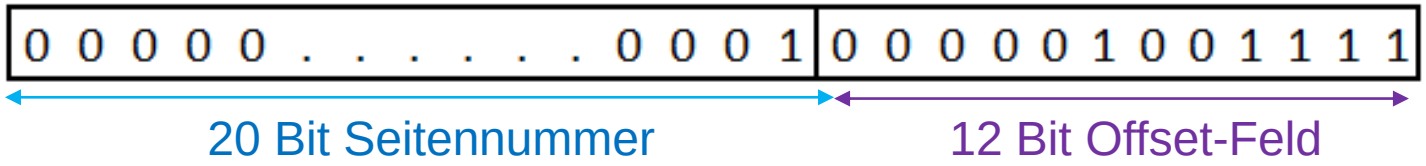
On Demand Paging

Grundlage für die Umsetzung einer teilweisen Seiteneinlagerung:

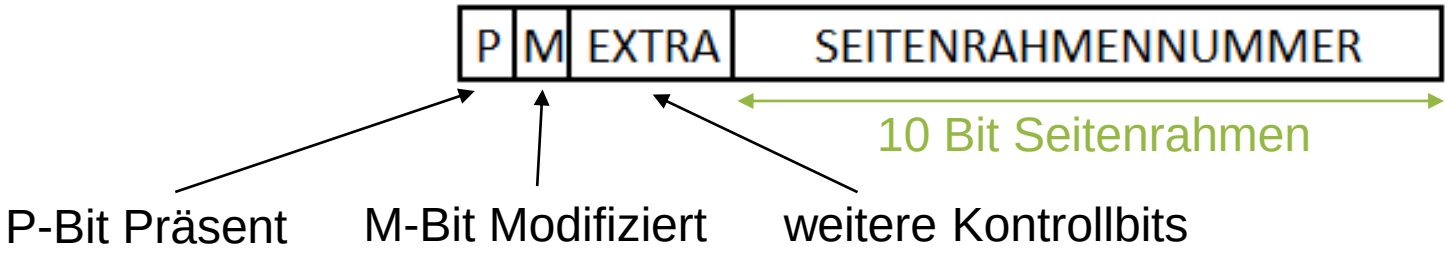
- Überwachung der Nutzung von einzelnen Seiten und deren aktueller Einlagerungszustand.
- dafür zusätzliche Angaben in der Seitentabelle eines Prozesses
- Strategien zur Ersetzung von Seitenrahmeninhalten
- effiziente Überwachung der Lokalität bei Seitenzugriffen

On Demand Paging

Durch die Aufteilung der Anzahl der Bit für Seiteneintrag und Offset kann eine Seitenanzahl beispielsweise von 20 Bits und ein Offset von 12 Bit für 32 Bit Systeme genutzt werden:



Der dazugehörige Eintrag in der Seitentabelle wird ergänzt um Status Bits für eingelagerte Seiten (P Präsent) und geänderte Seiten (M Modifiziert):



Funktionsweise On Demand Paging

- Prozess möchte Speicherbereich der Seitentabelle aufrufen der nicht eingelagert ist
- Prozess wird durch einen Seitenfehler (Page Fault) Interrupt unterbrochen
- Prozess wird eingelagert
- bei fehlender Speicherseite im Festspeicher 2 Varianten:
  - Seite wird im Hauptspeicher angelegt
  - bei fehlender Speicherseite wird Page Fault Error ausgegeben und Prozess gestoppt

Seitenersetzungsstrategien

- wenn kein freier Seitenrahmen verfügbar, auslagern von Seitenrahmen aus dem Hauptspeicher
- M-Bit gesetzt, Seitenrahmen in Festspeicher auslagern
- Seitentabelleneintrag entfernen
- Seitenrahmenverdrängung durch verschiedene Techniken möglich:
  - First in First Out, FiFo
  - Least Recently Used, LRU
  - Second Chance, Clock/Uhrzeiger
- danach Seite aus Festspeicher einlagern und Seitentabelle aktualisieren

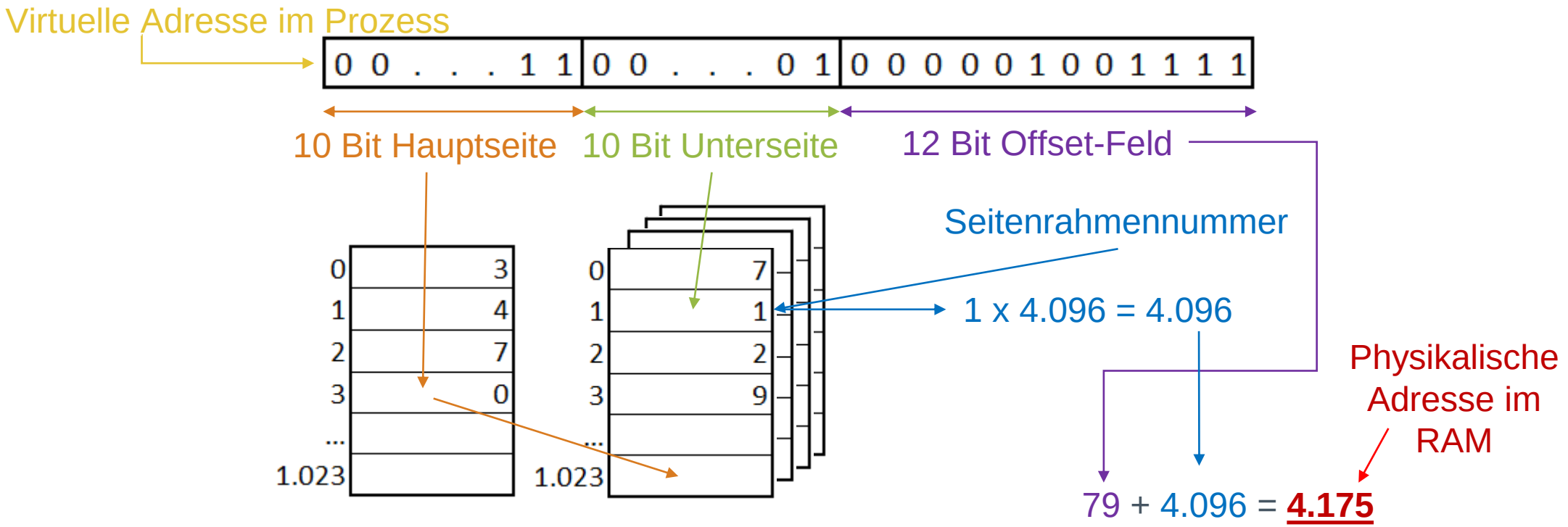
Mehrstufige Seitentabelle

Mehrstufige Seitentabellen wenden die Physical Address Extension (PAE) an:

- auch die Seitentabellen werden in Pages unterteilt und im virtuellen Speicher implementiert.
- Untertabellen sind als Pages entweder eingelagert oder ausgelagert abgelegt
- damit muss keine einzelne große Seitentabelle wie im Beispiel mit 2^{20} = 1 GiB = 1.048.576 Zeilen also 256 Seitenrahmen zu je 4KiB mehr verwaltet werden
- damit wird die Ablage der kompletten Seitentabelle im Hauptspeicher möglich, da nur zwei Tabellen zu 2^{10} = 1 KiB = 1.024 Einträgen benötigt wird

Mehrstufige Seitentabelle

Im Folgenden unser Beispiel mit 4 KiB = 4.096 großem Seitenrahmen und einer Seitentabelle von zweimal 2^{10} = 1 KiB = 1.024 Einträgen.



Translation Lookaside Buffer (TLB)

- Beim Paging werden zusätzliche Speicherzugriffe auf die Seitentabellen, die ebenfalls im Speicher abgelegt werden notwendig.
- Speicherzugriffe erfordern Zugriffszeit und Bremsen das System
- Beschleunigung nur durch zusätzlichen Cache für Adressübersetzung (Adressumsetzungspuffer, TLB)
- der TLB enthält Einträge der Seitentabellen, auf die zuletzt zugegriffen wurde gemäß dem örtlichen bzw. räumlichen Lokalitätsprinzip
- bei Zugriffen auf eine nicht im TLB enthaltene Seite wird die entsprechende Seite in den TLB eingetragen

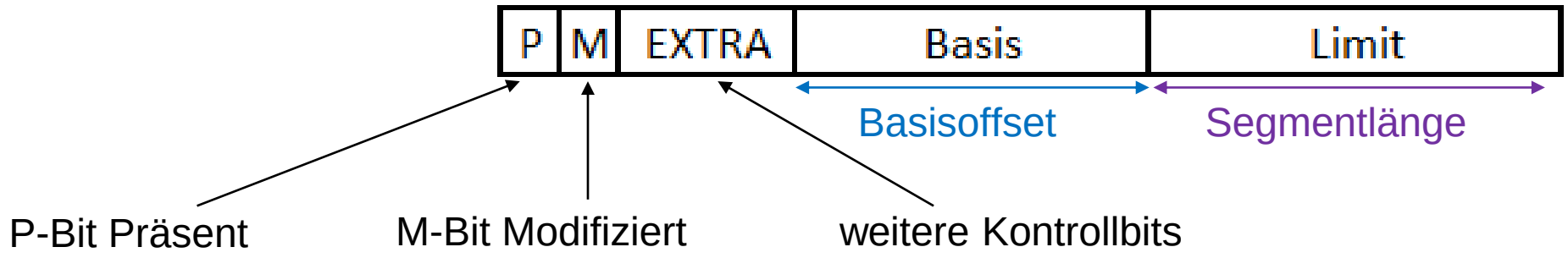
Segmentierung

Eine weitere Möglichkeit der Verwaltung des virtuellen Speichers bietet die Segmentierung an:

- Aufteilung virtueller Adressraumes Prozess in einzelne Segmente (Programmsegment, Datensegment, Stacksegment,etc...)
- Segmentgröße kann differenzieren
- Speicheranforderung pro Prozess nicht zwingend zusammenhängend und nicht ständig eingelagert
- Schutzmechanismus mittels Basis- und Limitregister pro Segment umsetzbar
- Speicheranpassung leicht pro Segment realisierbar

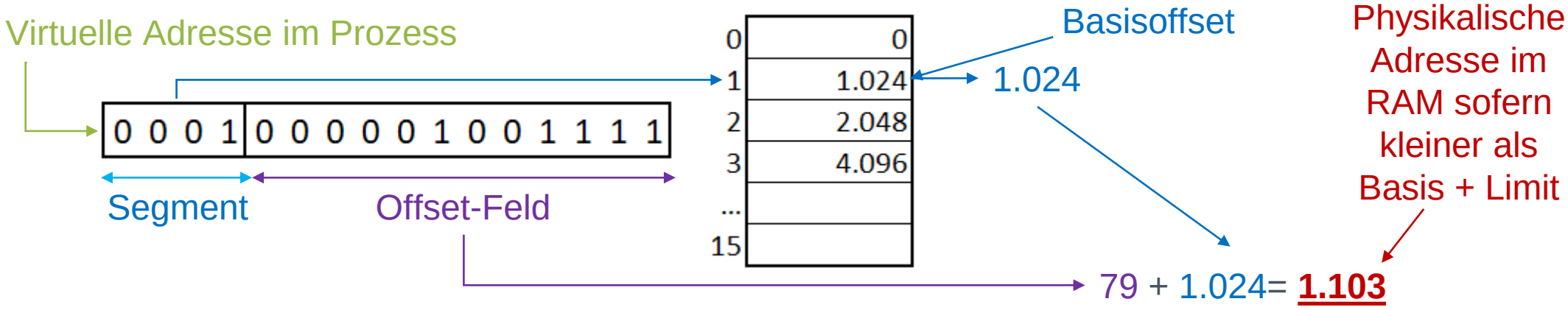
Segmentierung

Für jeden Prozess ist eine Segmenttabelle notwendig mit folgenden Eigenschaften eines Segmenttabelleneintrags:



Segmentierung

Die Umrechnung der virtuellen Adresse in die physikalische Adresse erfolgt folgendermaßen:



Segmentierung

Um große Segmente effizient zu verwalten ist eine Kombination von Segmentierung und Paging on Demand notwendig.

Dafür existieren zwei Realisierungsmöglichkeiten:

- eine Segmenttabelle pro Prozess und pro Segment eine eigene Seitentabelle (Unix)
- eine Segmenttabelle pro Prozess und eine einzelne Seitentabelle für alle Segmente (Windows)

PrePaging und Trashing

Überwachung der Lokalität bei Seitenzugriffen:

- Zeitlich: Zugriff auf häufig genutzte Seiten von Prozessen, dazu notwendig zusätzliche Überwachung des Nutzungsverhaltens von Seiten
- Räumlich: Datenzugriffe im unmittelbaren Umfeld der Daten (sequentiell in die Zukunft gerichtet), dabei vorausschauende Einlagerung von Seiten in die Seitenrahmen

Verhindert zu viele Seiteneinlagerung, Umlagerungen, Auslagerungen beim Kontextwechsel einzelner Prozesse, dem sogenannten Trashing (Flattern) durch:

- zu wenig Speicher
- zu viele speicherintensive Prozesse

Copy on write

Vorrangig eingesetzt beim Forking (Duplizieren) von Prozessen in Linux Umgebungen:

- verfügbar machen einer Speicherseite im Adressraum eines anderen Prozesses
- bei Lesendem Zugriff ist es nicht nötig, die Daten tatsächlich zu kopieren und ein weiteres Mal im Hauptspeicher anzulegen
- es genügt, wenn die beiden Prozesse auf ein- und dieselbe Speicherseite zugreifen
- wenn einer der beiden Prozesse die Daten zu ändern versucht, müssen diese tatsächlich (und dann auch nur teilweise) kopiert werden

Shared Memory

Optimierung der Speicherauslastung und der Kontextwechsel durch gemeinsam genutzte Speicherbereiche („Shared Memory“)

- Programmsegmente müssen nicht mehrfach im Hauptspeicher gehalten werden
- bei Anwendungen mit mehreren Daten separate Datensegmente
- Einbindung von Shared Librarys (Dynamic Link – DLL Windows / Shared Library - Shared Libs Linux)
- Gemeinsam genutzte Speichersegmente befinden sich in mehreren Seitentabellen unterschiedlicher Prozesse
- notwendiges weiteres Steuerbit zur Erkennung, sofern Prozess Speicher freigibt

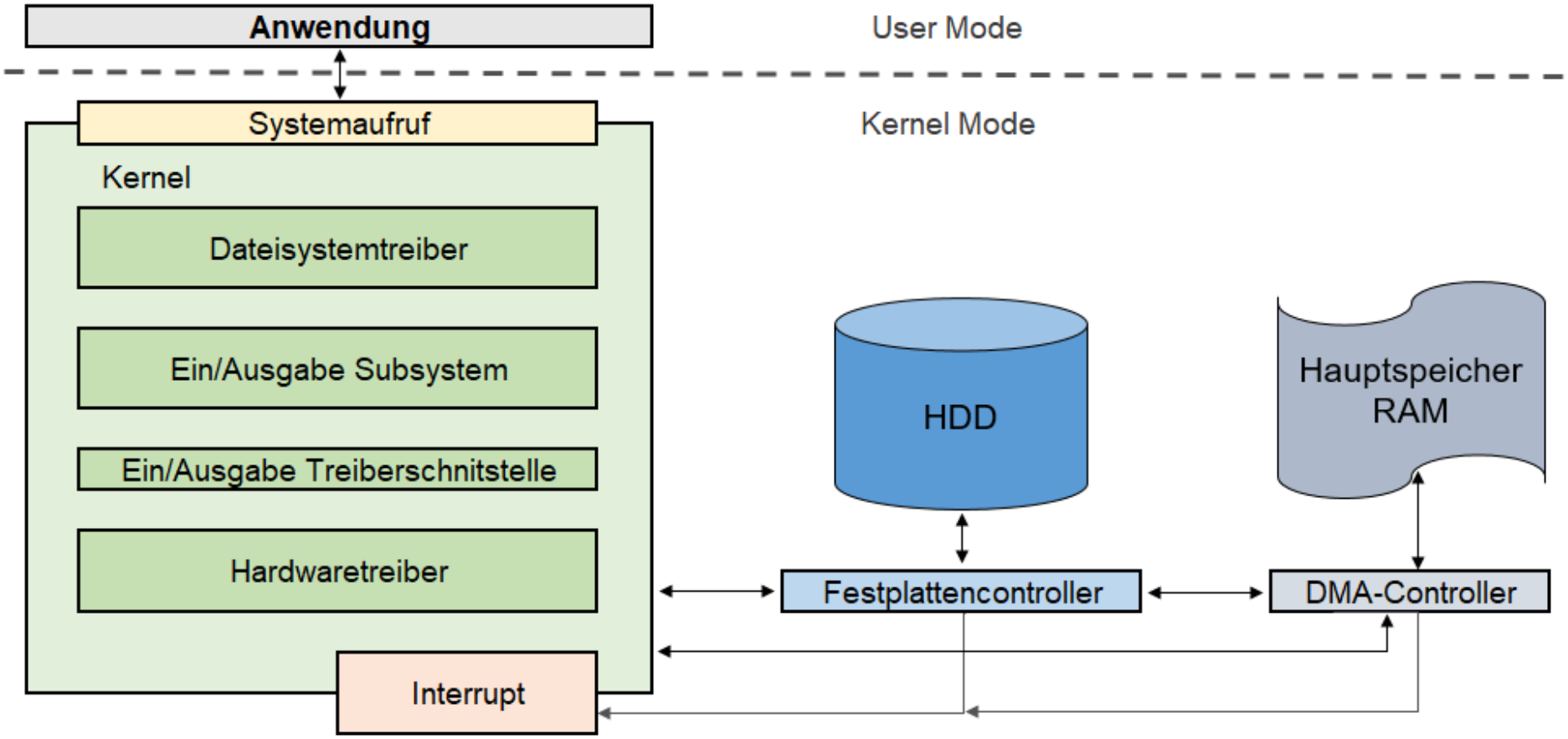
Direct Memory Access (DMA)

Für eine effiziente Ein- und Auslagerung von Daten aus dem Fest- in den Hauptspeicher wurden hardwareseitig Implementationen entwickelt, die einen direkten Zugriff auf den Hauptspeicher über das Bussystem ermöglichen (Direct Memory Access – DMA):

- der Prozessor und damit indirekt auch Programme können nicht direkt auf die Festplatte zugreifen
- durch Programmierung des DMA-Controller können die betreffenden Datenblöcke in vorher festgelegte Bereiche des Hauptspeichers kopiert werden
- während der DMA-Controller die Daten von der sehr langsamen Festplatte in den Hauptspeicher kopiert, kann die CPU bereits weitere Prozesse abarbeiten

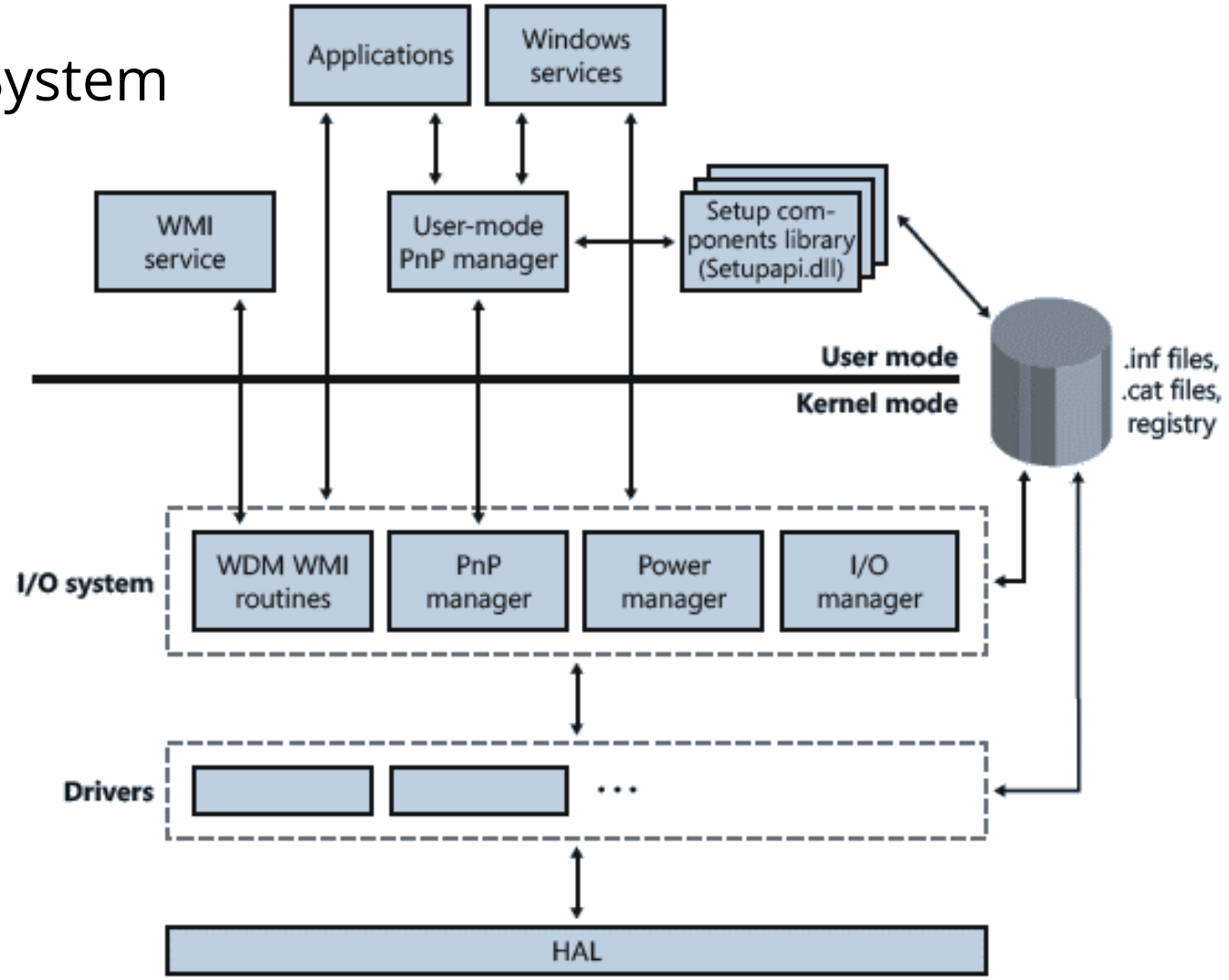
E/A Zugriffe auf die Festplatte

Zuordnung der Festspeicher E/A Geräte



E/A Zugriffe auf die Festplatte

Windows E/A System



Quelle: M. Valli

Zusammenfassung

ZUSAMMENFASSUNG

Die Einlagerung von Prozessen in den Hauptspeicher erfolgt in verschiedenen Betriebssystemkonfigurationen auf unterschiedlich Art und Weise.

Dabei spielte es in der Vergangenheit immer eine besondere Rolle den gering zur Verfügung stehenden flüchtigen Speicher effizient auszunutzen.

Dazu sind Techniken notwendig, um Prozesse bzw. Prozessbestandteile zu lokalisieren. Auch die Implementation von Speicherschutzmechanismen spielte hierbei eine entscheidende Rolle.

Die Einführung von virtuellen Speicherkonzepten ermöglichte es Prozessen mehr Speicher zu Verfügung zu stellen, als physikalisch vorhanden ist. Dabei müssen Methoden zur Verwaltung von Prozessbestandteilen und deren teilweise Ein- und Auslagerung eingesetzt werden, wie die Segmentierung oder das Paging.

Vielen Dank

Prof. Ronny Bodach

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

T +49 (0) 3727 58-1011
F +49 (0) 3727 58-21011
bodach@hs-mittweida.de
www.cb.hs-mittweida.de

Haus 8 | Richard-Stücklen Bau| Raum 8-205
Am Schwanenteich 6b| 09648 Mittweida

Felix Fischer

Hochschule Mittweida | University of Applied Sciences
Technikumplatz 17 | 09648 Mittweida
Fakultät Angewandte Computer- und Biowissenschaften

fische11@hs-mittweida.de
www.cb.hs-mittweida.de

HOCHSCHULE MITTWEIDA
University of Applied Sciences

hs-mittweida.de