Betriebssysteme

Speicherverwaltung

Von

Prof. Dr. Franz-Karl Schmatzer

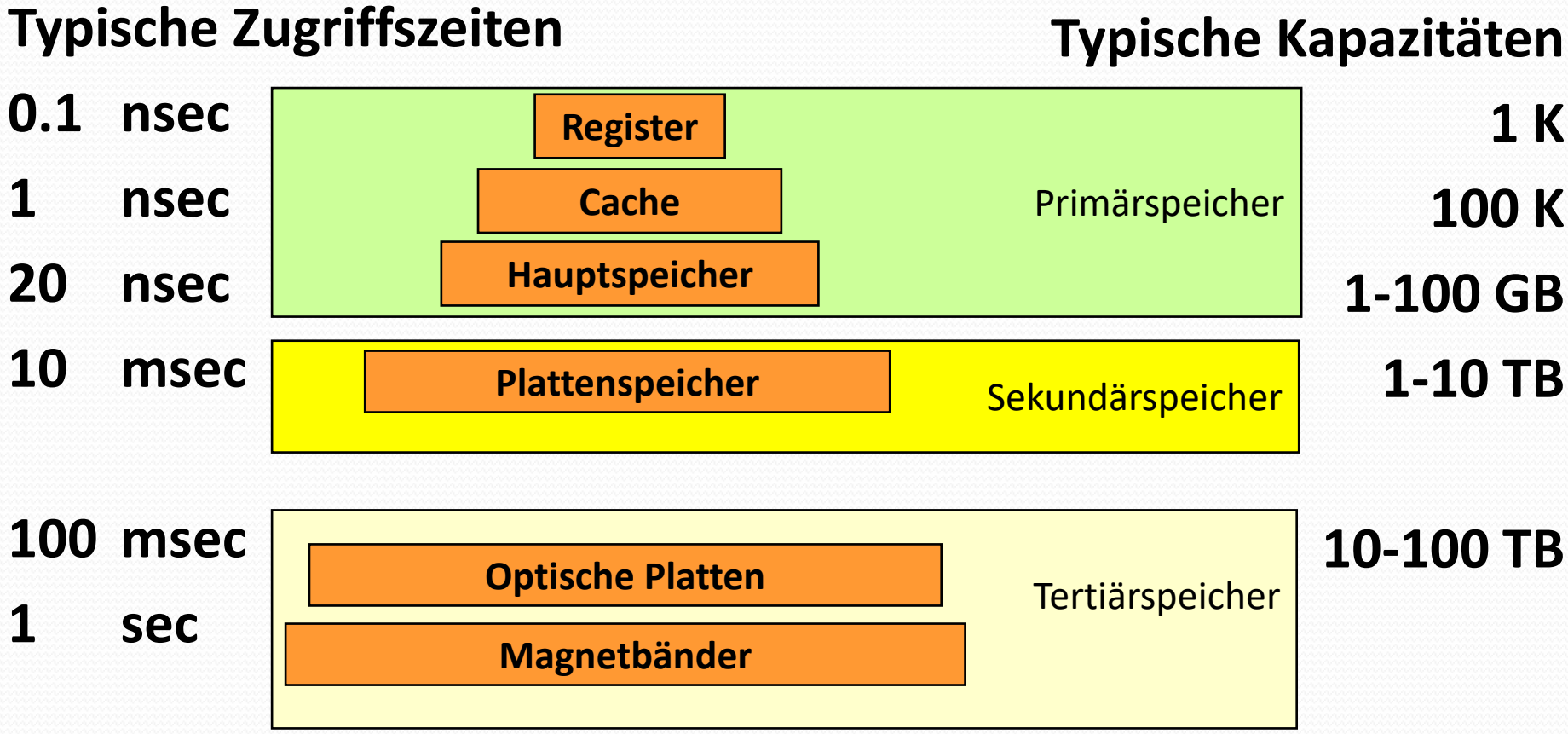
Literatur Verzeichnis

- Baun, Christian; Betriebssysteme kompakt; Springer Vieweg 2017
- Mandl, Peter; Grundkurs Betriebssysteme; 3.Aufl. 2013; Springer Verlag
- Tanenbaum, Andrew; Moderne Betriebssysteme; 3.Aufl. 2009; Pearson Studium
- Silberschatz et al.; Operating System Concepts; 7.ed; John Wiley 2005
- Siegert, H.J., Baumgarten U.; Betriebssysteme; 5.Aufl. 2001; Oldenbourg Verlag
- M.Russinovich, D.A.Solomon,A.Iomescu; Windows Internal Part 1; 7 Auflage, Microsoft Press 2017

Gliederung

- Speicherhierarchien
- Was macht eine Speicherverwaltung
- Lokalitätsprinzip
- Adressraumbelegung
- Mechanismen der Speicherverwaltung
- Freispeicherverwaltung
- Virtueller Speicher
- Virtuelle Adressierung

Speicher Hierarchie



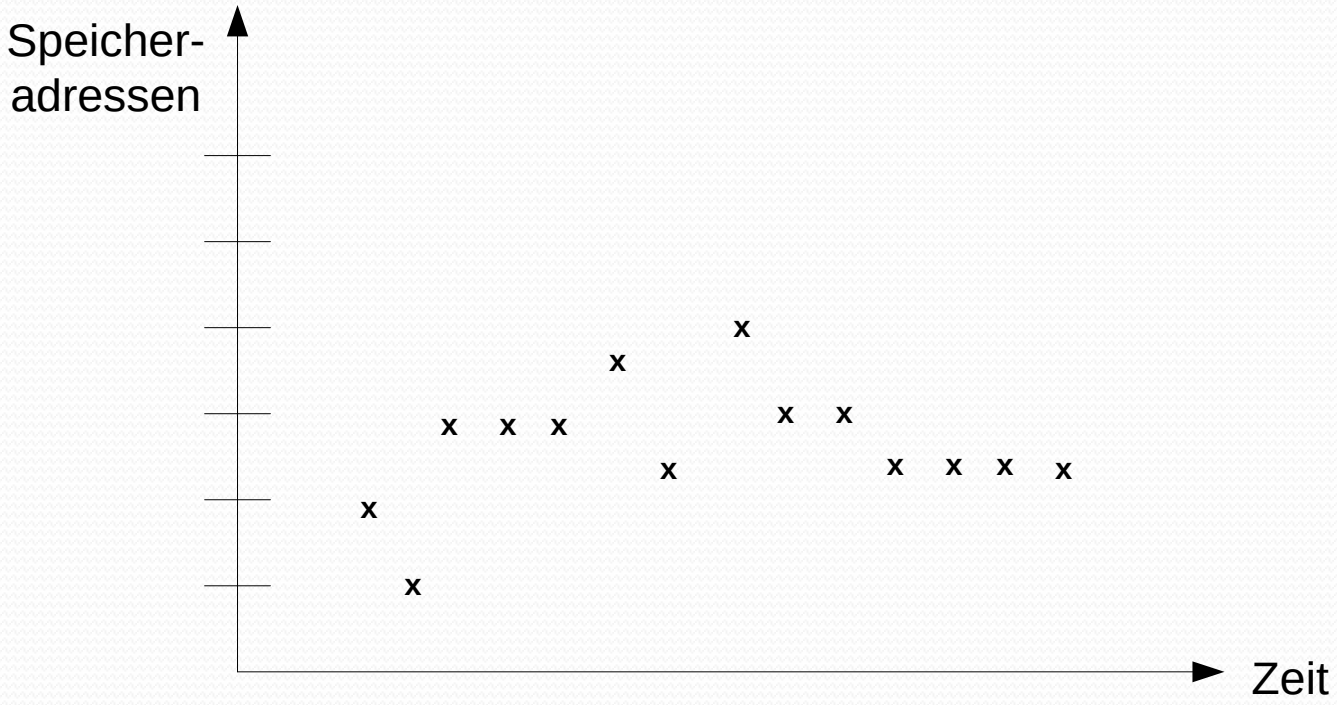
Was macht eine Speicherverwaltung?

- Versorgung der Prozesse mit dem Betriebsmittel „Arbeitsspeicher“ (Hauptspeicher)
- Verantwortliche Softwarekomponente ist der **Memory Manager (Speicherverwalter)**
- Der **Memory Manager** verwaltet die freien und belegten Speicherbereiche
- Dazu wird der Speicher entsprechend organisiert und adressiert.
- Heute geht man von dem Lokalitätsprinzip aus.
- Man unterscheidet zeitlich und örtliche Lokalität.

Lokalitätsprinzip

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

→ Diese sollten für den nächsten Zugriff bereitgehalten werden

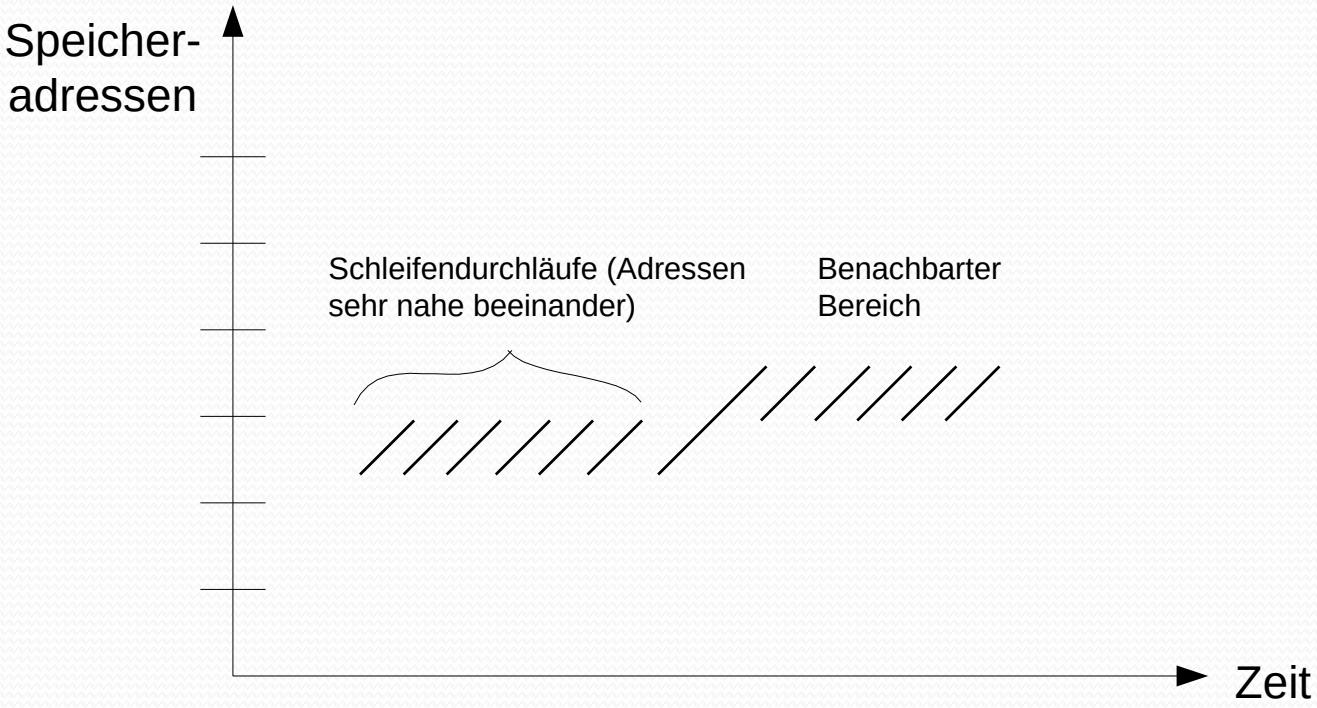


@Mandl

Lokalitätsprinzip

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

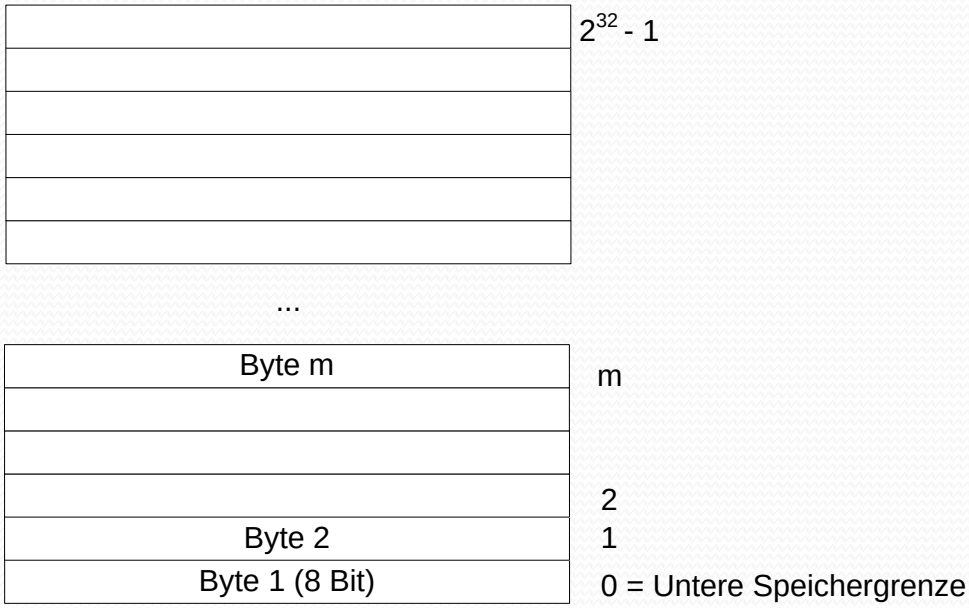
→ Benachbarte Daten beim Zugriff auch gleich in schnelleren Speicher laden



@Mandl

Adressen und Adressräume

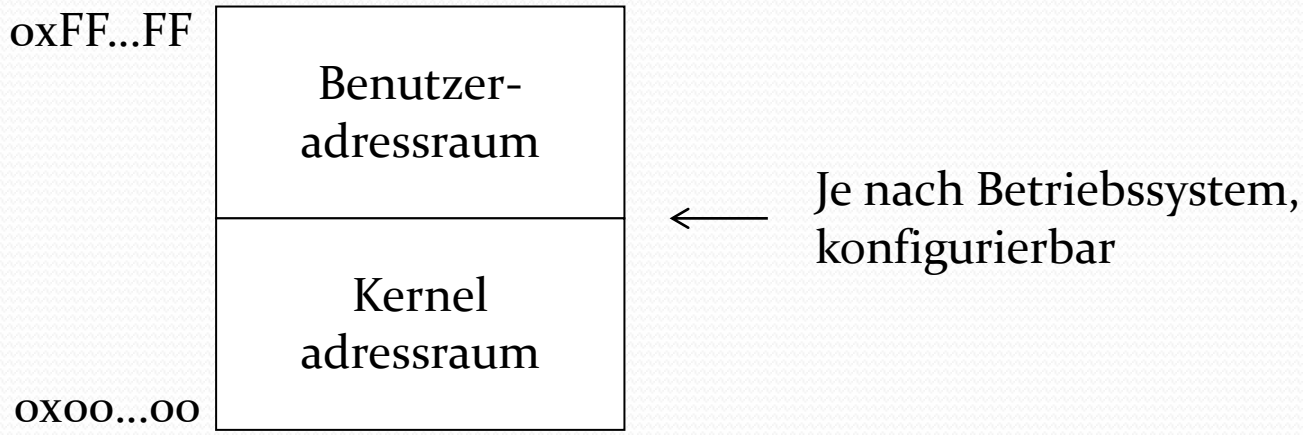
- Hauptspeicher ist in logisch adressierbare **Speicherstellen** unterteilt, meist byteweise (8 Bit)
- Ein Byte ist also die kleinste adressierbare Einheit
- 32-Bit-Adressen → 2^{32} adressierbare Bytes
- Ein **Adressraum** ist die Menge aller adressierbaren Adressen
  - 32-Bit-Adressen → {0, 1, 2, …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



@Mandl

Adressraumbelegung

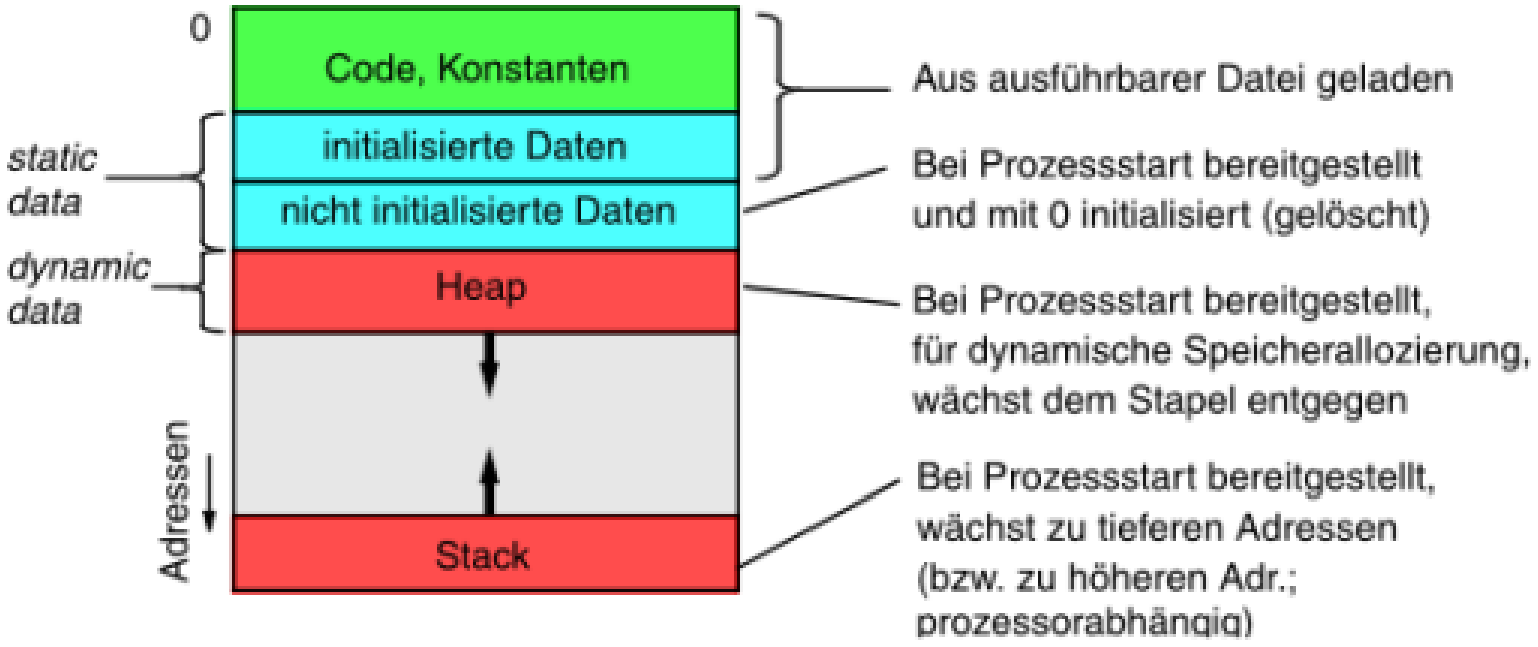
- Wird durch Adressraumbelegungsplan bestimmt
- Festlegung im Betriebssystem
- Ausrichtung auf Maschinenwörter wichtig wegen optimalem Zugriff



@Mandl

Adressraumbelegung

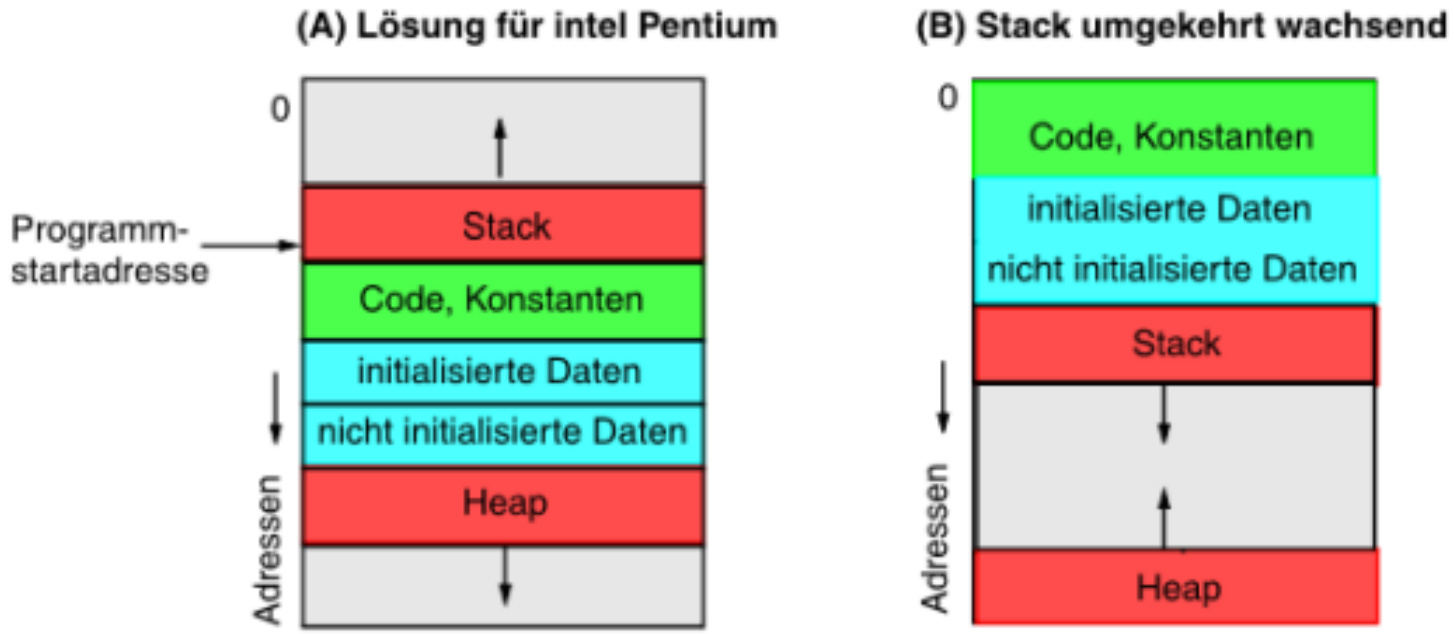
- Typische Adressraumbelegung von Prozessen
- Man hat eine Aufteilung in:
  - Code und Konstanten
  - Statische und dynamische Daten (Heap) und den Stack



@/studentenportal.ch/dokumente/bsys1/1450/

Adressraumbelegung

- Adressbereich der Anwendungsprogramme und Anwendungsdaten organisiert der Compiler/Interpreter bzw. das Laufzeitsystem
- Abhängig von der Programmiersprache.
- Es gibt daher mehrere Varianten:



@/studentenportal.ch/dokumente/bsys1/1450/

Mechanismen der Speicherverwaltung

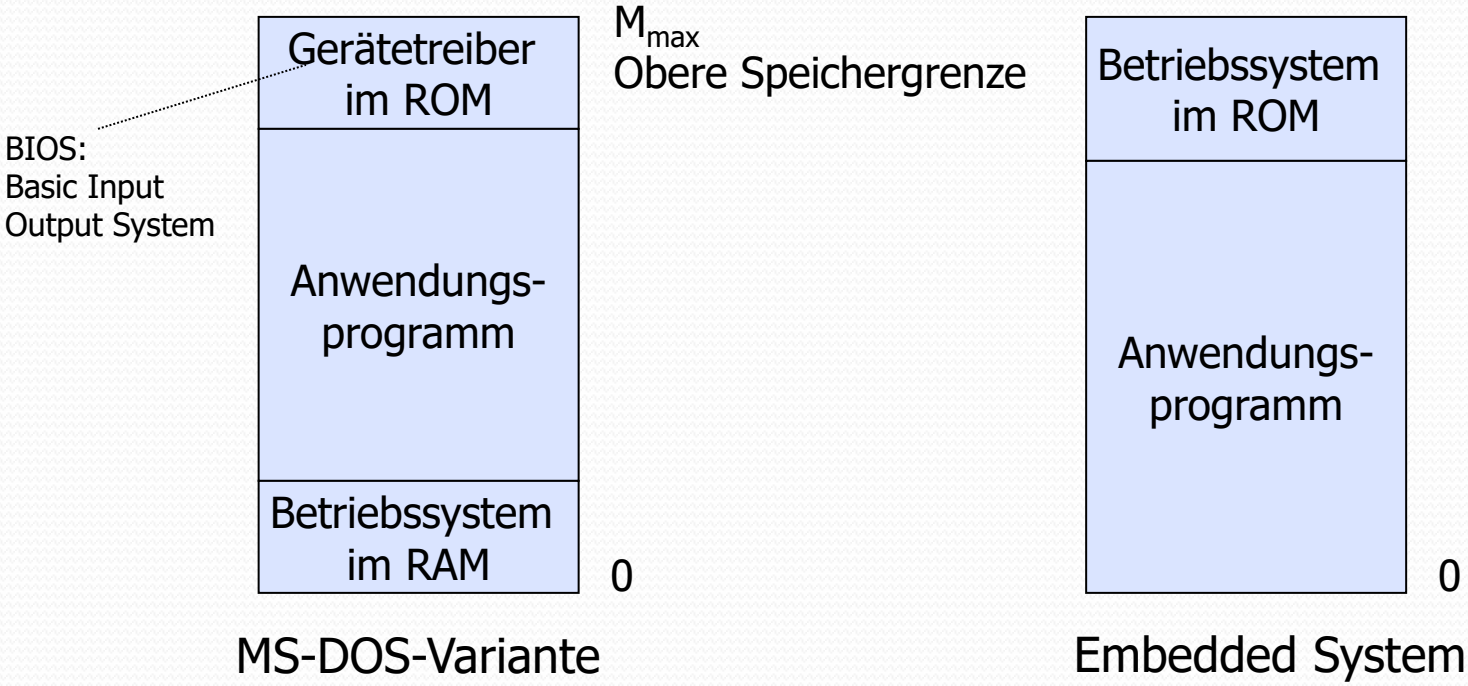
- Es gibt verschiedene Mechanismen für die Speicherverwaltung
- Historische Entwicklung:
  1. Speicherverwaltung bei Monoprogramming
  2. Speicherverwaltung mit festen Partitionen
  3. Overlay-Technik
  4. Swapping
  5. Virtueller Speicher

Aufgabe Speicherverwaltung

- Erläutern Sie die Speicherverwaltung
  - für das Monoprogramming
  - bei festen Partitionen
  - der Overlay-Technik
  - beim Swapping
- Geben Sie auch Beispiele für die Anwendungen an

Monoprogramming

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

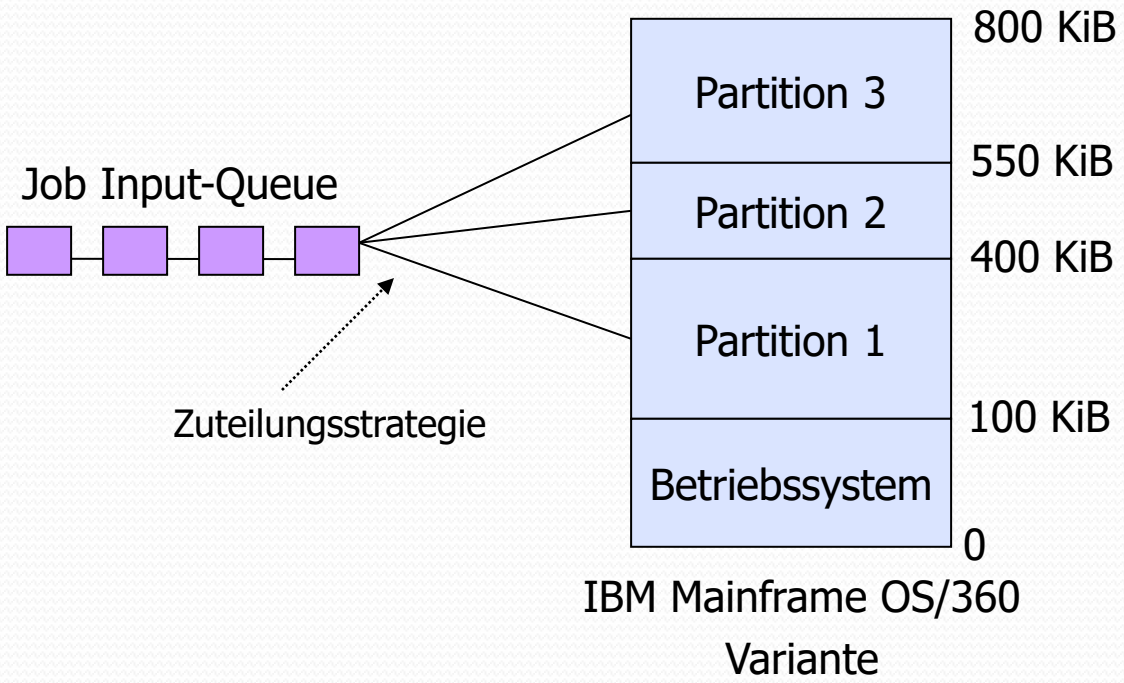


- **BIOS** = Programm zum Starten eines Rechnersystems, bis das Betriebssystem übernimmt. Es liegt in einem nicht flüchtigen ROM oder in einem Flashspeicher
- Weiterentwicklung von BIOS: **EFI** = Extensible Firmware Interface, unterstützt auch 64-Bit-Systeme

@Mandl

Feste 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

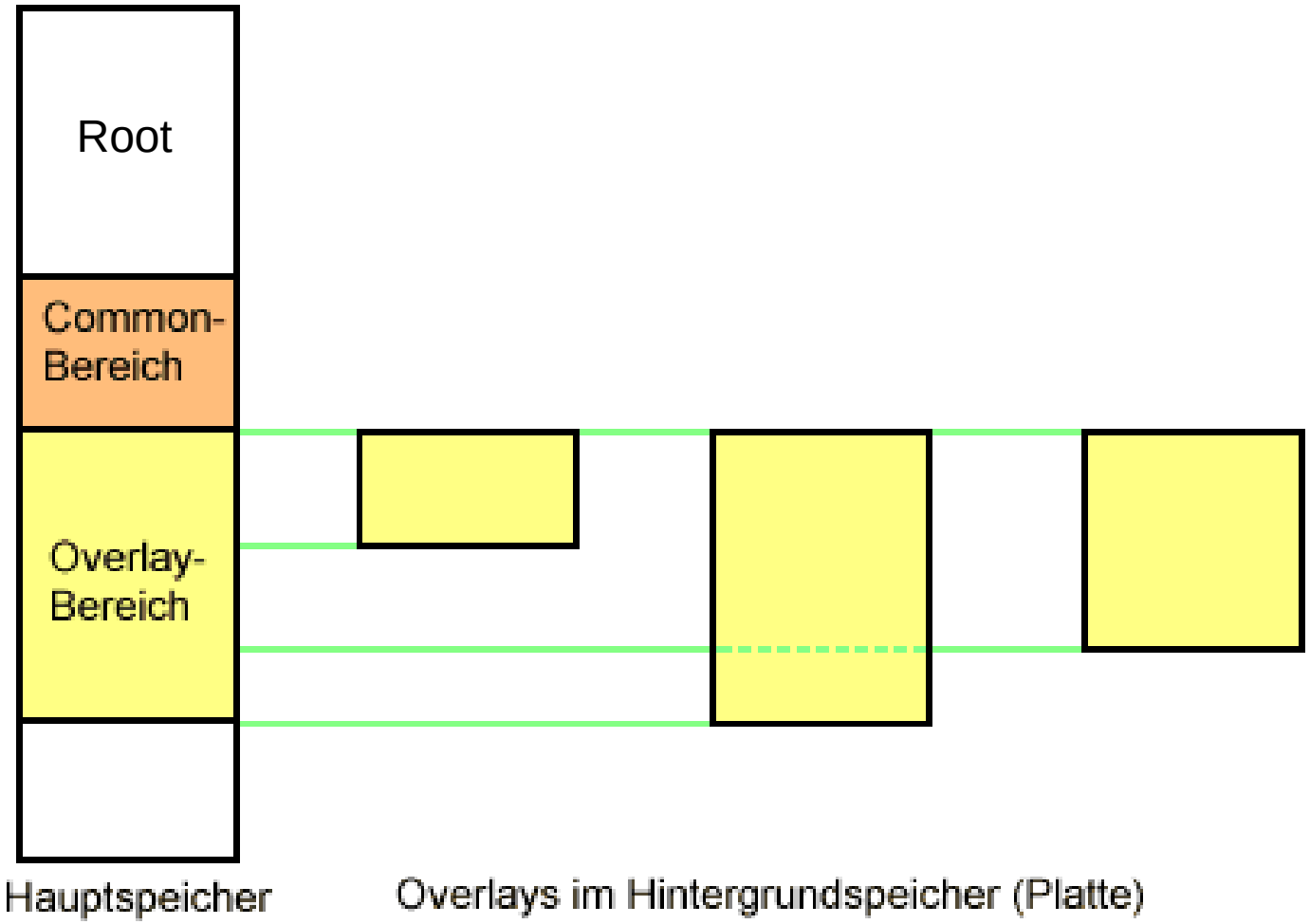


@Mandl

Overlay-Technik

- **Programme sind zu groß für den verfügbaren Arbeitsspeicher**
- Aufspaltung der Programme in Overlays, die auf der Festplatte gespeichert werden und dynamisch ein- und ausgelagert werden.
- Ausführung:
  - Zunächst wird Overlay 0 ausgeführt
  - Wenn Overlay 0 beendet ist, wird Overlay 1 ausgeführt, etc.
  - Problem: Programmierer müssen Programme in kleine, modulare Teile aufspalten
- Eine Overlay-Struktur besteht aus:
  - einem speicherresidenten Programmteil (root program),
  - aus mehreren Unterprogrammen, die sich gegenseitig überlagern, und
  - aus gemeinsamen Daten (Common), die von Aufruf zu Aufruf erhalten bleiben sollen

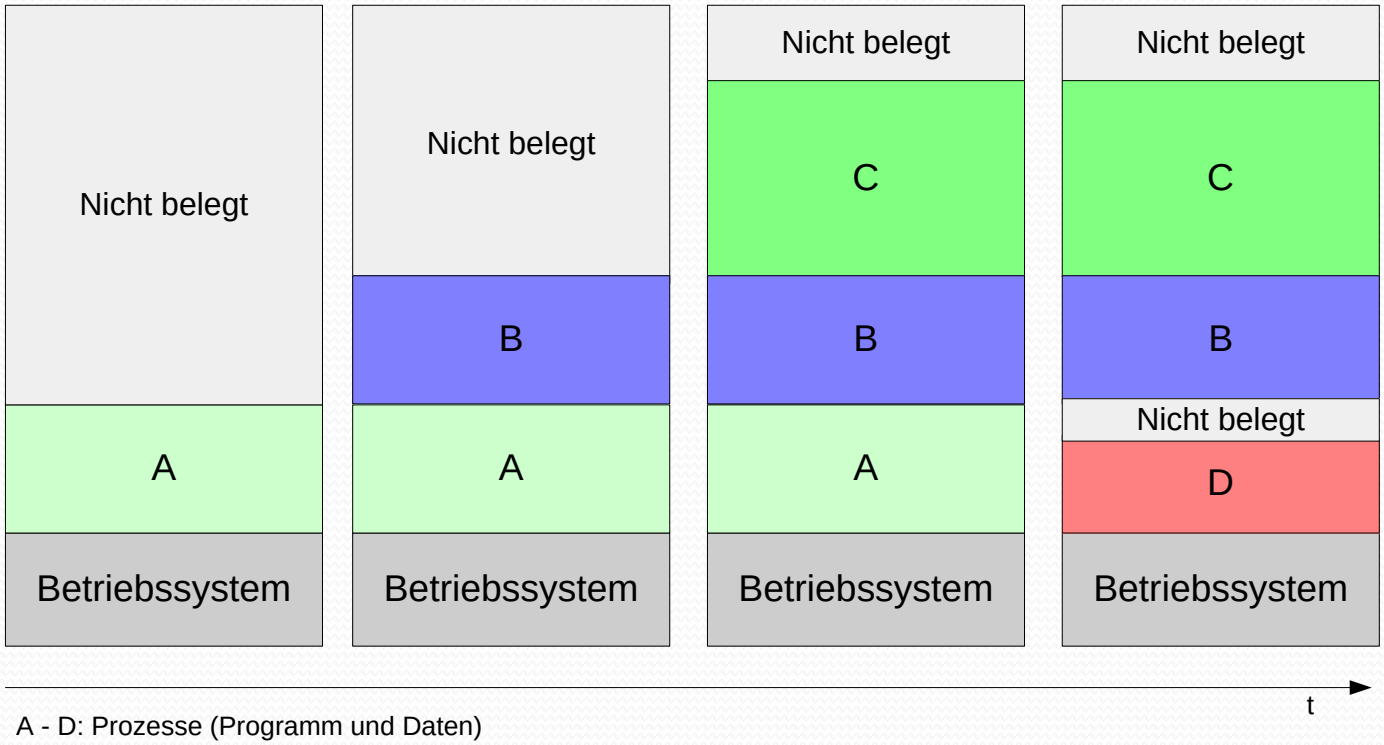
Overlay-Technik



Swapping I

- **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

Swapping II



A - D: Prozesse (Programm und Daten)

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

@Mandl

Die Freispeicherverwaltung

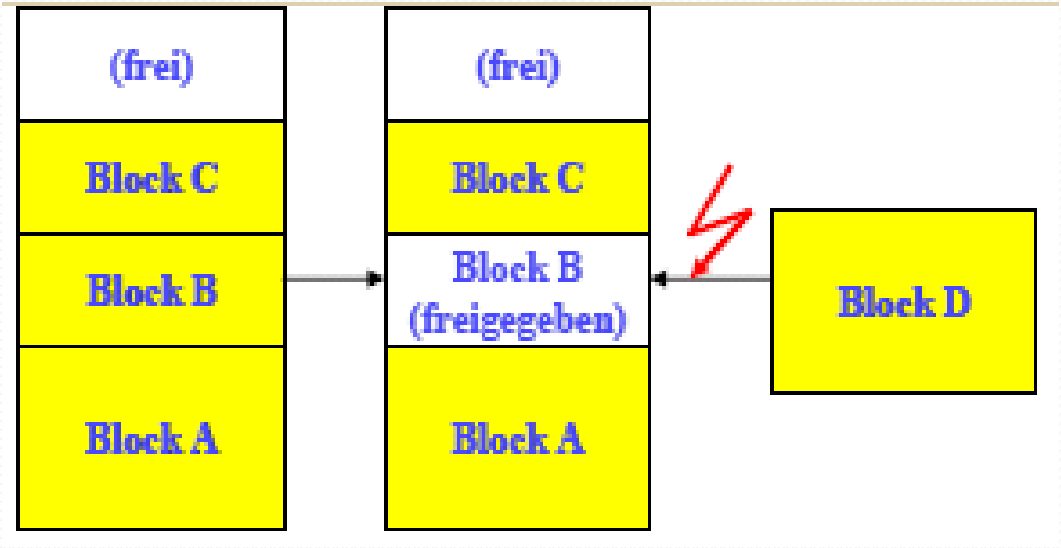
- Freispeicherverwaltung
  - Speicheranfrage (ob noch freier Speicher vorhanden ist)
  - Zuteilung eines Blocks gegebener Größe (Allokation)
  - Verlängerung eines bereits allozierten Blocks (ggf. mit Adresswechsel)
  - Rücknahme (mit Verschmelzung aneinandergrenzender Blöcke),
  - Kompaktifizierung (Lücken- oder Freispeichersammlung, garbage collection)

oft gekoppelt

Die Freispeicherverwaltung

- Verschnittproblem:

- interner Verschnitt: Speicherblöcke werden alloziert, die größer sind als gewünscht.
- externer Verschnitt: (Stückelung, Fragmentierung)



Die Freispeicherverwaltung

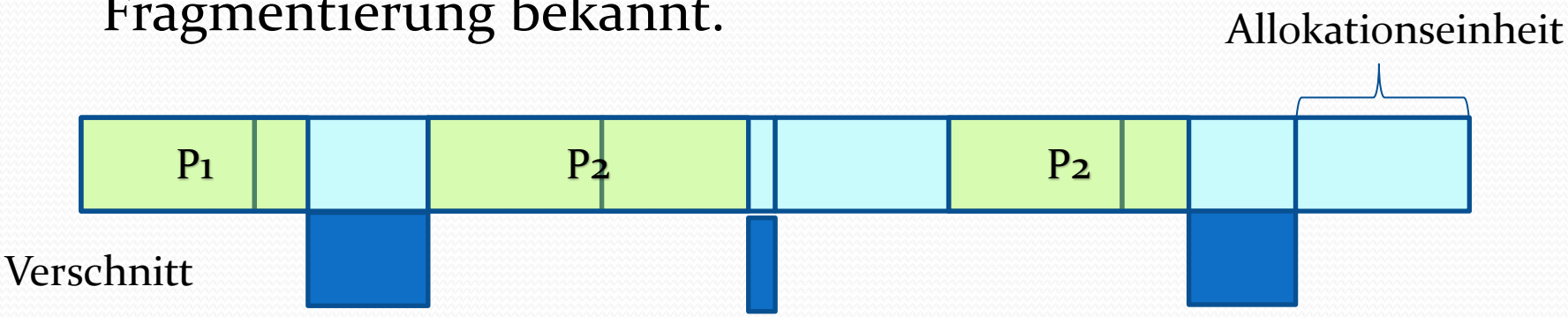
- Abhilfe gegen den externen Verschnitt Kachelung: Aufteilung des angeforderten Blocks in physikalisch nicht aneinandergrenzende Bereiche einheitlicher Größe (Pages, Kacheln von z.B. 512 byte oder 4 Kbyte). Dazu ist eine spezielle Form der virtuellen Adressierung nötig
- Lückensammlung: (garbage collection). Belegte Blöcke werden zusammengeschoben, so dass die Lücken größer werden bzw. eine einzige Lücke entsteht.
  - bei Bedarf (Erschöpfung des Vorrats) => Zeitaufwand?!
  - bei CPU-Leerlauf durch den Leerlaufprozess / nebenläufiger Kompaktifizierungsprozess

Verwaltung freier Speicherlücken

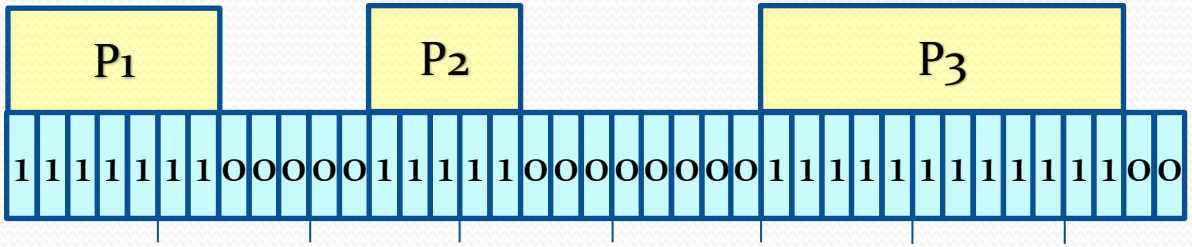
- Prozesse verändern während der Laufzeit ihre Größe
- Beim Einbringen von Prozessen oder wenn Prozesse dynamisch neuen Speicherplatz anfordern, muss das Betriebssystem den freien Speicherplatz verwalten.
- Es gibt 2 Verwaltungstechniken
  - Bitmaps
  - Verkettete Listen

Speicherverwaltung mit Bitmaps I

- Der Hauptspeicher wird in Allokationseinheiten unterteilt.
- Jede Einheit entspricht einem Bit in der Bitmap.
- Je kleiner die Allokationseinheiten sind, desto größer
- wird die Bitmap.
  - 0 = frei
  - 1 = belegt
- Große Allokationseinheit bedeutet Speicherverschwendung an den Rändern von Prozessen. Auch als interne Fragmentierung bekannt.



Speicherverwaltung mit Bitmaps II



- 11111110000011111100…
- Verkettete Liste (Prozess, Start, Länge)
- (P1,0,7)->(H,7,5)->(P2,12,5)->(H,17,8)->(P3,25,12)->…

Speicherbelegungstrategien

- Vergabestrategien:
  - **Sequentielle Suche**, erster geeigneter Bereich wird vergeben (First-Fit oder rotating First-Fit)
  - **Optimale Suche** nach dem passendsten Bereich, um Fragmentierung möglichst zu vermeiden (Best-Fit ,Worst-Fit)
  - **Zufällige Suche** (random-Fit)
  - **Buddy-Technik**: Schrittweise Halbierung des Speichers bei einer Hauptspeicheranforderung
    - Speichervergabe:
      - Suche nach kleinstem geeigneten Bereich
      - Halbierung des gefundenen Bereichs solange bis gewünschter Bereich gerade noch in einen Teilbereich passt
    - Bei Hauptspeicherfreigabe werden Rahmen wieder zusammengefasst:
      - Zurückgegebenen Bereich mit allen freien Nachbarbereichen (und deren Partnern) verbinden und zu einem Bereich machen

Prof. (emer.)
Donald E. Knuth
Stanford University

Die Freispeicherverwaltung

- **(rotating-) first-fit-Methode**

- Freiliste der Blöcke nach aufsteigenden Anfangsadressen geordnet
- Falls der geforderte Block x ≠ L dem Block der Freiliste ist, wird das Reststück auf der Freiliste belassen
- First fit: Suche beginnt jeweils beim vordersten Block, Rotating-first-fit arbeitet zirkulär

First Fit					
Start	200	250	400	450	700
300			100		
100	100				
150		100			
200				250	
500					200
300					

rotating First Fit					
Start	200	250	400	450	700
300			100		
100				350	
150					550
200	0				
500					50
300				150	

Die Freispeicherverwaltung

- **Best-fit**

- Freiliste nach steigenden Blocklängen geordnet, also $L_1 \leq L_2 \leq \ldots \leq L_M$
  Suche Index i, so daß $L_{i-1} < x <= L_i$
- Falls ein Restblock übrigbleibt, muss er neu in die Freiliste eingeordnet werden
- Problem: ?

- **Worst-fit**

- Suche unter den "passenden" Speicherbereichen, den größten
- Vorteil: ?

Die Freispeicherverwaltung

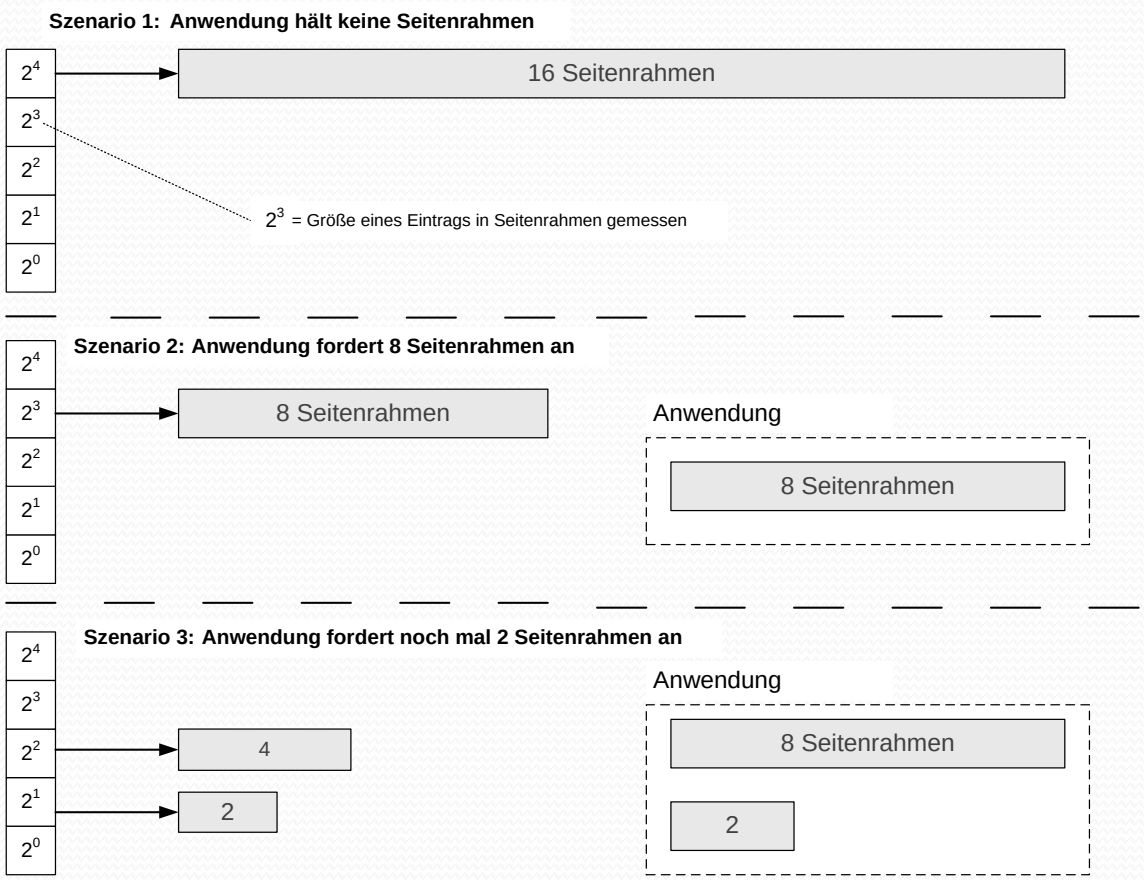
- **Best-fit**
  - Problem: best-fit erzeugt besonders viele nutzlos kleine Blöcke erzeugt
- **Worst-fit**
  - Vorteil: Verbleibende Reststücke können gut wieder genutzt werden

Best Fit					
Start	200	250	400	450	700
300			100		
100			0		
150	50				
200		50			
500					200
300				150	
Summe:	50	50	100	150	200

Worst Fit					
Start	200	250	400	450	700
300					400
100				350	
150			250		
200					200
500					
300					

Speicherbelegungstrategien:

Buddy-Technik



- Reduziert externe Fragmentierung auf Kosten einer verstärkten internen Fragmentierung!

Speicherbelegungstrategien:

Buddy-Technik

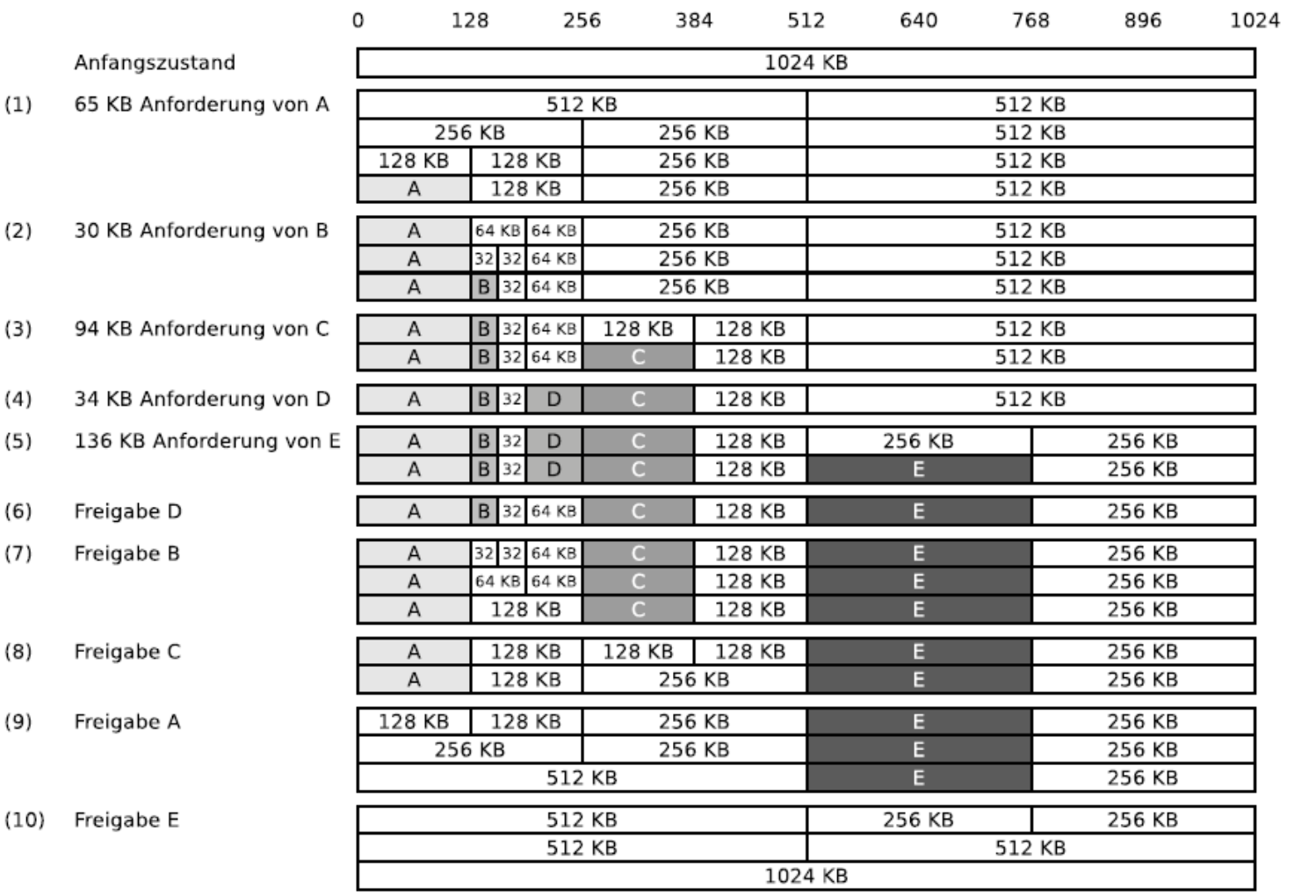


Abb. 5.4 Arbeitsweise der Buddy-Speicherverwaltung

@Baun

Aufgabe Freispeicherverwaltung

- Geben Sie für die folgende Blöcke 200,250,400,450, 600, die momentan in der Freispeicherliste stehen und den folgenden Anforderungen 350,150, 400, 300, 300, 150 die verbleibenden Blöcke an.
- Analysieren Sie die Blockzuteilung für die 4 Strategien:
  - first fit,
  - rotating first fit,
  - best fit und
  - worst fit

Lösung

First Fit

First Fit					
Start	200	250	400	450	600
350			50		
150	50				
400				50	
300					300
300					0
150		100			

Rotating First Fit

rotating First Fit						
Start	200	250	400	450	600	
350			50			
150				300		
400					200	
300				0		
300						Geht nicht
150						

Best Fit

Best Fit					
Start	200	250	400	450	600
350			50		
150	50				
400				50	
300					300
300					0
150		100			

Worst Fit

Worst Fit						
Start	200	250	400	450	600	
350					250	
150				300		
400			0			
300				0		
300						geht nicht
150						