



# Kapitel XIII

---

## Virtueller Speicher

- Bisher: Realer Speicher stellt physikalisch vorhandene Speicherkapazität für Nutzung durch Prozesse zur Verfügung
- In modernen Multiprogrammiersystemen jedoch: Verwendung der virtuellen Speichertechnik
- Begründung für die Nutzung der virtuellen Speichertechnik besteht in der Realisierung u. a. folgender Anforderungen:
  - Prozesse sollen ablauffähig sein, wenn sie sich auch nur partiell im Hauptspeicher befinden
  - Der Speicherbedarf eines Programms soll größer sein dürfen als die physikalisch zur Verfügung stehende Hauptspeicher-Kapazität
  - Partitionierung des Hauptspeichers im Multiprogrammiersystem soll für Programmierer transparent sein, d. h. dass Programmierer mit einem kontinuierlichen Speicherbereich beginnend bei Adresse 0 arbeiten, unabhängig davon an welcher Adresse der Speicherbereich physikalisch beginnt

- Aufgabe des Betriebssystems besteht im Rahmen der virtuellen Speichertechnik darin, aktuell benutzte Speicherbereiche von Prozessen im Hauptspeicher zu halten
- Nicht verwendete Speicherbereiche werden auf Sekundärspeicher ausgelagert
- Wichtige Annahme in diesem Zusammenhang:  
Auch verhältnismäßig große Programme benötigen nicht zu einem Zeitpunkt komplette Hauptspeicher-Kapazität auf einmal, sondern halten sich zur Laufzeit mit hoher Lokalität in gleichen Code- und Datenbereichen auf  
→ Genau dann kommen die Vorteile des virtuellen Speichers zur Geltung

- Mit Hilfe des virtuellen Speichers wird gegenüber einem Prozess die Illusion erzeugt, gesamten Hauptspeicher für sich alleine zu haben
- Zur Realisierung des virtuellen Speichers sind mehrere Komponenten relevant, auf die im Folgenden näher eingegangen wird

- Virtueller Adressraum:
  - Definiert den Speicherbereich, den ein Prozess zur Verfügung hat
  - Wird als „virtuell“ bezeichnet, da nicht physikalisch existent, sondern nur virtuell für den Prozess sichtbar
  - Sogenannter Memory-Manager bzw. Memory Management Unit (MMU) sorgt während der Laufzeit eines Prozesses für Abbildung des virtuellen Speichers auf realen Speicher
  - Im Rahmen der Abbildung von virtuellem Speicher auf realen Speicher:  
Betriebssystem muss notwendige Strategien implementieren, worauf im Folgenden eingegangen wird

- Strategien:

- Dienen der Verwaltung und dem optimalen Einsatz der virtuellen Speichertechnik
- Dazu Verwendung unterschiedlicher Strategien:
  - **Abrufstrategie (Fetch Policy):**  
Regelt Zeitpunkt des Einlesens von Speicherbereichen in den Hauptspeicher
  - **Speicherzuteilungsstrategie (Placement Policy):**  
Ermittelt Platzierung neu eingelesener Speicherbereiche im Hauptspeicher
  - **Austauschstrategie / Seitenersetzungs-Strategie:**  
Entfernt Speicherbereiche aus dem Hauptspeicher, sofern eine Verdrängung notwendig ist
  - **Aufräumstrategie (Cleaning Policy):**  
Sorgt dafür, dass im besten Fall zu jedem Zeitpunkt etwas Platz im Hauptspeicher vorhanden ist

- Seiten und Seitenrahmen:

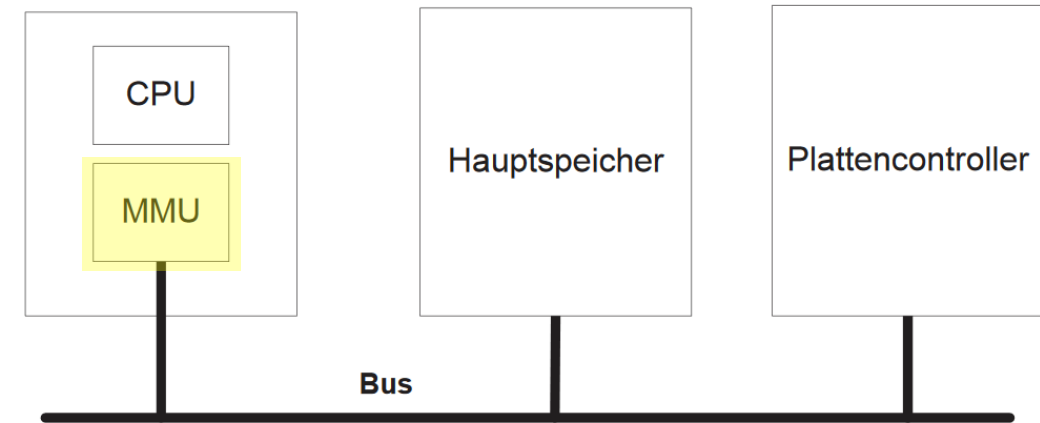
- Zusammenfassung der Speicherbereiche des virtuellen oder realen Adressraums zu Blöcken
- Blockgröße der Speicherbereiche von verwendeter Hardware abhängig
- Blöcke des virtuellen Adressraumes bezeichnet man als Seiten bzw. Pages
- Blöcke des realen Adressraumes bezeichnet man als Seitenrahmen bzw. Page Frames oder Frames
- Ausgewogene Größe der Seiten und Seitenrahmen wichtig vor folgendem Hintergrund:
  - Zu kleine Seitenrahmen erfordern mehr Ein- und Auslageroperationen
  - Zu große Seitenrahmen verschwenden Speicherplatz
- I. d R. sind Größen von 1, 4, 8, 16, 64 KiB vorzufinden

- Paging Area:
  - Auch bekannt als sogenannter Schattenspeicher
  - Der Speicherbereich innerhalb des Sekundärspeichers, der für ausgelagerte Seiten verwendet wird



## ■ Memory Management Unit (MMU):

- Heutzutage meistens Funktionseinheit des Prozessors und damit hardwaretechnisch umgesetzt
- Aufgabe der MMU besteht in der Umsetzung virtueller Adressen auf physikalische bzw. reale Adressen
- CPU sendet dabei die virtuelle Adresse an die MMU, welche anhand eines Algorithmus physikalische Adresse errechnet und über den Adressbus an den Hauptspeicher weiterleitet
- Weitere Aufgabe der MMU besteht in der Verwaltung der Paging Area, deren Größe durch Anzahl der Prozesse und weiteren Faktoren bestimmt wird

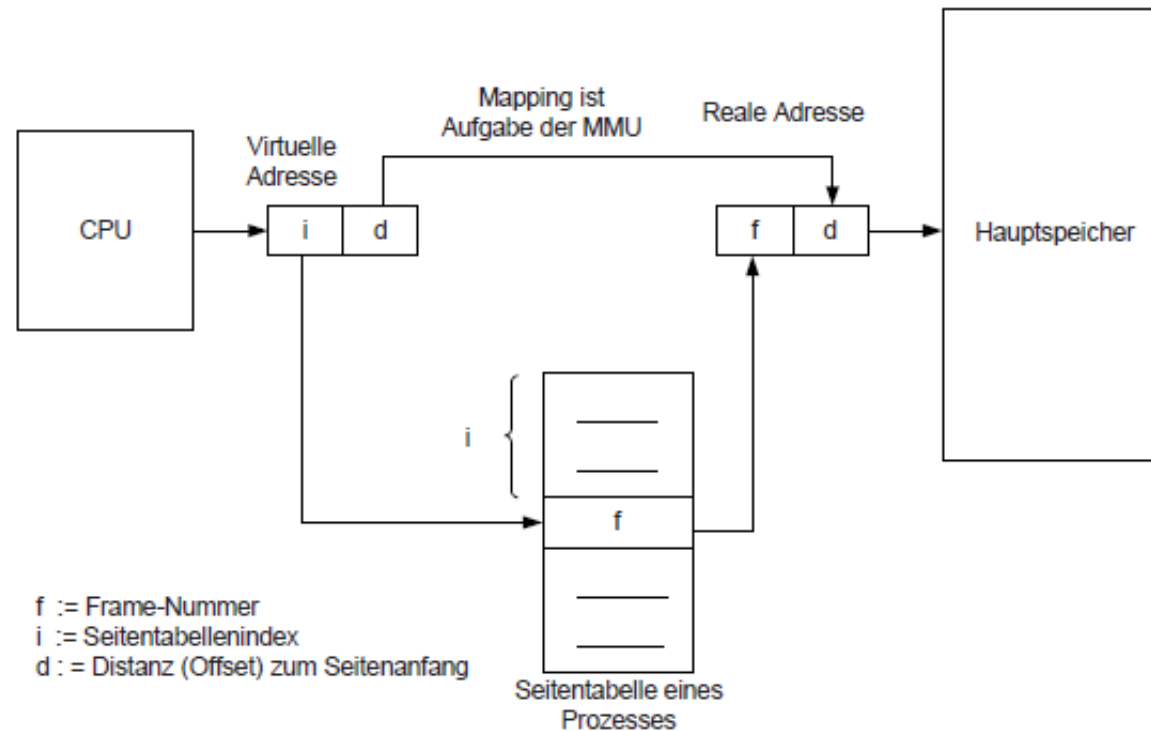


Quelle: [GB14]

- Seitentabellen (Page Tables):

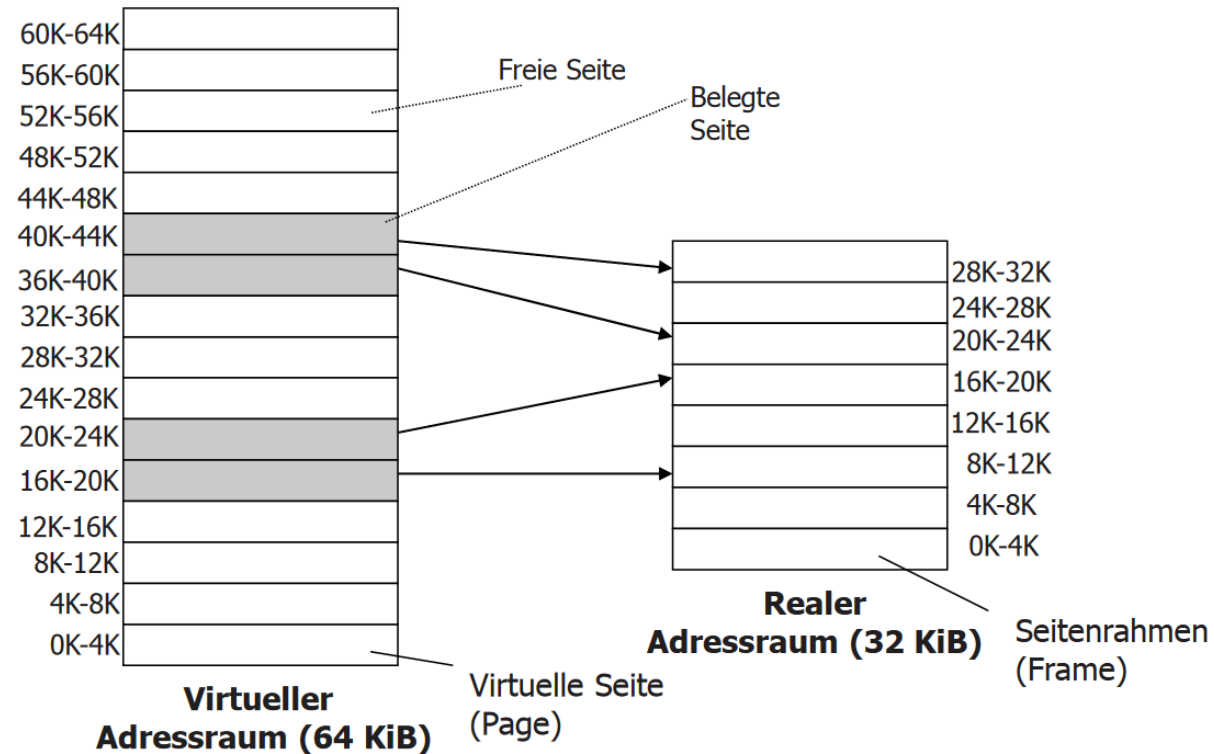
- Dienen der Verwaltung virtueller Adressräume in der MMU
- Enthalten Informationen darüber, wo Frames tatsächlich im Hauptspeicher vorzufinden sind
- Dazu: Zerlegung einer virtuellen Adresse in *Seitentabellenindex* und *Distanz*:
  - Seitentabellenindex gibt innerhalb der Seitentabelle den Index mit dem Verweis auf die reale Frame-Nummer an
  - Distanz gibt genaue Byteadresse innerhalb der Seite an
  - Veranschaulichung dazu auf nächster Folie
- Zuordnung: 1 Seitentabelle  $\Leftrightarrow$  1 Prozess

- Adressumsetzung bei der virtuellen Adressierung



Quelle: [GB14]

- Abbildung Seite auf Seitenrahmen

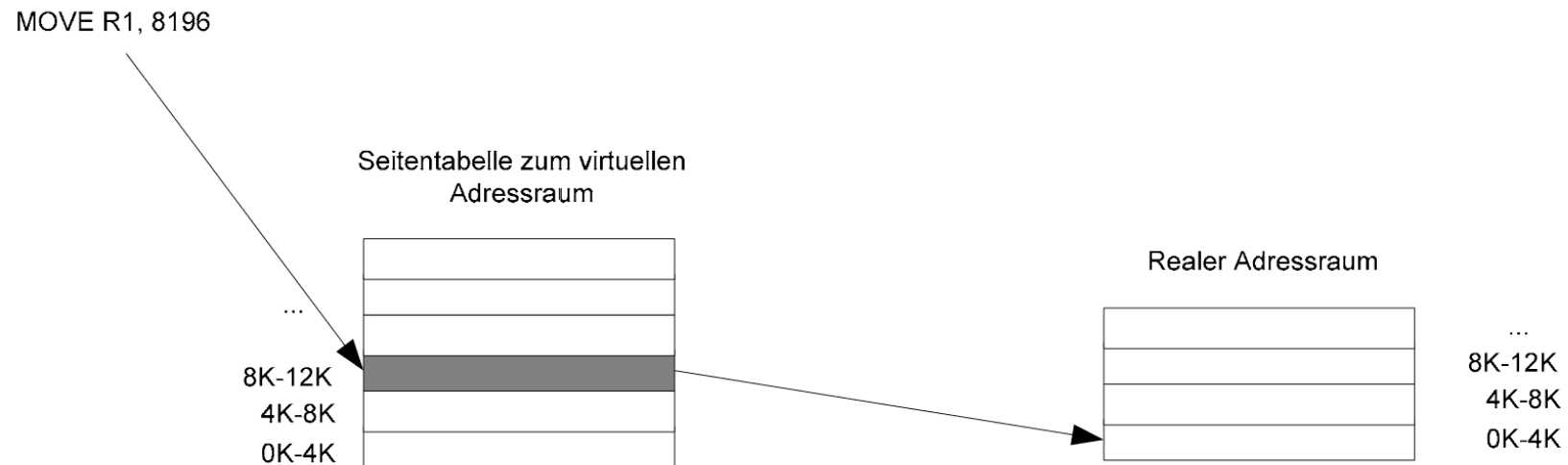


Quelle: [GB14]

- Zum tatsächlichen Zugriff auf Hauptspeicher-Inhalt:  
Memory-Manager setzt virtuelle Adresse in reale Adresse um über Funktion der Form:

$f(\text{virtuelle Adresse}) \rightarrow \text{reale Adresse}$

- Innerhalb der Funktion folgende Abbildung:

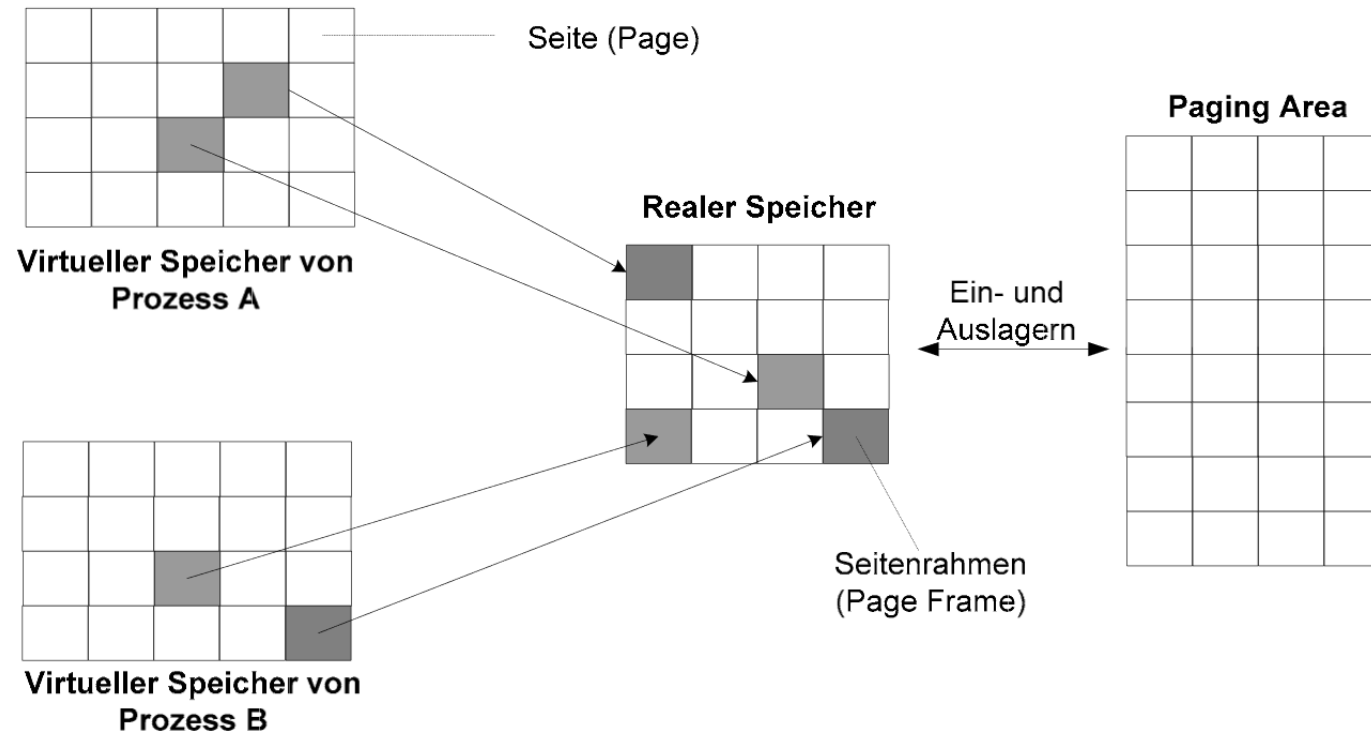


Adressumsetzung:  
Adresse 4 im 1. Frame wird angesprochen.

Quelle: [GB14]

- Zum Zeitpunkt der Befehlsausführung ist es notwendig, dass alle relevanten Seiten mit dem Programmcode und den zu verwendenden Daten im realen Speicher verfügbar sind
- Ist dies nicht der Fall, wird von der MMU sogenannter *Seitenfehler* bzw. *Page Fault* ausgelöst, welcher eine Unterbrechung des laufenden Prozesses nach sich zieht
- Ablauf bei einem Page Fault:
  1. Adresse, die den Page Fault ausgelöst hat, wird vermerkt und abgelegt (bei Intel-Prozessoren z. B. im Kontrollregister CR2)
  2. Betriebssystem geht in Kernel Mode über und führt Interruptroutine zur Bearbeitung eines Seitenfehlers aus, wodurch die Seite in einen Frame geladen wird. Dabei Berücksichtigung von:
    - Seitenersetzungsstrategie
    - Vergabestrategie
  3. Prozess erhält Prozessor in Abhängigkeit des CPU-Schedulings wieder zugeteilt

- Schematische Darstellung: Grundprinzip der virtuellen Speichertechnik



Quelle: [GB14]

- Seitenersetzung ist das Verfahren, welches eine Seite zur Verdrängung auswählt, sofern keine Seitenrahmen mehr zur Verfügung stehen
- Die zur Verdrängung ausgewählte Seite wird in die Paging Area ausgelagert, sodass Platz für eine neue Seite verfügbar wird
- Die Strategie zur Auswahl der zu verdrängenden Seite bezeichnet man als *Seitenersetzungsstrategie* bzw. *Replacement-Strategie*
- Optimalfall: alle zukünftigen Seitenzugriffe können bereits vorab bestimmt werden  
→ Dadurch rechtzeitige Seitenersetzung möglich, sodass Anzahl der Page Faults minimal ist
- Tatsächlich realisierbar: Sogenannte bedarfsgerechte Strategien, welche erst dann ausgeführt werden wenn Anforderung zur Seiteneinlagerung existiert, aber kein Seitenrahmen mehr verfügbar ist → Sogenanntes *Demand-Paging*



- Betrachtung des optimalen Algorithmus zur Seitenersetzung nach Belady (1996)
  - Grundgedanke:

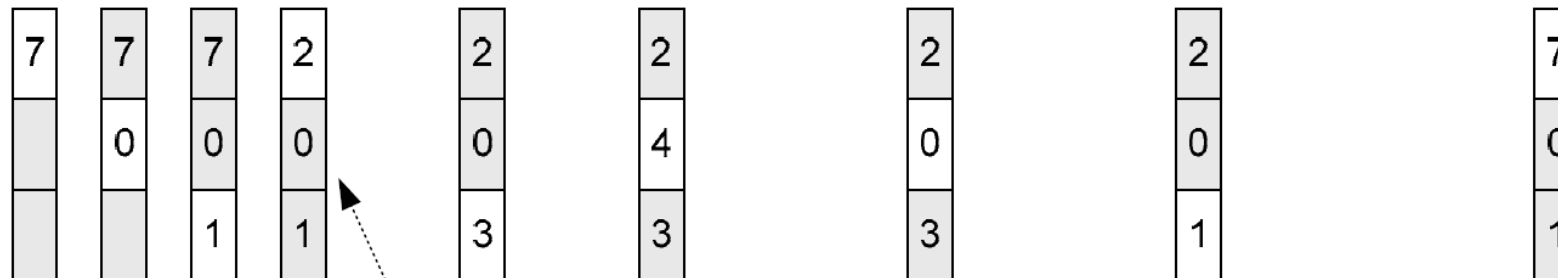
Optimal wäre, wenn zukünftige Seitenzugriffe aller Prozesse bereits im Vorfeld bekannt sind.  
Dadurch: Anzahl der Page Faults wird minimal gehalten.
  - Folglich:

Optimaler Algorithmus wählt die Seitenrahmen zur Ersetzung aus, die am spätesten von allen aktuell belegten Seitenrahmen wieder benötigt werden.

- Funktionsweise des optimalen Algorithmus zur Seitenersetzung:

Referenz-Reihenfolge

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1



3 Seitenrahmen

Seite 7 wird am spätesten  
wieder benötigt, daher Auslagerung

Quelle: [GB14]

- Bewertung des Algorithmus:
  - Sehr aufwändige bzw. unmögliche Realisierung, da es nicht praktikabel ist das Verhalten eines Prozesses im Hinblick auf zukünftige Speicherzugriffe zuverlässig zu bestimmen
  - Verfahren funktioniert nur für deterministische Programme, deren Seitenanforderungen bereits von vornherein bekannt sind (vgl. hierzu auch: CPU-Scheduling: Shortest Job First Algorithmus)
- Daher: Entwicklung verschiedener Seitenersetzungsalgorithmen, die für verschiedene Anwendungen möglichst optimale Lösung darstellen
- Dazu zählen z. B. die folgenden Algorithmen: Not Recently Used (NRU), FIFO, Second Chance, Clock Page, Least Recently Used (LRU), Not Frequently Used (NFU)

## ▪ Not Recently Used (NRU) Algorithmus:

- Idee:

Seiten, die in letzter Zeit nicht genutzt wurden, sind Kandidaten für Verdrängung

- Funktionsweise:

Markierung der Seiten mit einem Referenced Bit (R-Bit) und einem Modified-Bit (M-Bit).

R- und M-Bit werden in Seitentabelleneinträgen der virtuellen Adressräume verwaltet.

Ein Eintrag hat folgende Gestalt:

...	R	M	...	Frame-Nummer
-----	---	---	-----	--------------

*Quelle: [GB14]*

- M-Bit wird gesetzt, wenn ein Seitenrahmen sich verändert
- R-Bit wird gesetzt, wenn lesender Zugriff auf Seitenrahmen erfolgt
- Setzen der Bits geschieht durch Hardware, Zurücksetzung mittels Software auf Kernel-Ebene
- In bestimmten Abständen: Löschen des R-Bit, sodass nur bei Seiten das R-Bit gesetzt ist, die in der letzten Zeit auch benutzt wurden
- Eine Seite kann demzufolge folgende Bit-Zustände aufweisen:
  - $R = 0, M = 0 \rightarrow$  wird als erstes ausgelagert
  - $R = 0, M = 1$
  - $R = 1, M = 0$
  - $R = 1, M = 1 \rightarrow$  wird als letztes ausgelagert

- Seitenersetzung erfolgt nach folgendem Prinzip:
  1. Sobald Seitenersetzung fällig, prüft der Memory Manager ob eine Seite existiert, die weder modifiziert noch referenziert wurde
    - Falls ja: Seite wird ausgewählt
    - Falls nein: gehe zu Schritt 2
  2. Prüfe, ob nicht referenzierte, aber bereits modifizierte Seite existiert
    - Falls ja: Seite wird ausgewählt
    - Falls nein: gehe zu Schritt 3
  3. Prüfe, ob referenzierte, aber nicht modifizierte Seite existiert
    - Falls ja: Seite wird ausgewählt
    - Falls nein: gehe zu Schritt 4
  4. Wähle eine Seite, die bereits referenziert und modifiziert wurde

- Eine zur Ersetzung ausgewählte Seite mit gesetztem M-Bit muss vor der Verdrängung in die Paging-Area zurückgeschrieben werden → sonst Datenverlust
- Seiten, die vor längerer Zeit verändert wurden, werden „schlechter“ behandelt als Seiten, die erst vor kurzem referenziert wurden
- Annahme dabei: Seiten, die erst vor kurzem referenziert wurden, werden mit hoher Wahrscheinlichkeit nochmals benötigt (vgl. dazu nochmals: Lokalitätseffekt)

- Auch als *Placement* bezeichnet
- Über Laufzeit eines Systems hinweg kann es passieren, dass freie Seitenrahmen über gesamten physikalischen Adressraum verteilt sind
- Problem dabei: Für neue Speicheranforderungen nahezu keine zusammenhängenden Speicherbereiche mehr vorhanden  
→ Sogenannter Fragmentierter Speicher
- Eine Speicherfragmentierung sollte aus Leistungsgründen vermieden werden.  
Deshalb: Verwendung von Speicherbelegungs- und Vergabestrategien
- Zunächst jedoch: Unterscheidung zwischen interner und externer Fragmentierung



- Interne vs. externe Fragmentierung:

- Interne Fragmentierung:

- Ein Prozess erhält einen größeren Speicherbereich zugewiesen, als dieser tatsächlich benötigt.

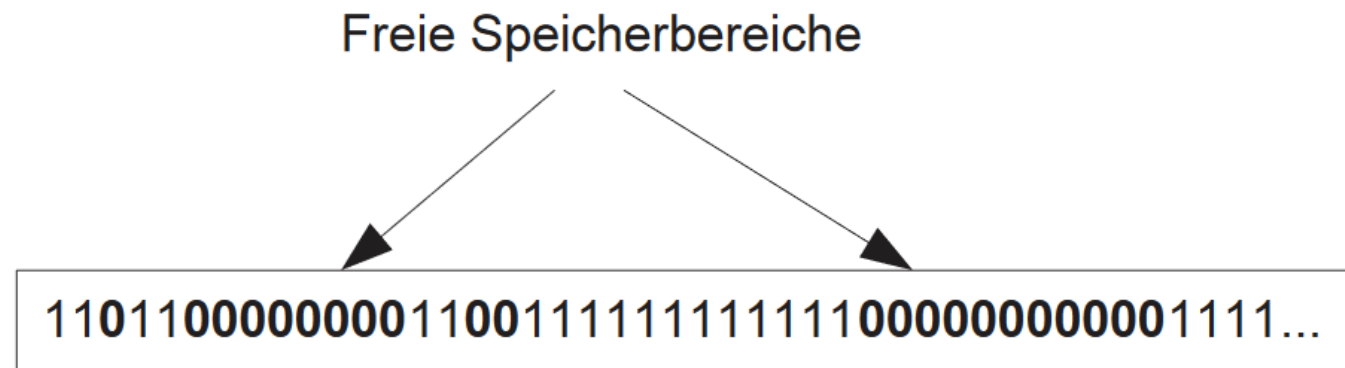
- Der zugewiesene Speicher bleibt in Teilen ungenutzt, wodurch sich einzelne Fragmente bilden.

- Externe Fragmentierung:

- Entsteht, wenn prozessübergreifend verfügbare Speicherbereiche immer kleiner werden.

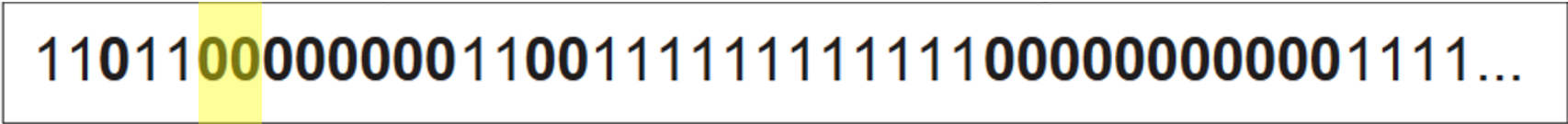
- Dadurch: Aufwändigere Speicherzuweisungen, da für eine Speicheranforderung mehrere einzelne Speicheranforderungen notwendig.

- Speicherbelegungsstrategien bzw. Placement Policies:
  - Werden verwendet, um freie Speicherbereiche schnell aufzufinden
  - I. d. R. Abbildung mit Hilfe einer einfachen Bit-Map, die der Memory Manager verwaltet
  - Dabei: Ist ein Bit auf 1 gesetzt, ist der Seitenrahmen belegt, ansonsten ist er frei
  - In diesem Zusammenhang: Aufeinanderfolgenden Nullen kennzeichnen freie Speicherbereiche



Quelle: [GB14]

- Speichervergabestrategien bzw. Placement Strategies:
  - Legen fest, welche Seitenrahmen für die Einlagerung neuer Seiten verwendet werden
  - Auf mögliche Strategien wird im Folgenden eingegangen
  
- Speichervergabestrategie: First Fit
  - Sequenzielle Suche nach dem erstbesten passenden Speicherbereich
  - Beispiel: Seitenrahmen 4 KiB, Suche nach 8 KiB Speicherbereich



1 1 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 ...

Quelle: [GB14]

## ▪ Speichervergabestrategie: Best Fit

- Ausschöpfende Suche nach dem passendsten Bereich zur Vermeidung von Fragmentierung
- Beispiel: Seitenrahmen 4 KiB, Suche nach 8 KiB Speicherbereich



A horizontal bar representing memory layout. It is divided into segments of varying sizes, each labeled with a binary string. From left to right, the segments are: '110110000000011' (16 bits), '00' (2 bits, highlighted in yellow), '111111111111' (12 bits), '000000000000' (12 bits), and '1111...' (4 bits). The yellow highlight indicates the best fit for an 8 KiB request.

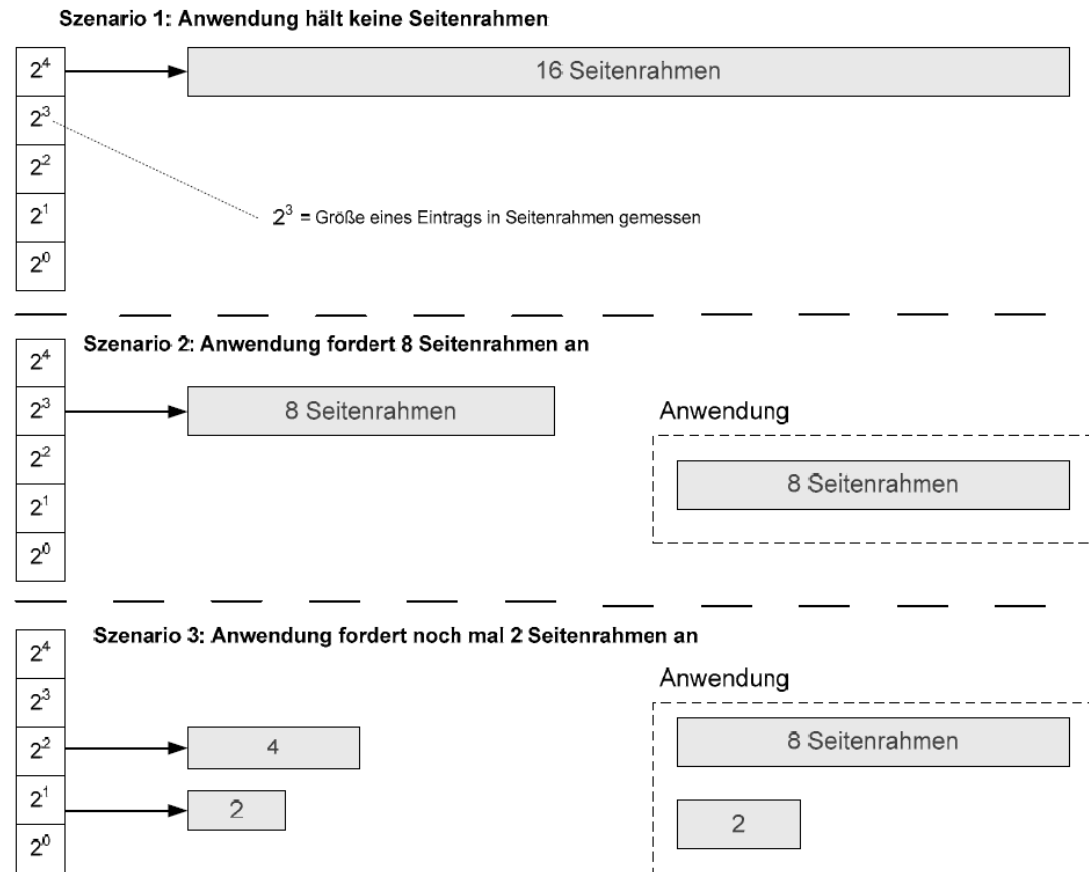
11011000000001100111111111111000000000001111...

## ▪ Speichervergabestrategie: Buddy-Technik bzw. Halbierungsverfahren

- Sieht schrittweise Halbierung des Speichers vor und sucht dabei nach dem kleinsten geeigneten Bereich
- Suche ist dann beendet, wenn Bereich gefunden wurde, in den die neuen Seiten bestmöglich hineinpassen
- Bei Speicherfreigabe: Seitenrahmen werden wieder zusammengefasst, sodass größere verfügbare Speicherbereiche entstehen

Quelle: [GB14]

- Schematische Darstellung: Buddy-Technik

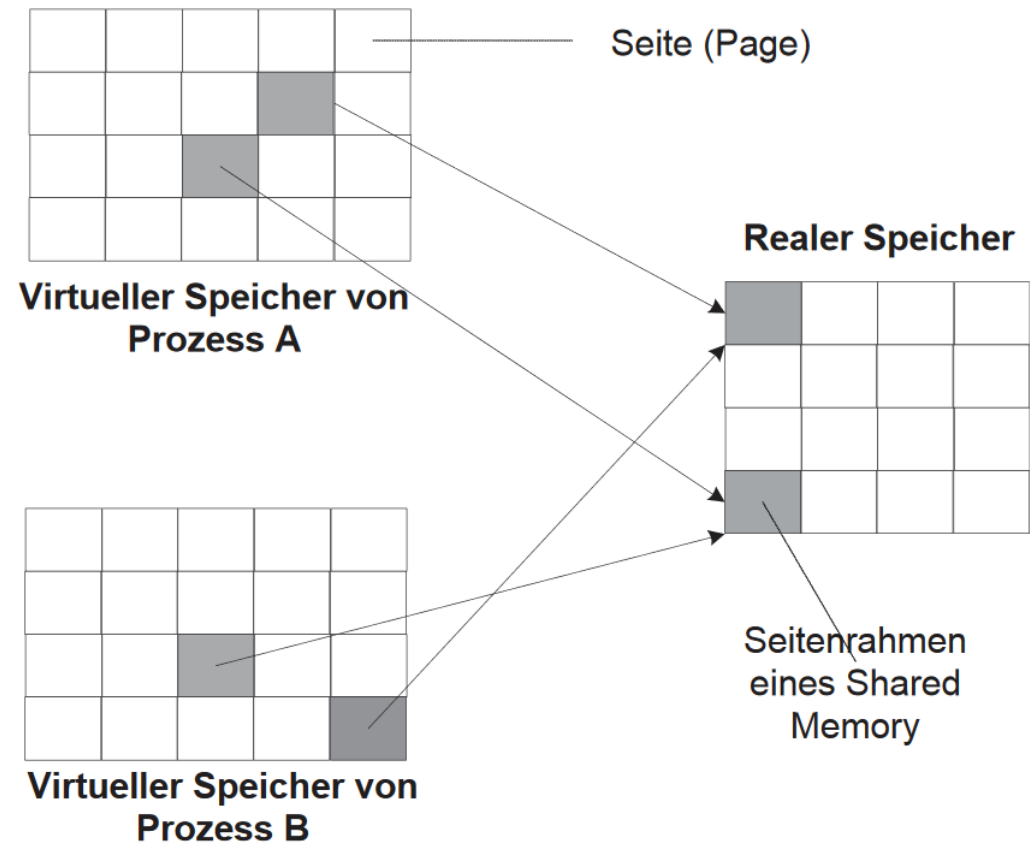


Quelle: [GB14]

- Auch als *Cleaning-Strategien* bezeichnet
- Realisieren „Sauberhaltung“ des Speichers, indem festgelegt wird zu welchem Zeitpunkt eine Seite auf die Paging-Area verschoben wird
- Dazu Unterscheidung von zwei Strategien:
  - Entladestrategie: Demand Cleaning
    - Entladen (bzw. Auslagern) einer Seite erfolgt nach Bedarf; immer dann, wenn belegter Seitenrahmen benötigt wird
  - Entladestrategie: Precleaning
    - Verfahren, bei dem veränderte Seiten präventiv zurückgeschrieben werden
    - Dadurch: Betroffene Seitenrahmen sofort für Seitenersetzung verfügbar, Zurückschreiben entfällt

- Wdh. aus Abschnitt Prozessverwaltung:  
Shared Memory kann zur Realisierung der Interprozesskommunikation eingesetzt werden
- Bisher nicht beantwortet:  
Wie wird Shared Memory abgebildet?
- Hierbei hilft virtuelle Speichertechnik, indem prozessübergreifend gemeinsam genutzte Speicherbereiche einmal in den Hauptspeicher geladen werden und die jeweiligen Seitentabelleneinträge der Prozesse darauf referenzieren

- Schematische Darstellung: Abbildung des Shared Memory



Quelle: [GB14]



- Aufgabe der Speicherverwaltung beim Shared Memory besteht lediglich in der Bereitstellung eines gemeinsamen Adressbereichs
- Daher: Synchronisation der Datenzugriffe ist umzusetzen, z. B. mit:
  - Semaphore
  - Monitor
- Gemeinsam genutzte Codeteile müssen wiedereintrittsfähig (auch: reentrant, threadsafe) sein:  
Ein Thread stört einen anderen, der denselben Code ausführt solange nicht, bis dieser fertig ist

- Alle Abbildungen, sofern nicht anders angegeben aus [MB17]

- [BS17]           Betriebssysteme – Grundlagen und Konzepte, Rüdiger Brause, 4. Auflage  
Springer Vieweg Verlag, 2017  
ISBN: 978-3-662-54099-2
  
- [GB14]           Grundkurs Betriebssysteme, Peter Mandl, 4. Auflage  
Springer Vieweg Verlag, 2014  
ISBN: 978-3-658-06217-0
  
- [BK17]           Betriebssysteme Kompakt, Christian Baun, 1. Auflage  
Springer Vieweg Verlag, 2017  
ISBN: 978-3-662-53142-6

- [MB17]            Moderne Betriebssysteme, Andrew S. Tanenbaum & Herbert Bos, 4. Auflage  
Pearson Studium, 2017  
ISBN: 978-3-86894-270-5
  
- [MS12]            Multicore-Software, Urs Gleim & Tobias Schüle  
dpunkt.verlag, 2012  
ISBN: 978-3-89864-758-8
  
- [BS15]            Betriebssysteme, Eduard Glatz, 3. Auflage  
dpunkt.verlag, 2015  
ISBN: 978-3-86490-222-2

- [TI11]            Technische Informatik, Günther Kemnitz, Band 2  
  
                         Springer Heidelberg Dordrecht London New York, 2011  
  
                         ISBN: 978-3-642-17446-9