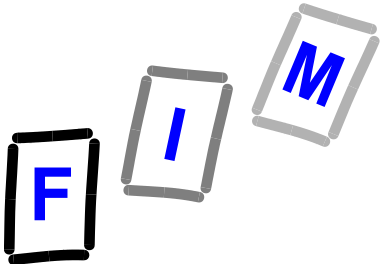
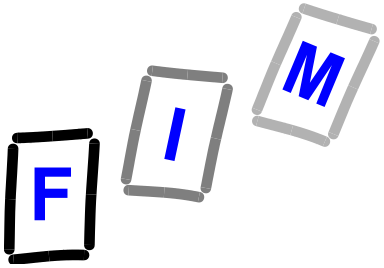


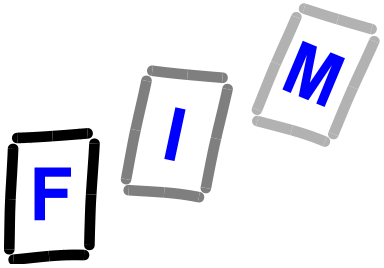
Betriebssysteme

Teil 10 B:
Fragen rund um
Seitenfehler



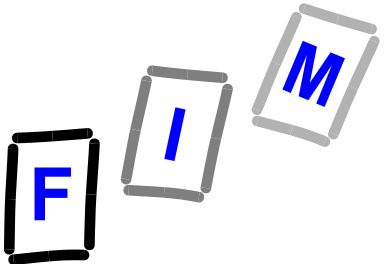
Überlegungen

- **Wenn wir einige Seiten eines Programms in den Speicher laden, brauchen wir eine Strategie, welche Seiten als nächstes geladen werden sollen**
  - ➡ **Hoffentlich wird jene Seite geladen, die im nächsten Befehlszyklus adressiert wird**
- **Später: Wenn wir eine Seite brauchen, die noch nicht geladen wurde, müssen wir sie sofort laden**
  - ➡ **In welchen Seitenrahmen?**
  - ➡ **Wenn alle Seitenrahmen voll sind, was dann?**



Grundsätze

- **Ladestrategie** (engl: fetch policy)
  - ➡ Laden bei Bedarf
  - ➡ Vorschau-Laden
- **Platzierung** (engl: placement strategy)
  - ➡ In welchen Rahmen (frame) soll eine Seite gelegt werden?
- **Austauschstrategie** (engl: replacement strategy, deutsch auch: Seitenersetzungs-Strategie)
  - ➡ Welche Seite ist auszulagern, wenn der Speicher voll ist?
- **Zuweisungsalgorithmus** (engl: allocation algorithm)
  - ➡ Welcher Prozess bekommt wie viele Seitenrahmen?



Laden bei Bedarf “Demand Paging”

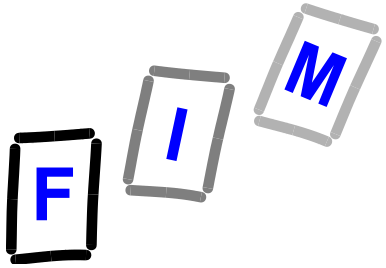
Paging on Demand
ist eine spezielle und üblicherweise *die* Ladepolitik

- **Warte bis die Seite gebraucht wird (die Seite referenziert wird)**
  - » **Überprüfe, ob diese Seite schon geladen wurde Geschieht durch Überprüfung des Valid Bits in der Seitentabelle**
  - » **Falls nicht, lade die Seite: Hole sie (engl: fetch) von der Festplatte/Pagefile**
- **Oft mit Vorschaufunktion kombiniert**
  - » **Nicht nur die benötigte, sondern einige andere (die „benachbarten“, „nächsten“) werden „auf Verdacht“ mitgeladen**

Beispiel Windows: Mitschreiben der anfangs benötigten Seiten für spätere Bootvörgänge/Programmstarts

Das Prinzip von Zugriff bei Bedarf

- **Was Du heute kannst besorgen, das verschiebe nicht auf morgen, wenn Du es auch auf übermorgen verschieben kannst. „Lazy Swapper“**
- **Denn vielleicht stellt sich dann heraus, dass Du es überhaupt nicht machen musst!**

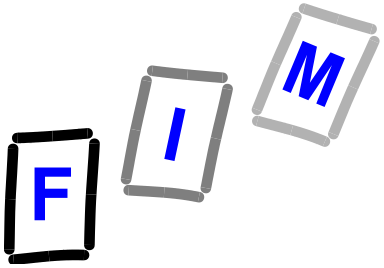


Seitenfehler
(engl.: page fault)

Falls ein Prozess eine Seite anspricht (Seitenreferenz), die zur Zeit nicht im Hauptspeicher ist, dann kommt es zu einem Seitenfehler („page fault“)

- **Solch eine Seite bezeichnet man als “invalid page”**
- **Dies wird angezeigt durch einen Seitenfehlerinterrupt (=Trap)**
- **Nun wird der "lazy swapper" gefordert:**
  - **Die richtige Seite wird eingelesen**
  - **In welchen Seitenrahmen?**

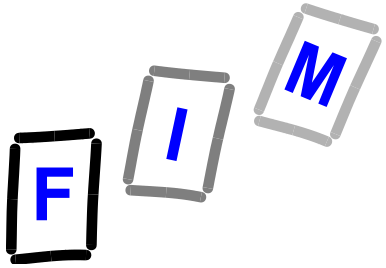
    → Ersetzungsstrategie



Seitenfehler

Kurz:

Ein Seitenfehler ist ein Verweis/Referenz auf eine ungültige Seite



Seitenladen beim Programmstart

1 Lazy Swapper (hier: BS-Komponente zum Laden einer Seite) lädt die erste benötigte Seite (und vielleicht auch ein paar „benachbarte“ Seiten)

2 Programm/Prozess wird gestartet

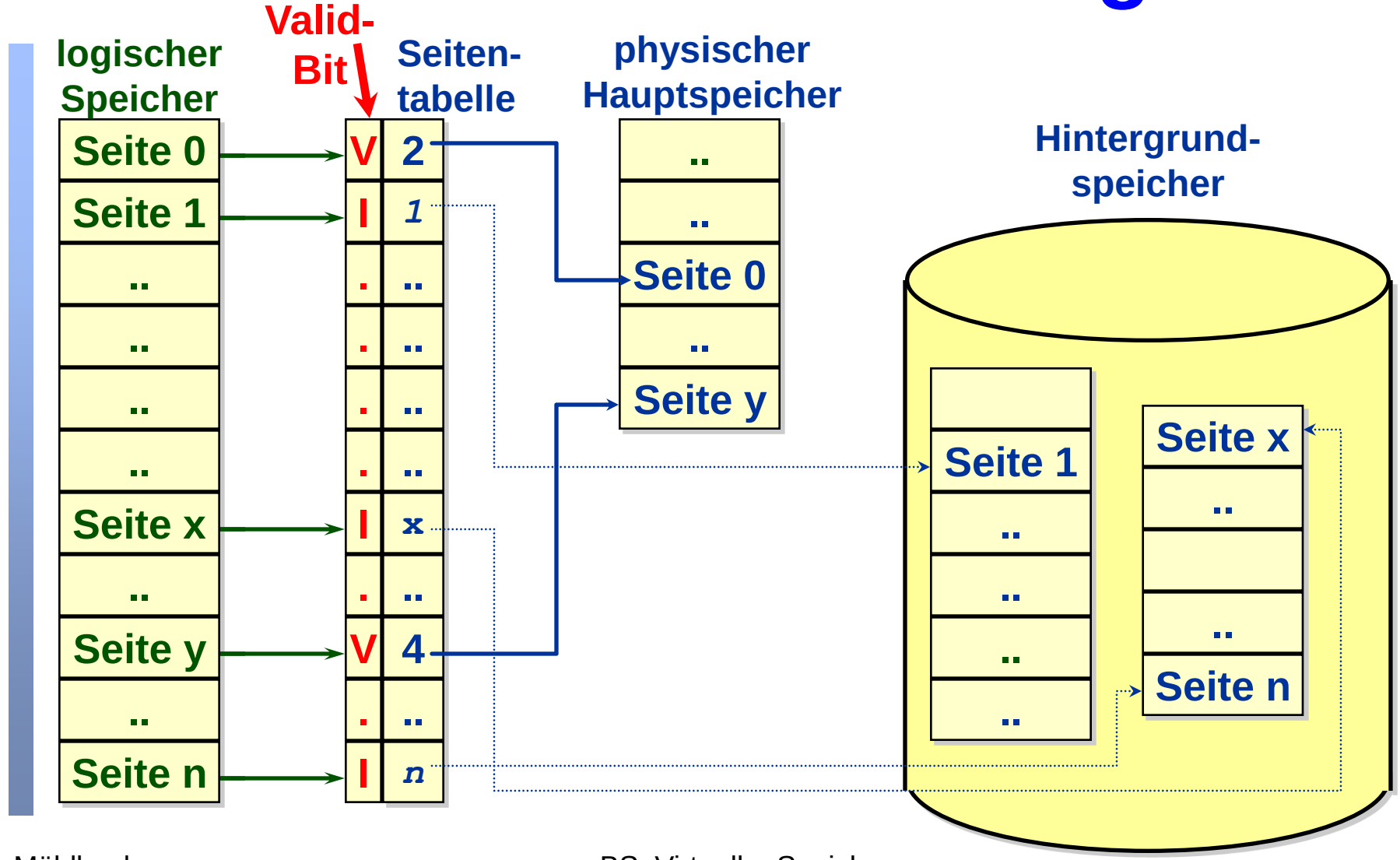
……...

x Spricht das Programm eine noch nicht vorhandene Seite an, so kommt es zu einem *Seitenfehler*

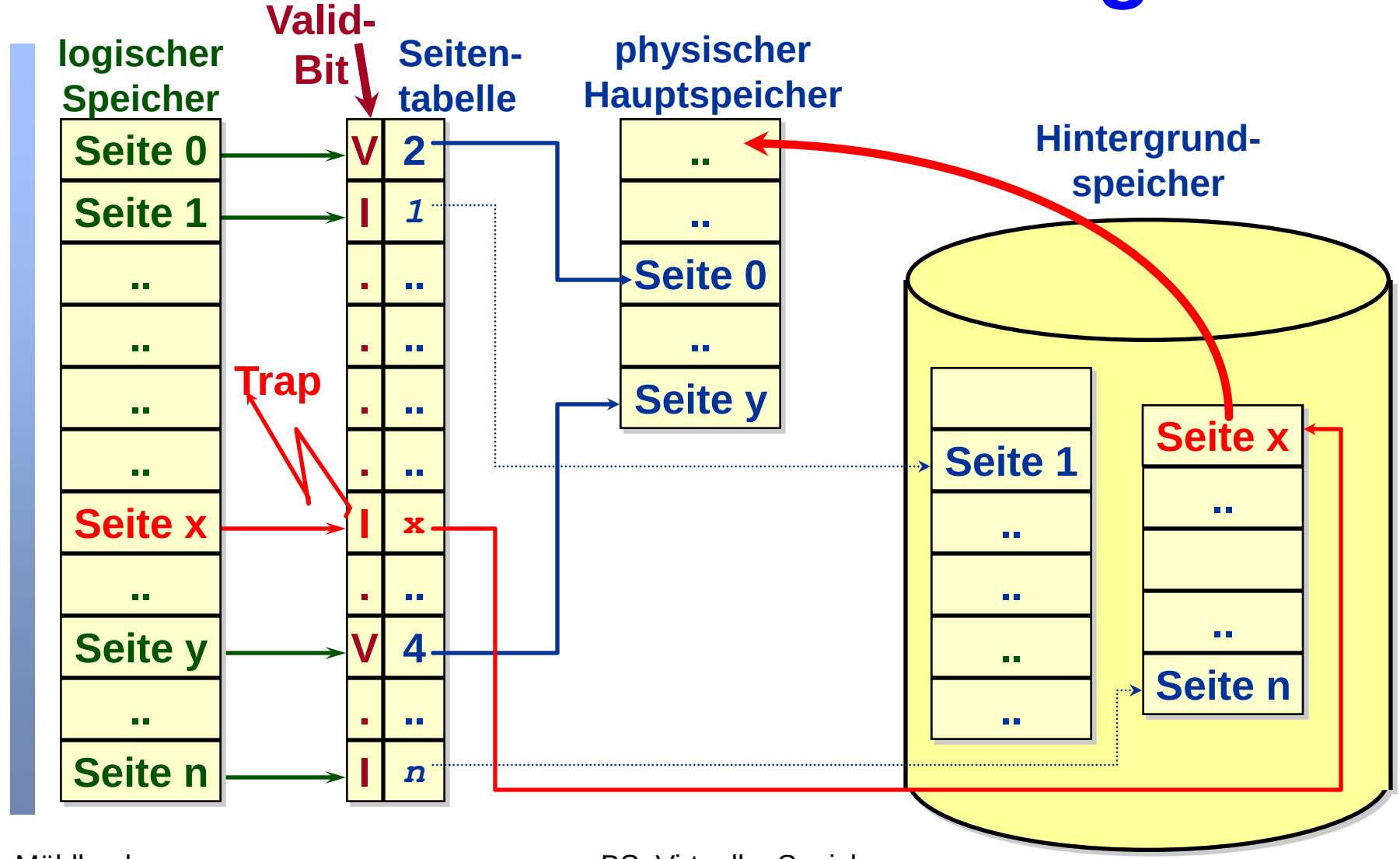
x+1 Seitenfehler löst einen Trap aus (Interrupt)

x+2 Die benötigte Seite ist nachzuladen

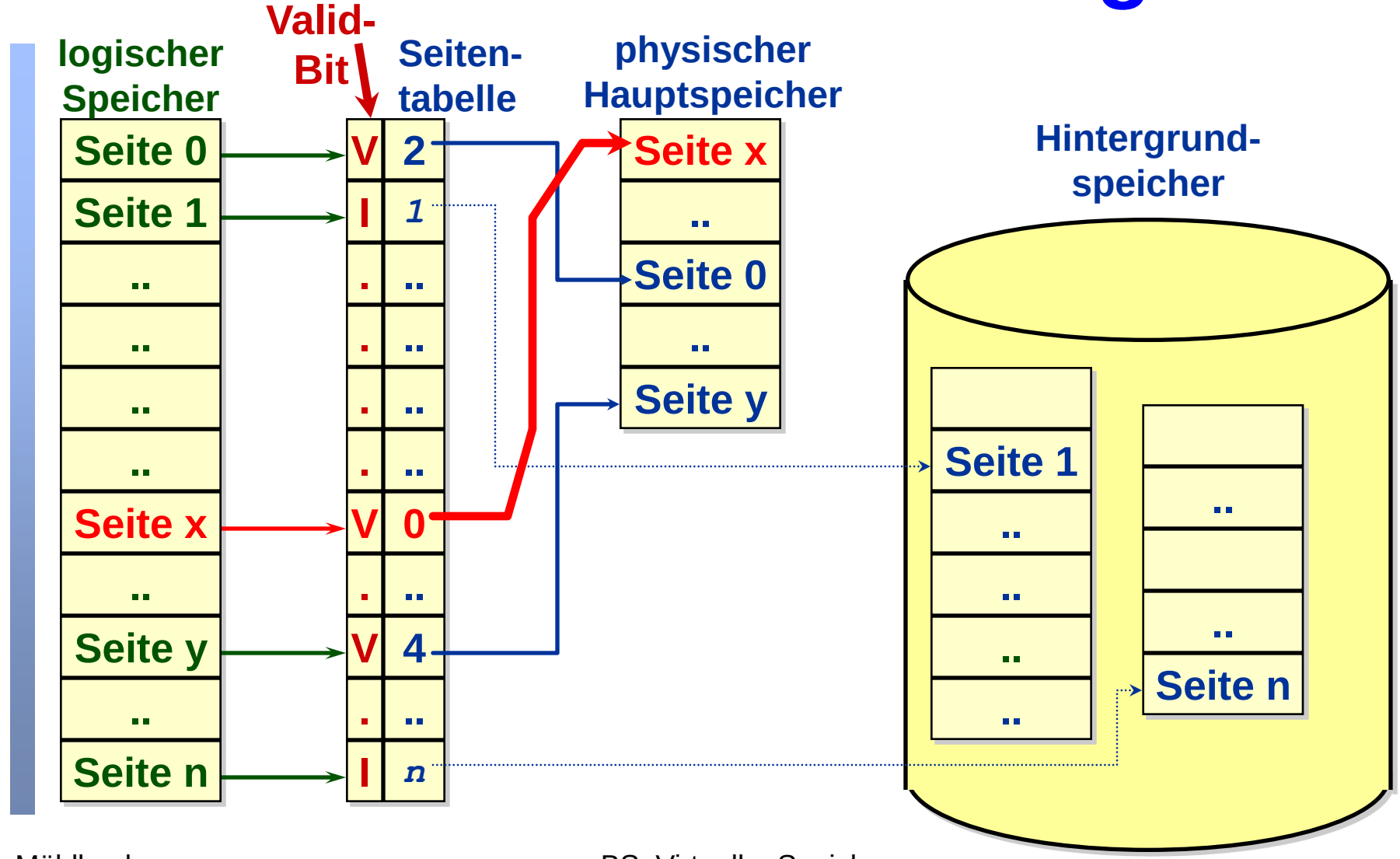
Wie wird ein Seitenfehler ausgelöst?



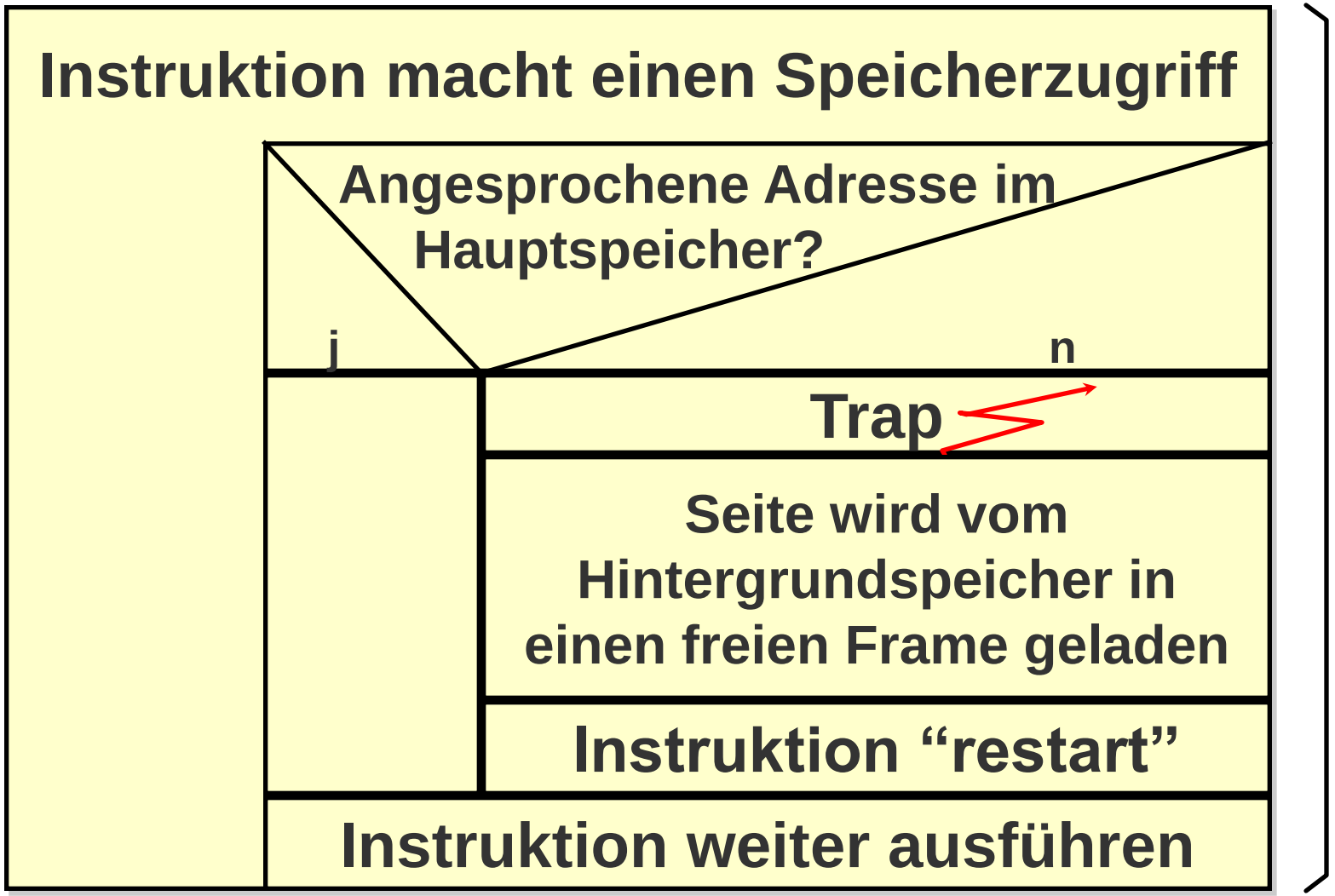
Wie wird ein Seitenfehler ausgelöst?

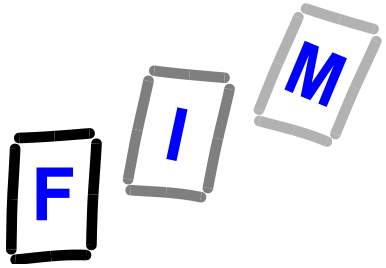


Wie wird ein Seitenfehler ausgelöst?



Ablauf Speicherzugriff (vereinfacht)





Seitenfehler-Aktionen des BS (stark vereinfacht)

1) Seitenfehler Interrupt bedienen

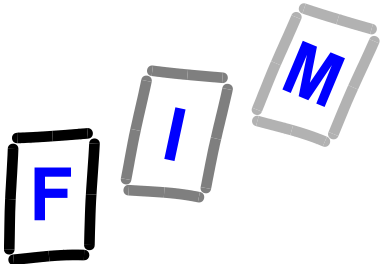
   z.B. 1 - 100 Mikrosekunden

2) Lesen der Seite von der Disk (z.B.)

Disk Wartezeit	~2 ms
Disk Suchzeit	~10 ms
Disk Transferzeit	~1 ms

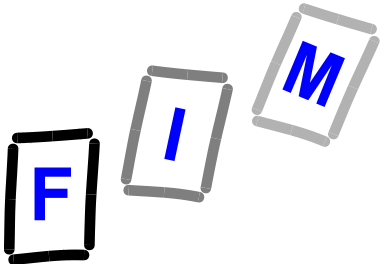
3) Prozess wieder starten

   z.B. 1 - 100 Mikrosekunden



Reaktion des BS auf Seitenfehler (noch unvollständig)

0. **Maschineninstruktion im Prozess *„PagefaultProcess“* spricht eine Adresse an, die derzeit nicht im Hauptspeicher ist**
1. **Trap zum Betriebssystem**
2. **Register und Prozess-Status speichern**
3. **Erkennen der Interrupt Quelle**
4. **Seitenfehler analysieren (Legal? Wo auf Disk?)**
5. **Festplatten-Leseoperation auslösen**
6. **Während des Wartens andere Prozesse zum Zug kommen lassen (Scheduling)!**



Reaktion des BS auf Seitenfehler

(noch unvollständig)

7. Interrupt von der Festplatte (I/O vollständig)
8. Register und Prozess-Status speichern
9. Erkennen der Interrupt Quelle
10. Seitenverzeichnis(se) korrigieren
11. Warten, bis *„PagefaultProcess“* die CPU wieder erhält
12. Register und Prozess-Status wiederherstellen
13. Neustart des unterbrochenen Maschinenbefehls

Überlegungen zur Speicher-Zugriffszeit

Kenn	Beschreibung	Ordnung	z.B.
WZZ	Effektive Zugriffszeit	??	?
SZZ	Speicher-Zugriffszeit	5-50 ns	10ns
p	Wahrscheinlichkeit eines Seitenfehlers	[0..1]	?
SFZ	*Seitenfehlerzeit*	*5-35 ms*	*10ms*

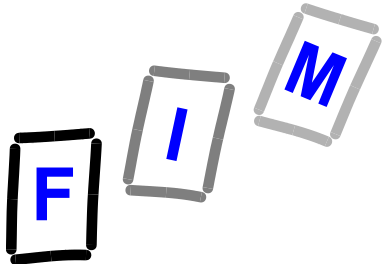
WZZ = SZZ * (1-p) + SFZ * p

Vorgabe z.B.: Effektive Zugriffszeit sollte höchstens um 20 % länger als Speicher-Zugriffszeit dauern

$12 = 10 * (1 - p) + 10*10^6 * p$

$p <= 0{,}0000002$ (Größenordnung 10^{-7})

(ohne Berücksichtigung TLB-Miss)



Überlegungen zur Speicher-Zugriffszeit

p<=0,0000002 → alle 5.000.000 Speicherzugriffe ein Seitenfehler

Annahme (sehr grob und teilweise falsch!):

Eine Seite sind 4096 Bytes

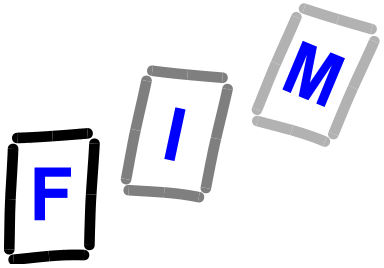
Ein Befehl benötigt 2 Bytes

(Fast) Jeder Befehl verursacht einen Speicherzugriff

Dann sind das ca. 2.000 Speicherzugriffe pro Speicherseite Code

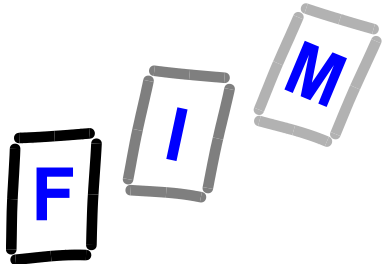
Ergebnis: Alle 2.500 Speicherseiten Programmcode darf ein Speicherfehler passieren!

Das müssen schon viele Schleifen sein (selbe Seite erzeugt viele, hoffentlich gleiche, Speicherzugriffe), oder die Vorhersage der benötigten Seiten ist SEHR gut!



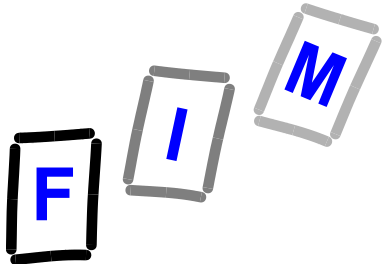
Konsequenzen

- **Die effektive Zugriffszeit ist direkt proportional zur Seitenfehler-Rate**
- **Spontaner Lösungsansatz:**
  - ➡ **Vergrößere den Speicher, damit er mehr Seiten aufnehmen kann, was wiederum zu weniger Seitenfehlern führen sollte**
- **Aber Software „verhält sich wie Gas“**
  - ➡ **Sie wird jeden Speicher füllen, neue Versionen von Software brauchen immer mehr Speicher**
  - ➡ **Wir möchten immer den Multitasking-Grad erhöhen, indem wir viele Prozesse gleichzeitig laufen lassen**



Problem: Hauptspeicher zu klein (1)

- **Ausgangspunkt**
  - ➡ **Viele Prozesse parallel (hoher Multitasking-Grad)**
  - ➡ **Logischer Adressraum ist nicht durch physikalischen Hauptspeicher beschränkt (→ Virtueller Speicher)**
- **Konsequenz:**
  - ➡ **Auch wenn bei jedem Prozess bei weitem nicht alle Seiten eingelagert werden müssen, so „geht der phys. Hauptspeicher über“**
  - ➡ **Engl: Memory Overloading**

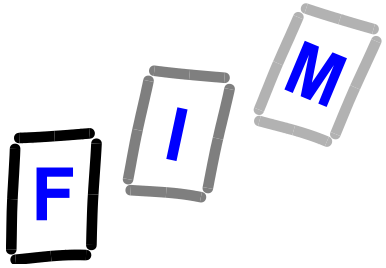


Problem:
Hauptspeicher zu klein (2)

- **Lösung:**

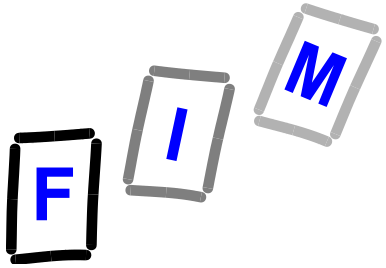
  Seitenrahmen im phys. Haupspeicher müssen freigegeben werden, obwohl der Prozess noch läuft (→ Seitenaustausch ist notwendig)

- **Konsequenz:**
  - ➡ **Eventuell müssen Seiten während des Prozessablaufs mehrmals eingelagert werden**
  - ➡ **Modifizierte Seiten müssen ausgelagert werden (Rückschreiben auf die Festplatte)**



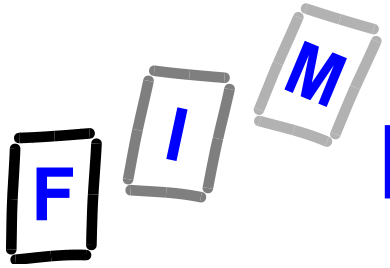
Demand paging soll effektiv sein

- **Grundsätzlich sind die Daten/Code durch das Dateisystem organisiert, und die Seiten werden von dort geladen**
  - ➡ **Dateisuche braucht zu viel Zeit**
- **Verwende einen speziellen „Swapbereich“ während der Laufzeit für Seiten(rahmen), die ein- und ausgelagert werden müssen**
  - ➡ **Page-File**
  - ➡ **Einfacher organisiert, damit schnellerer Zugriff**



Auslagern modifizierter Seiten

- Nur modifizierte Seitenrahmen müssen auf den Page File zurückgeschrieben werden
- Read-Only bzw. Execute-Only-Seitenrahmen z.B. sind nicht zurückzuschreiben
- Bei Seitenrahmen mit Daten hilft uns ein zusätzliches Bit im Seitenverzeichnis, welches anzeigt, ob auf diesem Seitenrahmen eine Schreiboperation stattgefunden hat
  - ➡ DIRTY-Bit



Einträge im Seitenverzeichnis
(Page Table Entries)

Dirty-Bit (CPU bzw. MMU-Hardware)

- Wurde seit dem Einlagern auf diese Seite eine Schreiboperation ausgeführt?

Valid-Bit

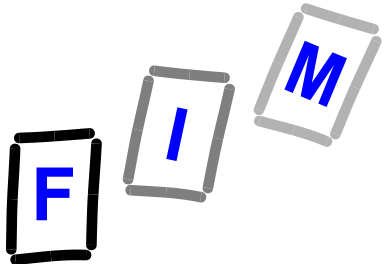
- Befindet sich die Seite im Hauptspeicher?

Seitennummer

- Wo liegt die Seite im Hauptspeicher
  (bzw. wo liegt sie auf der Disk im Page File)

Bits, die für Sicherheit/Schutz verwendet werden

…….



Damit weitere Aufgaben des paging systems

1) Seitenfehler Interrupt bedienen
2) Wenn noch ein freier Frame verfügbar, dann verwende diesen; weiter bei 5
3) Bestimme eine Seite, die ausgelagert werden soll: „victim page“
4) Falls dirty-bit gesetzt, schreibe die Seite auf die Disk
5) Lesen der Seite von der Disk
6) Seitenverzeichnisse korrigieren
7) Prozess wieder starten (fortsetzen)