

Speicherarchitektur

Speicherarchitektur

Die Bedeutung des Speichers in der Mikroarchitektur

Die Harvardarchitektur

Differenz von den vielen Lebewesen

Der Adressraum

Die Leistung (Performance des Speichers)

Die Technologische Barriere

Architektur vs. Technologie

Caches:

Stromber

Architektur

Zugriffspunkthalle

Speicherbandbreite

Die Speicherkette

Zugriffstyp, Band und Verarbeitung

Speicherzugriff (Memory Controller)

Sprüngeverwaltung (Schnittstelle zu Betriebssystem)

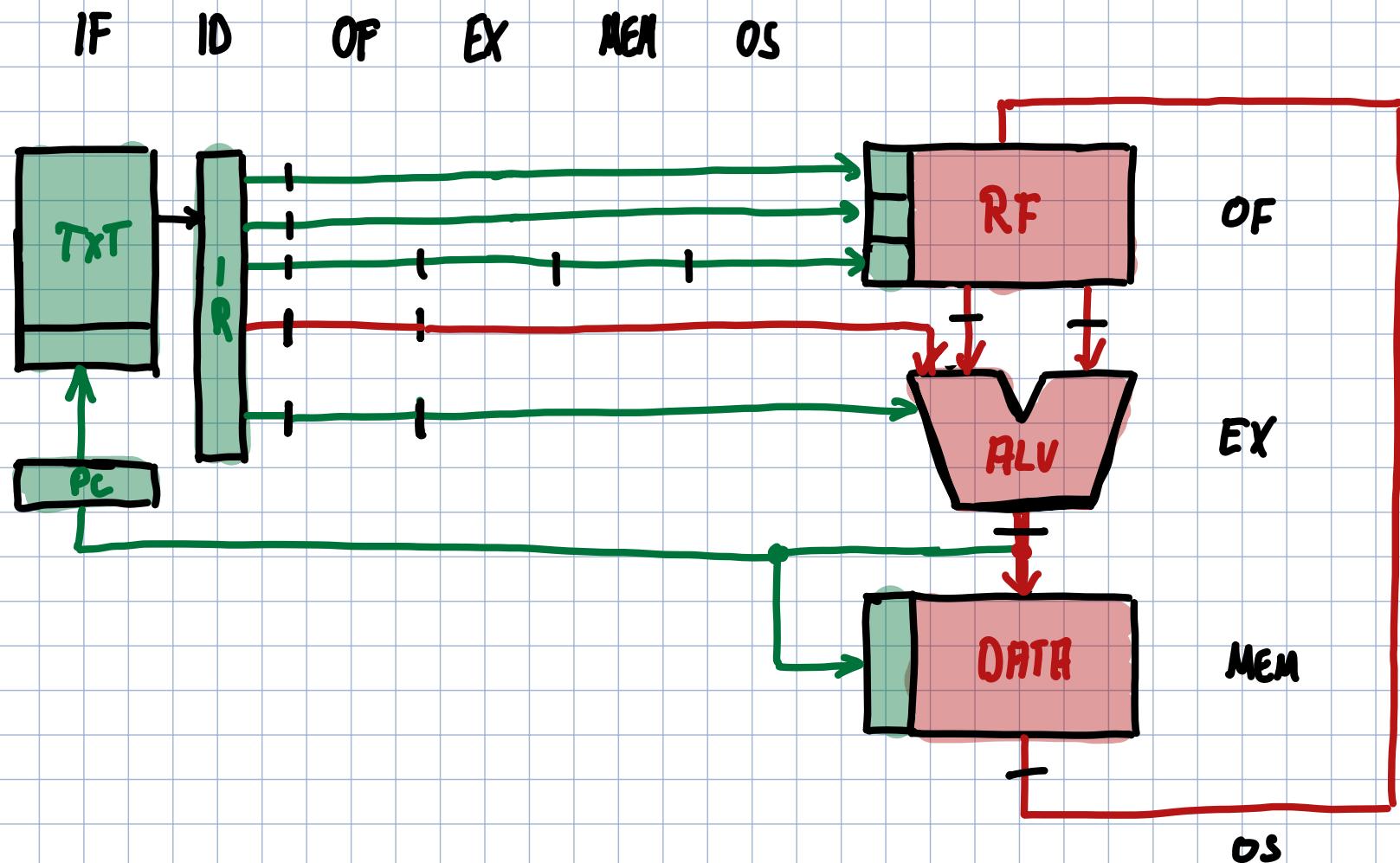
Overlays

Segmentverwaltung

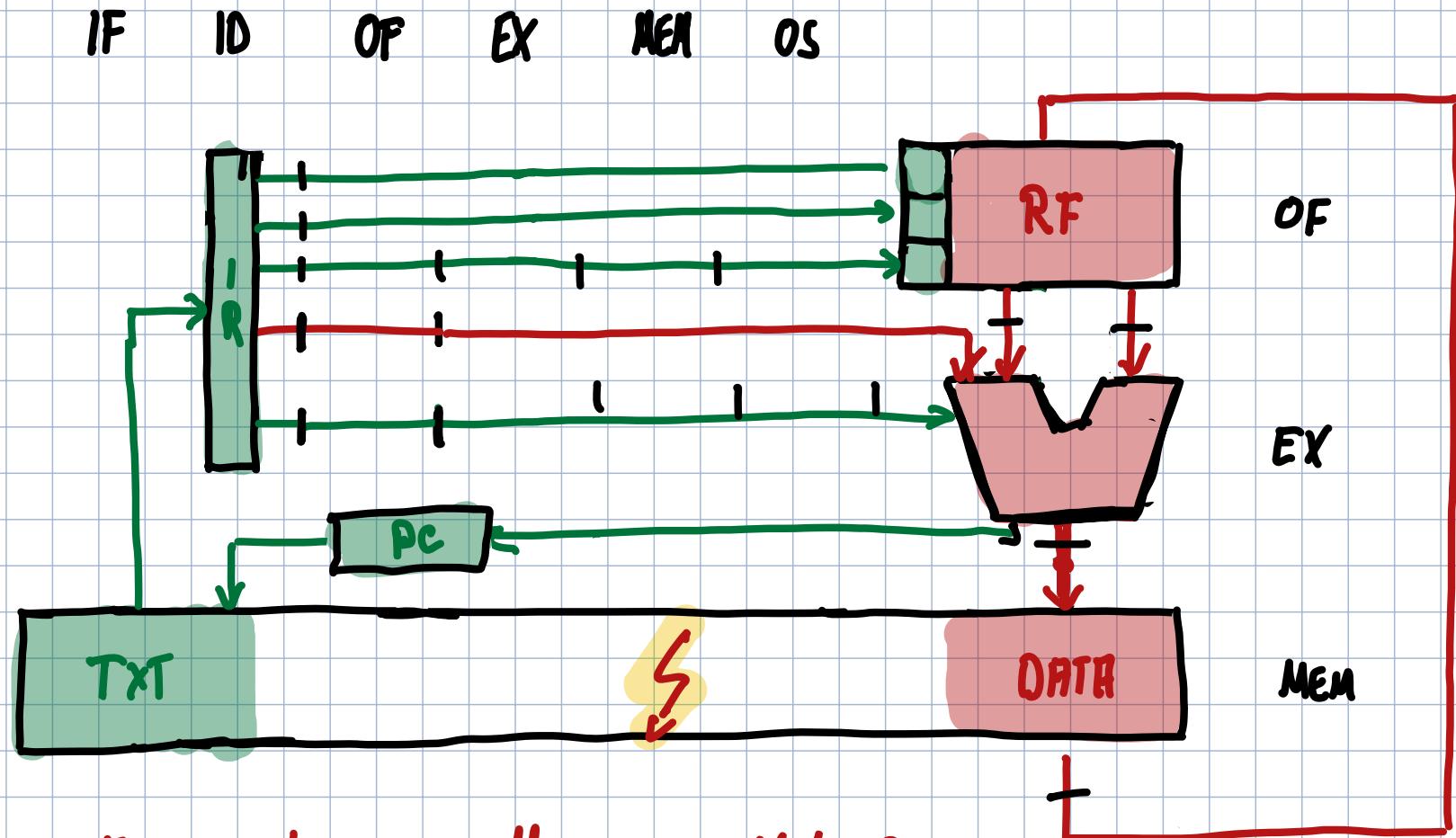
Slittenverwaltung

Speicherverwaltungshardware (Memory Controller)

Mikroarchitektur mit Flipflopsversatzung als hazard-Architektur:

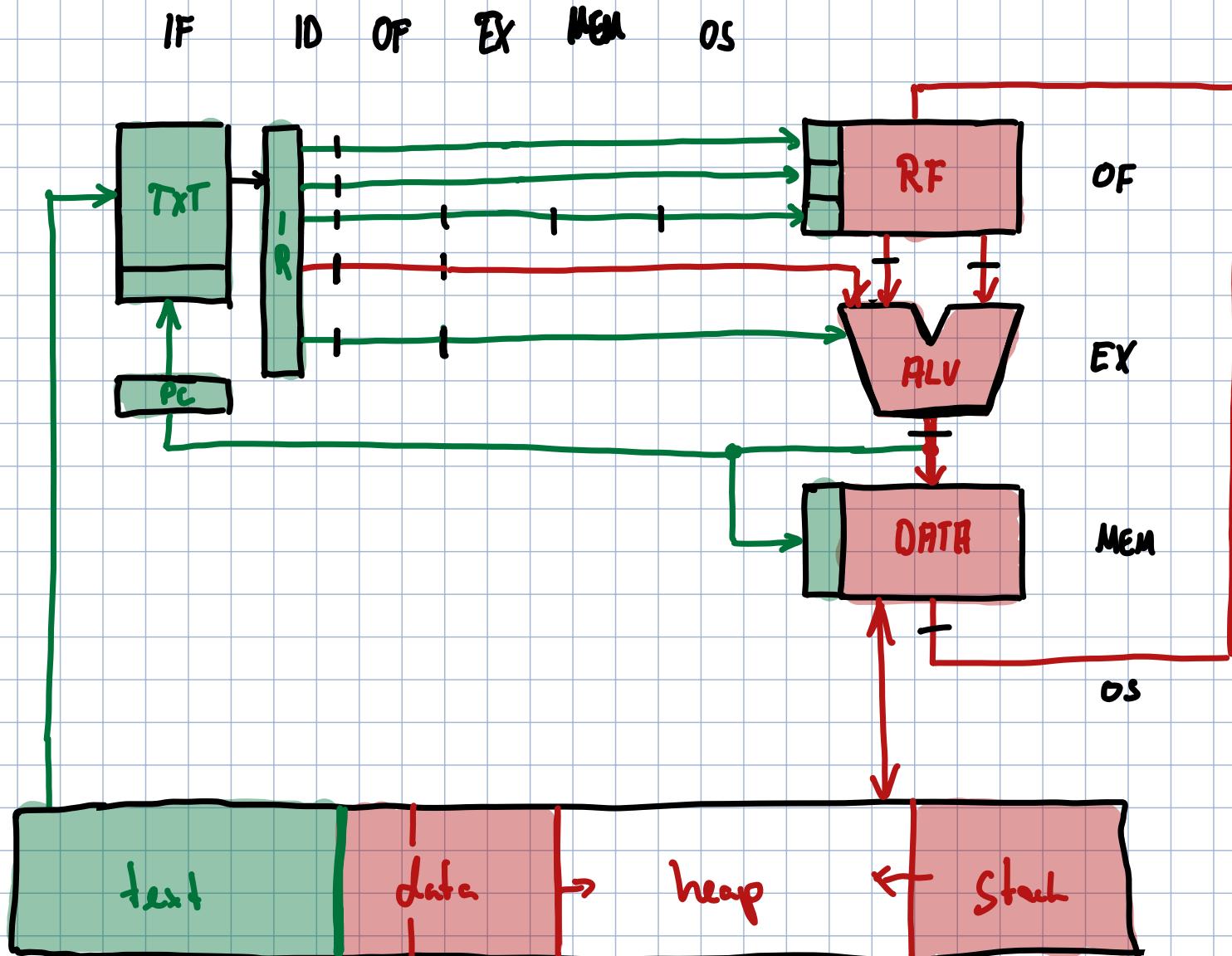


Mikroarchitektur mit Fließbandarzähler als Pionier- (v. Numann) - Architektur:



Gleichzeitige Zugriff unterschiedliche Phasen
auf die gleichen Speicher, ist ohne hohen
HW-Refence nicht möglich.

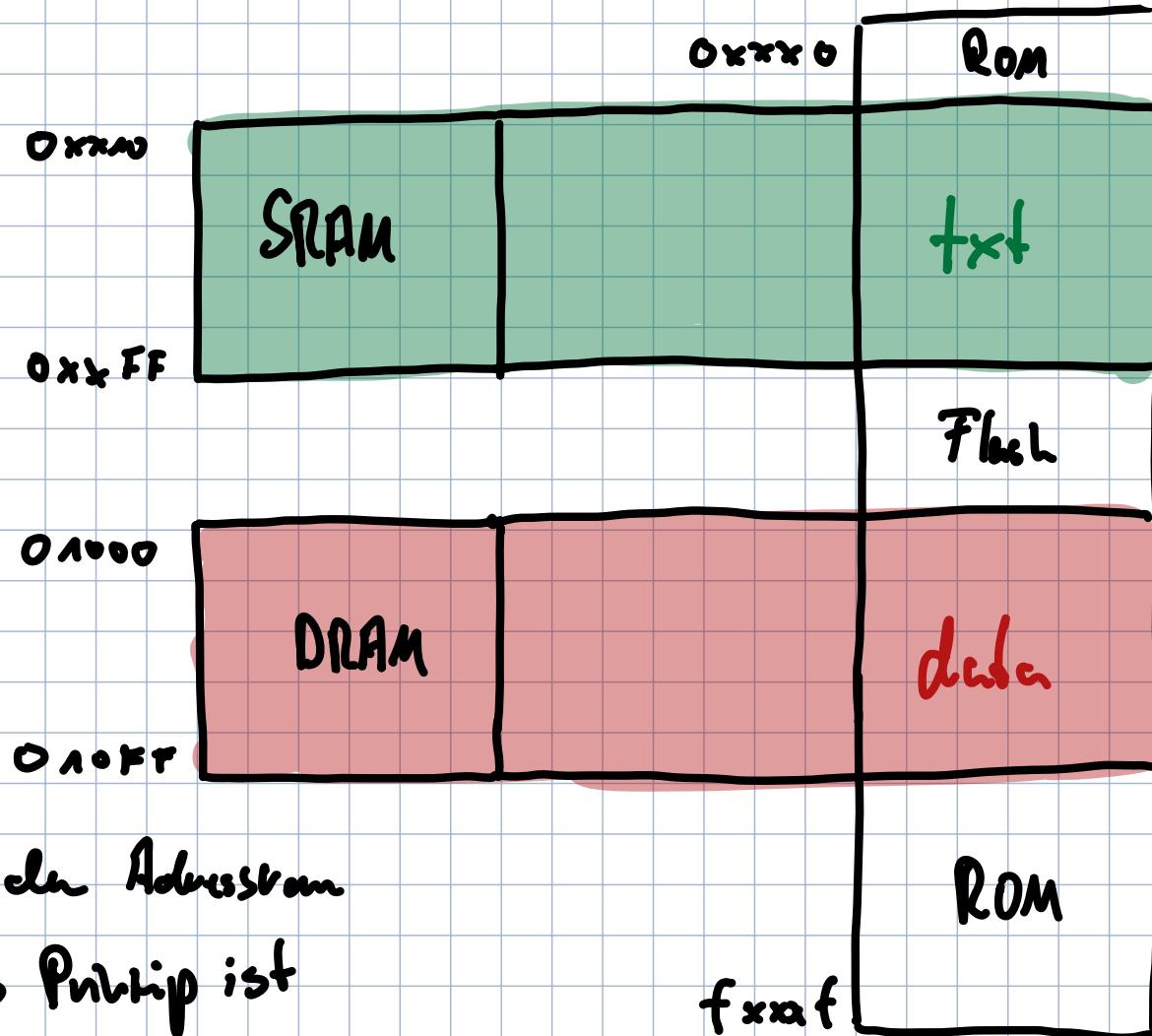
Mikroarchitektur mit Flipboard verarbeitung und kombinierten Harvard-, Princeton - Architektur



Notwendig sind kleine lokale Speicher, die für den Zugriff der jeweiligen Phase reserviert sind. Die Hardware muß daher sicherstellen, daß wir die Modelle der jeweiligen Phasen auf den lokalen Speichern abgreifen können.

Adressraum: Scratch Pad

Nur einheitl. Adress Spez.

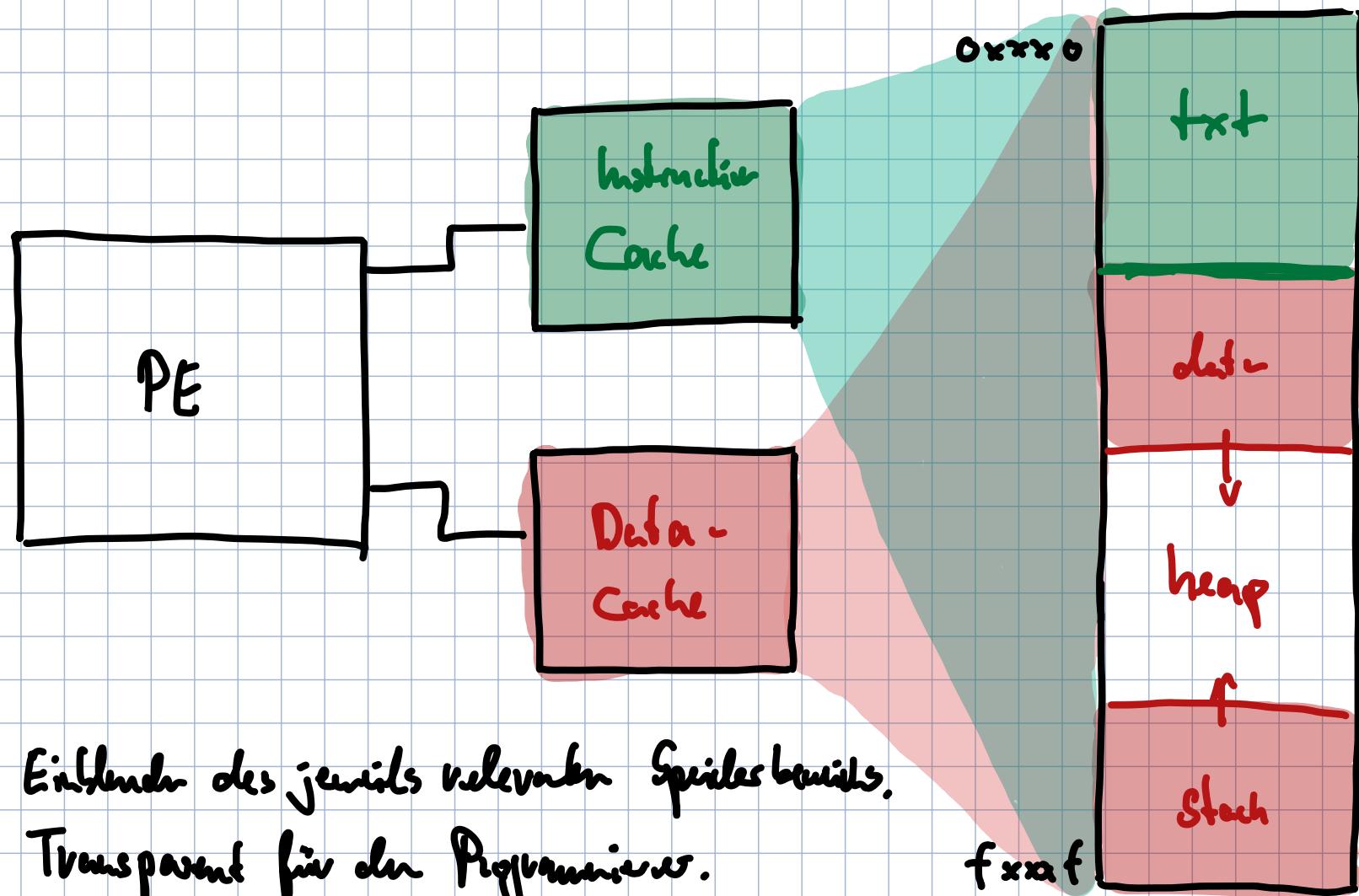


Dieses Modell separiert den Adressraum
in separate Teile. Das Prinzip ist
nicht transparent für die Programmierung.

Adressraum : Cache

Uniform Adress Space

Lokaler Speicher



Einindruck des jeweils relevanten Speicherbereichs.
Transparent für den Programmierer.

Hwang - Architektur vs. Princeton (v. Neumann) Architektur

Speicheranfließung zwischen
Programm und Daten fest

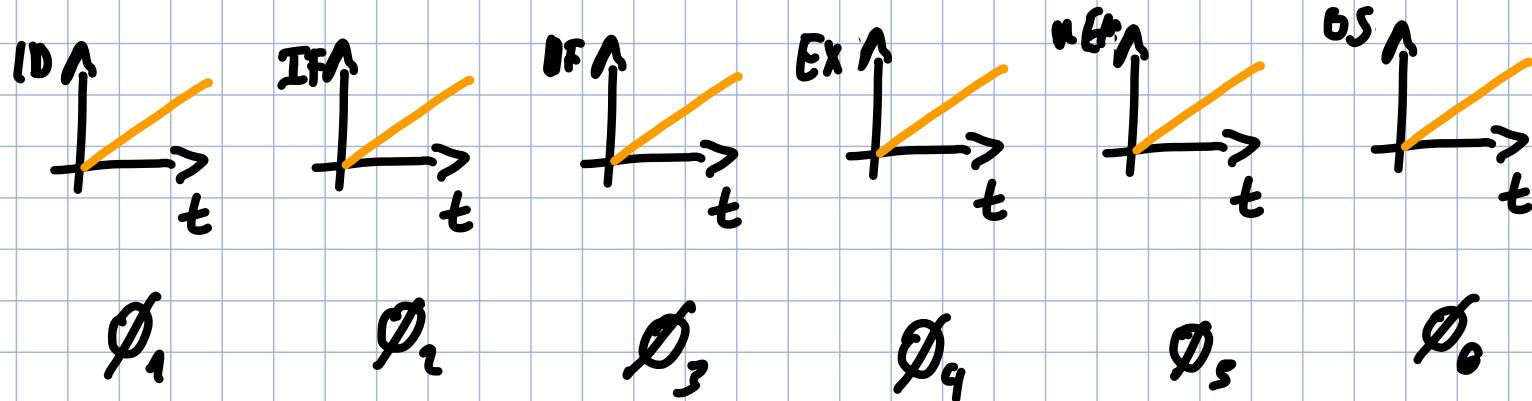
Speicheranfließung variabel

Keine Ingriffskontrolle

v. Neumann Flaschenhals

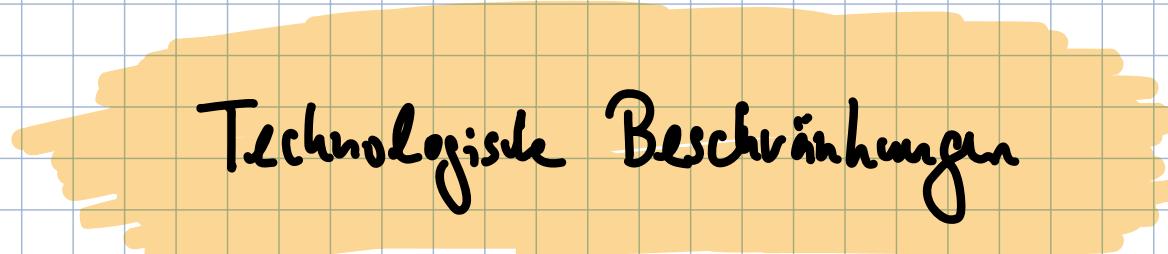
Doch: Kombination aus beiden Architekturen:

Speicher



$$P = \min_{\phi} \{ P_\phi \}$$

Zugriff auf den Speicher muss die gleiche Zugriffszeit wie die Bearbeitung der anderen Phasen haben.



Technologische Beschränkungen

Gekürzter Laufzeit

Springantwort $U_a(t) = U_e(0) e^{-\frac{t}{\tau}}$ $U_a(t) \approx U_e(0) (1 - e^{-\frac{t}{\tau}})$

Bitwechsel ($L \rightarrow H \vee H \rightarrow L$) bei $U_a(t) = \frac{1}{2} U_e(0)$

$$\cancel{U_e(0)} e^{-\frac{t}{\tau}} = \frac{1}{2} \cancel{U_e(0)} \quad \cancel{U_e(0)} (1 - e^{-\frac{t}{\tau}}) = \frac{1}{2} \cancel{U_e(0)}$$

$$-\frac{t}{\tau} = \ln\left(\frac{1}{2}\right)$$

$$-t = \tau (-\ln(2))$$

$$\ln(1) - \frac{t}{\tau} = \ln(2^{-1})$$

$$\cancel{\ln(1)} + \ln(2) = \frac{t}{\tau}$$

$$t = \tau \ln(2)$$

$$t = -\tau \ln(2)$$

$$\tau = RC$$

Abschätzung des Laufzeit eines Signals (Elektrotechnische Nähe)

Aluminium Leiter in einem Siliziumprozess (CMOS) 28 nm

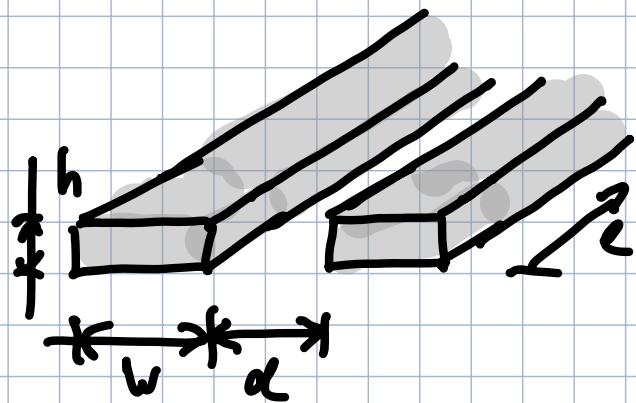
Kleinste Leiterbreite nach Design Rules (DR) : 40 nm W

Kleiner Abstand - " - : 40 nm d

Annehmen: Fläche F : 40 nm

Zwei parallele Leiterbahnen: DRAM-Zelle: $6F^2$

SRAM-Zelle: $140F^2$



$$R^0 = \rho \frac{l}{A} = \rho \frac{l}{h \cdot w}$$

$$C^0 = \epsilon_0 \epsilon_r \frac{A}{d} = \epsilon_0 \epsilon_r \frac{l \cdot w}{d}$$

$$\tilde{\tau} = R^0 C^0 = \rho \frac{l}{h \cdot w} \cdot \epsilon_0 \epsilon_r \frac{l \cdot w}{d} = \rho \epsilon_0 \epsilon_r \frac{l^2}{d^2} *$$

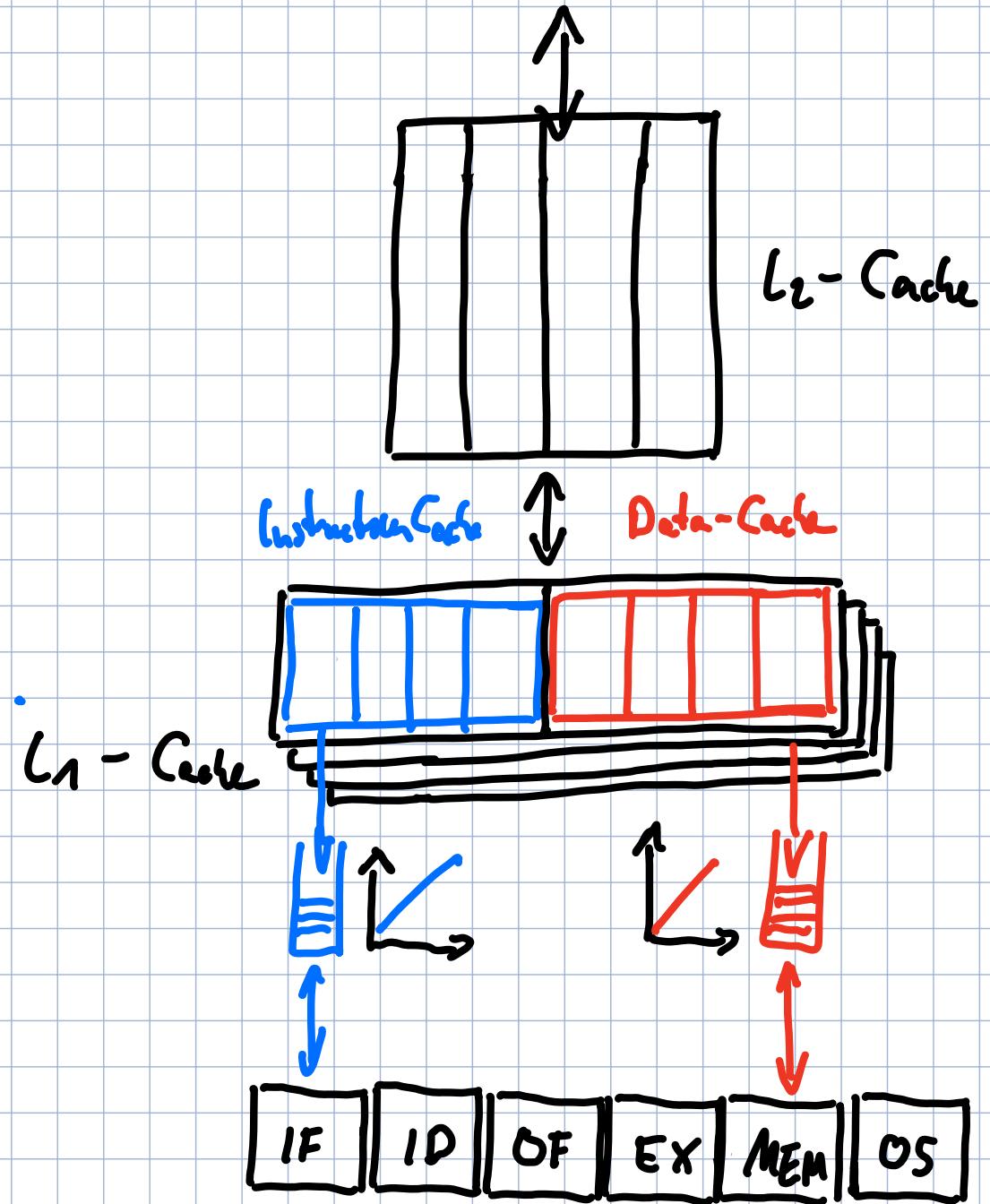
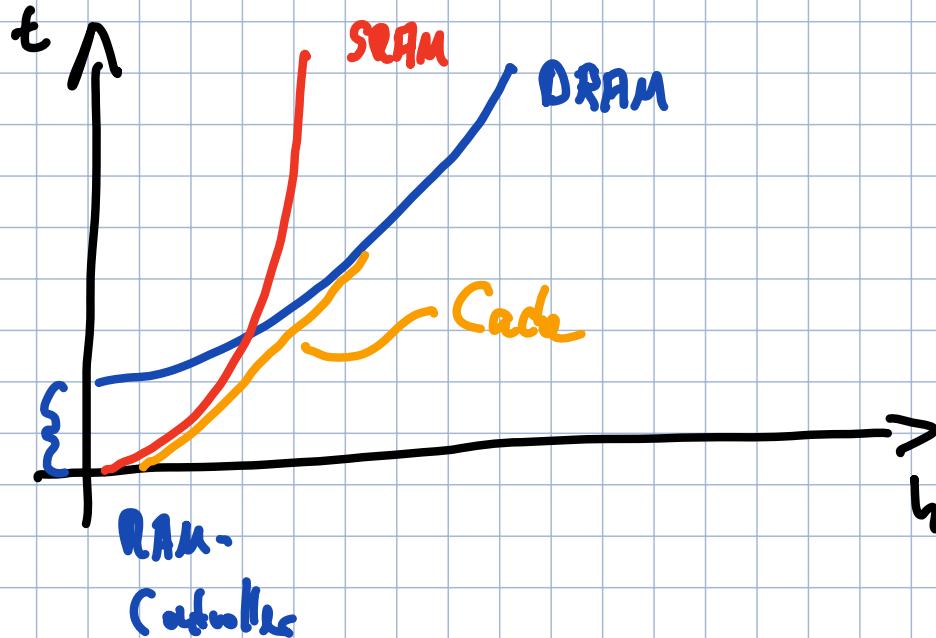
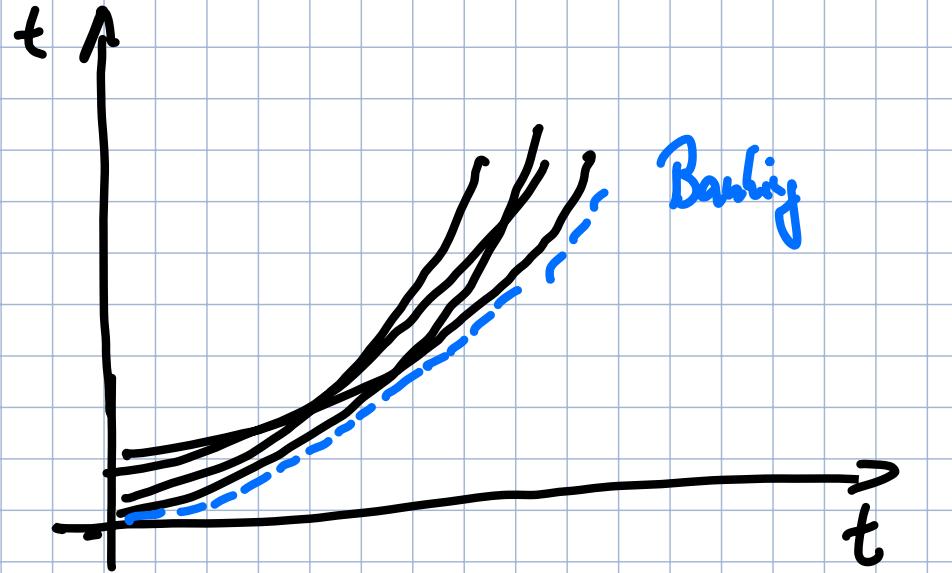
mit $h=d=40 \text{ nm}$ $l^2=d^2=40 \text{ nm}^2$

* $h=d$

folgt $\tilde{\tau} = 728 \cdot 10^{-15} \text{ s} = 0,728 \text{ ps}$

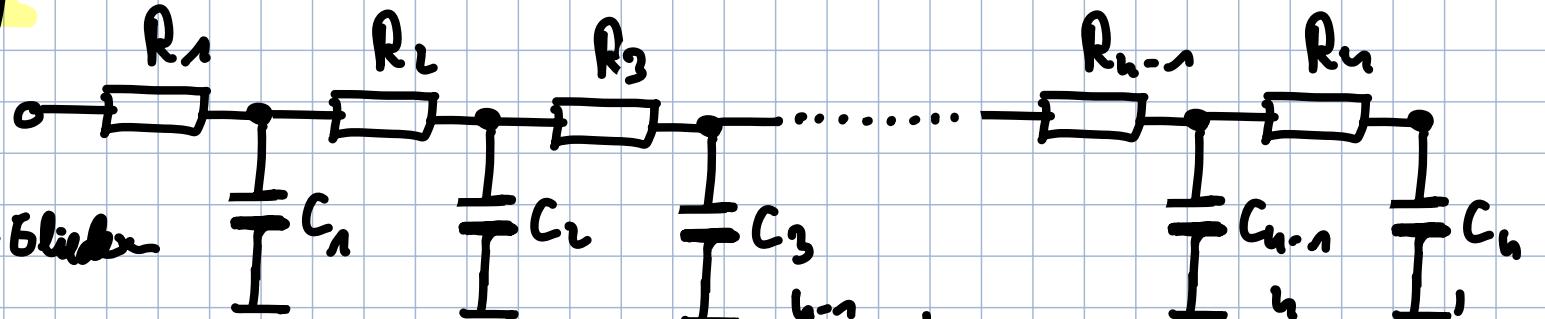
Diese 0,7 ps müssen dann mit l^2 multipliziert werden.

Wirkung des Caches:



Elmaw Delay

Modell:



Kette von RC-Gliedern

$$\text{mit der Zeitkonstante } \tau = RC \text{ folgt} \quad \tau = \sum_{i=1}^n \sum_{j=1}^i R_{ij} C_i = \sum_{i=1}^n C_i \sum_{j=1}^i R_j \\ = R_1 C_1 + (R_1 + R_2) C_2 + (R_1 + R_2 + R_3) C_3 + \dots \\ + \left[\sum_{i=1}^n R_i \right] C_n$$

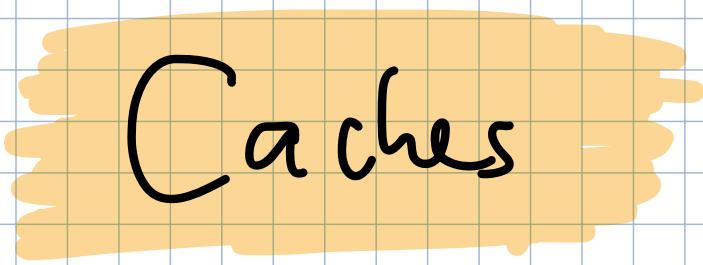
Bei einer (einfach) j:de $R_1 = R_2 = \dots = R_n = R^0$ $C_1 = C_2 = C_3 = \dots = C_n = C^0$

$$\tau = RC + 2RC + 3RC + \dots + nRC$$

Mit der Gaußschen Summenformel

$$1 + 2n + 3n + \dots + n = \frac{n(n+1)}{2}$$

$$\tau = \frac{n(n+1)}{2} RC = \frac{n^2+n}{2} ln(2)$$



Caches

Caches: Begriffe

Assoziativität :

Freiheitsgrad bei der Platzierung
von Speicherblöcken im Cache

Cache Block

und

!

Cache Line

Menge (Set) :

Eine Menge von Blöcken im Cache
mit dem gleichen Cache - Index

Etikett (Tag) :

Bezeichner zu einem Speicherblock
im Cache zu finden.

Eine hohe Assoziativität reduziert die Zahl der Sets und erhöht die Zahl an Tags (mehr RAM \rightarrow weniger).

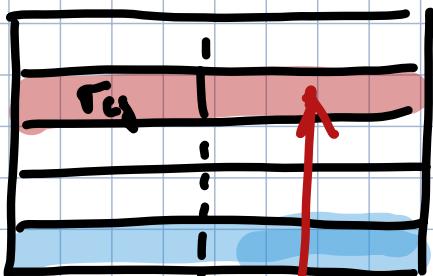
Caches : Strukturen und Organisation

Bereich des Kurses

Cache

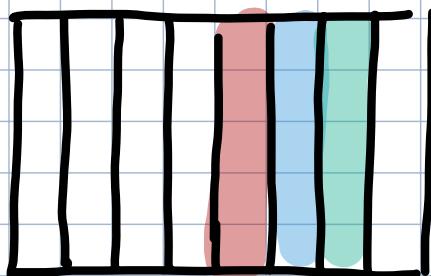
Voll-Assoziativ

n -Blöcke
 $n = 8$



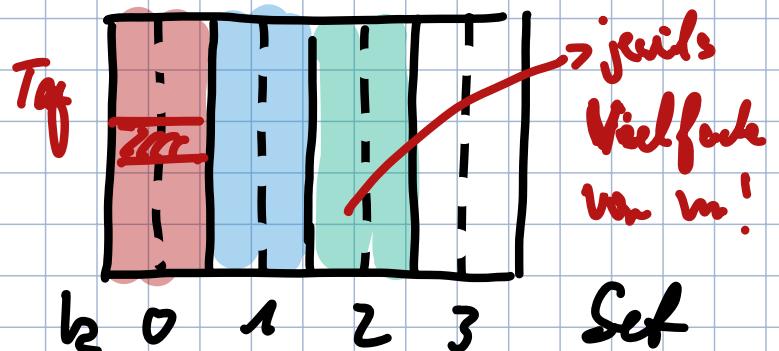
Direct Mapped

$i = 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$



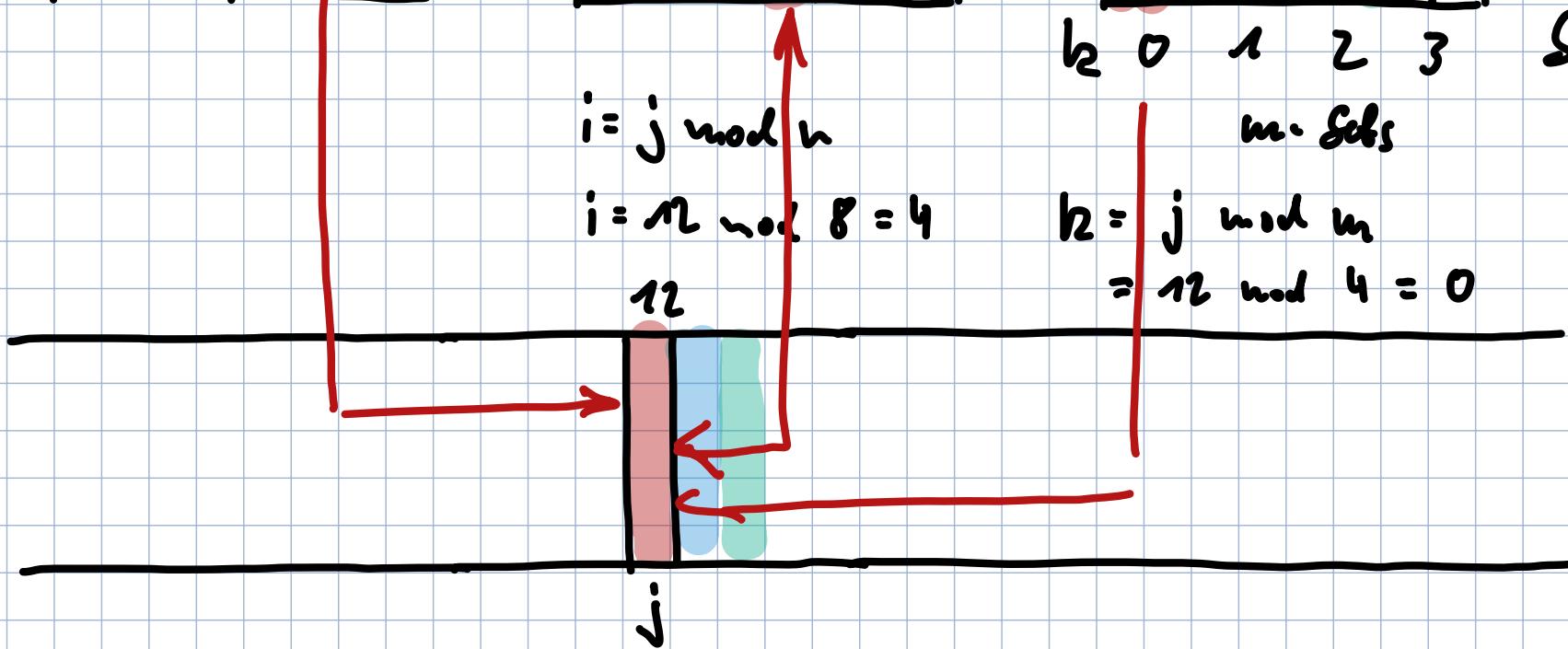
Two-Way-Assoziativ

$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7$



$$i = j \bmod n$$

$$i = 12 \bmod 8 = 4$$



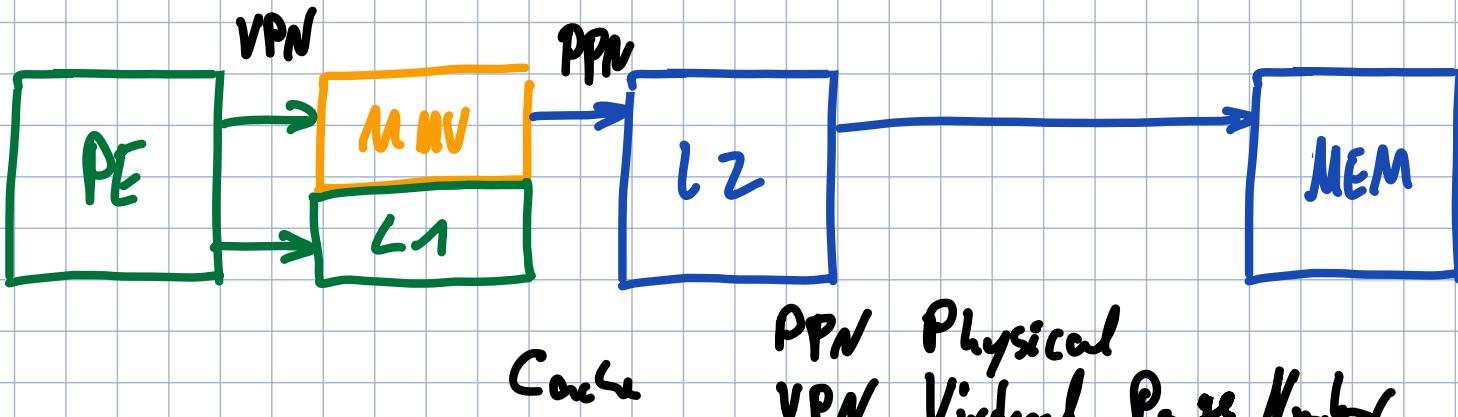
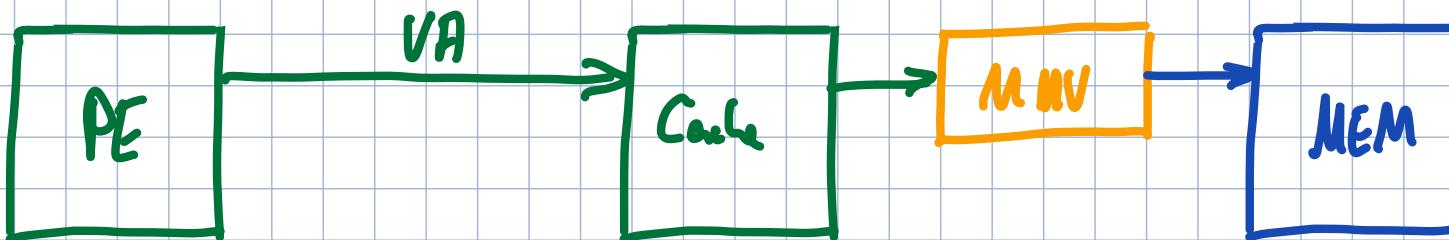
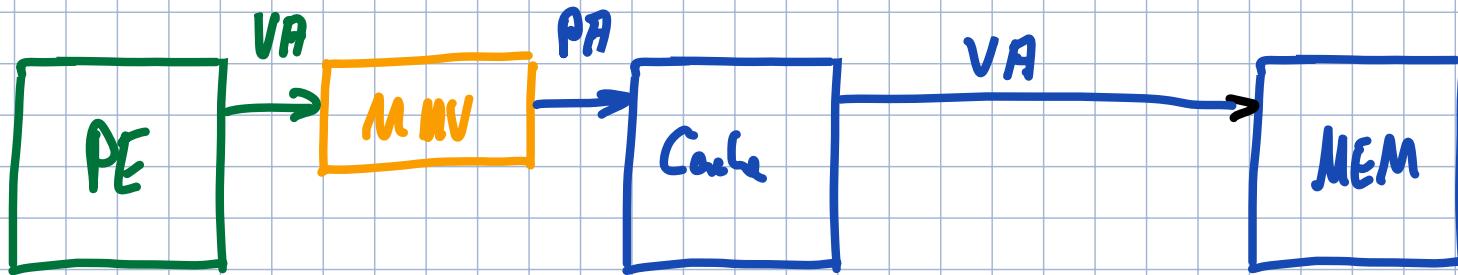
m -Sets

$$b_2 = j \bmod m$$

$$= 12 \bmod 4 = 0$$

Cache Architekturen:

VA = Virtuelle Adresse
PA = Physische Adresse



PPN Physical
VPN Virtual Page Number

Caches : Adressformat



Block Offset

Vollassoziativer Cache

Directed Mapped Cache

großer Index

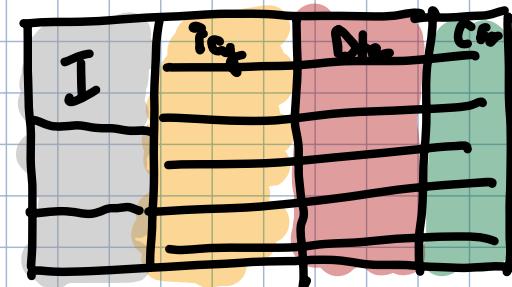
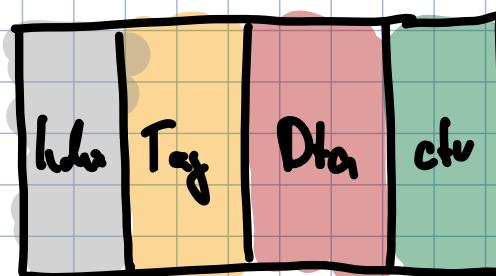
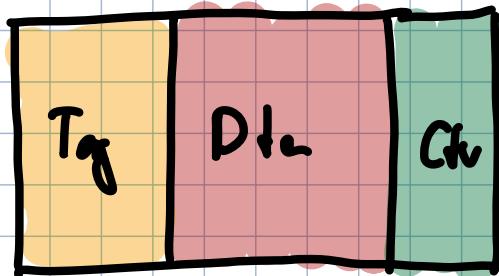
Assoziativ (Lat.
Verknüpfung!)

Assoziativität: Führt bei der Platzierung im Cache

Voll Assoziativ

Direct Mapped Cache

Mehr Assoziativ



Begriffe

Cache hit

miss

miss penalty

flush Leren des Cache

hit rate Trefferrate bessre Wahrscheinlichkeit
miss rate Fehlerrate.

capacity miss

Ein Block war schon im Cache
wurde aber entfernt wurde.

conflict miss

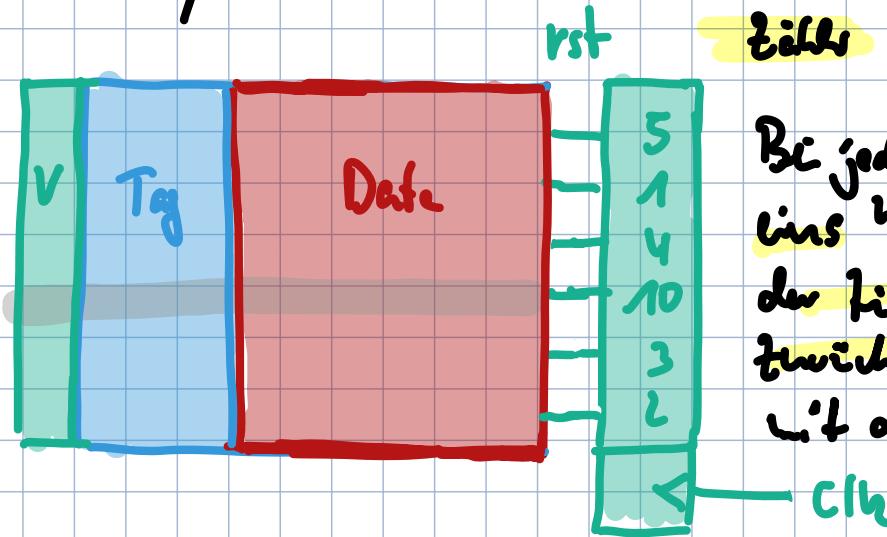
Zwei in Speicher an unterschiedlichen
Stellen liegende Objekte haben in
Cache an die gleiche Stelle.

cold start miss (compulsory)

Cache ist always leer.

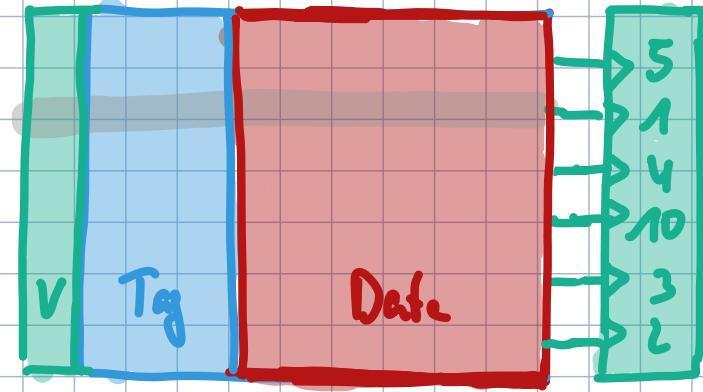
Ersatzstrategie

Least Recently Used (LRU): Die älteste Cache-line wird ersetzt.



Bei jedem Takt zählt jede Zelle eins hoch. Wenn der Schreiber der Zelle schreibt, dann erhöht er die Zelle zweimal. Ersatzt wird die Zelle mit den höchsten Wert.

Least Frequently Used (LFU): Die am wenigst benutzte Cache-line wird ersetzt.

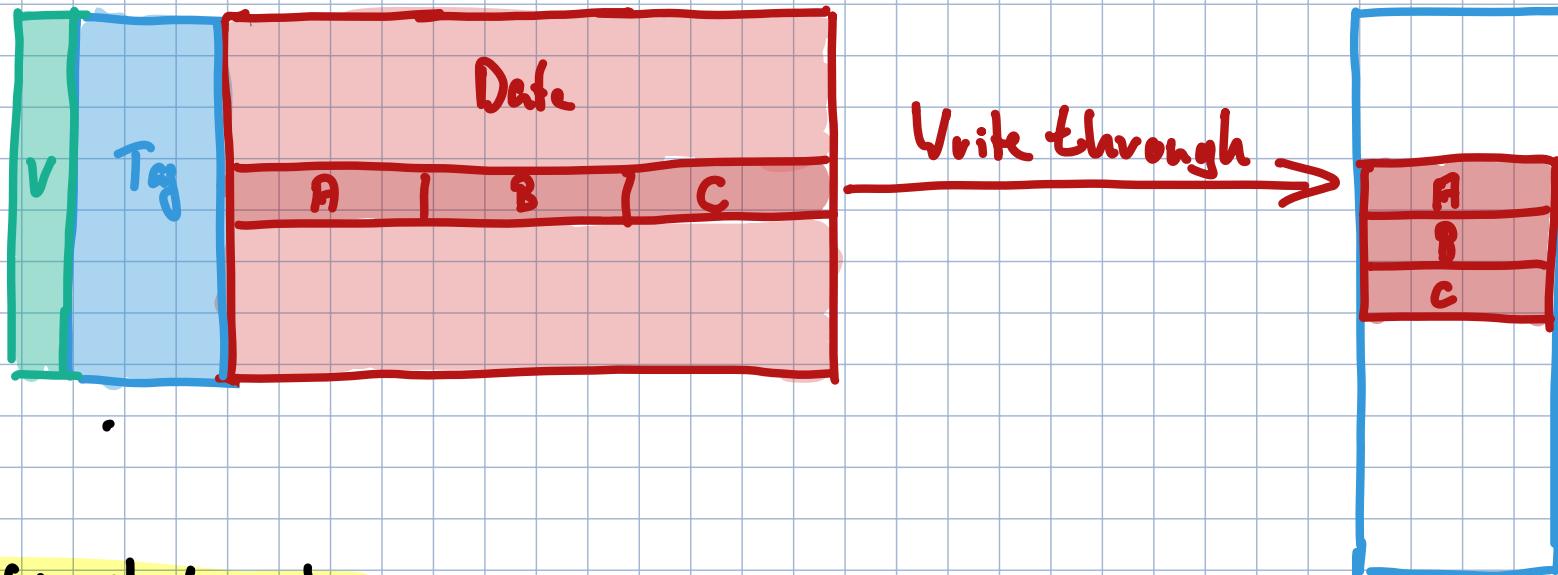


Bei jedem Zugriff zählt der Zähler einer Zelle hoch. Ersatz wird die Zelle mit den niedrigsten Wert.

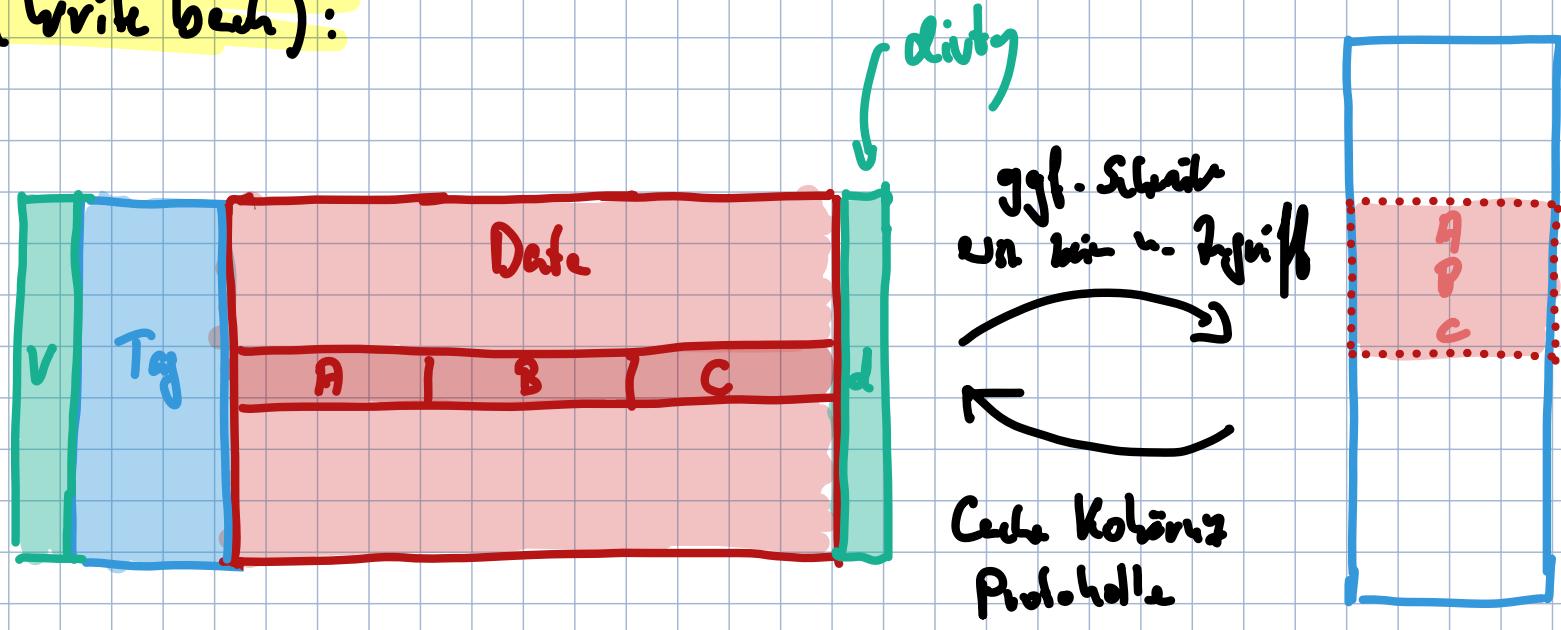
Schreiben der Daten (write allocate)

Durchgängiges Schreiben (write through):

write non-allocate: Cache Slink!
bei miss low!

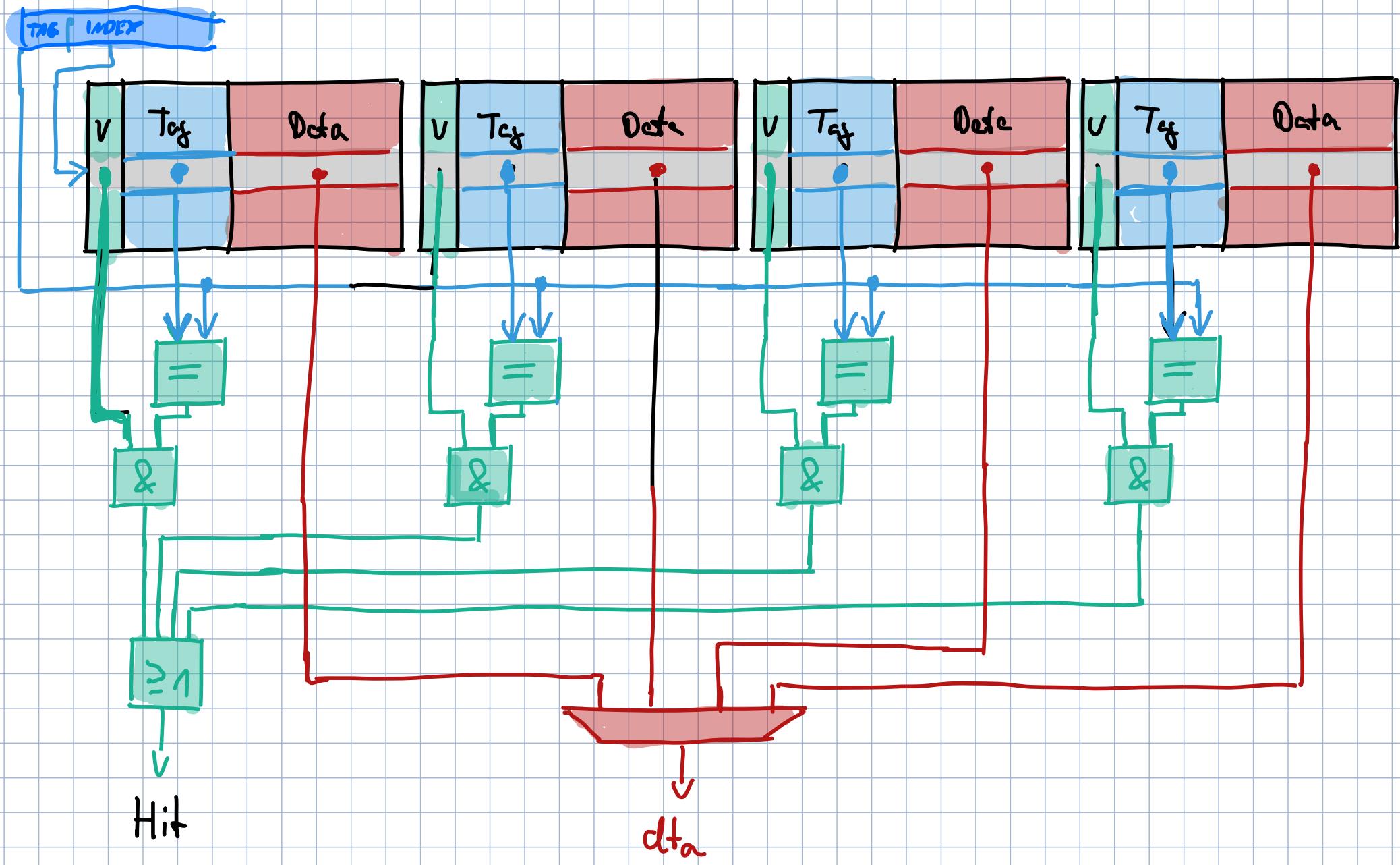


Zurückschreiben (write back):

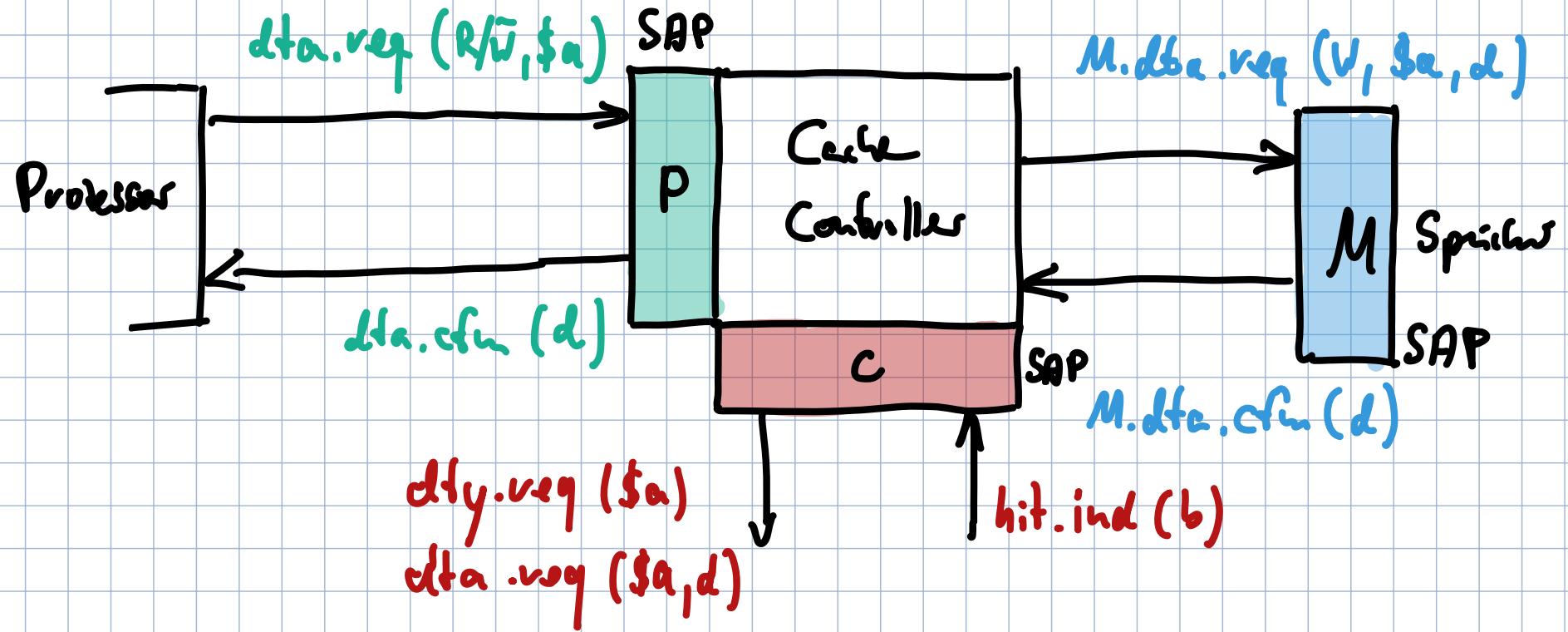


Viewfach - Disjunktiv - Cache

(4-Way Set Associative Cache)



Kommunikationsmodell (nach OSI) des Cache-Controllers



Service Access Point (SAP)

Cache Controller

Zustandsübergangstabelle

I P.dta.req (R, \$a)

P.dta.req (V, \$a, d)

HR

HV

HR h.ind (1)

h.ind (0)

P.dta.cfin (d)

M.dta.req (R, \$a)

I

A

HV h.ind (1)

h.ind (0)

C.dta.req (\$a)

C.dta.req (\$a, d)

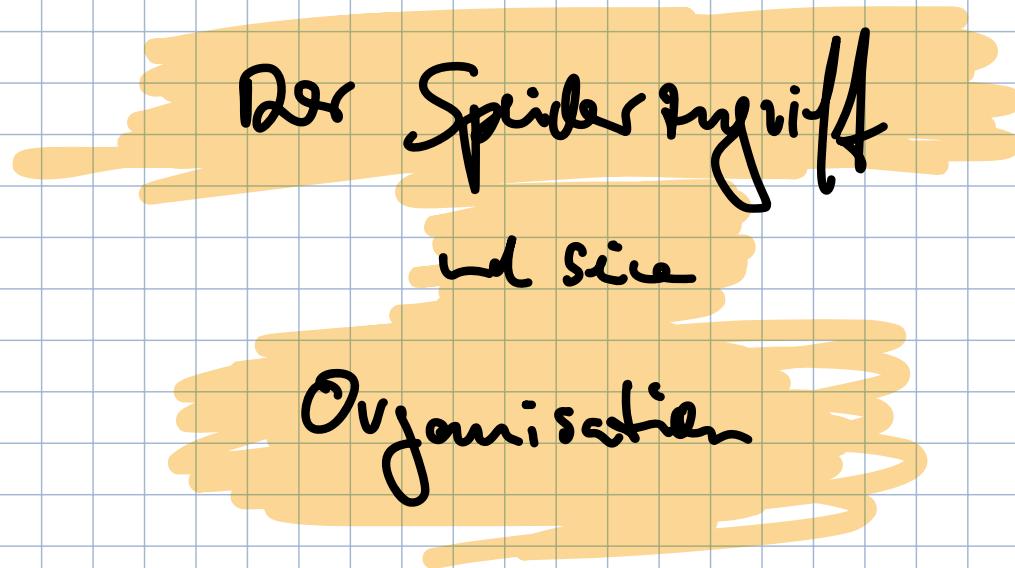
I

A M.dta.cfin (d)

C.dta.req (\$a, d)

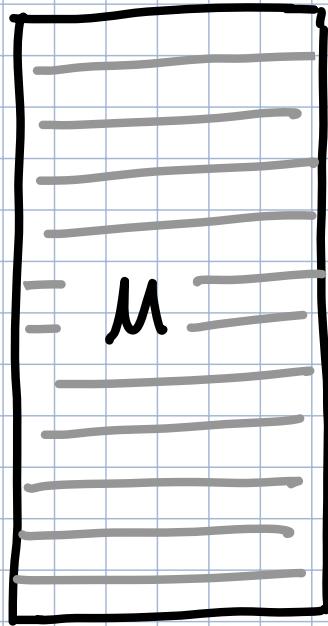
P.dta.cfin (d)

I



Speicherarchitektur: Wahlweise Speicher (Random Access Memory, RAM)

Ist Speicher nicht einfach ein Feld von Einträgen?

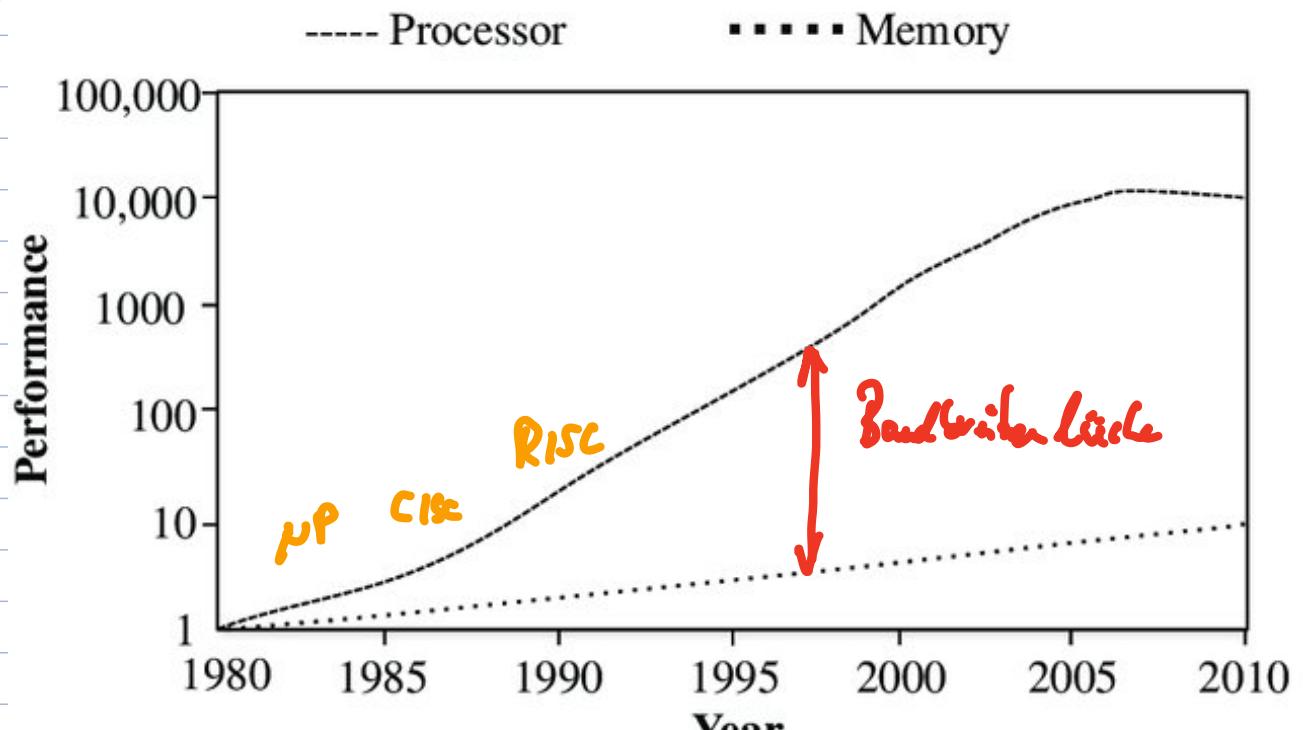


Feld (Array von Speicherplätzen)

$M[i]$

$O(n)$

(Memory) Index i ist die Adresse.



Reduzierung der Zugriffslücke durch Architektur.

Einsatz für die Lücke!

Bandbreitengrenzen

Wann ist die Zugriffssicherheit beschränkt?

- Technologische Gründe

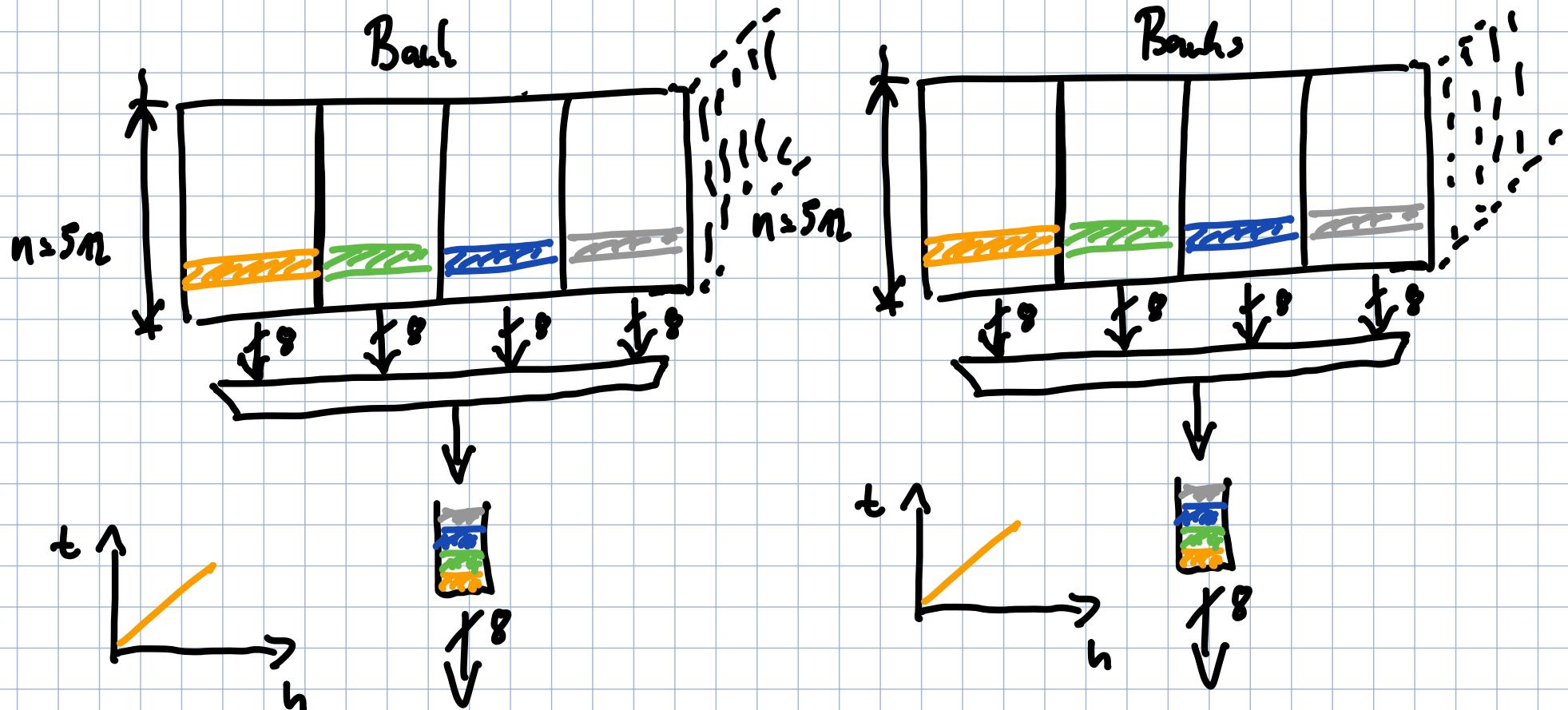
Zugriffsmögl. vs. Kosten

- Architektonische (Strukturelle) Gründe

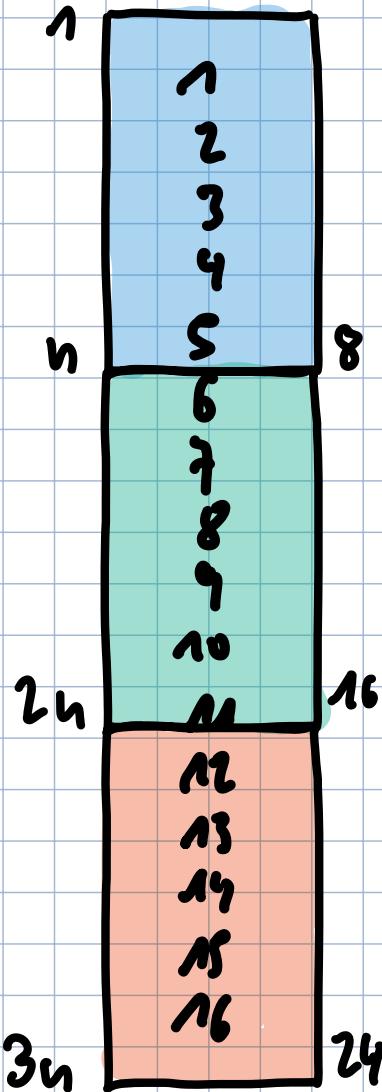
Gleichzeitige Anforderung von Daten

Gleichzeitiger Zugriff auf Program und Daten

Banking in DRAM : Streams

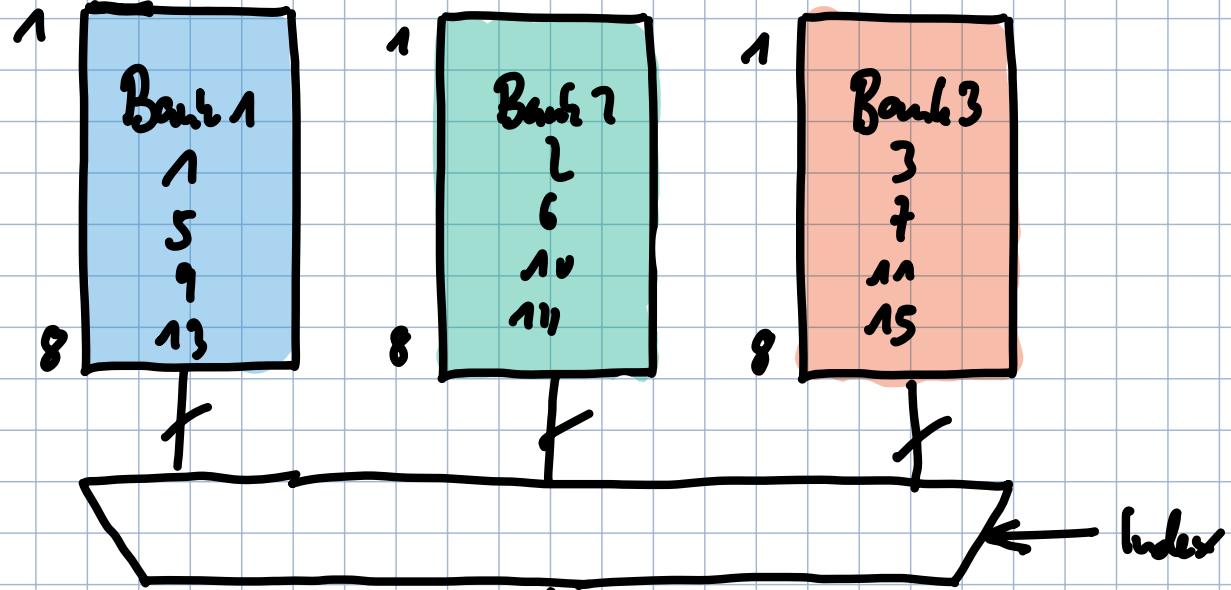


Zugriffssitz in Speicher



$$n = 8$$

Verschwücher Speicher (Chaining)



$$T = \frac{24(24+1)}{2} \ln(2)$$

$$= \underline{\underline{300 \ln(2)}} !$$

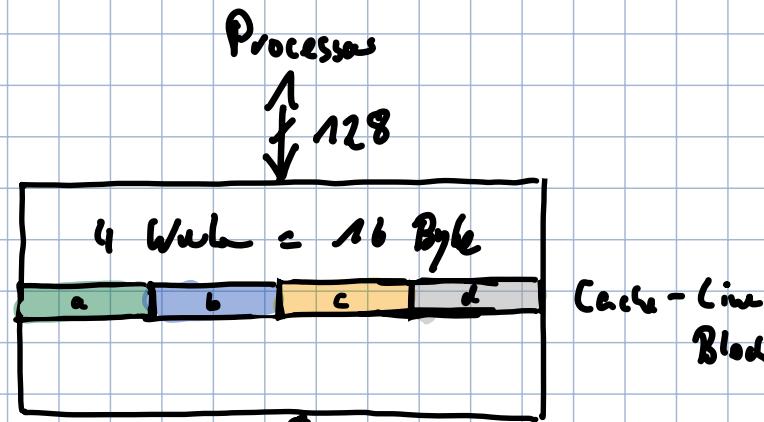
$$T = \frac{8(8+1)}{2} \ln(2)$$

$$= \underline{\underline{36 \ln(2)}}$$

1 Iter

Spieldurchlauflängen (lives leaving)

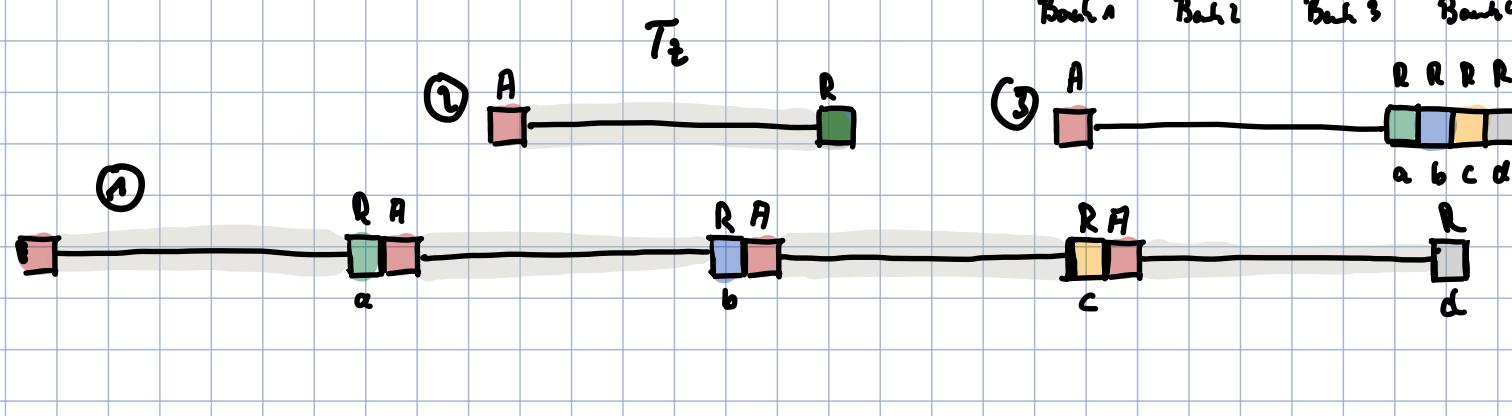
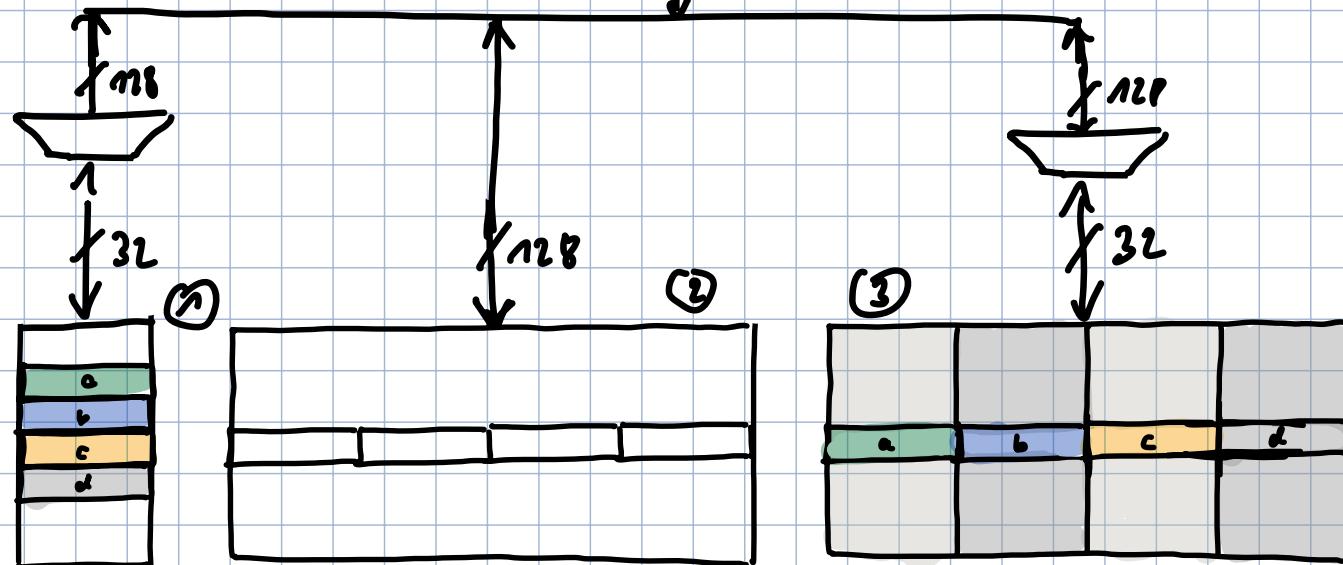
1 Wort = 4 Bytes



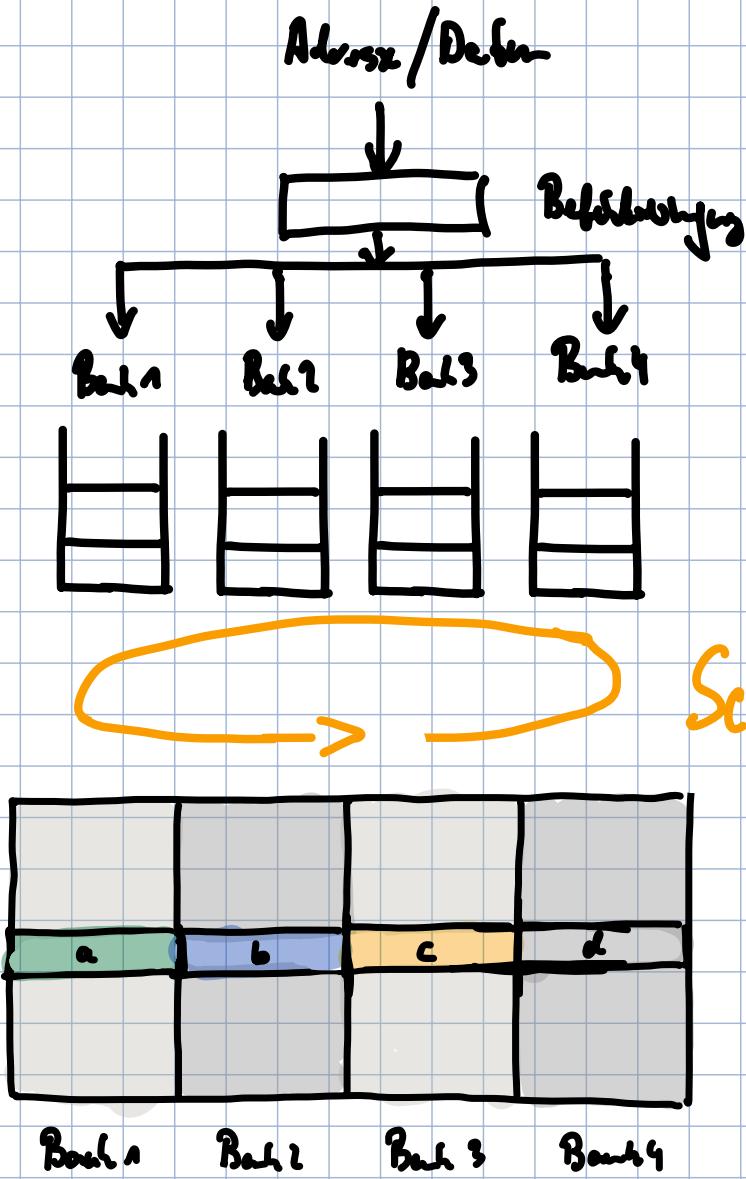
A Adresse anlegen

R Lese

T_2 Spield Zugriff



Speicherzugriffssortierung (Memory Control)



- Erhält Speicherbefehle
- Lesen / Schreiben / Refresh
- Verwaltet Datenstruktur

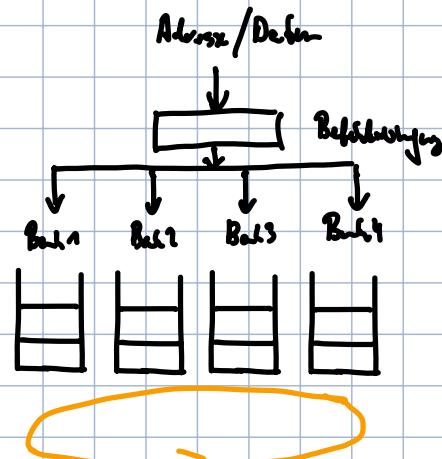
jede Bank ist ein physischer Speicherkanal

Schreiber

DRAM - Chip

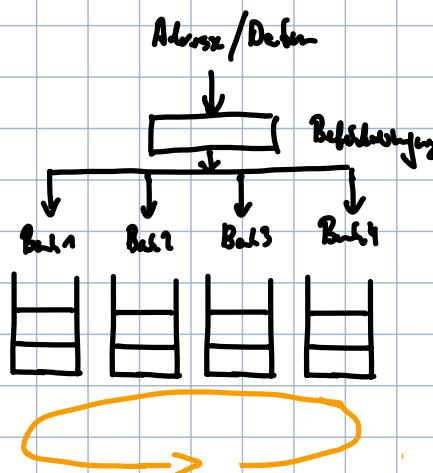
Logische Sprungbeschleunigung:

Kern 1



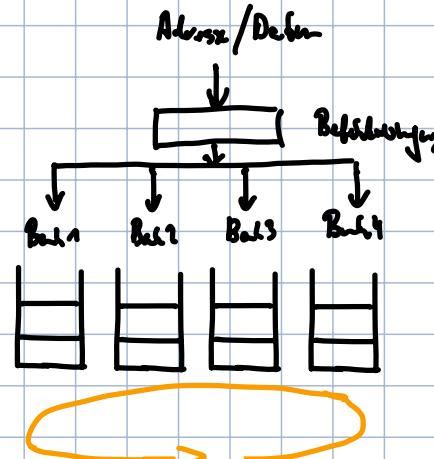
| | a | b | c | d |
|--------|---|---|---|---|
| Biel 1 | | | | |
| Biel 2 | | | | |
| Biel 3 | | | | |
| Biel 4 | | | | |

Kern 2



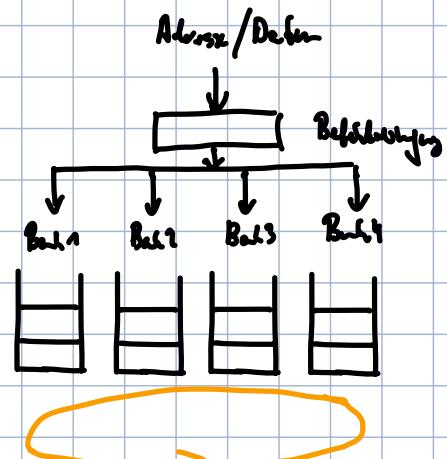
| | a | b | c | d |
|--------|---|---|---|---|
| Biel 1 | | | | |
| Biel 2 | | | | |
| Biel 3 | | | | |
| Biel 4 | | | | |

Kern 3



| | a | b | c | d |
|--------|---|---|---|---|
| Biel 1 | | | | |
| Biel 2 | | | | |
| Biel 3 | | | | |
| Biel 4 | | | | |

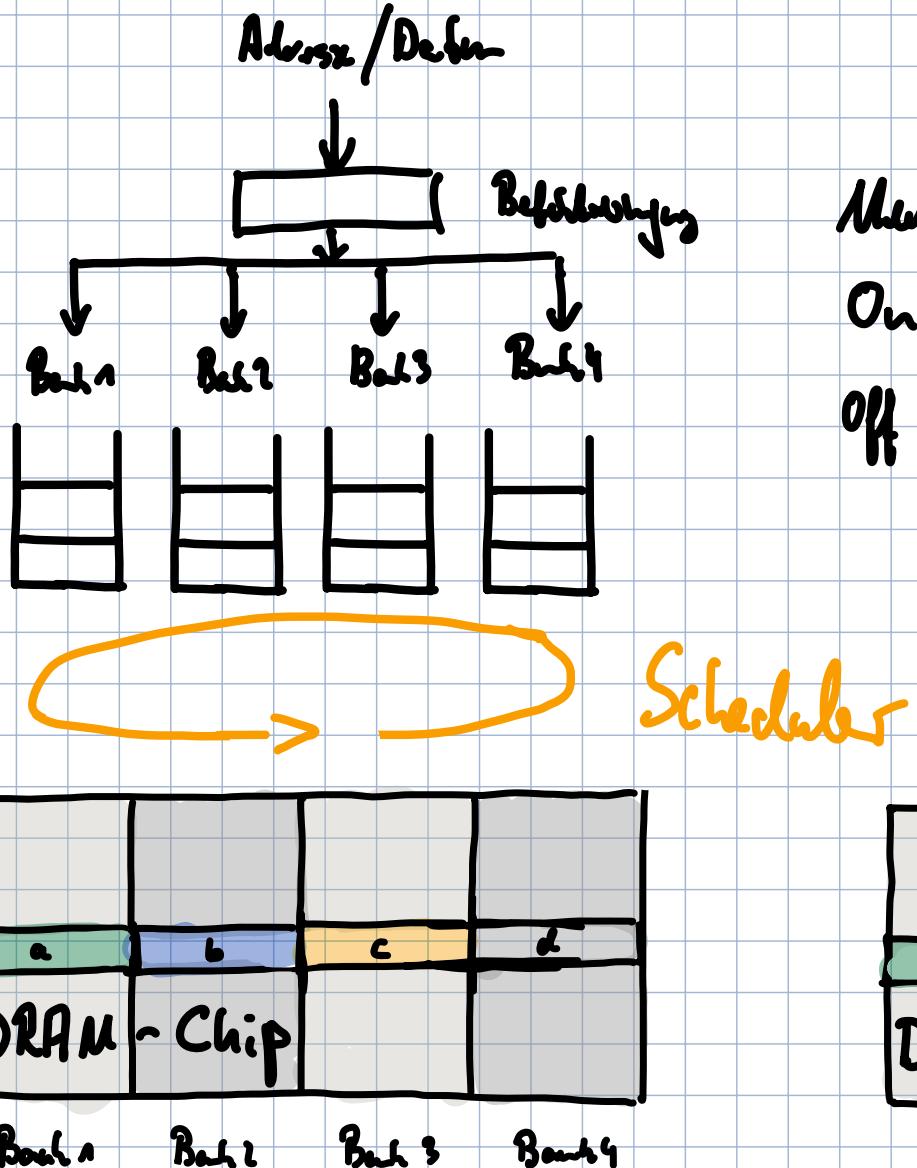
Kern 4



| | a | b | c | d |
|--------|---|---|---|---|
| Biel 1 | | | | |
| Biel 2 | | | | |
| Biel 3 | | | | |
| Biel 4 | | | | |

- Erhöht Speicherleistung
Lesen / Schreiben / Refreshen
- Kürzt Lesezeit Distanz

Das Banking kann erreicht werden durch Ranks.



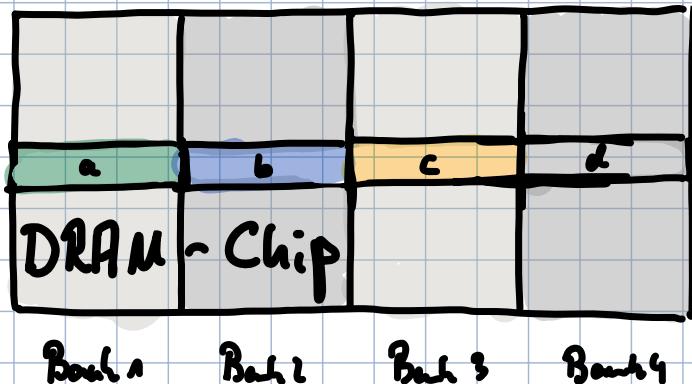
Mehr:

On Chip: Banking

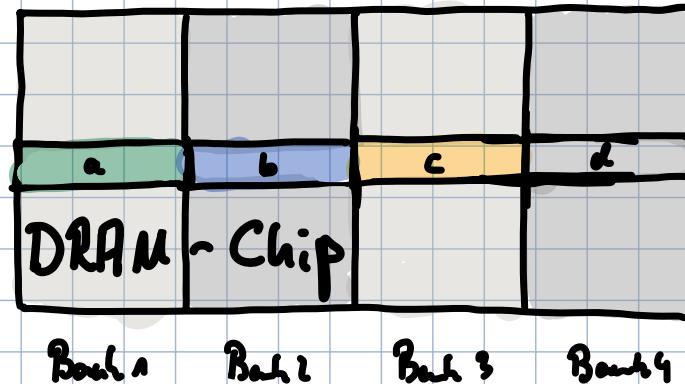
Off Chip : Rowycling

IC

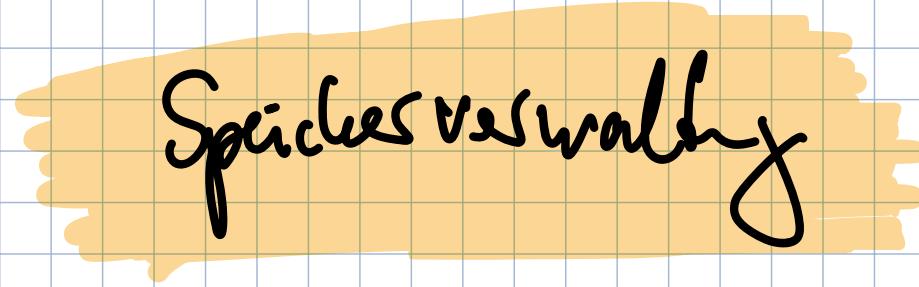
Platine



Rank 1



Rank 2



Speicher verwaltung

Speichermodell

Problem:

- i) Die Anzahl der Speicherplätze ist durch die Zugriffsszeit begrenzt:

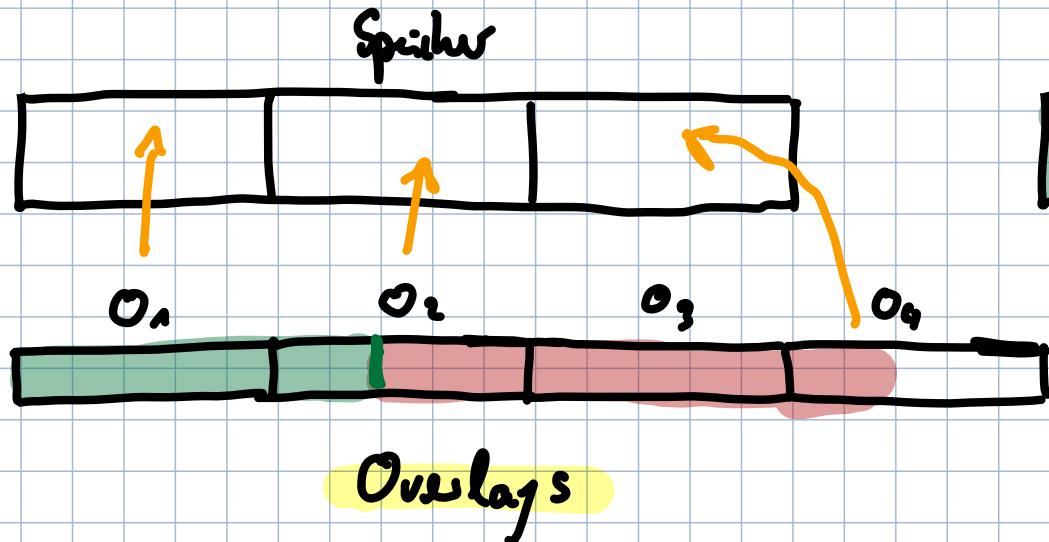
$$T = \frac{n(n+1)}{2} \ln(2)$$

- ii) Programme sind größer als der zu Verfügung stehende Platz im Hauptspeicher. Teile müssen daher ausgelagert werden.

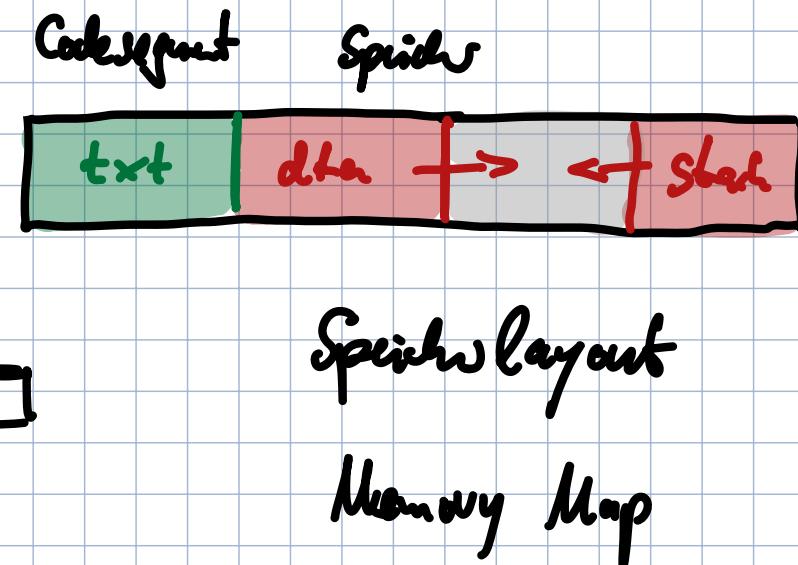
- iii) Im Hauptspeicher sollen gleichzeitig mehrere Programme vor gehalten werden. Zwischen den Programmen soll nahtlos umgeschaltet werden.

Speicherverwaltung: Overlays

Programm ist größer als der Speicher

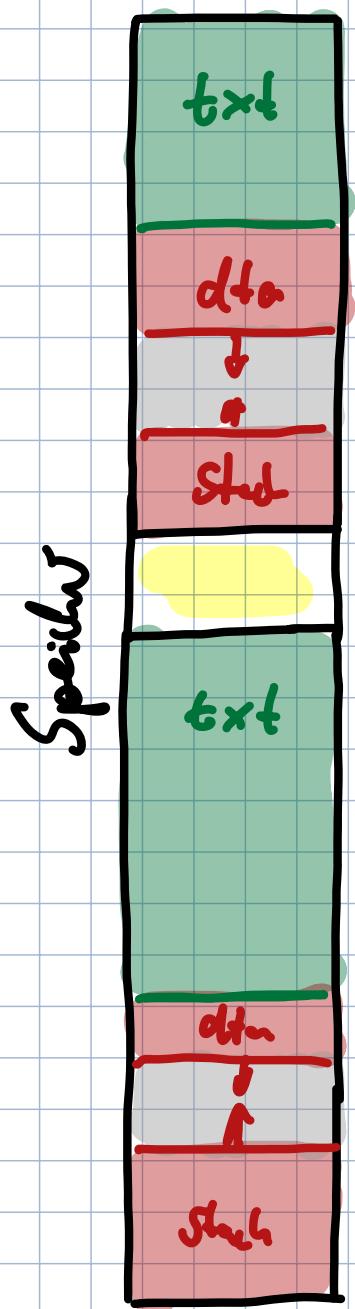


Programm ist kleiner als der Speicher



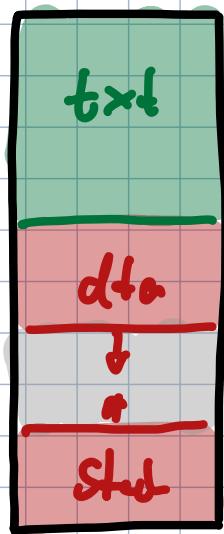
Overlays sind Programmteile, die als junks in der Speicher passen. Der Programmierer ist selbst für die Verwaltung zuständig.

Splittung verdeckt: Segmentation

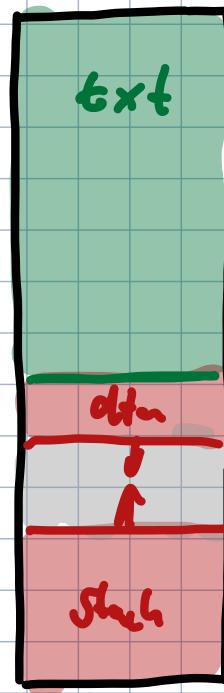


Segmente können von Betriebssystem verwaltet werden.
Allerdings besteht die Gefahr der Fragmentierung des Speichers.

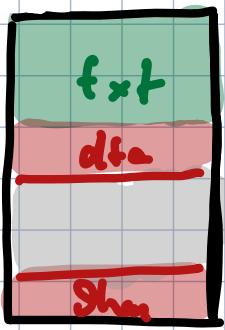
Program 1



Program 2



Program 3



Program
Heap

statische + dynamische Daten

Speicherverwaltung: Seiten



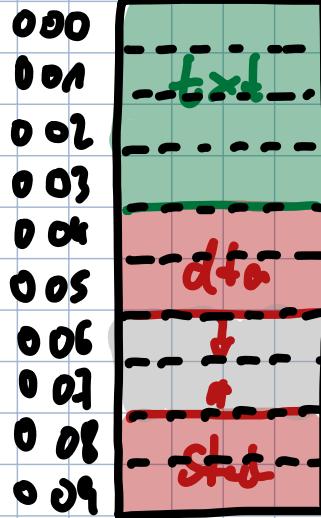
Blöcke
vs
Siten

Bsp. Programm 1

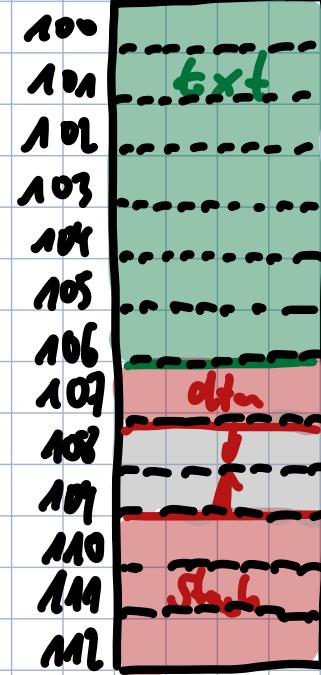
Verwaltung über Seiten-tabelle:

| | |
|-----|----|
| 000 | 10 |
| 001 | 1L |
| 002 | 18 |
| 003 | 1 |
| 004 | 15 |
| 005 | 5 |
| 006 | 13 |
| 007 | 2 |
| 008 | 3 |
| 009 | 24 |

Programm 1



Programm 2



Programm 3



Seitenummern

Aufbau eines **Virtuellen Adressrraumes!**

Programmadressen \neq Speicheradressen

Verwaltung durch **Hardware (MMU)**.

Sprachurvalhy: Virtueller Adressraum:

Physische Adressraum



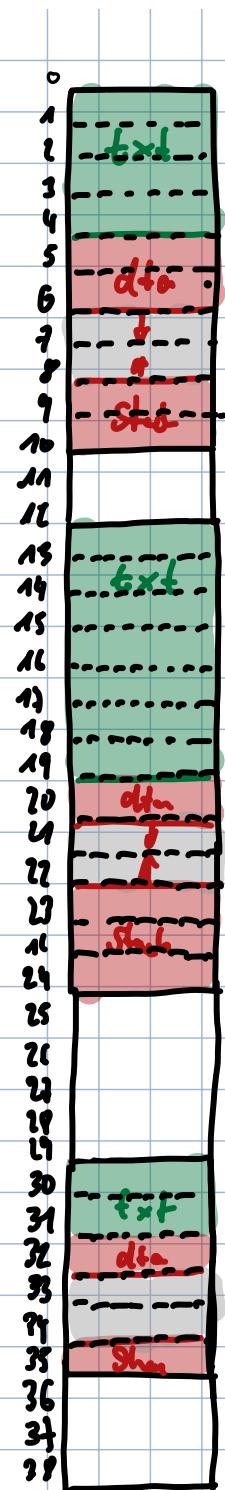
Sichturvalhy

Seiten-tabelle

| | |
|----|----|
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| | : |
| 20 | 5 |
| 21 | 10 |
| 22 | 6 |
| 23 | 7 |

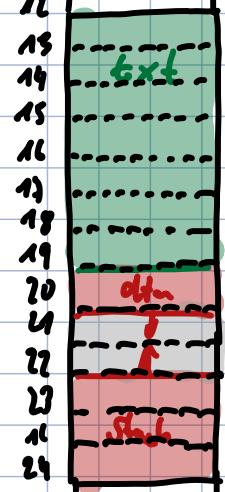
Segment-tabelle

| | Seit | Einträge |
|-------|------|----------|
| S_1 | 1 | 9 |
| S_2 | 13 | 13 |
| S_3 | 30 | 4 |



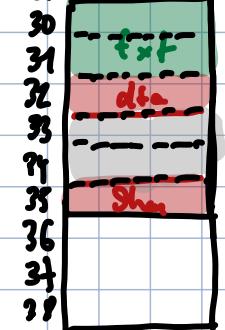
S_1

Program 1



S_2

Program 2

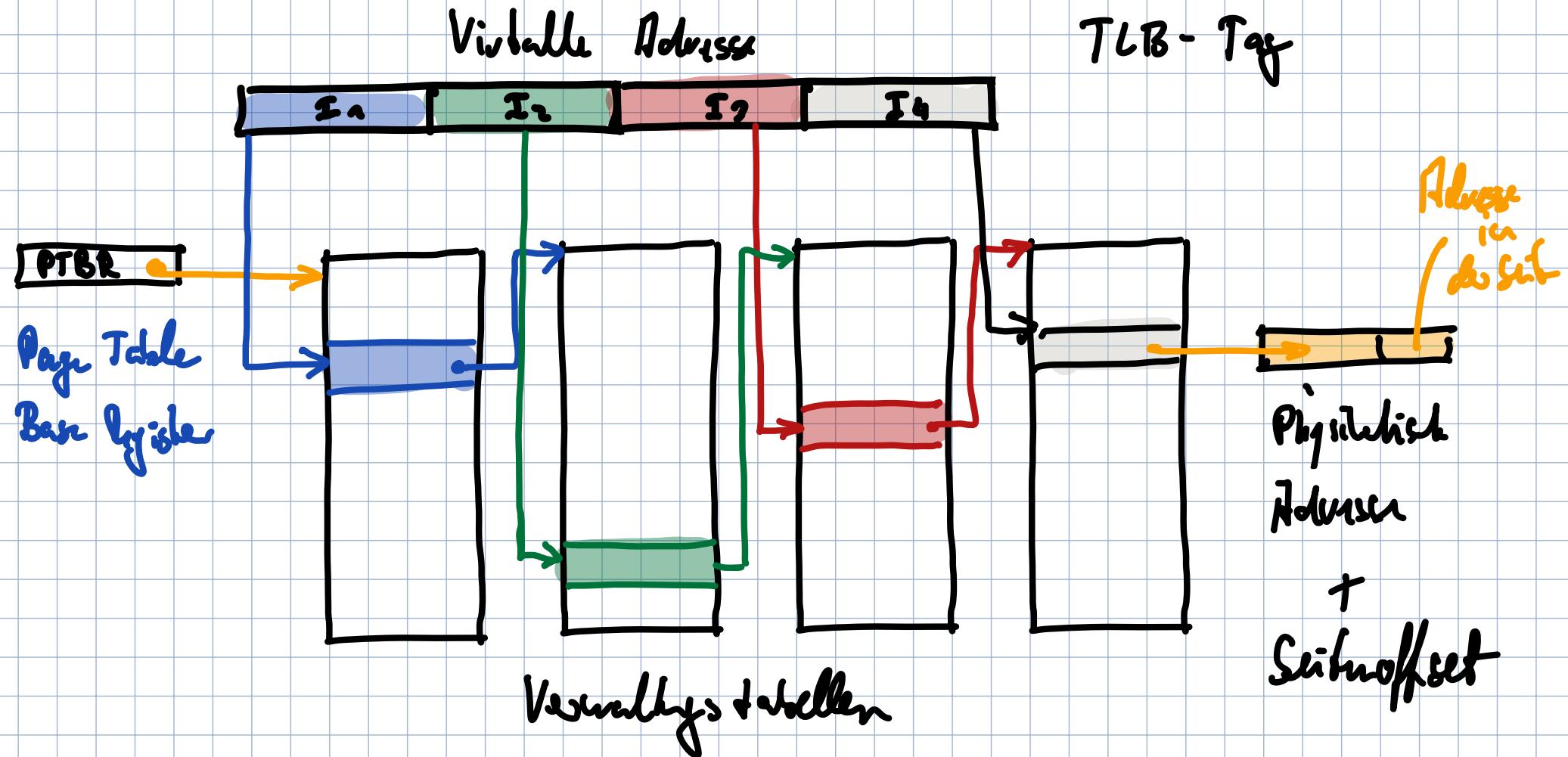


S_3

Program 3

Virtueller
Adressraum

Speicherverwaltung: Mehrstufige Seitenverwaltung



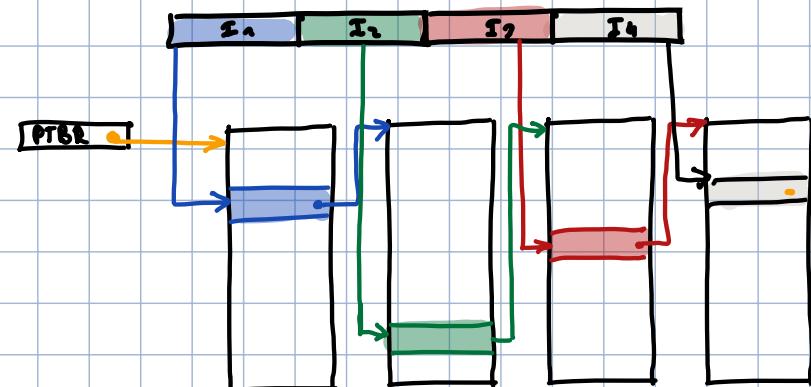
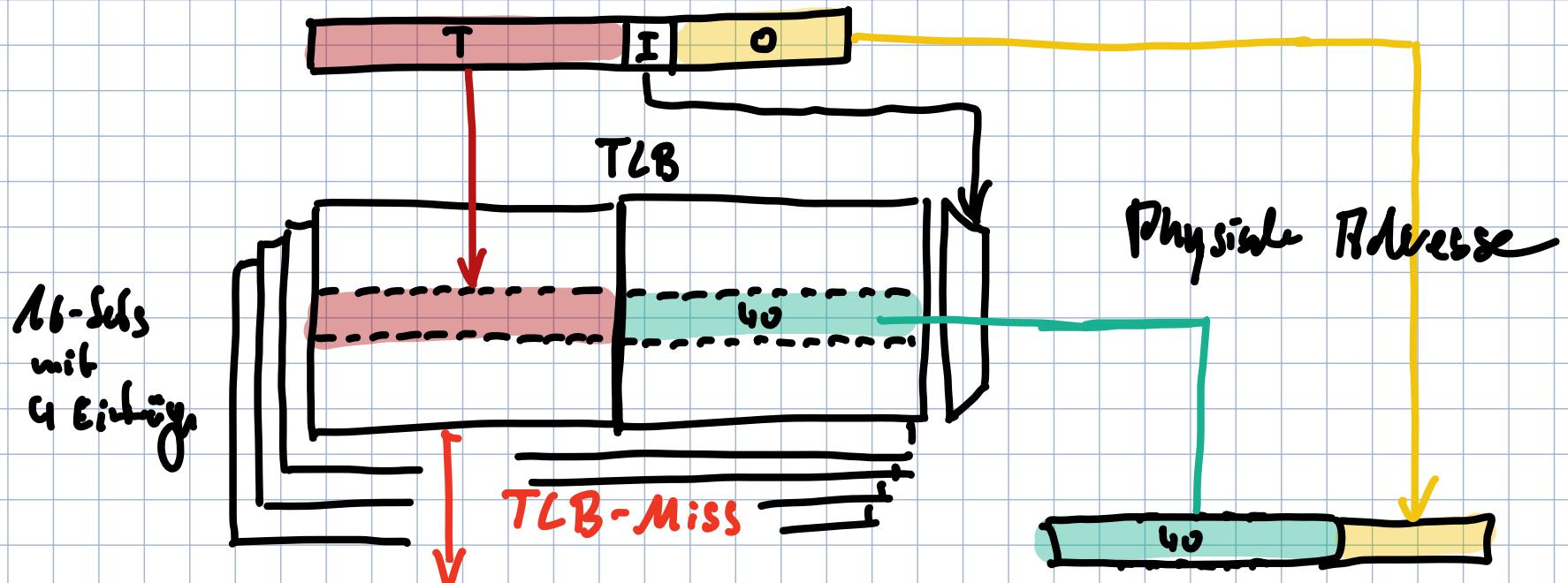
Die Seitenverwaltung besteht aus einer mehrstufigen Tabelle. Ein Eintrag in der virtuellen Adresse weist quindi auf die nächst Tabelle.

Speicherwaltung: Hardware zur Umsetzung der Seitenadressen

MMU

Memory Management Unit

Virtuelle Adresse



Seitenstabell im Speicher

Translation Look Aside
Buffer (TLB)
Cache für die Speicher-
waltung.