Übung 5

Karlsruher Institut für Technologie

Cache Speicher

- Speicherhierarchie
- Aufbau und Funktionsweise
- Aufgaben

Speicherhierarchie

Ein technologisch einheitlicher Speicher mit ***kurzer Zugriffszeit*** und ***großer Kapazität*** ist aus ***Kostengründen*** i. A. nicht realisierbar

Lösung:

Schichtenweise Anordnung verschiedener Speicher und Verschiebung der Information zwischen den Schichten (Speicherhierarchie)

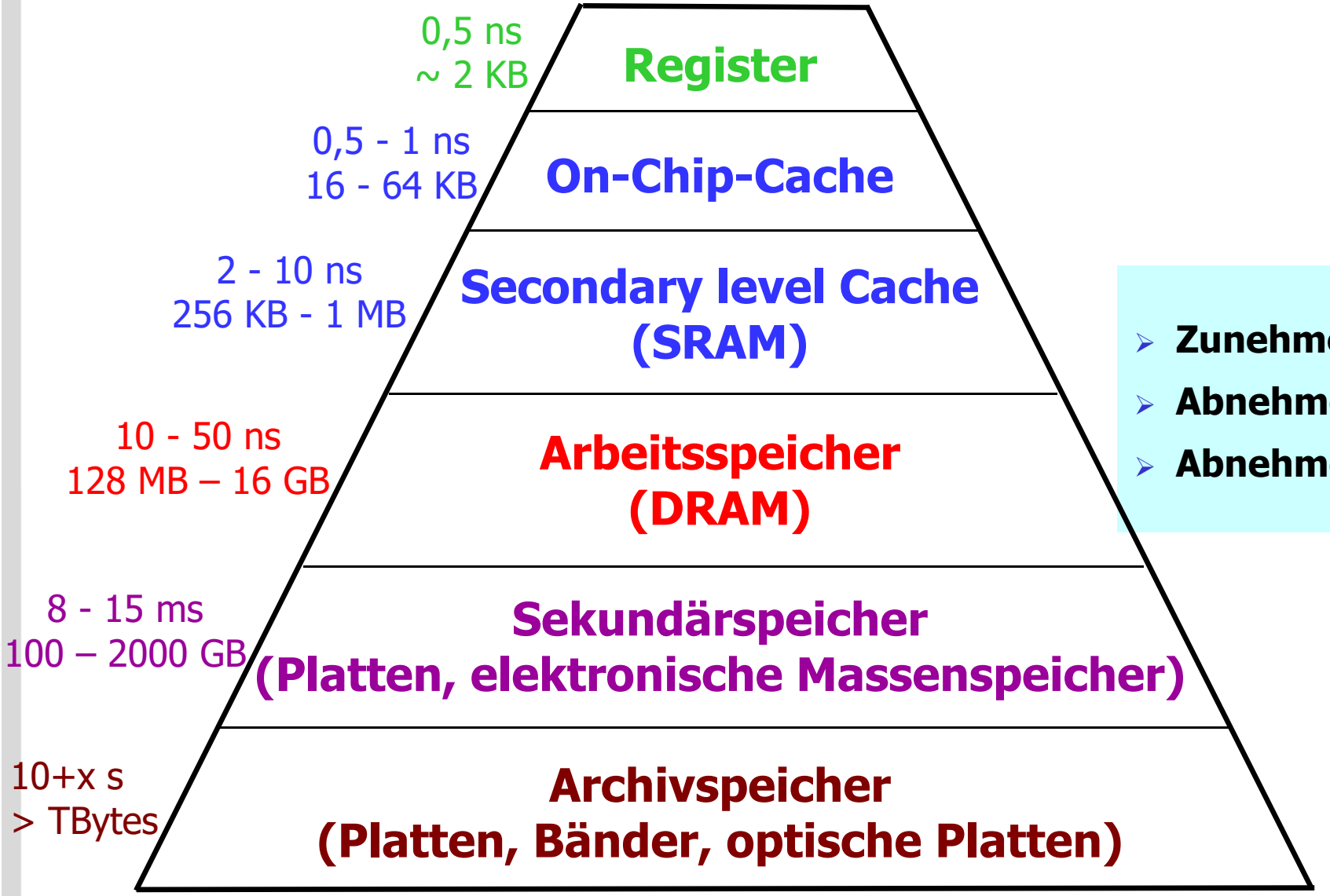
Speicherhierarchie

Speicherhierarchie zum Ausgleich der unterschiedlichen Zugriffszeiten der CPU und des Hauptspeichers.

2 Strategien:

- **Cache-Speicher:**
  Kurze Zugriffszeiten
  ➔ Beschleunigung des Prozessorzugriffs
- **Virtueller Speicher:**
  Vergrößerung des tatsächlich vorhandenen Hauptspeichers (z.B. bei gleichzeitiger Bearbeitung mehrerer Prozesse)

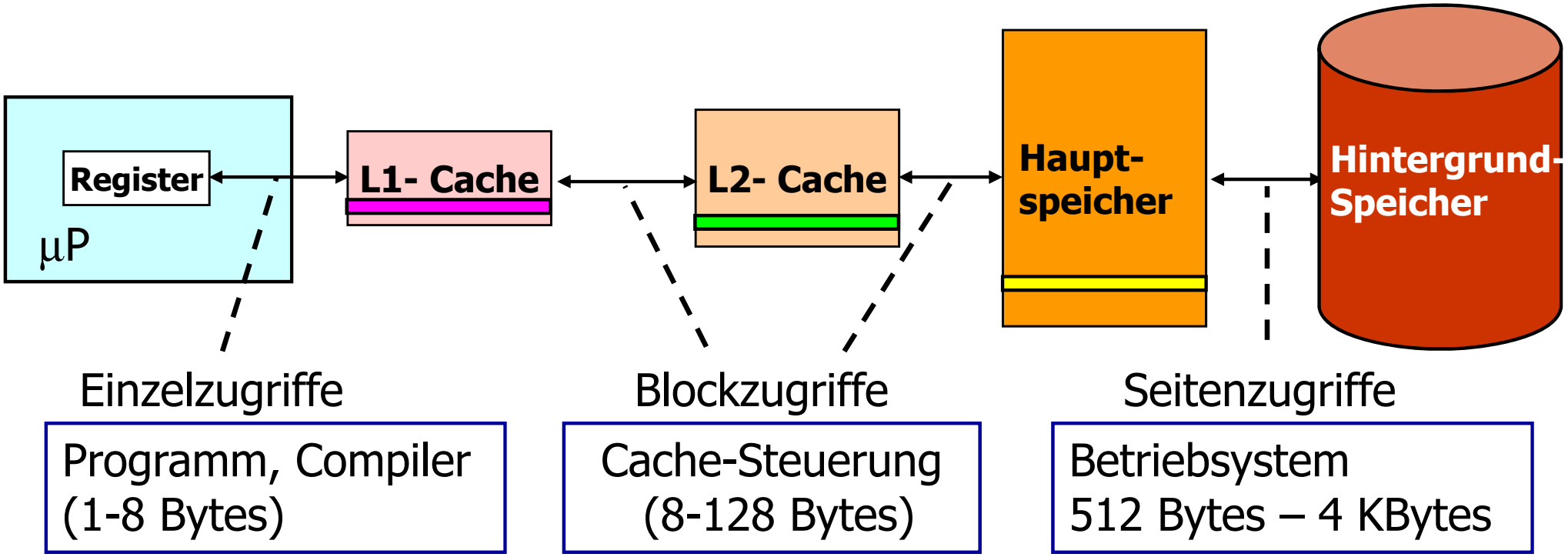
Speicherhierarchie



- Zunehmende Kosten/Byte
- Abnehmende Kapazität
- Abnehmende Zugriffszeit

Speicherhierarchie

Daten werden nur zwischen aufeinanderfolgenden Ebenen der Speicherhierarchie kopiert.

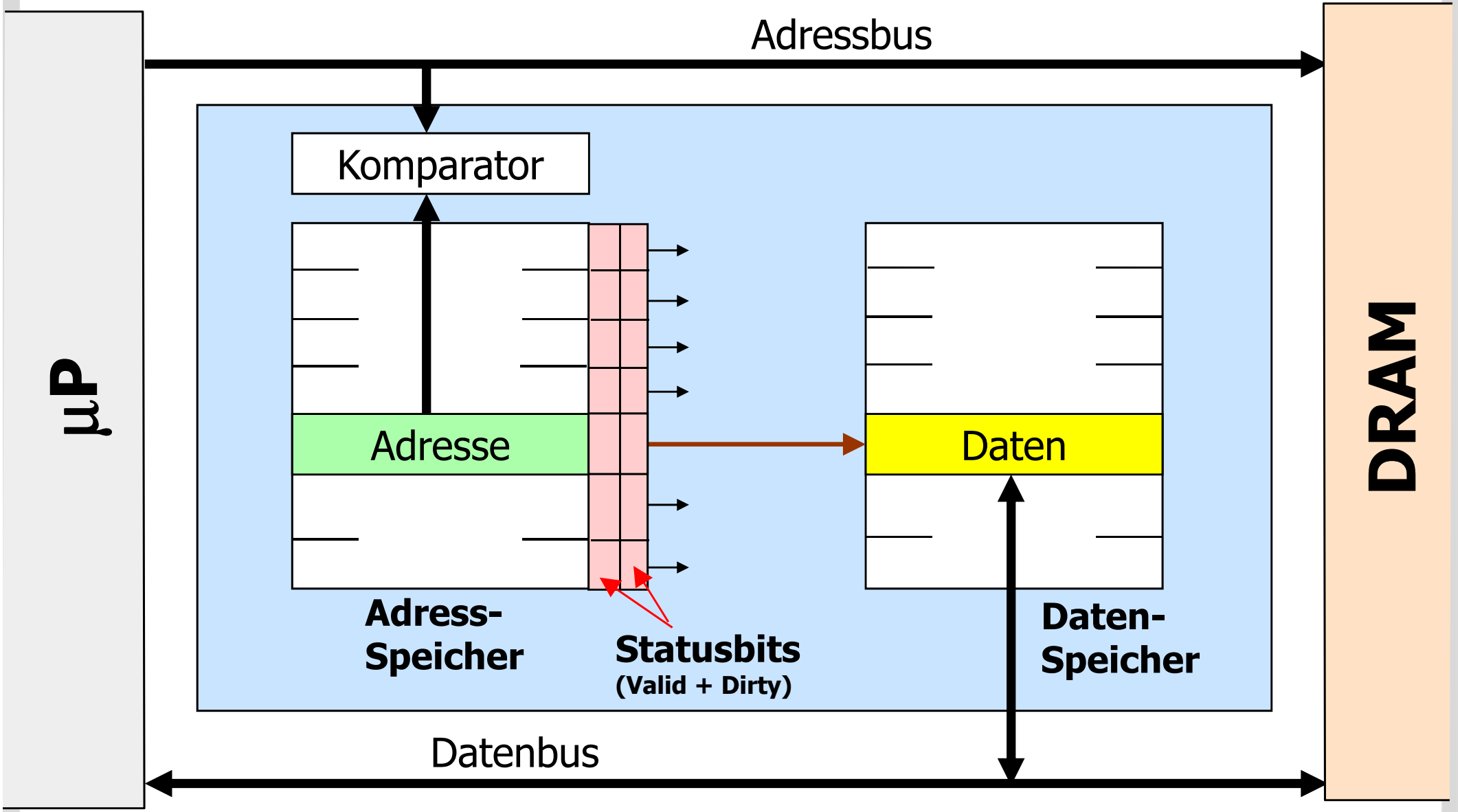


Wieso kommt es zur einer Leistungssteigerung?

Ein CPU-Cache-Speicher bezieht seine Effizienz im wesentlichen aus der **Lokalitätseigenschaft** von Programmen (locality of reference), d. h. es werden bestimmte Speicherzellen bevorzugt und wiederholt angesprochen (z. B. Programmschleifen)

- **Zeitliche Lokalität:** Die Information, die in naher Zukunft angesprochen wird, ist mit großer Wahrscheinlichkeit schon früher einmal angesprochen worden (Schleifen).
- **Örtliche Lokalität:** Ein zukünftiger Zugriff wird mit großer Wahrscheinlichkeit in der Nähe des bisherigen Zugriffs liegen (Tabellen, Arrays).

Aufbau eines Cache-Speichers



Arbeitsweise eines Cache-Speichers

- Cache-Steuerung prüft, ob
  - Der zur Speicheradresse gehörende Hauptspeicherinhalt als Kopie im Cache steht **(Bedingung 1)** und
  - Dieser Cache-Eintrag durch das Gültigkeits-Bit (Valid-Bit) als gültig gekennzeichnet ist **(Bedingung 2)**
- Prüfung führt zu einem Cache-Treffer oder zu einem Fehlzugriff.
- Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.

Arbeitsweise eines Cache-Speichers

- Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.

Lesezugriffe (read miss)

- Lesen des Datums aus dem Hauptspeicher und Laden des Cache-Speichers
- Kennzeichnen der Cache-Eintrag als gültig (V-Bit setzen)
- Speichern der Adressinformation im Adress-Speicher des Cache-Speichers

Arbeitsweise eines Cache-Speichers

- Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.

Schreibzugriffe (write miss)

Aktualisierungsstrategie bestimmt, ob

- der entsprechende Block in den Cache geladen und dann mit dem zu schreibenden Datum aktualisiert wird oder, ob
- nur der Cache aktualisiert wird und der Hauptspeicher unverändert bleibt

Arbeitsweise eines Cache-Speichers

- Cache-Treffer (Cache-hit, read hit, write hit):
  - Beide Bedingungen 1 und 2 sind erfüllt
  - Read-Hit: Cache-Datum → CPU
  - Write-Hit:
    - Write-through: CPU-Datum → Cache, Speicher
    - Write-back: CPU-Datum → Cache & D = 1

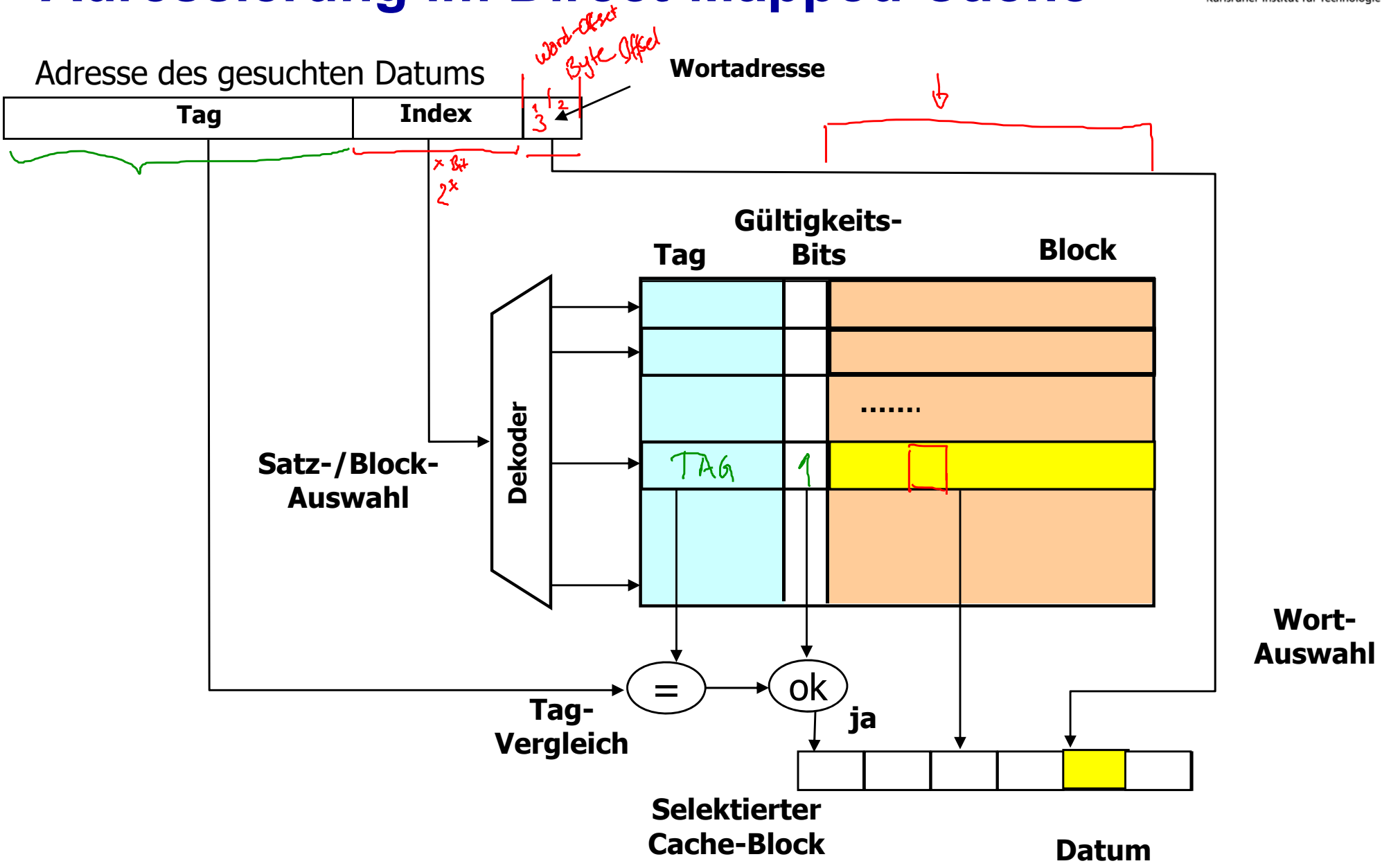
Cache-Strukturen

Wie wird festgestellt, ob die benötigten Daten im Cache sind und falls ja wie können diese gefunden werden?

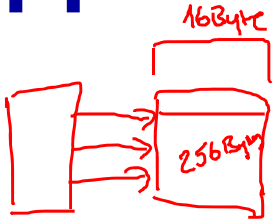
3 Techniken für den Adressvergleich ➔ 3 Cache-Typen:

- **Fully Associative Cache**
- **Direct Mapped Cache**
- **n-Way Set Associative Cache**

Adressierung im Direct-Mapped Cache



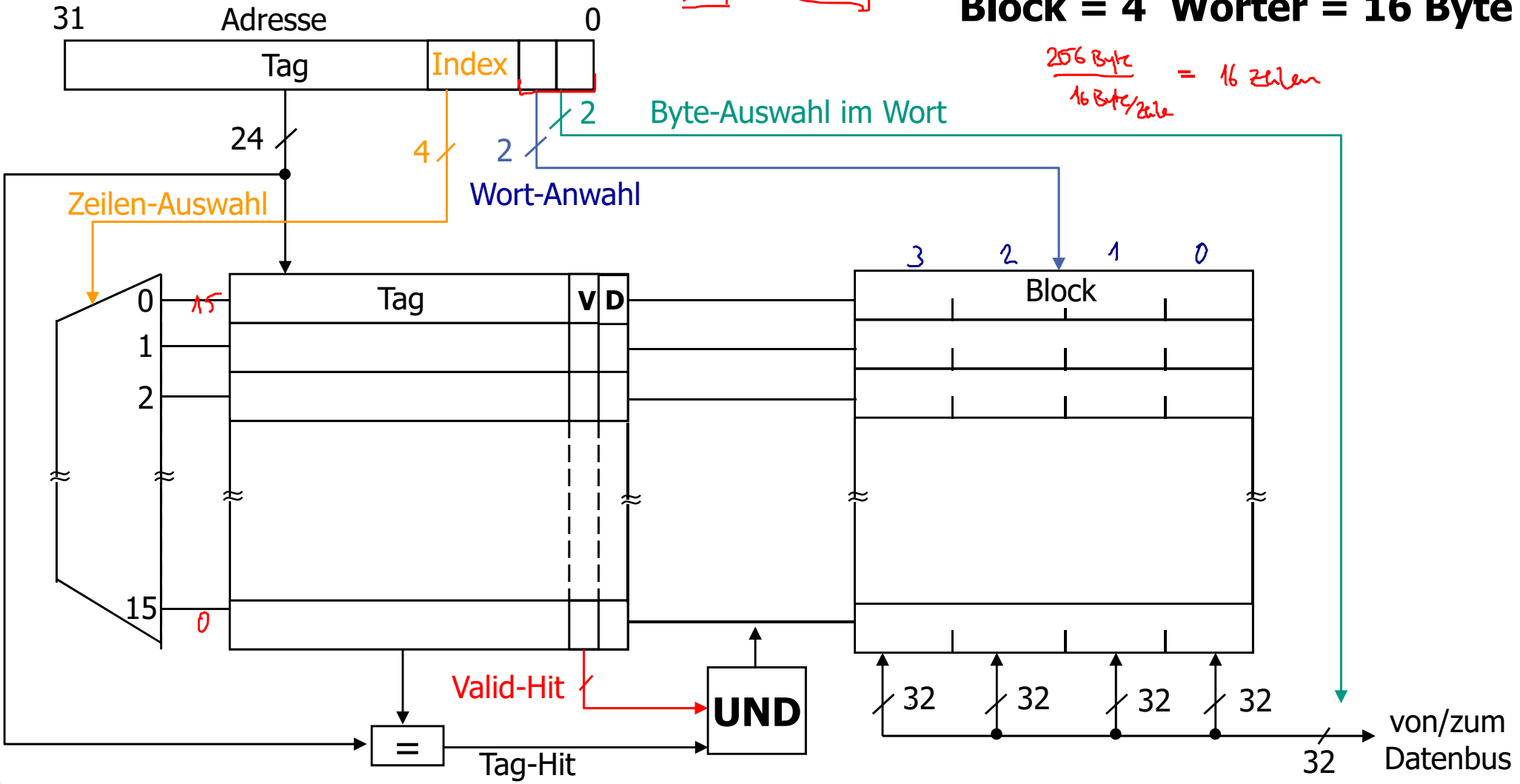
Beispiel: Direct-mapped-Cache



Kapazität: 256 Byte

Block = 4 Wörter = 16 Byte

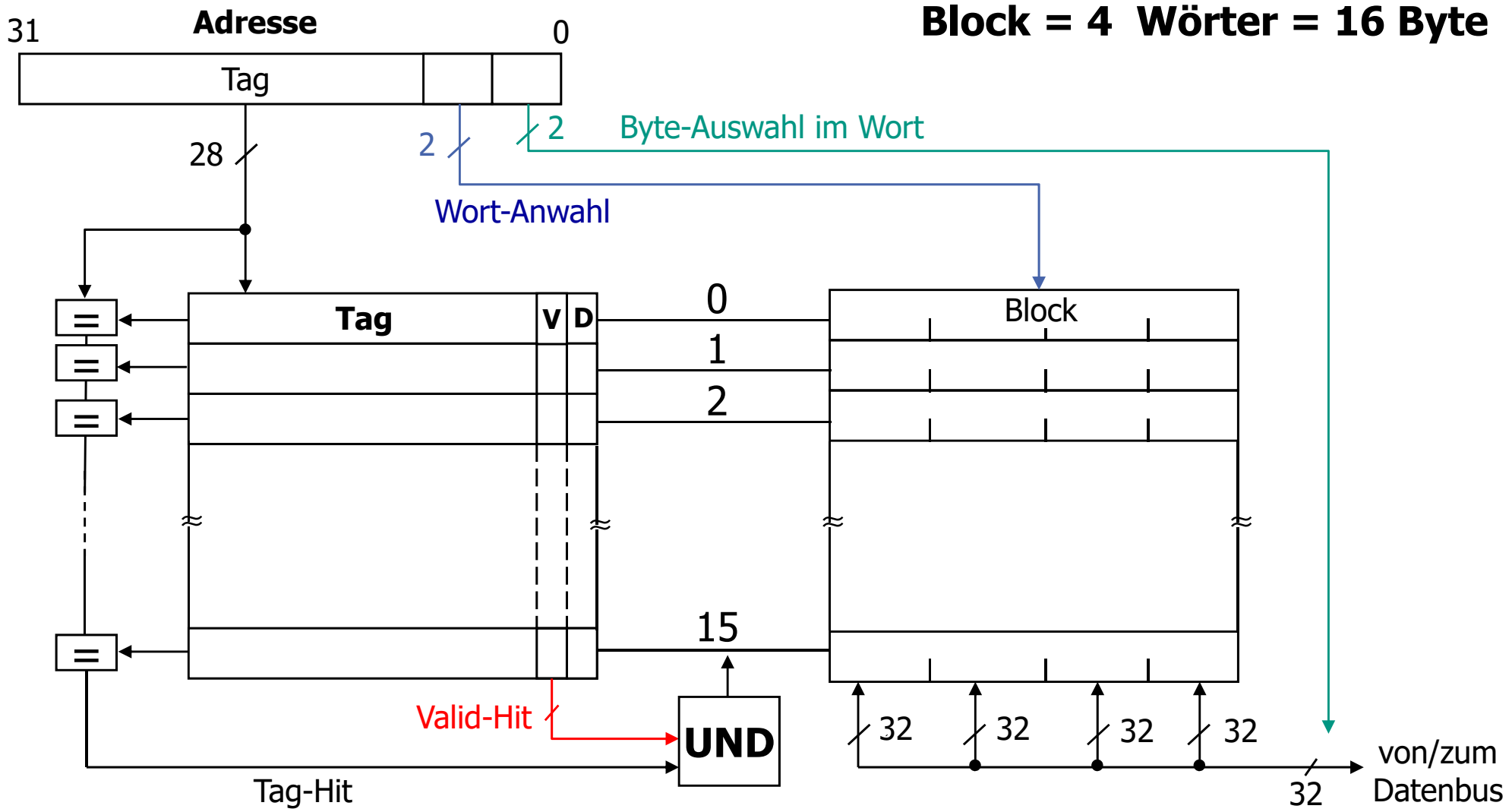
256 Byte / 16 Byte/Zeile = 16 Zeilen



Beispiel: Vollassoziativer Cache

Kapazität: 256 Byte

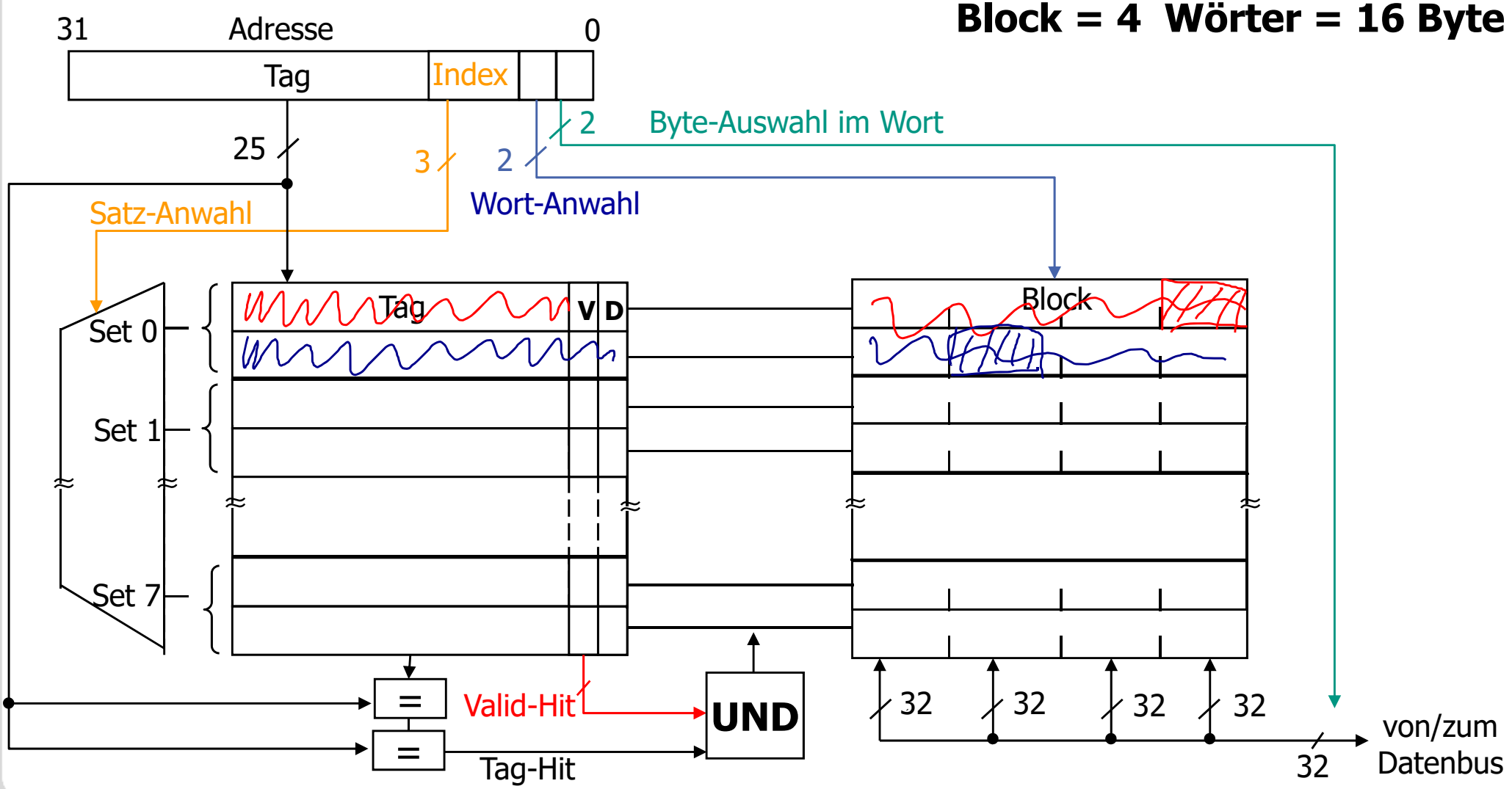
Block = 4 Wörter = 16 Byte



Beispiel: 2-fach satzassoziativer Cache

Kapazität: 256 Byte

Block = 4 Wörter = 16 Byte



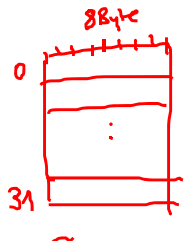
Aufgabe 1

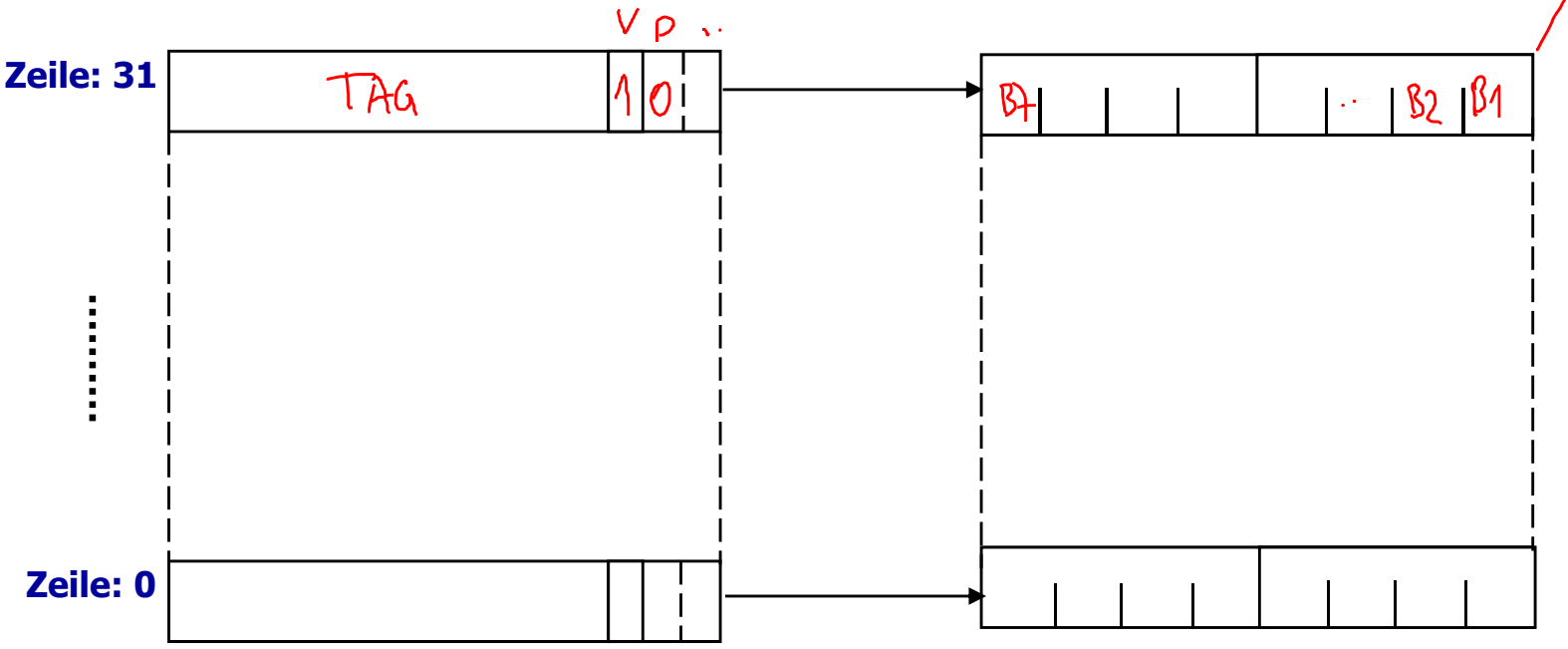
In einem Mikroprozessorsystem mit 32-bit-Datenzugriff auf den Hauptspeicher ist ein Daten-Cache vorhanden. Das Laden des Caches erfolgt in Blöcken von je **acht** Bytes, d. h. von zwei Wörtern. Die Hauptspeicheradresse umfasst 24 Bits; die Cache-Kapazität beträgt **256** Bytes.

1. Geben Sie die Anzahl der Cache-Zeilen an und skizzieren Sie die Unterteilung der Hauptspeicheradresse für einen *direct-mapped*-Cache (DM) , vollassoziativen Cache (AV) und 4-way-set-assoziativ-Cache (A4).

Lösung 1.1

$$\text{Cache-Zeilen} = \frac{\text{Cache-Kapazität}}{\text{Blockgröße}} = \frac{256}{8} = 32 \text{ Zeilen}$$

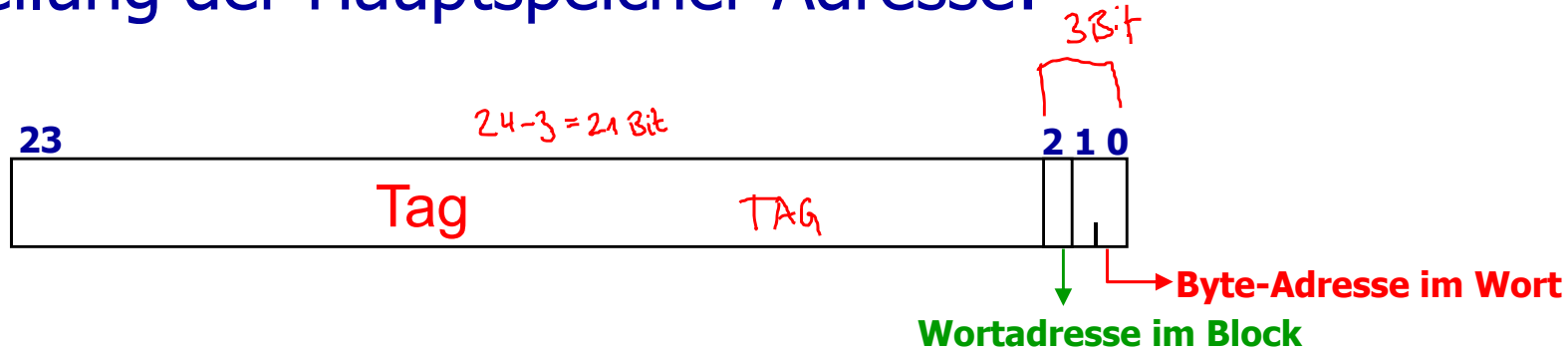




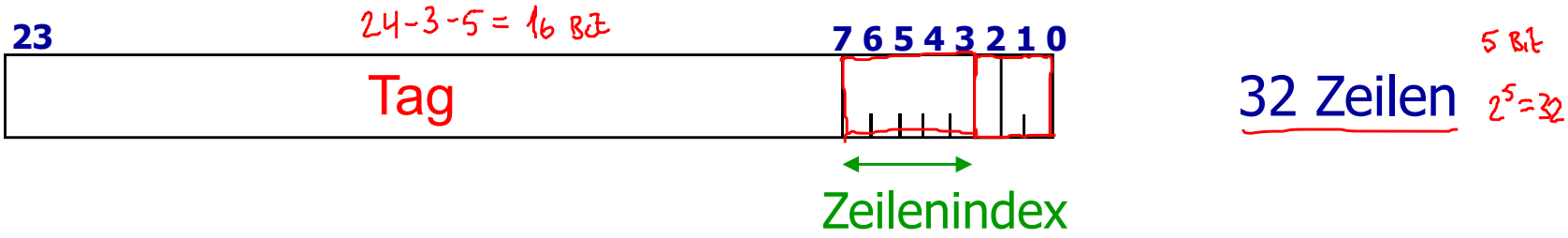
Lösung 1.1

Unterteilung der Hauptspeicher-Adresse:

AV

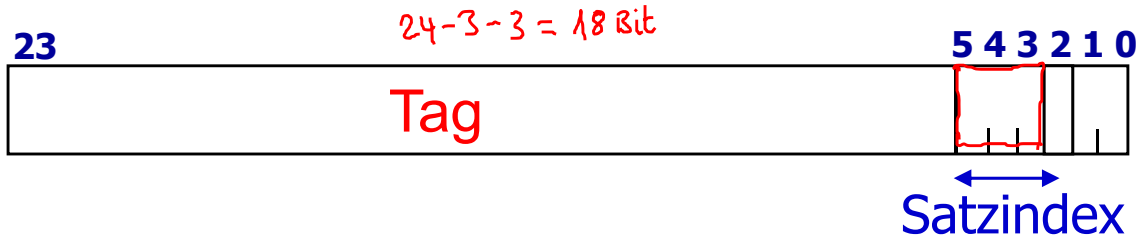


DM



32 Zeilen

A4



$\frac{32}{4} = 8$ Sätze

32 Zeilen → 8 Sätze

$2^3 = 8$

Aufgabe 1.2

2. Geben Sie für die drei Cachespeicher die Anzahl der benötigten Vergleicher und die zu vergleichende Bitzahl an.

Anzahl der zu vergleichenden Bits = Tag-Länge

32	Vergleicher	Tag-Länge
VA	32	24 - 3 = 21
DM	1	24 – 8 = 16
A4	4	24 – 6 = 18

Aufgabe 1.3

3. Welche Zeilen sind bei diesen Cachespeichern hinsichtlich der Blockersetzung als zusammengefasst zu betrachten, z. B. für einen Alterungsmechanismus nach dem LRU-Prinzip. Least Recently Used = LRU

- **VA:** Zeile 0 bis 31
- **DM:** nur die durch den Zeilenindex ausgewählte Zeile
- **A4:** jeweils die 4 Zeilen des durch den Satzindex angewählten Satzes

Aufgabe 2

In die drei Cachespeicher der Aufgabe 1 sind drei Hauptspeicherblöcke mit den folgenden in hexadezimaler Schreibweise angegebenen Adressen in der angeführten Reihenfolge zu laden:

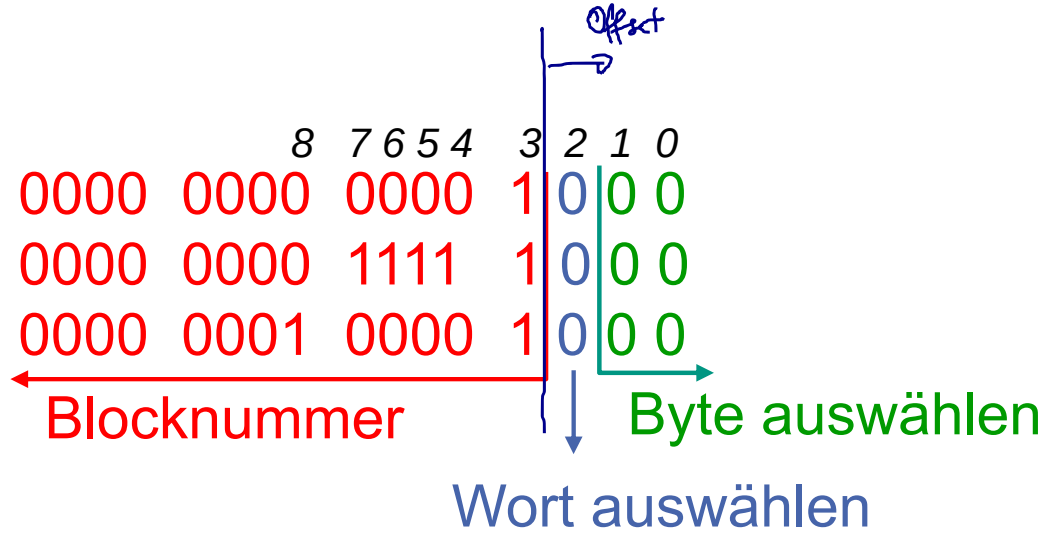
0x000008 0x0000F8 0x000108

Die Caches seien zu Beginn leer, und das Laden soll jeweils in die Zeile mit der niedrigsten verfügbaren Zeilennummer erfolgen. Geben Sie bei der Zuordnung von Blöcken zu Zeilen nicht die Adressen sondern die Blocknummern an.

Lösung 2

Block: 2 Wörter = 8 Byte

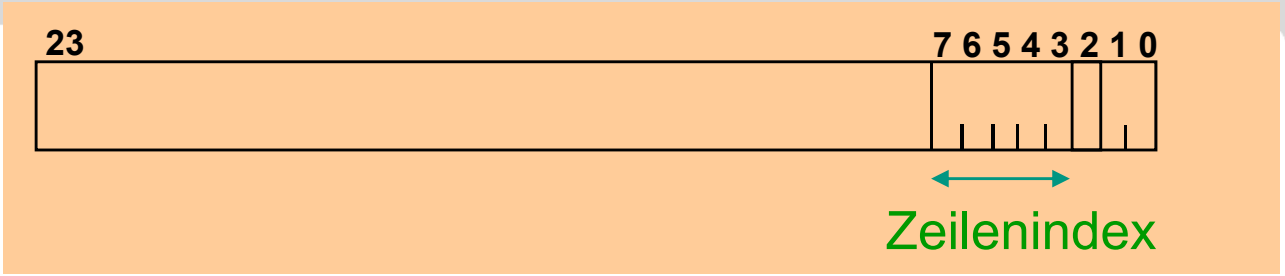
Adressen:



Blocknummer: 21 Bit lang

0x000008 → Block 0x000001 = 1_{10}
0x0000F8 → Block 0x00001F = 31_{10}
0x000108 → Block 0x000021 = 33_{10}

Lösung 2



Block: 2 Wörter = 8 Byte

Adressen:

	8	7 6 5 4	3	2	1	0
0x000008 :	0000 0000 0000 0000	0000	1	0	0	0
0x0000F8 :	0000 0000 0000 0000	1111	1	0	0	0
0x000108 :	0000 0000 0000 0001	0000	1	0	0	0

Blocknummer

Byte auswählen

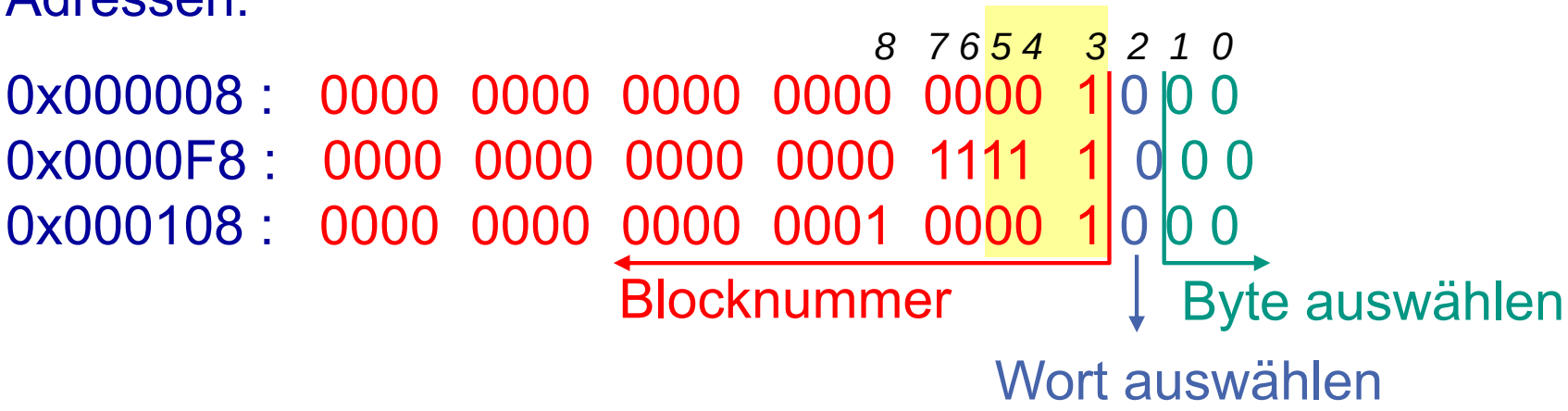
Wort auswählen

Zeilennummer: 5 Bit lang (Bits 3-7)

0x000008 → Zeile 00001 = 1_{10}
0x0000F8 → Zeile 11111 = 31_{10}
0x000108 → Zeile 00001 = 1_{10}

Block: 2 Wörter = 8 Byte

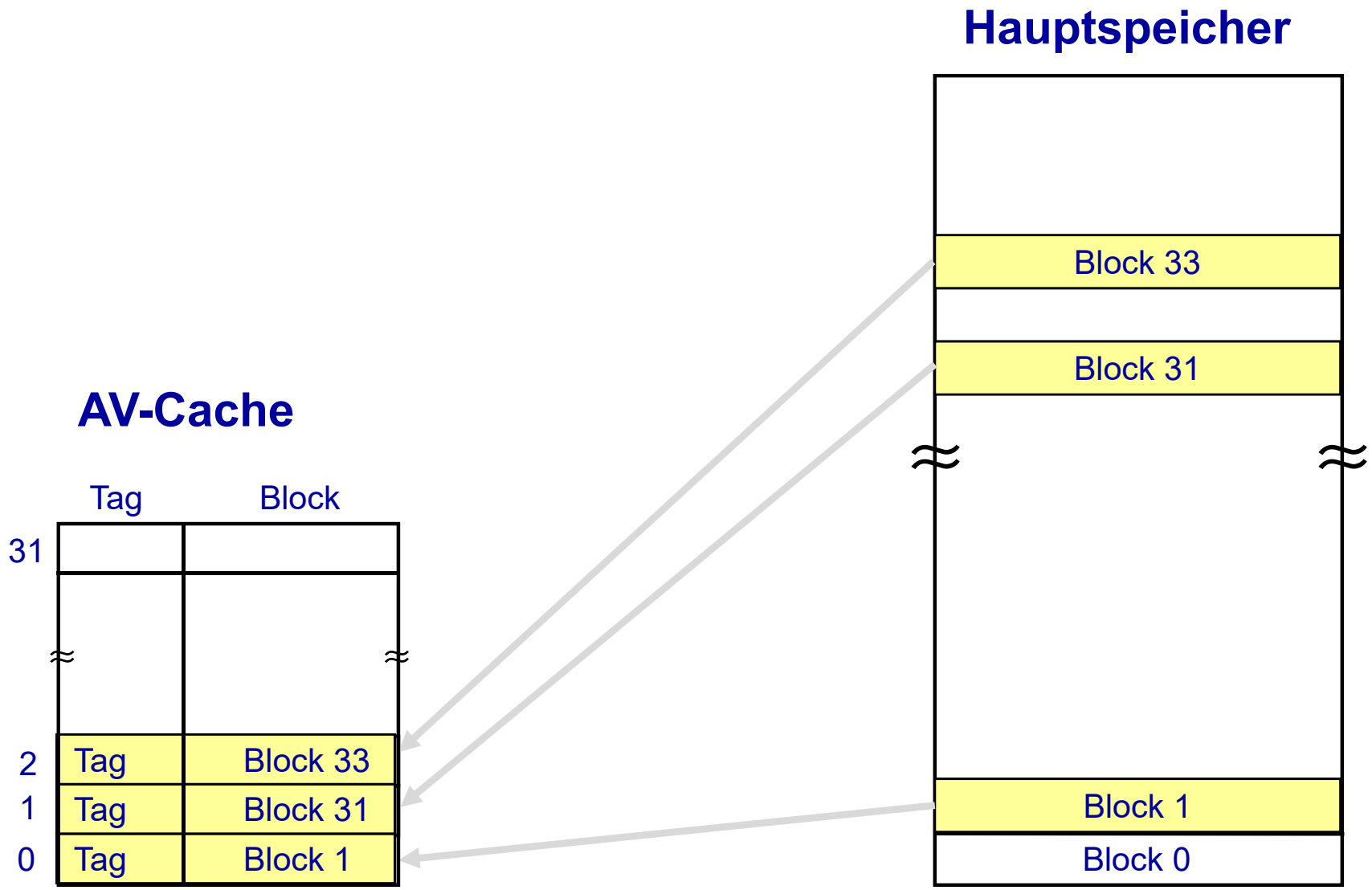
Adressen:



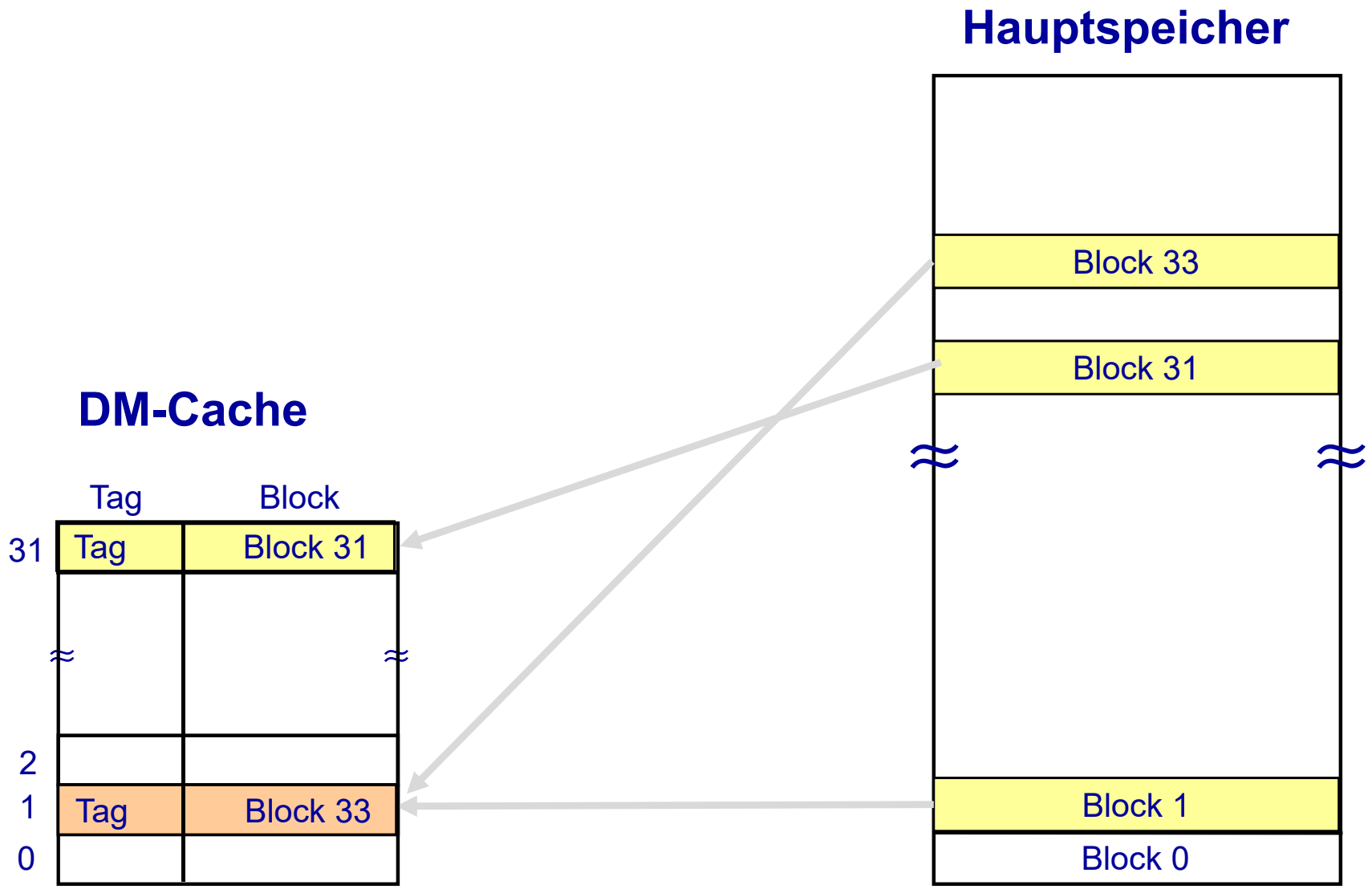
Satznummer: 3 Bit lang (Bits: 3 -5)

0x000008 → Satz 001 = 1_{10}
0x0000F8 → Satz 111 = 7_{10}
0x000108 → Satz 001 = 1_{10}

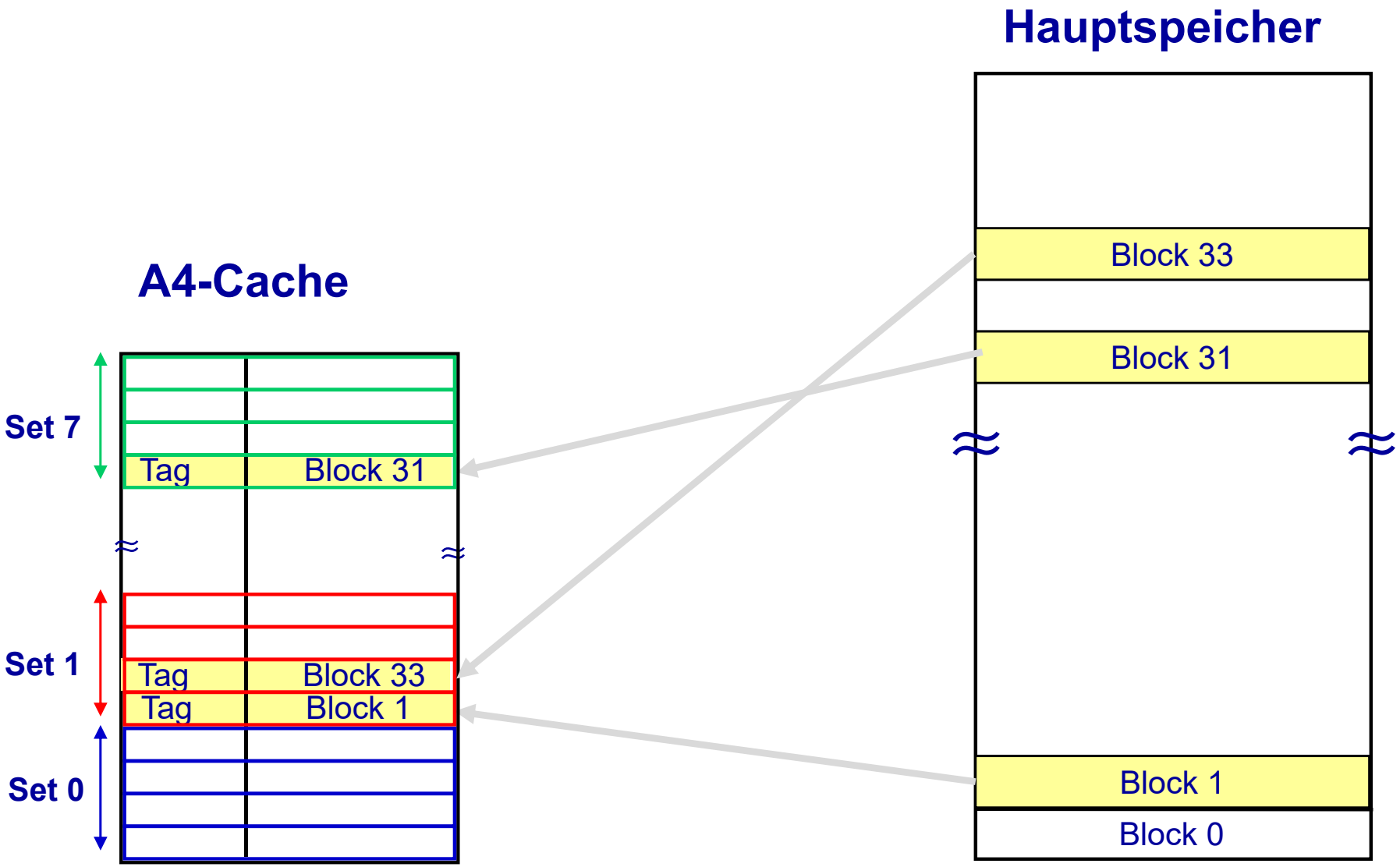
AV-Cache


Hauptspeicher
Block 33
Block 31
Block 1
Block 0
AV-Cache
Tag
Block
31
2
Tag
Block 33
1
Tag
Block 31
0
Tag
Block 1

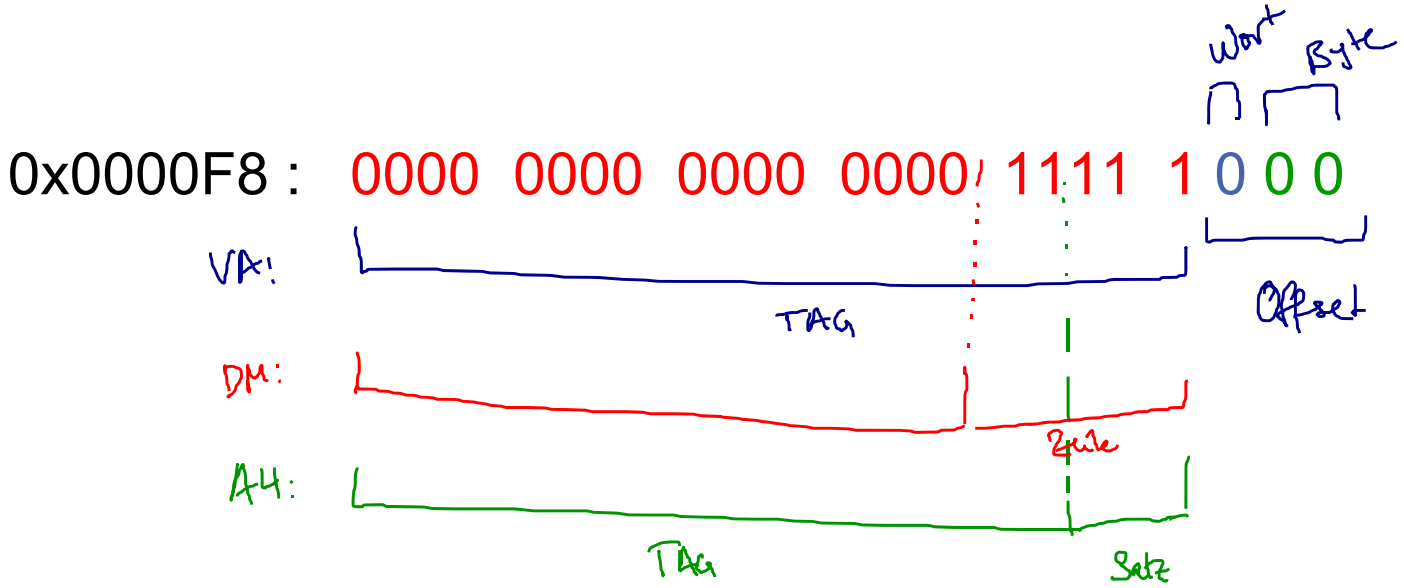
DM-Cache


Hauptspeicher
Block 33
Block 31
Block 1
Block 0
DM-Cache
Tag
Block
31
Tag
Block 31
2
1
Tag
Block 33
0

A4-Cache


Hauptspeicher
A4-Cache
Block 33
Block 31
Block 1
Block 0
A4-Cache
Set 7
Tag
Block 31
Set 1
Tag
Block 33
Tag
Block 1
Set 0

Wohin wird ein Block abgebildet?



Wohin wird ein Block abgebildet?

Blocknummer = (Hauptspeicheradresse) **div** (Blockgröße)

Zeilennummer = (Blocknummer) **mod** (Zeilenanzahl)

Satznummer = (Blocknummer) **mod** (Satzanzahl)

- 8_{16}
  Blocknummer: 8_{16} div 8 = 1
  Zeilennummer: 1 mod 32 = 1
  Satznummer: 1 mod 8 = 1
- $F8_{16}$
  Blocknummer: $F8_{16}$ div 8 = 31
  Zeilennummer 31 mod 32 = 31
  Satznummer: 31 mod 8 = 7
- 108_{16}
  Blocknummer: 108_{16} div 8 = 33
  Zeilennummer 33 mod 32 = 1
  Satznummer: 33 mod 8 = 1

Aufgabe 3

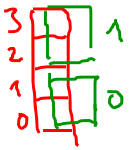
Gegeben seien drei Cache-Speicher DM, A2 und AV, die jeweils **vier** Cache-Blöcke besitzen mit je **einem Byte**. Der Cache DM ist als direkt-abgebildeter Cache (direct-mapped) organisiert; Cache A2 als 2-fach satzassoziativer Cache (2-way-set-associativ); Cache AV ist vollassoziativ (fully-associative). Bei den Cachespeichern A2 und AV soll die „least recently used"-Ersetzungsstrategie LRU angewendet werden.

Nehmen Sie an, die Cachespeicher seien zu Beginn leer, und es soll eine Serie von einzelnen Bytes mit den folgenden 32-Bit-Adressen gelesen werden:

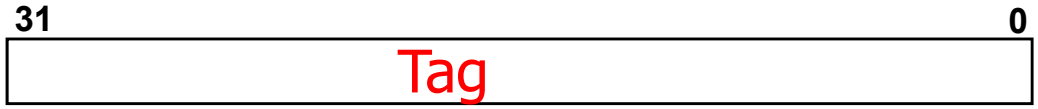
9, 42, 13, 9, 23, 12, 13, 42

Aufgabe 3

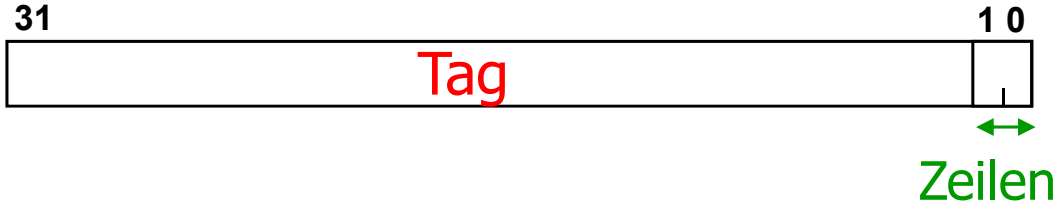
1. Geben Sie für die drei Cachespeicher an, wie viele Bits zur Verwaltung eines Cacheblocks benötigt werden. Dabei sollen für den Zustand des Cache-Blocks zwei Statusbits verwendet werden (Valid-Bit und Dirty-Bit).



AV

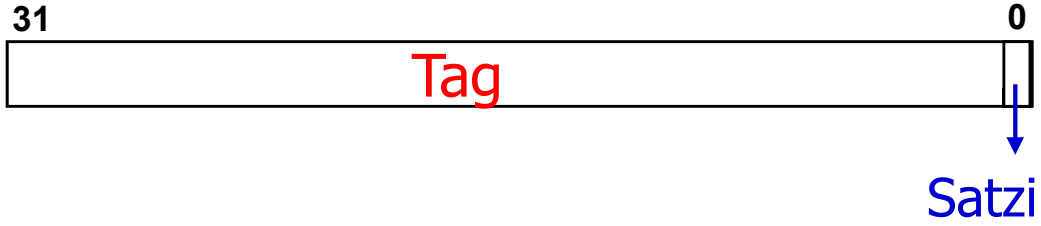


DM



4 Zeilen

A2



4 Zeilen → 2 Sätze

Lösung 3.1

	Tag-Länge	Tag+Statusbits
VA	32	32 + 2 = 34
DM	30	30 + 2 = 32
A2	31	31 + 2 = 33

Anzahl der notwendigen Bits zur Verwaltung eines Cacheblocks

Lösung 3.2

2. Geben Sie für die drei Cachespeicher die Anzahl der erforderlichen Vergleicher und die jeweils zu vergleichende Bitanzahl an.

	Vergleicher	Tag-Länge
VA	4	32
DM	1	30
A2	2	31

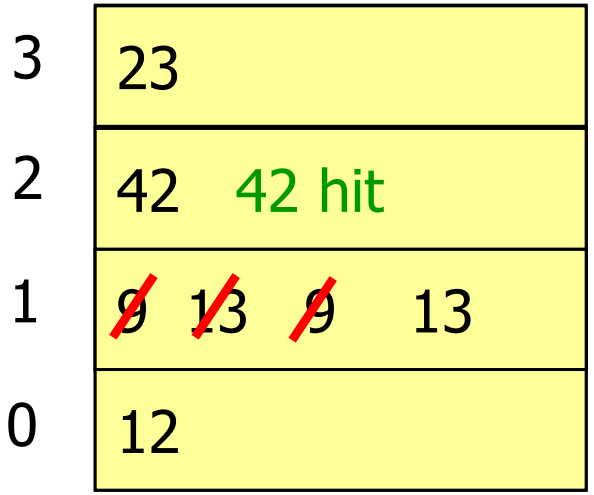
Lösung 3.3

3. Geben Sie nun tabellarisch für jeden Cache an, ob es sich beim Lesezugriff auf die jeweilige Adresse um einen Cache-Hit oder um einen Cache-Miss handelt.

Lesezugriffe auf die Adressen (Hier: Adresse = Blocknummer)

DM: Zeilennummer = (Blocknummer) mod (Zeilenanzahl)

9 mod 4	= 1	Miss
42 mod 4	= 2	Miss
13 mod 4	= 1	Miss
9 mod 4	= 1	Miss
23 mod 4	= 3	Miss
12 mod 4	= 0	Miss
13 mod 4	= 1	Miss
42 mod 4	= 2	Hit

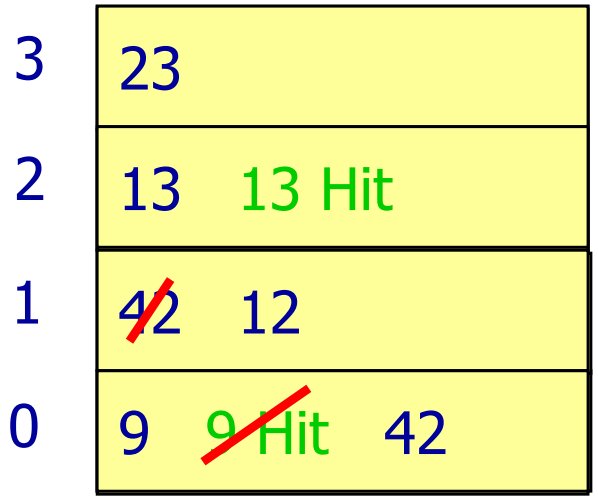


Lösung 3.3

AV:

9 → Zeile 0	Miss
42 → Zeile 1	Miss
13 → Zeile 2	Miss
9 in Zeile 0	Hit
23 → Zeile 3	Miss
12 → Zeile 1	Miss
13 in Zeile 2	Hit
42 in Zeile 0	Miss

Cache voll → Ersetzungsstrategie LRU

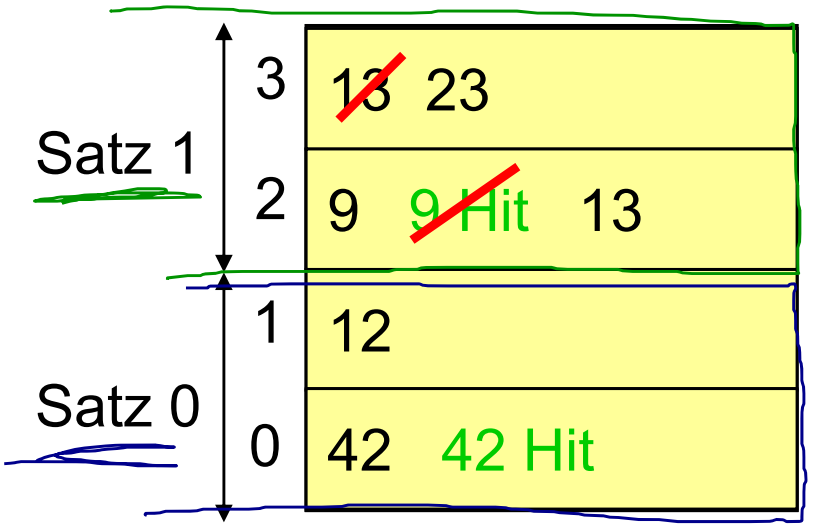


Lösung 3.3

A2: Satznummer = (Blocknummer) mod (Satzanzahl)

9 mod 2	= 1	Miss
42 mod 2	= 0	Miss
13 mod 2	= 1	Miss
9 mod 2	= 1	Hit
23 mod 2	= 1	Miss
12 mod 2	= 0	Miss
13 mod 2	= 1	Miss
42 mod 2	= 0	Hit

Bei vollem Satz: LRU



Aufgabe 4

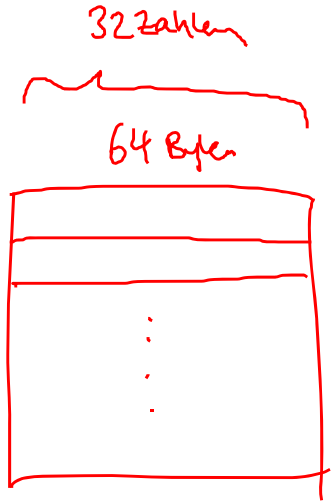
Cache Hit/Miss-Rate

Programmschleife zur Multiplikation von 512 16-Bit-Zahlen

```
for (mul=1, j=0; j<512; j++) mul *=g[j];
```

Vollassoziativer Daten-Cache (am Anfang leer) mit 64 Bytes pro Cache-Zeile.

➔ 32 Zahlen pro Cache-Zeile



Lösung 4

	Miss	Hit
`mul = 1`	1 1	
`j = 0`	1 1	
`loop: read j`		1 1 512
`if (j >= 512) exit`		
`else`		
`read g[j]`	1 16	1 496
`read mul`		1 1 512
`compute mul *g[j]`		
`write mul`		1 1 512
`read j`		1 1 512
`compute j+1`		
`write j`		1 1 512
`jump to loop`		
	18	3056
1. Durchlauf 2. Durchlauf	0,59 %	99,41 %