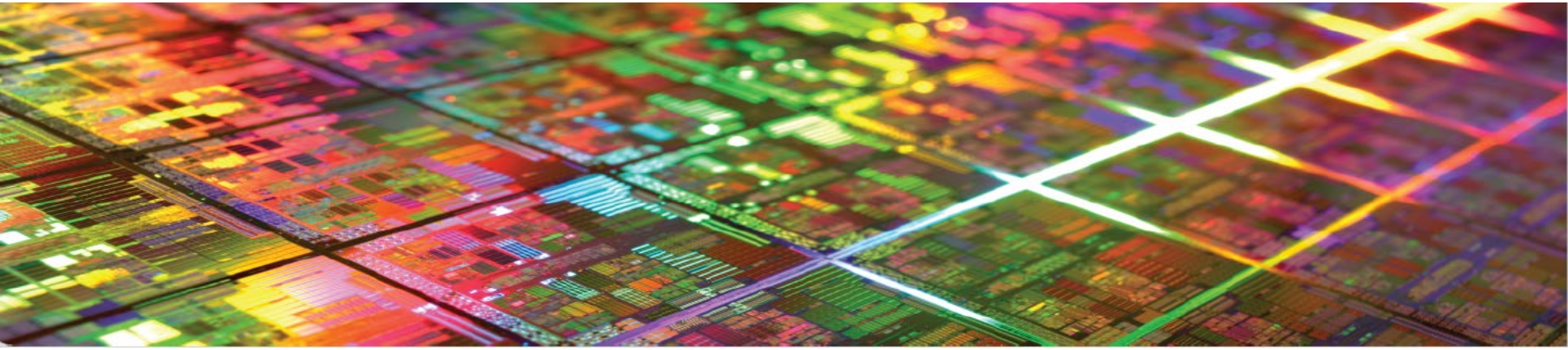
Karlsruher Institut für Technologie

Rechnerorganisation

Prof. Dr. Wolfgang Karl

Vorlesung im Wintersemester 2025/2026 – Foliensatz: RO25-FS12

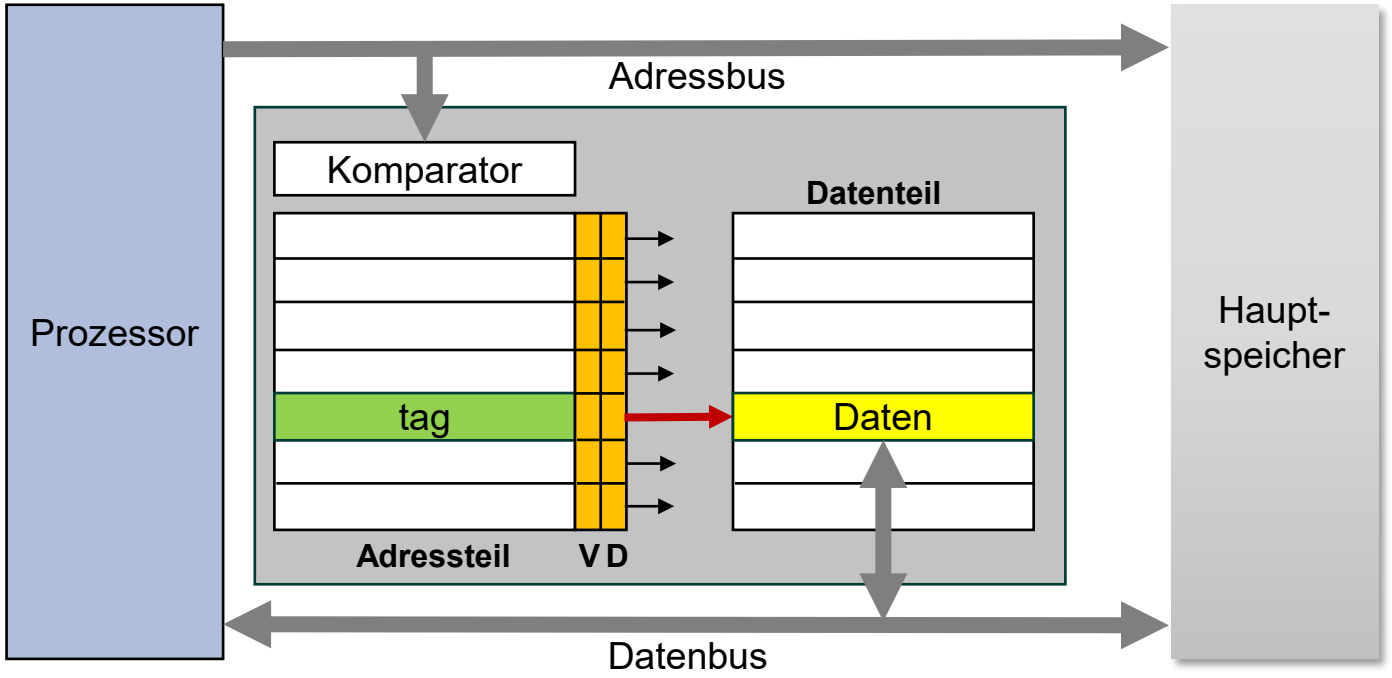


Kapitel 8
Cache-Speicher

- Speicherhierarchie
- Systemaufbau mit Cache-Speicher
- Grundlegende Arbeitsweise
- **Cache-Organisationsformen**
- Grundlegende Fragen beim Entwurf
- Gültigkeitsproblem

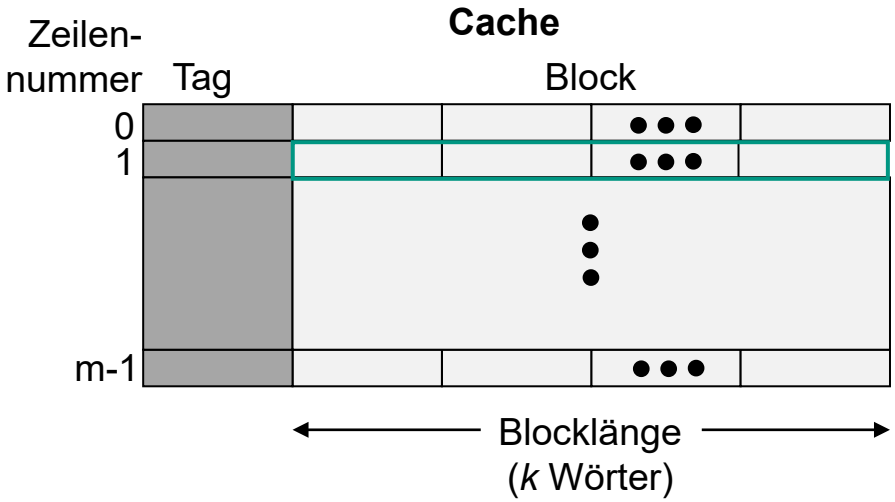
Cache-Speicher

Aufbau eines Cache-Speichers

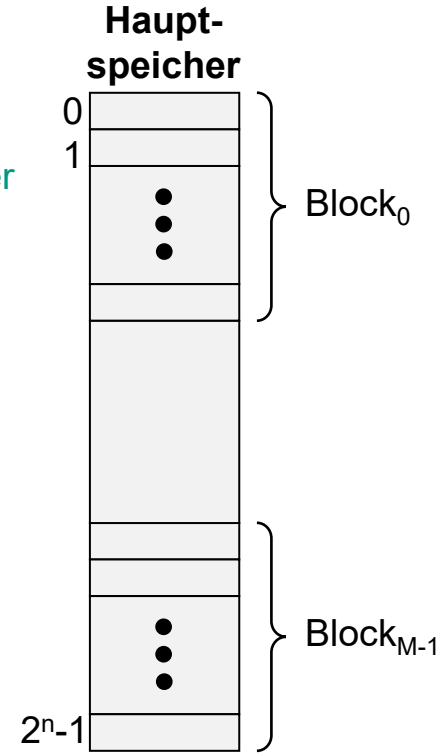


Cache-Speicher

Grundlegende Fragen

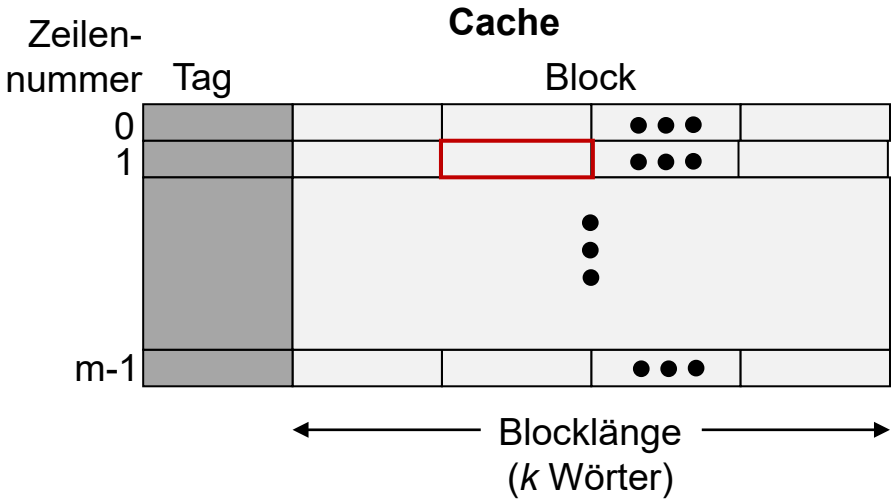


Jede Cache-Zeile kann den Inhalt verschiedener Blöcke enthalten.

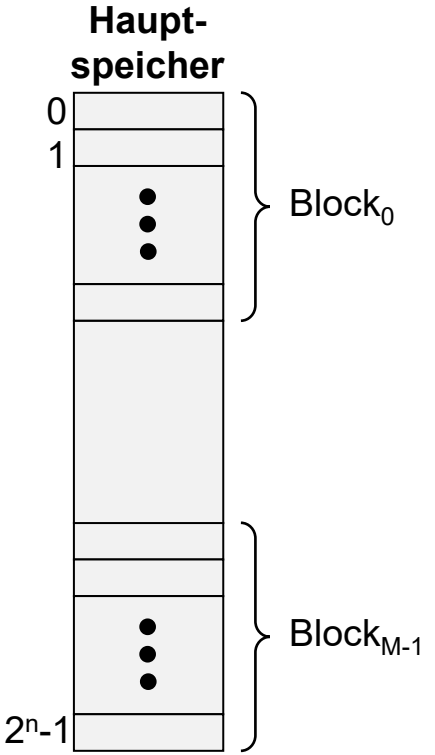


Cache-Speicher

Grundlegende Fragen

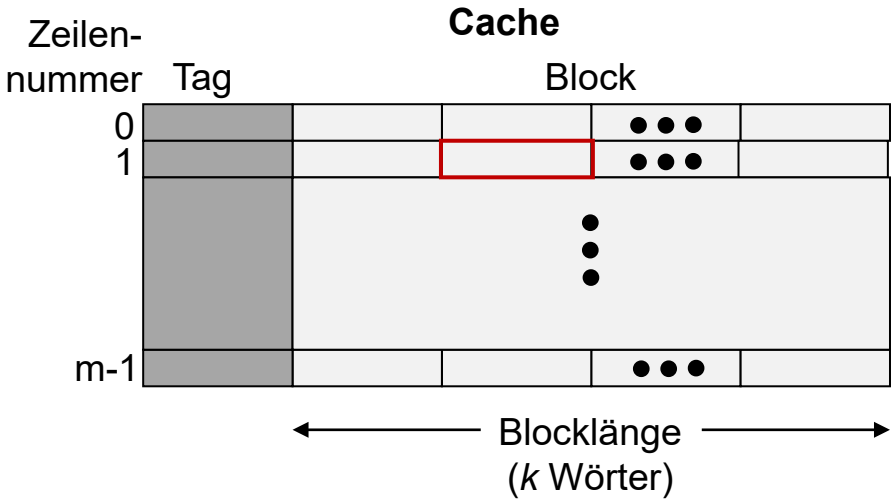


Wie können wir wissen, ob ein Objekt im Cache dem angeforderten Wort entspricht?

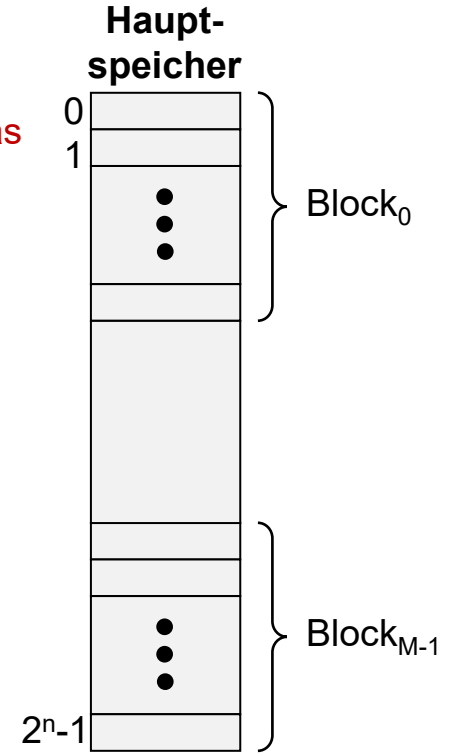


Cache-Speicher

Grundlegende Fragen

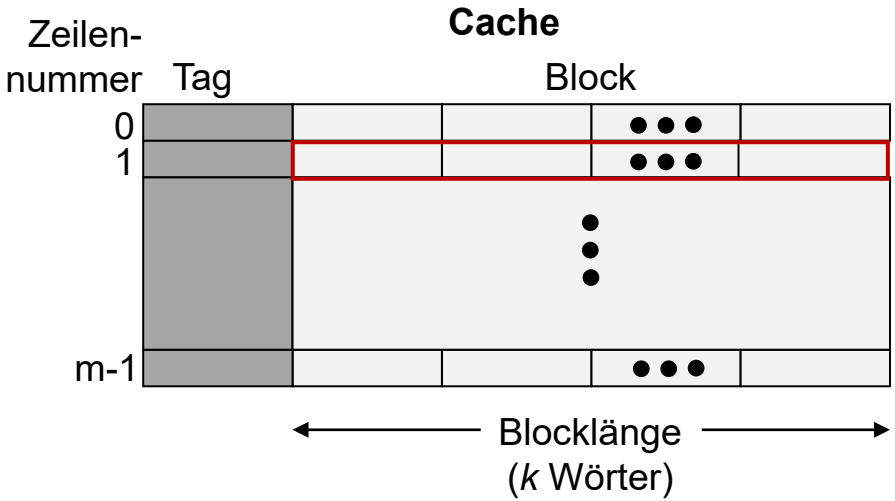


Woher wissen wir, ob das angeforderte Wort im Cache ist oder nicht?

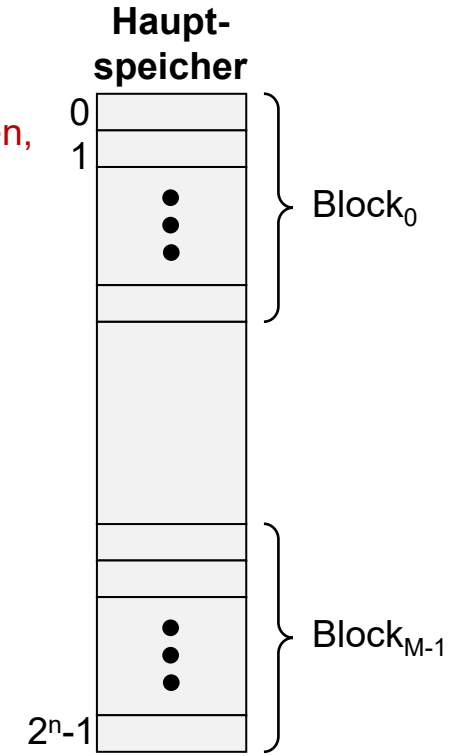


Cache-Speicher

Grundlegende Fragen

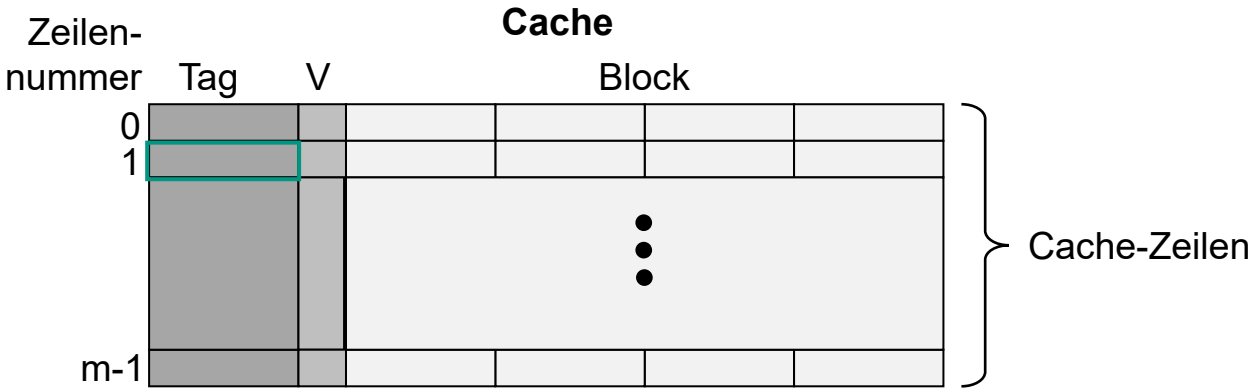


Wie können wir erkennen, ob eine Cache-Zeile gültige Informationen enthält oder nicht?



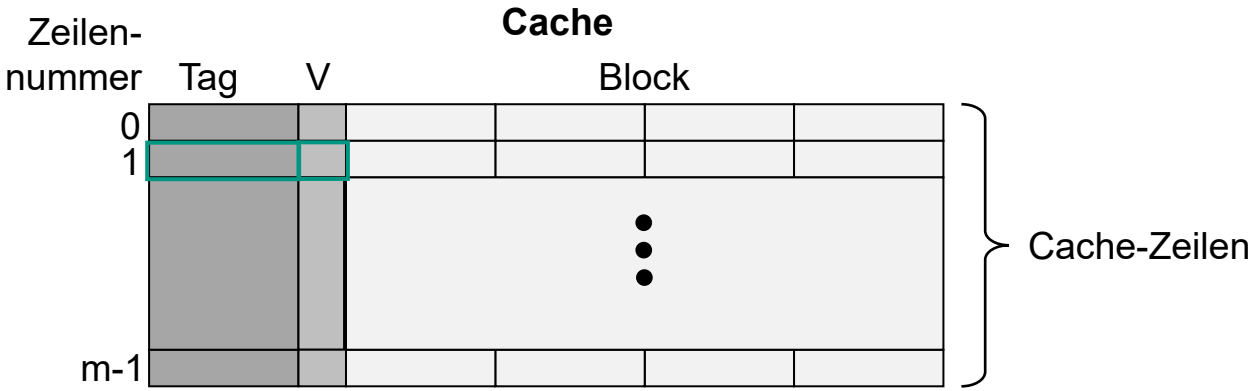
Cache-Speicher

- **Antworten**
  - **Tag** im Adressteil einer Cache-Zeile
    - identifiziert, welcher Block in der Zeile enthalten ist;
    - ist die notwendige Adressinformation, mit der erkannt werden kann, ob der Block in der Cache-Zeile dem Speicherblock mit dem angeforderten Wort entspricht;



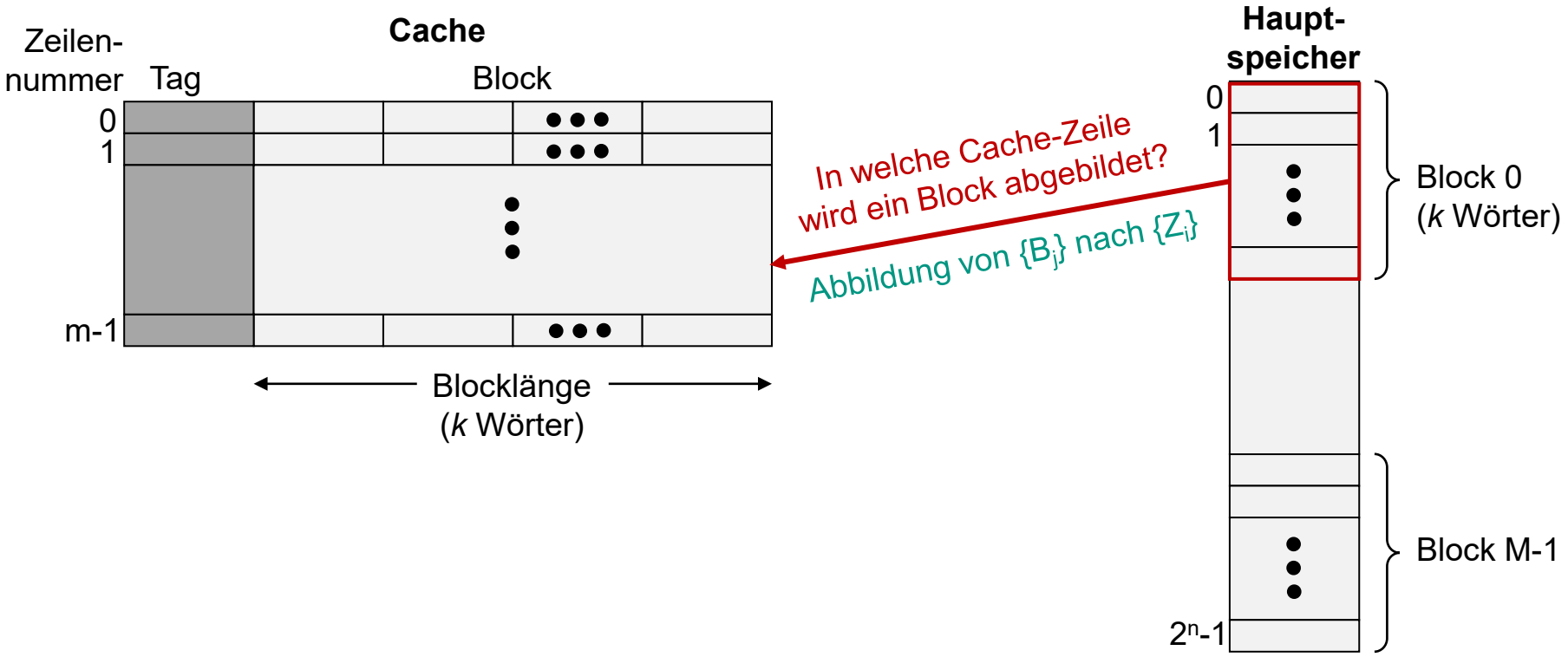
Cache-Speicher

- **Antworten**
  - **Gültigkeitsbit (Valid-Bit, V)**
    - zeigt an, ob die Zeile gültige Daten enthält oder nicht.



Cache-Speicher

Grundlegende Fragen

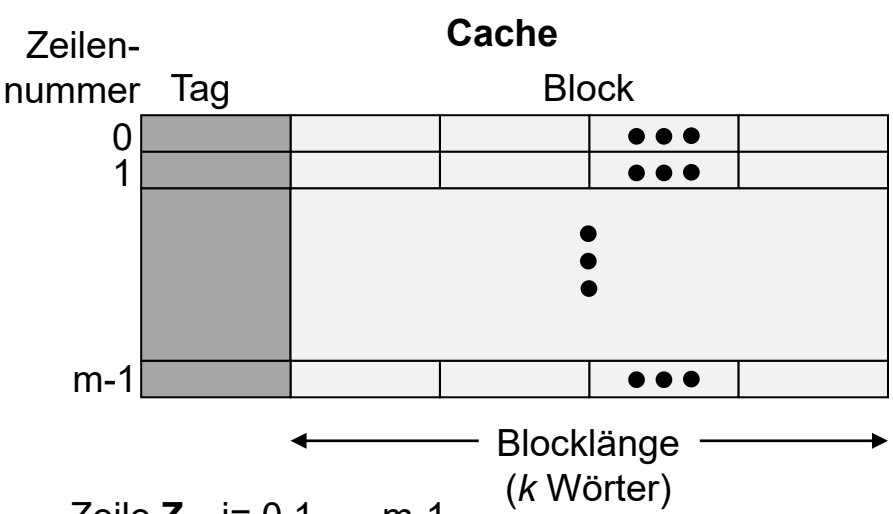


Cache-Speicher

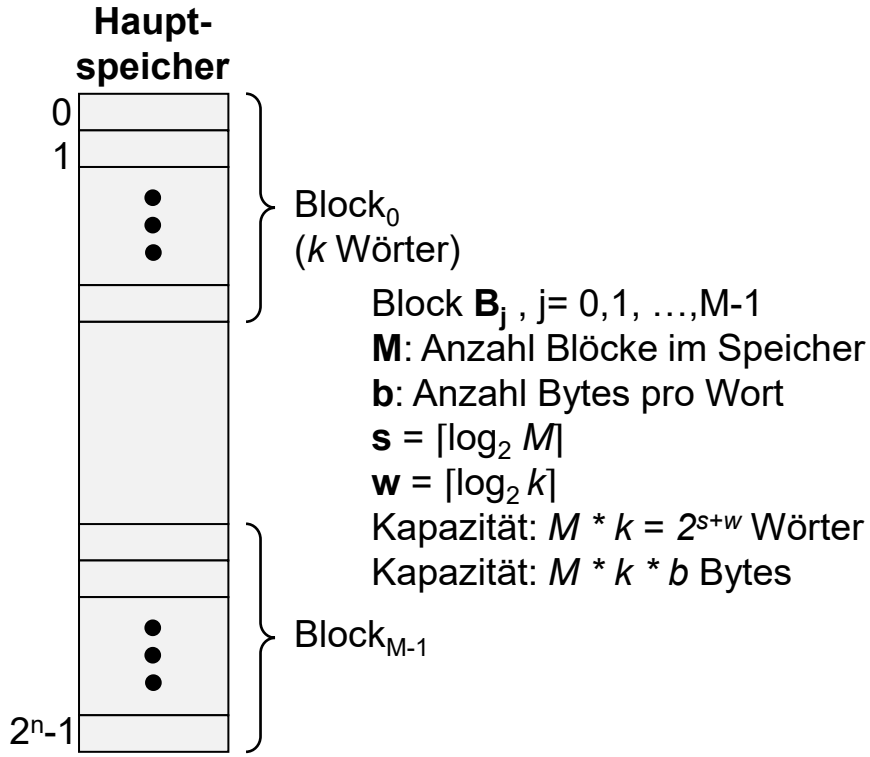
- **Antwort**
  - **Cache-Organisationsformen:**
    - **Direkt abgebildeter Cache (direct mapped cache)**
    - **Voll -assoziativer Cache (fully associative cache)**
    - **n-fach satz-assoziativer Cache (n-Wege mengenassoziativer Cache, n-way set-associative cache)**

Cache-Speicher

Cache-Parameter



Zeile $\mathbf{Z_i}$, i= 0,1, …,m-1
m: Anzahl Zeilen im Cache
r = $\lceil \log_2 m \rceil$
w = $\lceil \log_2 k \rceil$
Kapazität: $m * k = 2^{r+w}$ Wörter



Cache-Organisationsformen

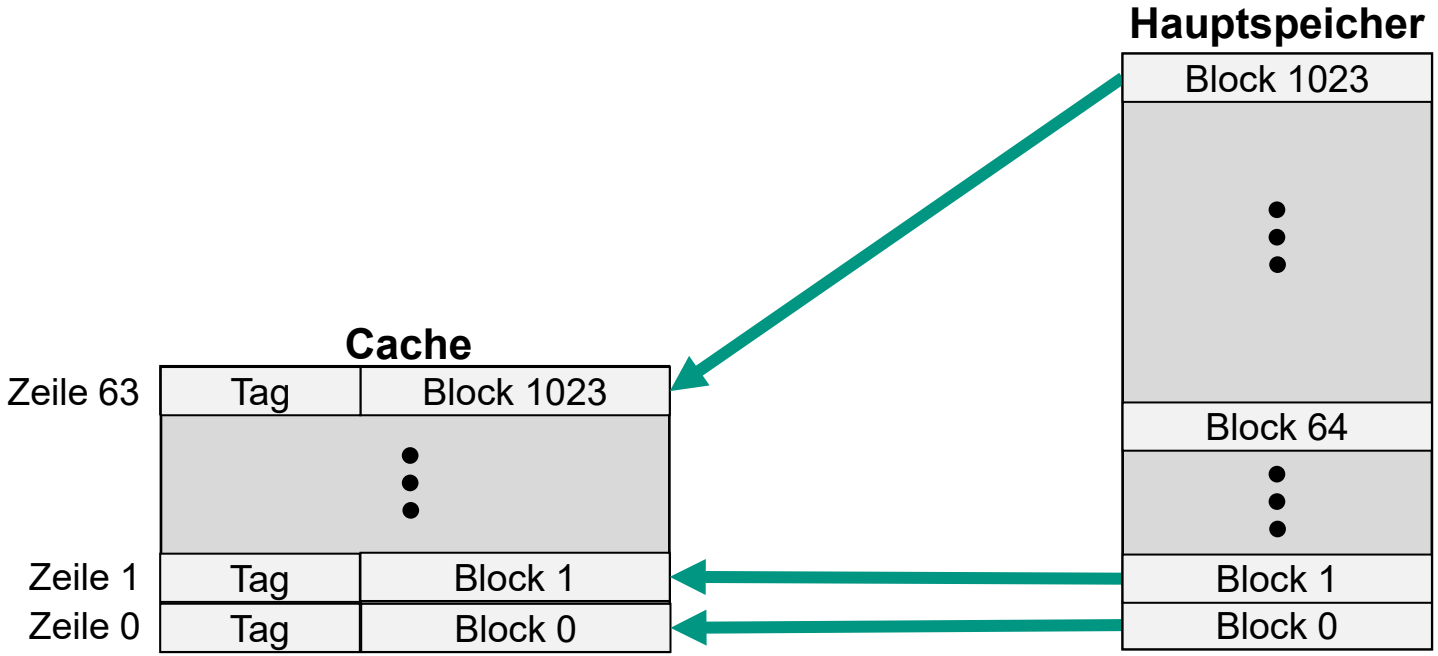
- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - Jedem Block im Hauptspeicher ist eine eindeutige und feste Zeile im Cache zugeordnet.
  - Ein Speicherblock wird auf eine Zeile des Cache-Speichers gemäß folgender **Abbildungsvorschrift** abgebildet:

$$(\text{Block} - \text{Adresse}) \bmod (\text{Anzahl der Zeilen im Cache})$$

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**

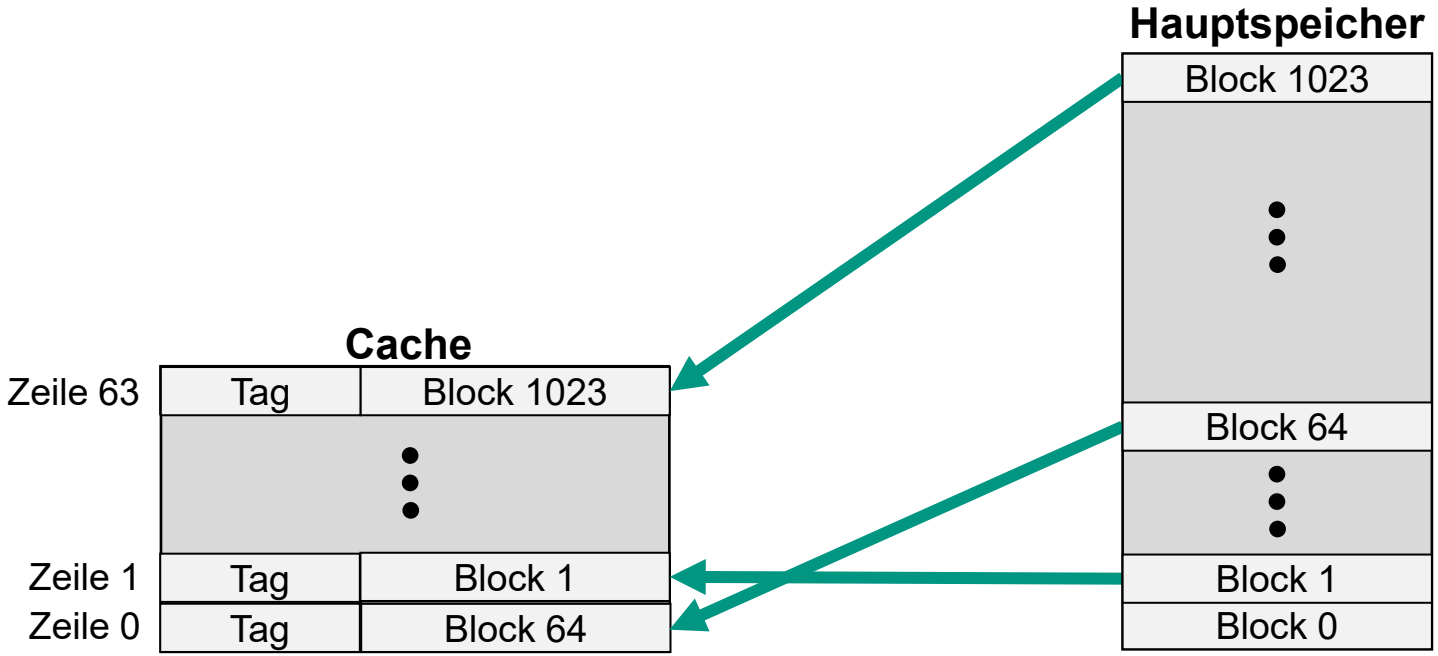
Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2



Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**

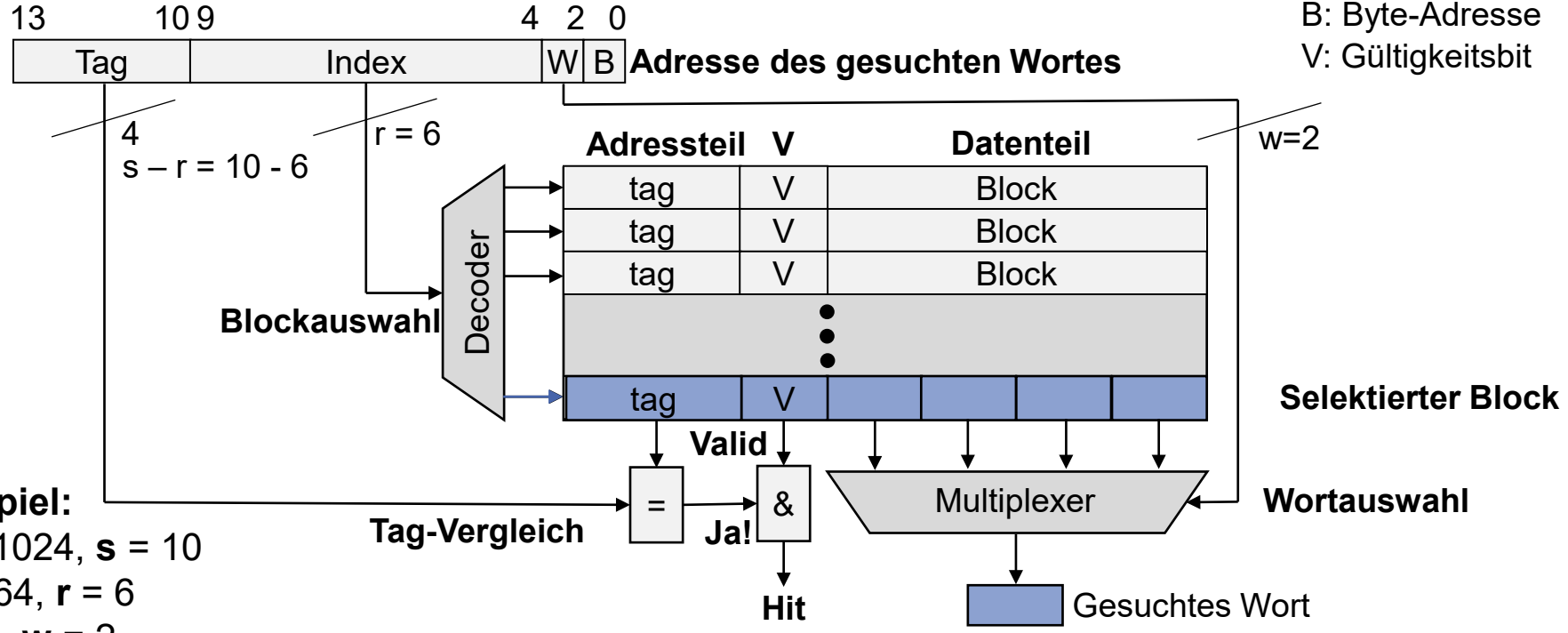
Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2



Cache-Organisationsformen

Direkt abgebildeter Cache (direct-mapped cache, DM)

W: Wortadresse
B: Byte-Adresse
V: Gültigkeitsbit



Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Aufteilung der Adresse:**
    - **Index-Feld**
      - Dient zur Auswahl der Cache-Zeile;
      - Ein Cache hat **m** Cache-Zeilen mit $\mathbf{m = 2^r}$; Index-Feld ist **r** Bit breit
    - **Tag-Feld:**
      - Wert wird mit dem Adressetikett (tag) in der ausgewählten Cache-Zeile verglichen;
      - Breite des Tag-Feldes ist $\mathbf{a - (r + w + 2)}$, mit **a** = Anzahl Adressbits
    - **Wortadresse:**
      - Adresse eines Worts im Speicherblock;
    - **Byte-Adresse**
      - Falls die Wörter im Speicher an Wortgrenzen ausgerichtet abgelegt sind, kann die Byte-Adresse ignoriert werden.

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**
    - Wie viele Bits sind erforderlich für einen direkt abgebildeten Cache mit $\mathbf{16\ KB}$ Daten und Blöcken mit jeweils $\mathbf{k = 2^w = 2^2 = 4}$. Die Speicheradresse umfasst $\mathbf{a = 64\ Bits}$?
    - **Antwort:**
      - Jeder Block enthält $\mathbf{k = 4}$ Wörter, d. h. es gibt $\mathbf{k = 2^r = 2^{10} = 1024\ Blöcke}$.
        $\mathbf{16\ KB}$ entspricht $\mathbf{2^{r+w} = 2^{12} = 4096\ Wörter}$.
      - Der Datenteil einer Cache-Zeile umfasst $\mathbf{4 * 32\ Bits = 128\ Bits}$. Die Breite des Adressteils ist $\mathbf{a - (r + w + 2) = 64 + (10 + 2 + 2)Bits}$ und einem Valid-Bit.
      - Die Größe des Cache-Speichers ist:

        $\mathbf{2^{10} * (4 * 32 + (64 - 10 - 2 - 2) + 1) = 179\ KBits = 22,4\ KB}$

      - Der Cache benötigt 1,4-mal mehr Bits als für die Speicherung der Daten notwendig ist.

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**
    - Ein Cache hat $\mathbf{m = 64\ Zeilen}$ Zeilen. Der Datenteil umfasst $\mathbf{16\ Bytes}$. Auf welche Zeilenadresse wird der Speicherblock mit der Byte-Adresse $\mathbf{1200}$ abgebildet?
    - **Antwort:**
      - Die Abbildungsvorschrift lautet:

$$(\text{Block} - \text{Adresse})\ \text{modulo}\ (\text{Anzahl der Zeilen im Cache})$$

      - Die Blockadresse ist:

$$\frac{\text{Byte} - \text{Adresse}}{\text{Bytes pro Block}}$$

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**
    - Ein Cache hat $\mathbf{m = 64\ Zeilen}$ Zeilen. Der Datenteil umfasst $\mathbf{16\ Bytes}$. Auf welche Zeilenadresse wird der Speicherblock mit der Byte-Adresse $\mathbf{1200}$ abgebildet?
    - **Antwort:**
      - Der Block mit dieser Block-Adresse ist der Block, der alle Adressen enthält zwischen

$$\left\lfloor \frac{\text{Byte} - \text{Adresse}}{\text{Bytes pro Block}} \right\rfloor \times \text{Bytes pro Block}$$

und

$$\left\lfloor \frac{\text{Byte} - \text{Adresse}}{\text{Bytes pro Block}} \right\rfloor \times \text{Bytes pro Block} + (\text{Bytes pro Block} - 1)$$

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Beispiel:**
    - Ein Cache hat $\mathbf{m = 64\ Zeilen}$ Zeilen. Der Datenteil umfasst $\mathbf{16\ Bytes}$. Auf welche Zeilenadresse wird der Speicherblock mit der Byte-Adresse $\mathbf{1200}$ abgebildet?
    - **Antwort:**
      - Das Wort mit der Speicheradresse $\mathbf{1200}$ ist im Block mit der Blocknummer

$$\left\lfloor \frac{1200}{16} \right\rfloor = 75$$

      - und wird auf die Cache-Zeile $\mathbf{m = (75\ modulo\ 64) = 11}$ abgebildet.
      - Auf diese Cache-Zeile werden alle Adressen zwischen $\mathbf{1200}$ und $\mathbf{1215}$ abgebildet.

Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Nachteile:**
    - Starre Abbildungsfunktion:
      - Bei Blöcken (z.B. 0, 64, 128, ...), die auf dieselbe Cache-Zeile abgebildet werden, ergeben sich Kollisionen, obwohl andere Blöcke im Cache vielleicht frei sind.
      - Bei einem abwechselnden Zugriff auf Speicherblöcke, deren Adressen den gleichen Index-Teil haben, erfolgt laufendes Überschreiben des gerade geladenen Blocks in der ausgewählten Cache-Zeile; es kommt zum „**Flattern**“ (**thrashing**)

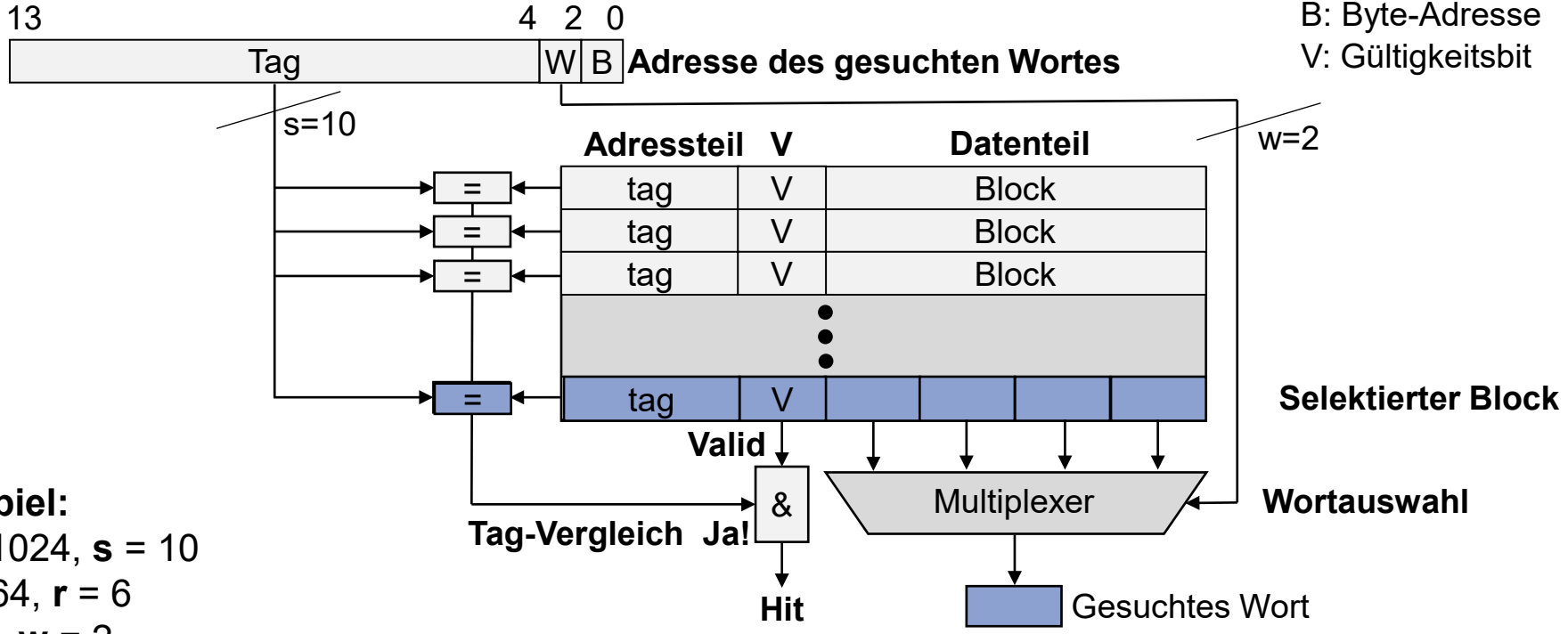
Cache-Organisationsformen

- **Direkt abgebildeter Cache (direct-mapped cache, DM)**
  - **Vorteile:**
    - Geringer Hardwareaufwand: nur ein Vergleicher!
      - Nur auf 1 tag muss zugegriffen werden
      - Das Tag-Feld ist relativ kurz.
      - Es ist keine Ersetzungsstrategie erforderlich, weil die direkte Abbildung keine Alternativen zulässt.
    - Der Zugriff erfolgt schnell, weil das Tag-Feld parallel mit dem zugehörigen Block gelesen werden kann

Cache-Organisationsformen

Vollassoziativer Cache (fully associative cache, VA)

W: Wortadresse
B: Byte-Adresse
V: Gültigkeitsbit



Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2

Cache-Organisationsformen

- **Vollassoziativer Cache (fully associative cache, VA)**
  - **Aufteilung der Adresse:**
    - **Tag-Feld:**
      - Wert wird mit dem Adressetiketten (tags) aller Cache-Zeilen verglichen;
      - Breite des Tag-Feldes ist $\mathbf{a - (w + 2)}$, mit $\mathbf{a}$ = Anzahl Adressbits;
    - **Wortadresse:**
      - Adresse eines Worts im Speicherblock
    - **Byte-Adresse**
      - Falls die Wörter im Speicher an Wortgrenzen ausgerichtet abgelegt sind, kann die Byte-Adresse ignoriert werden

Cache-Organisationsformen

- **Vollassoziativer Cache (fully associative cache, VA)**
  - Paralleler Vergleich aller Adressetiketten (tags) mit der Speicherdresse des gesuchten Wortes in einem einzigen Taktzyklus;
  - **Vorteile:**
    - Ein Block kann auf jede Zeile des Cache-Speichers abgebildet werden.
    - Sehr gute Cache-Ausnutzung, da völlig freie Wahl bei der Auswahl einer Cache-Zeile, deren Inhalt zu ersetzen ist;

Cache-Organisationsformen

- **Vollassoziativer Cache (fully associative cache, VA)**
  - Paralleler Vergleich aller Adressetiketten (tags) mit der Speicherdresse des gesuchten Wortes in einem einzigen Taktzyklus;
  - **Nachteile:**
    - Hoher Hardwareaufwand (paralleler Zugriff auf alle tags):
      - Für jede Cache-Zeile ist ein Vergleicher notwendig.
      - Ist für kleine Cachespeicher sinnvoll realisierbar;
      - Die große Flexibilität der Abbildungsvorschrift erfordert eine weitere Hardware-Logik, welche die Ersetzungsstrategie implementiert.
      - **Ersetzungsstrategie bestimmt,** welcher Cache-Zeile mit einem neuen Block überschrieben werden soll, wenn der Cache voll ist.

Cache-Organisationsformen

- **n-fach satzassoziativer Cache (n-way set-associative Cache, nA)**
  - Mehrere Cache-Zeilen werden zu einem Satz (Set) zusammengefasst.
  - Der Satz, in der ein Block abgelegt wird, wird ausgewählt durch die Abbildungsvorschrift:

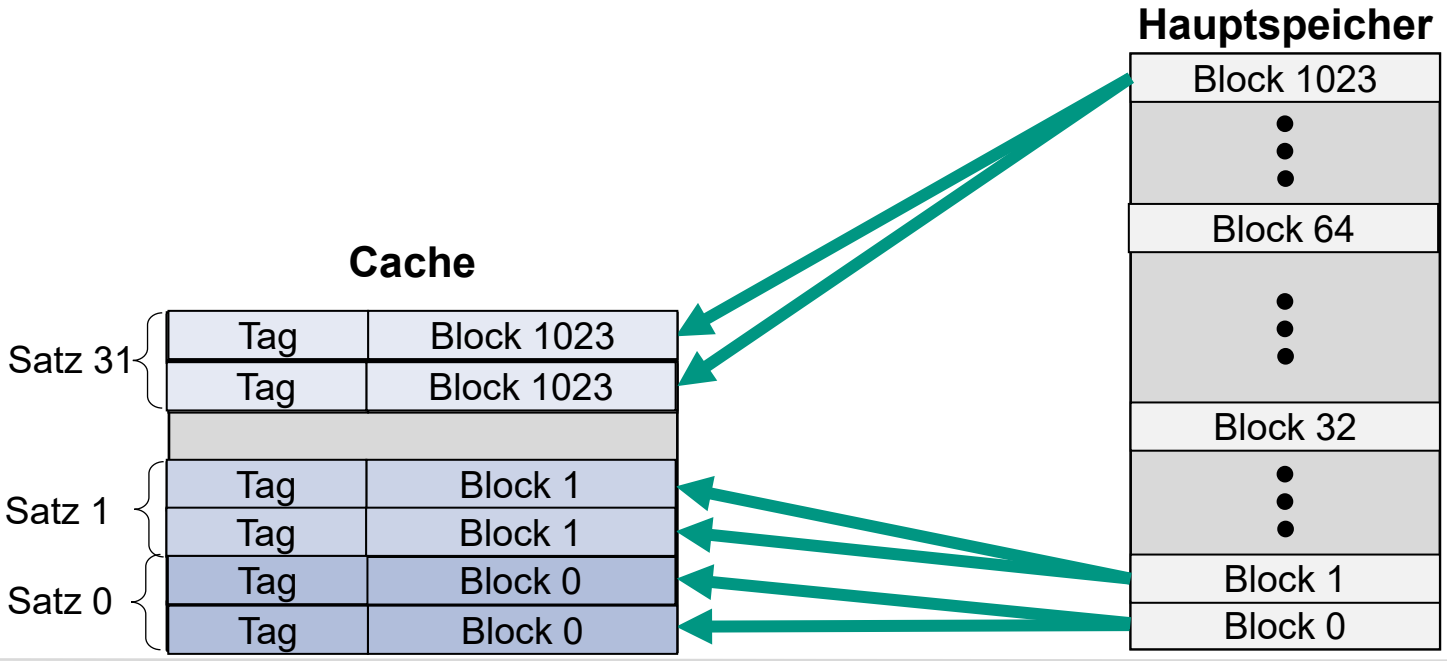
$$(\text{Block} - \text{Adresse}) \text{ modulo } (\text{Anzahl p Sätze des Cache} - \text{Speichers})$$

  - Innerhalb eines Satzes gibt es dann **n** Wege (Ways).
    - der Block wird auf eine der Cache-Zeilen eines Satzes abgebildet.

Cache-Organisationsformen

- **n-fach satzassoziativer Cache (n-way set-associative Cache, nA)**
  - **Beispiel: 2-fach satzassoziativer Cache**

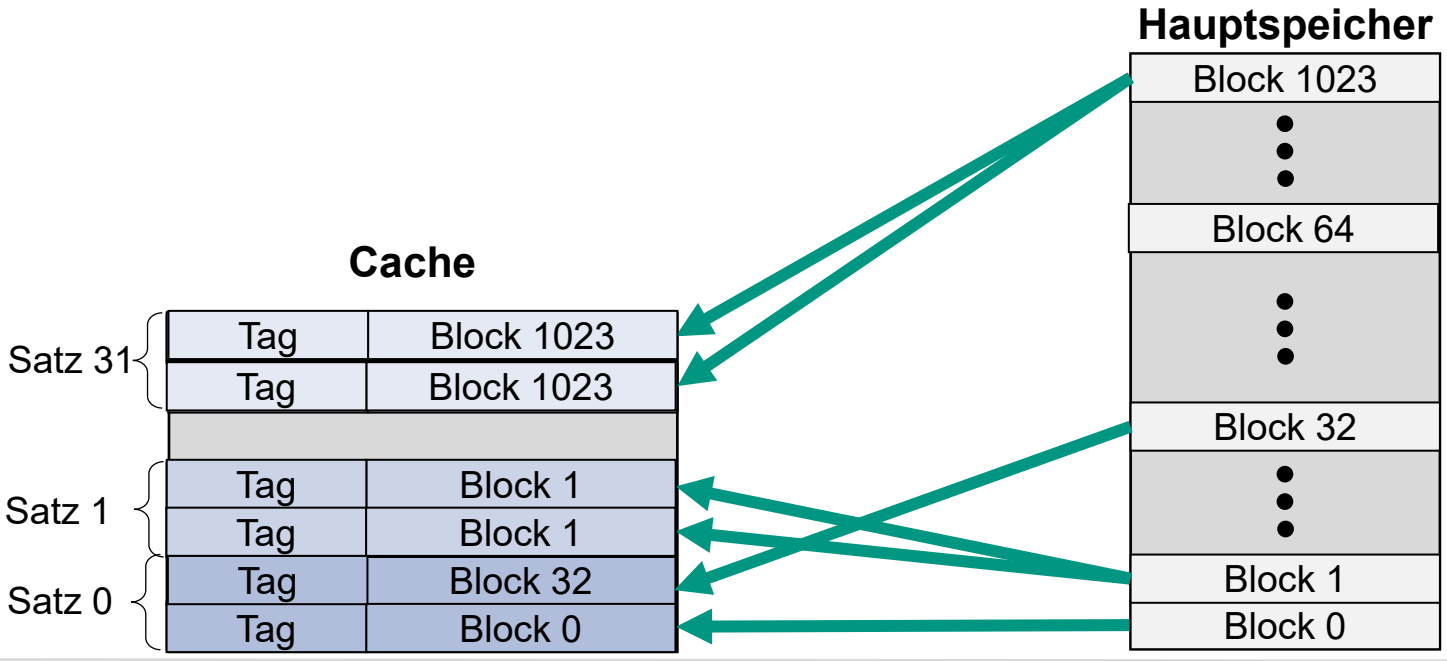
Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2
n = 2, **p** = 32



Cache-Organisationsformen

- **n-fach satzassoziativer Cache (n-way set-associative Cache, nA)**
  - **Beispiel: 2-fach satzassoziativer Cache**

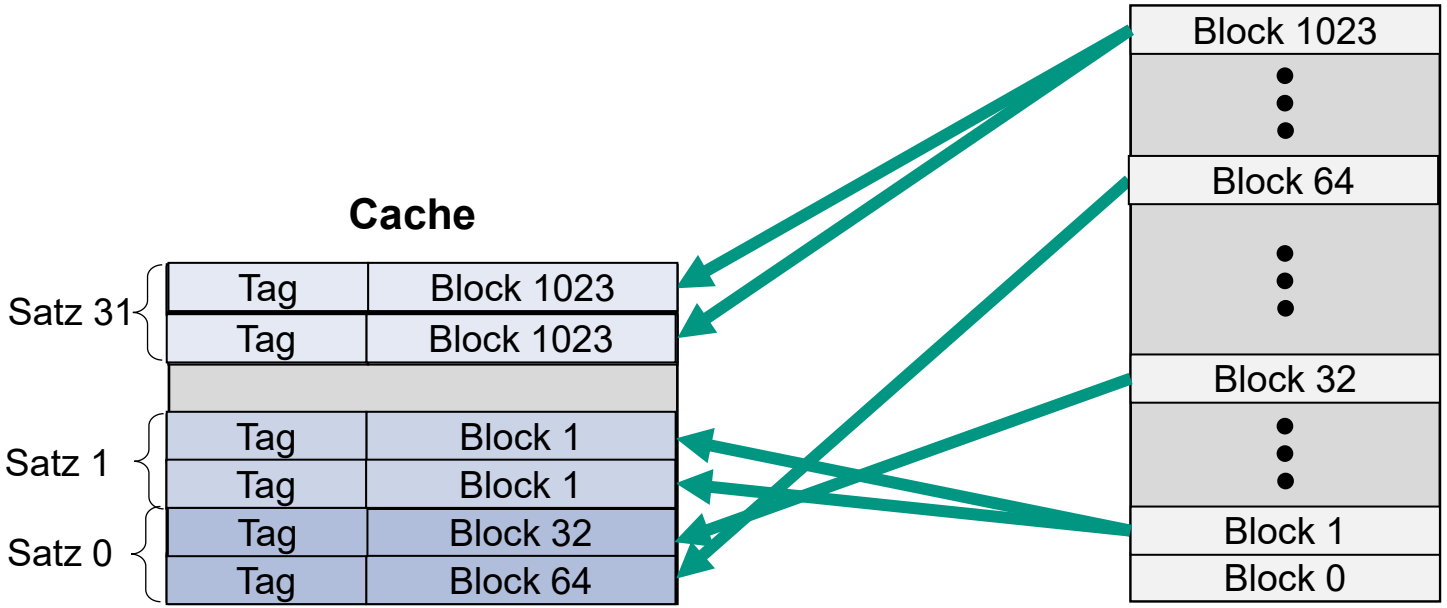
Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2
n = 2, **p** = 32



Cache-Organisationsformen

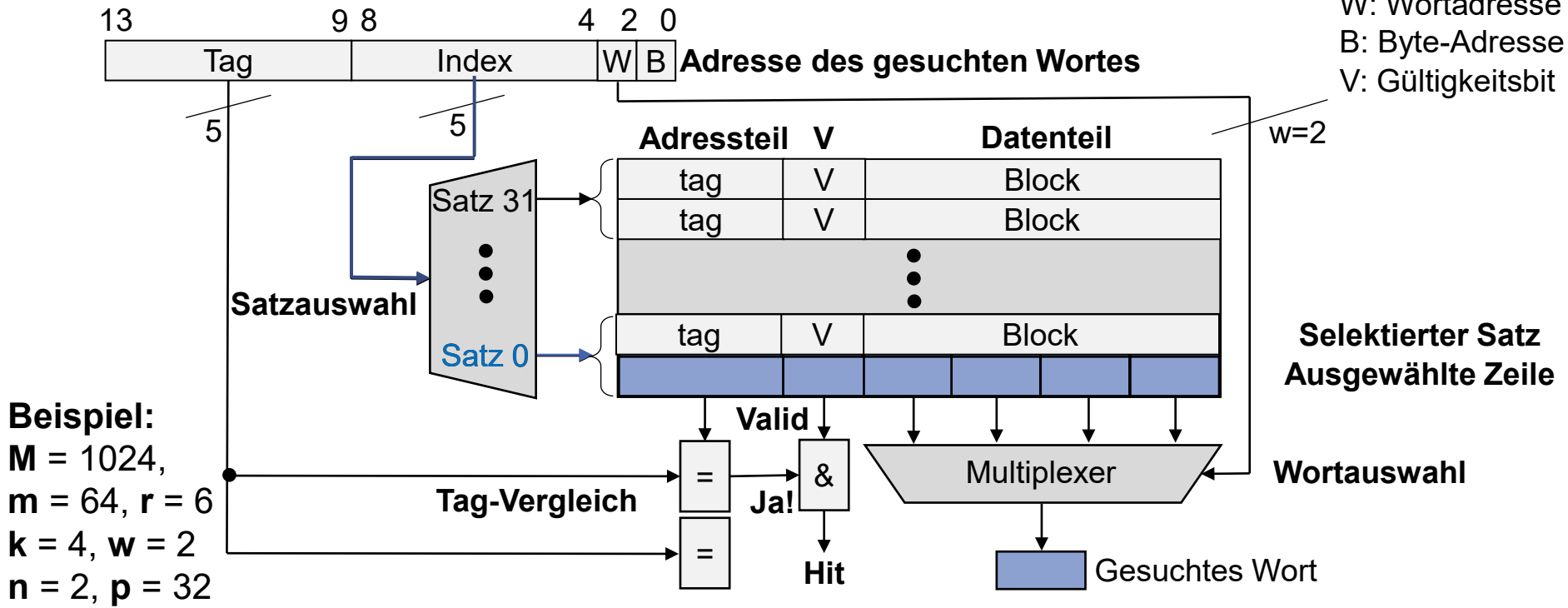
- n-fach satzassoziativer Cache (n-way set-associative Cache, nA)
  - Beispiel: 2-fach satzassoziativer Cache

Beispiel:
M = 1024, **s** = 10
m = 64, **r** = 6
k = 4, **w** = 2
n = 2, **p** = 32



Cache-Organisationsformen

n-fach satzassoziativer Cache (n-way set-associative Cache, nA)



Cache-Organisationsformen

- **n-fach satzassoziativer Cache (n-way set-associative Cache, nA)**
  - **Aufteilung der Adresse:**
    - **Index-Feld**
      - dient zur Auswahl des Satzes.
      - Ein Cache hat $\mathbf{m}$ Cache-Zeilen mit $\mathbf{m = 2^r}$;
      - **Assoziativität:** $\mathbf{n = 2^j}$, mit $\mathbf{0 \leq j \leq r}$.
      - Breite des Index-Feldes ist $\mathbf{r - j}$.
    - **Tag-Feld:**
      - Wert wird mit dem Adressetiketten (tags) in den Cache-Zeilen des ausgewählten Satzes verglichen;
      - Breite des Tag-Feldes ist $\mathbf{a - r + j - w - 2}$, mit $\mathbf{a}$ = Anzahl Adressbits

Cache-Organisationsformen

- **n-fach satzassoziativer Cache (n-way set-associative Cache, nA)**
  - Verbesserte Trefferrate gegenüber direkt abgebildetem Cache, da hier eine Auswahl möglich ist;
    - Ein zu verdrängender Eintrag kann unter allen Zeilen eines Satzes ausgewählt werden;
    - Die Wahrscheinlichkeit von Kollisionen ist kleiner.
  - Zum Auffinden eines Datums werden alle Tags des indizierten Satzes parallel verglichen.
    - Der Aufwand steigt mit der Zahl der Zeilen in einem Satz.
    - Für eine große Anzahl an Zeilen in einem Satz nähert sich der HW-Aufwand dem des voll-assoziativen Caches.
  - ➢ Kompromiss zwischen direkt abgebildetem Cache und voll-assoziativem Cache;

Cache-Organisationsformen

- **Ersetzungsstrategien**
  - bestimmen, für welche Cache-Zeile der Inhalt nach einem Fehlzugriff ersetzt werden soll, damit der neu zu ladende Block abgelegt werden kann.
  - werden in Hardware implementiert;

Ersetzungsstrategien

- **Direct-mapped Cache**
  - Bei einem Fehlzugriff ist für einen angeforderten Block die Cache-Zeile durch die Abbildungsvorschrift eindeutig festgelegt und der Block, der diese Cache-Zeile belegt muss ersetzt werden.
  - Keine Ersetzungsstrategie notwendig!

Ersetzungsstrategien

- **Vollassoziative oder n-fach satz-assoziative Cache-Speicher**
  - Für einen angeforderten Block kann die Cache-Zeile gewählt werden, in der dieser abgelegt werden kann.
    - Bei einem vollassoziativen Cache kann jede Zeile ausgewählt werden.
    - Bei n-fach satz-assoziativen Cache-Speichern können nur die Blöcke des ausgewählten Satzes ausgewählt werden.

Ersetzungsstrategien

- **LRU (least recently used):**
  - Ersetze den Block eines Satzes, auf den am längsten nicht mehr zugegriffen worden ist.

  - **Implementierungsvariante für 2-fach satzassoziative Cache-Speicher**
    - Verwendung eines **USE-Bit**s für jede Cache-Zeile;
      - Bei einem Zugriff auf eine Cache-Zeile eines Satzes wird das USE-Bit dieser Zeile gesetzt und das USE-Bit der anderen Zeile zurückgesetzt.
      - Wenn eine Zeile in einem Satz ausgewählt werden soll, dann wird die Zeile ausgewählt deren USE-Bit nicht gesetzt ist und der Block dieser Zeile wird ersetzt.
      - Führt zu einer vergleichsweisen hohen Trefferrate;

Ersetzungsstrategien

- **LRU (least recently used):**
  - **Implementierungsvariante für n-fach satzassoziative Cache-Speicher**
    - Hält für jeden Satz die Zugriffsordnung auf dessen Zeilen mit Hilfe von Zeitstempeln (Zähler) fest.
      - Bei einem Zugriff auf eine Cache-Zeile wird dessen Zeitstempel inkrementiert.
      - Wenn ein Block ersetzt werden soll, dann wird die Zeile eines Satzes mit dem kleinsten Zeitstempel ausgewählt.
    - Hoher Hardware-Aufwand für $n \geq 4$:
      - Anzahl Bits pro Zeitstempel?
      - Das Auffinden des kleinsten Zeitstempels erfordert für jeden Satz eine Vergleicher-Einheit.

Ersetzungsstrategien

- **LRU (least recently used):**
  - **Implementierungsvariante für vollassoziative Cache-Speicher**
    - Der Cache-Controller verwaltet eine verkettete Liste mit Indices auf alle Zeilen im Cache.
      - Wenn auf einen Block zugegriffen wird, dann wird dessen Zeile auf den Anfang der Liste gesetzt.
      - Wenn ein Block ersetzt werden soll, dann wird die Zeile ausgewählt, die am Ende der Liste steht.
    - Hoher Aufwand, da bei jedem Zugriff auf eine Zeile des Satzes die Liste neu geordnet werden muss.

Ersetzungsstrategien

- **LRU (least recently used):**
  - **Fazit:**
    - Für Cache-Speicher mit einer Assoziativität $n \geq 4$ ist die Implementierung der LRU-Strategie mit einem hohen Hardware-Aufwand verbunden, um das Zugriffsverhalten genau festzuhalten.

Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - Approximation des LRU-Algorithmus
    - Verwendung von Näherungsmaßen für das Alter der Werte im Cache anstelle des exakten Alters der jeweiligen Werte im Cache;
  - Ersetze den Block eines Satzes, auf den *wahrscheinlich* am längsten nicht mehr zugegriffen worden ist.

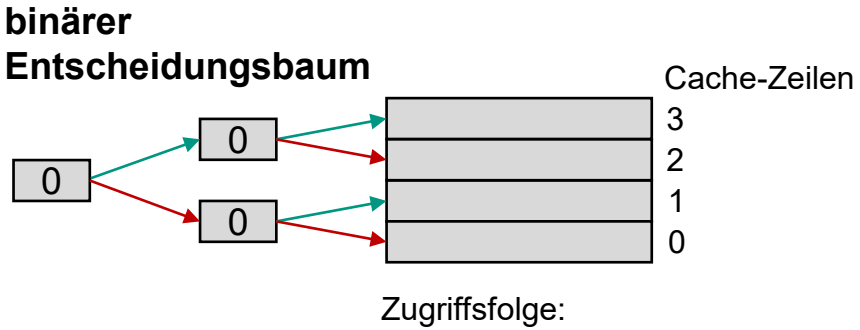
    Oder:
  - Ersetze den Block eines Satzes, auf den *wahrscheinlich* nicht zuletzt zugegriffen worden ist.

Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - Für jeden Satz eines satzassoziativen Cache-Speichers wird ein binärer Baum verwaltet.
      - Die Blattknoten des binären Baums sind den Zeilen des Satzes zugeordnet.
      - Der Wurzelknoten und die Zwischenknoten enthalten jeweils ein Bit, das anzeigt, welcher Unterbaum die aktuelleren und welcher Unterbaum die älteren Werte enthält.
    - Bei einem Treffer werden die Werte der Knoten auf dem Pfad vom Blattknoten zum Wurzelknoten invertiert.
      - Alle Knoten sind dann so gesetzt, dass sie auf einen Unterbaum zeigen, der nicht der zuletzt traversierte Weg ist. Damit wird gesichert, dass die zuletzt zugegriffenen Blöcke wahrscheinlich nicht ersetzt werden.
    - Wenn ein Block aus dem Hauptspeicher in den Cache geladen und ein Block ersetzt werden soll, dann wird in dem entsprechenden Satz die Zeile ausgewählt, die dem Blattknoten auf dem aktuell angezeigten Weg zugeordnet ist.

Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel:**

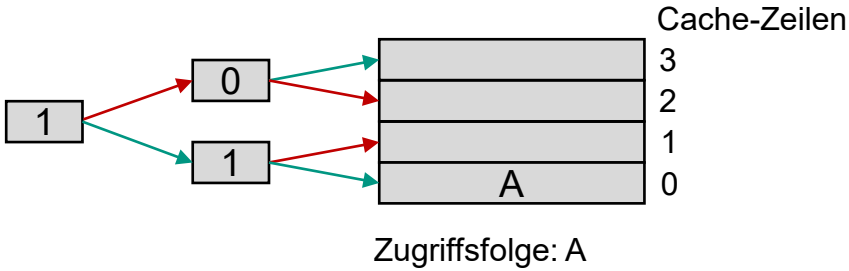


1: Auf eine Cache-Zeile des unteren Teilbaums wurde vor kurzem zugegriffen und auf eine Zeile oberen Teilbaum wurde zu einem früheren Zeitpunkt zugegriffen.

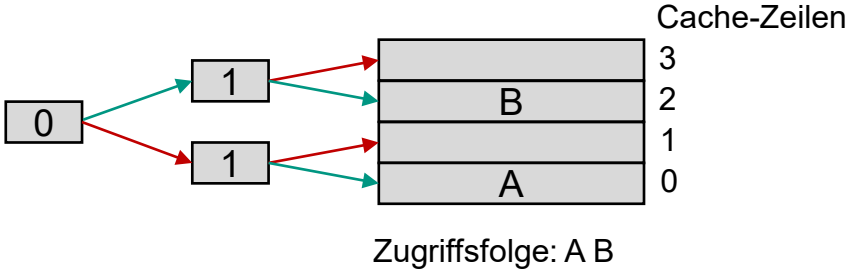
0: Auf eine Cache-Zeile des oberen Teilbaums wurde vor kurzem zugegriffen und auf eine Zeile des unteren Teilbaums wurde zu einem früheren Zeitpunkt zugegriffen.

Ersetzungsstrategien

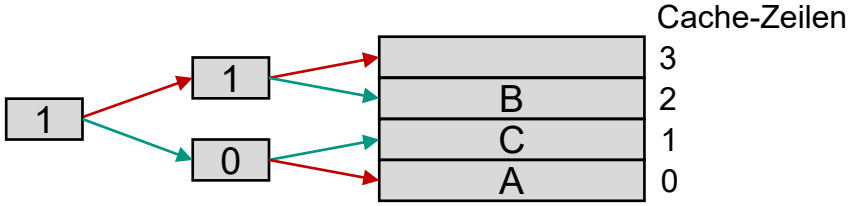
- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel:**



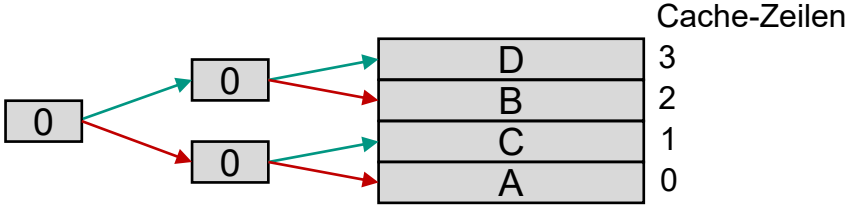
Zugriffsfolge: A



Zugriffsfolge: A B



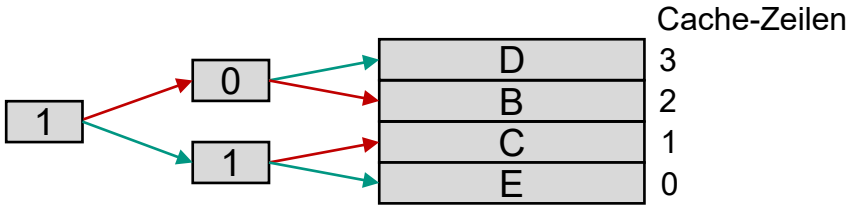
Zugriffsfolge: A B C



Zugriffsfolge: A B C D

Ersetzungsstrategien

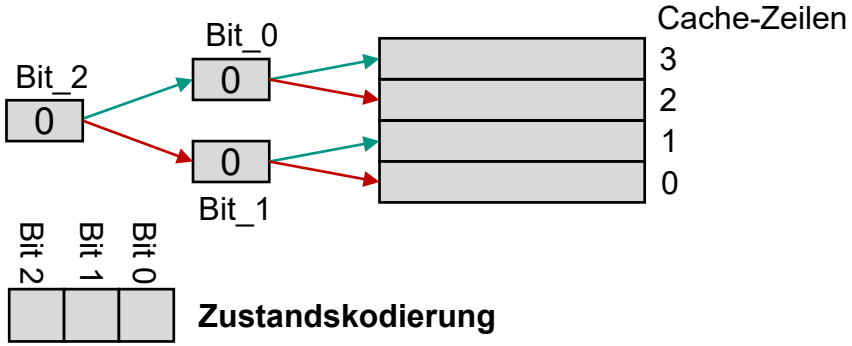
- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel:**



Zugriffsfolge: A B C D E

Ersetzungsstrategien

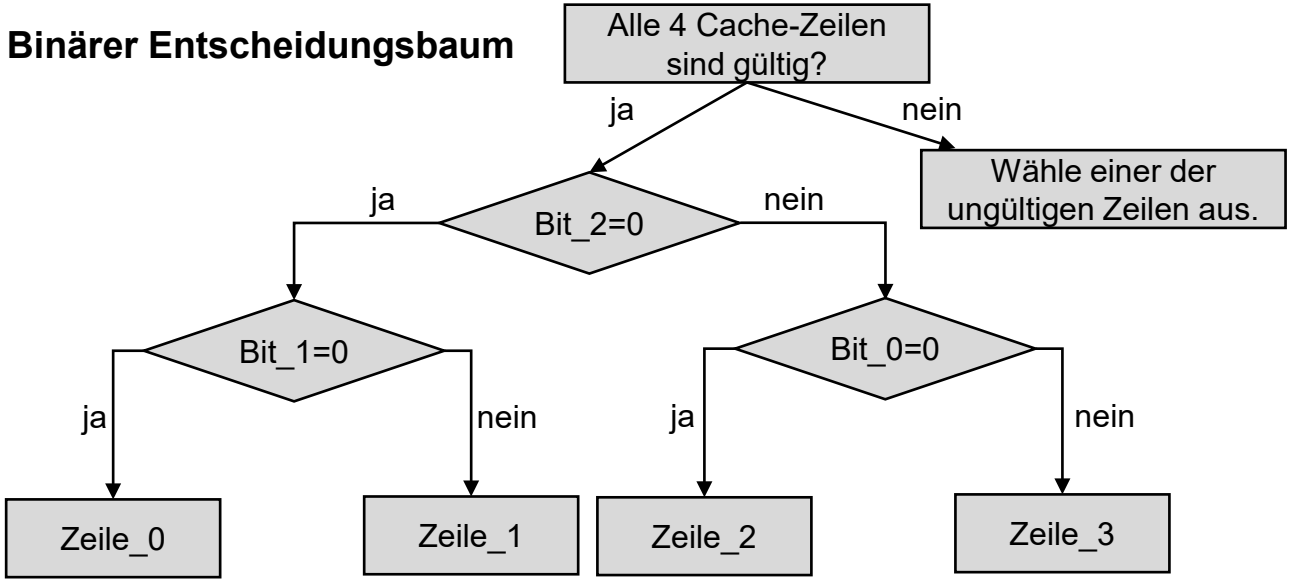
- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel: 4-fach satzassoziative Cache-Speicher**
      - Pro Satz 3 Bits für Zustandskodierung notwendig
      - Jedes Bit repräsentiert einen Knoten (Verzweigungspunkt) im binären Entscheidungsbaum



Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel: 4-fach satzassoziative Cache-Speicher**

Binärer Entscheidungsbaum



Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Implementierung mit Hilfe eines binären Baums (Tree-PLRU)**
    - **Beispiel: 4-fach satzassoziative Cache-Speicher**
      - Aus binären Entscheidungsbaum kann ein Zustandsautomat abgeleitet werden, kann in HW implementiert werden kann.

Zustandstabelle

Zustand	Ersetze
00x	Zeile_0
01x	Zeile_1
1x0	Zeile_2
1x1	Zeile_3

x: don‘t care

Zustandsübergänge

Zugriff auf	Folge-zustand
Zeile_0	11*
Zeile_1	10*
Zeile_2	0*1
Zeile_3	0*1

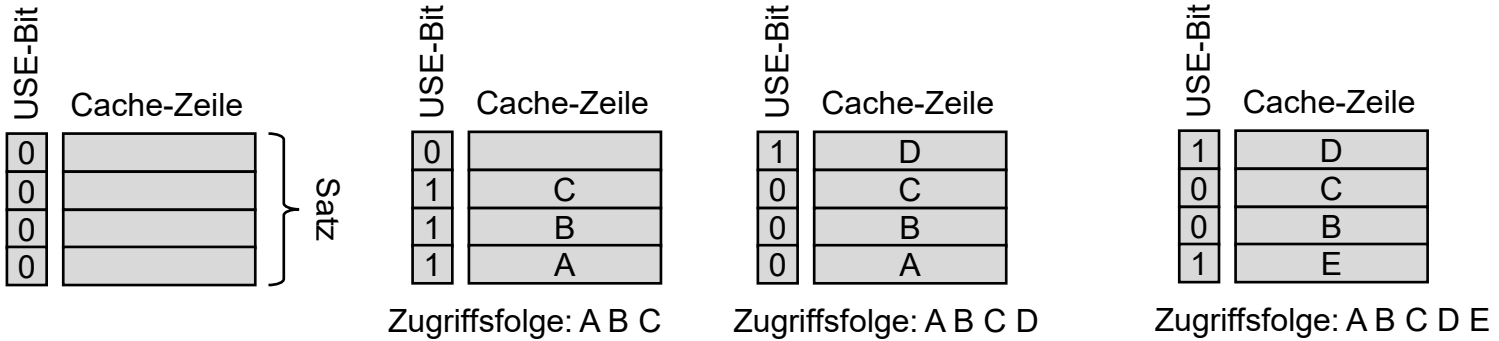
*: bleibt unverändert

Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Beispiel: Implementierung mit Hilfe eines USE-Bits (Bit-PLRU)**
    - Für jeden Satz des Cache-Speichers wird eine Menge von USE-Bits verwaltet, wobei ein USE-Bit jeweils einer Zeile des Satzes zugeordnet ist.
    - Bei einem Zugriff auf einen Satz wird das USE-Bit, das der Zeile zugeordnet ist, die den gewünschten Block enthält, gesetzt.
    - Wenn das letzte verbliebene USE-Bit, das zurückgesetzt ist, gesetzt wird, werden alle anderen USE-Bits zurückgesetzt.
    - Wenn ein Block ersetzt werden muss, dann wählt die Cache-Steuerung den Block der Cache-Zeile, dessen zugeordnetes USE-Bit zurückgesetzt ist.
    - Wenn mehrere USE-Bits zurückgesetzt sind, dann erfolgt die Auswahl zufällig.
      - Beispiel: Die „am weitesten unten angeordnete“ Zeile wird ausgewählt, dessen USE-Bit, zurückgesetzt ist.

Ersetzungsstrategien

- **LRU-Approximation (Pseudo-LRU, PLRU):**
  - **Beispiel: Implementierung mit Hilfe eines USE-Bits (Bit-PLRU)**



Ersetzungsstrategien

- **LRU / LRU-Approximationen**
  - **Fazit:**
    - LRU-Approximationen liefern annähernd so gute Leistungsergebnisse wie LRU.

    ➢ In Prozessoren werden überwiegend LRU-Approximationen anstelle von LRU-Algorithmen implementiert.

Cache-Organisationsformen

- **Ersetzungsstrategien**
  - Es sind eine Vielzahl von Ersetzungsstrategien untersucht worden, die Nutzungsinformationen für die Auswahl eines zu ersetzenden Blocks verwenden:
  - **Beispiele:**
    - **FIFO (first-in, first-out):**
      - Ersetze den Block eines Satzes, der am längsten im Cache ist.
      - Kann einfach mit Hilfe eines Ringpuffers oder round-robin-Strategie implementiert werden.
    - **LFU (least frequently used):**
      - Ersetzte den Block eines Satzes, auf den am wenigsten häufig zugegriffen worden ist.
    - **MFU (Most Recently used)**
      - Ersetze den Block eines Satzes, auf den erst kürzlich zugegriffen worden ist.

Cache-Organisationsformen

- **Ersetzungsstrategien**
  - **Random, zufälliges Ersetzen:**
    - Ersetze den Block einer zufällig ausgewählten Zeile eines Satzes.
    - Einfache Implementierung in der Hardware;
    - Die Fehlzugriffsraten (Leistung) sind im Allgemeinen nur geringfügig höher als bei LRU oder anderen auf Nutzungsinformationen basierenden Verfahren.

Cache-Organisationsformen

- **Ersetzungsstrategien**
  - **Bélády's Algorithmus** (László Bélády, IBM Systems Journal, 1966)
    - Ersetze den Block eines Satzes, der am längsten nicht mehr gebraucht werden wird.
      - Nur theoretisch relevant, weil Zukunftswissen erforderlich;
      - Aber, er liefert ein optimales Ergebnis, womit die Qualität anderer Strategien bei Experimenten bewertet werden kann.

Cache-Organisationsformen

- **Ersetzungsstrategien**
  - **Fazit:**
    - Bei Cache-Speichern mit großen Kapazitäten sinkt grundsätzlich die Fehlzugriffsrate und eine einfach zu implementierende Ersetzungsstrategie liefert hinreichend gute Leistungsergebnisse.