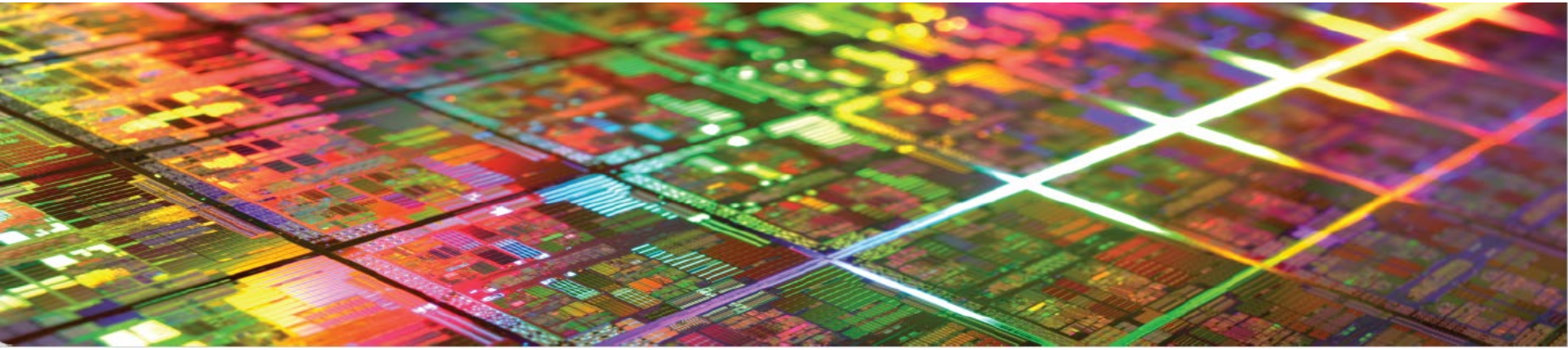
Karlsruher Institut für Technologie

Rechnerorganisation

Prof. Dr. Wolfgang Karl

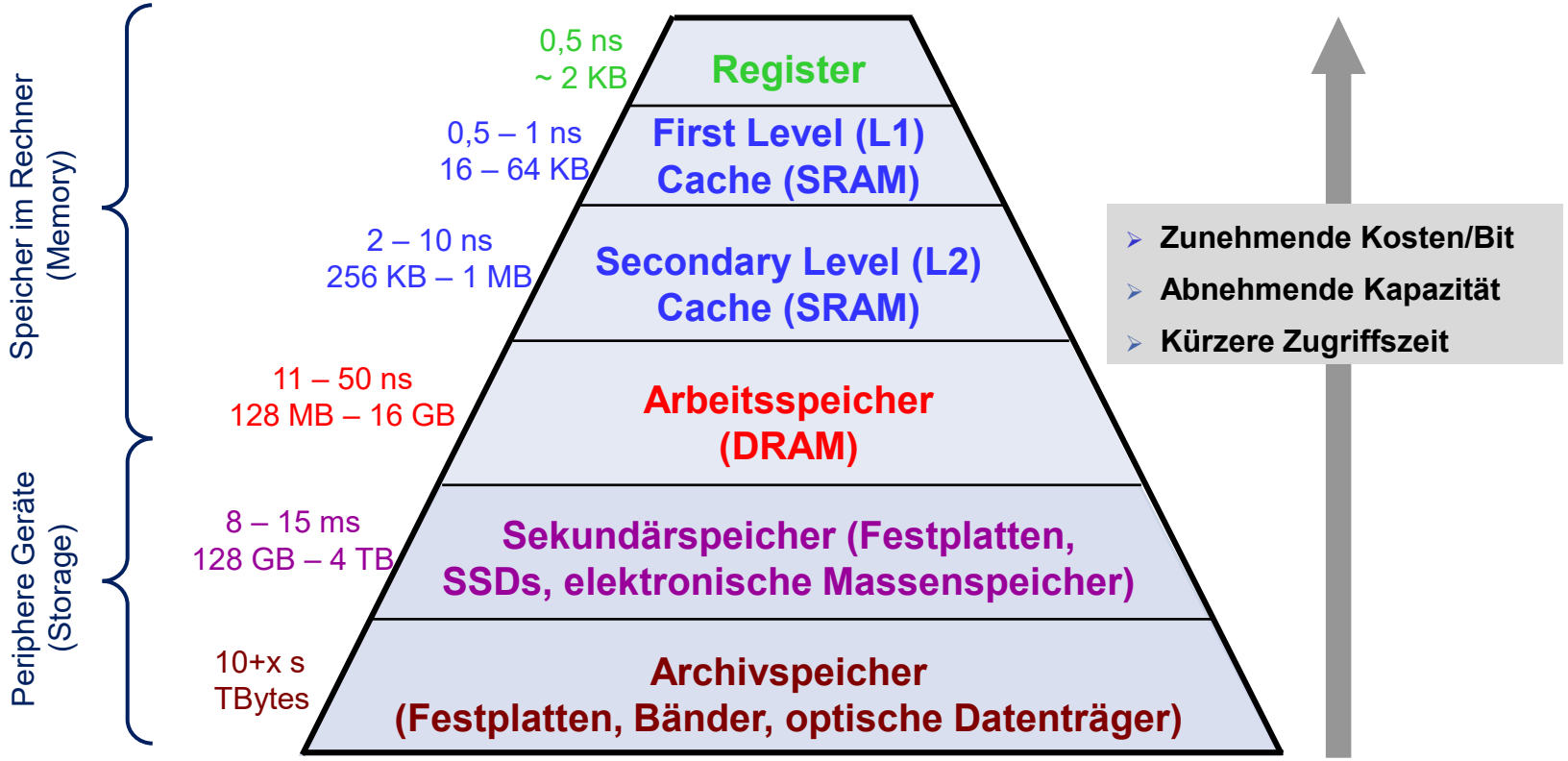
Vorlesung im Wintersemester 2025/2026 – Foliensatz: RO25-FS11



Kapitel 8
Cache-Speicher

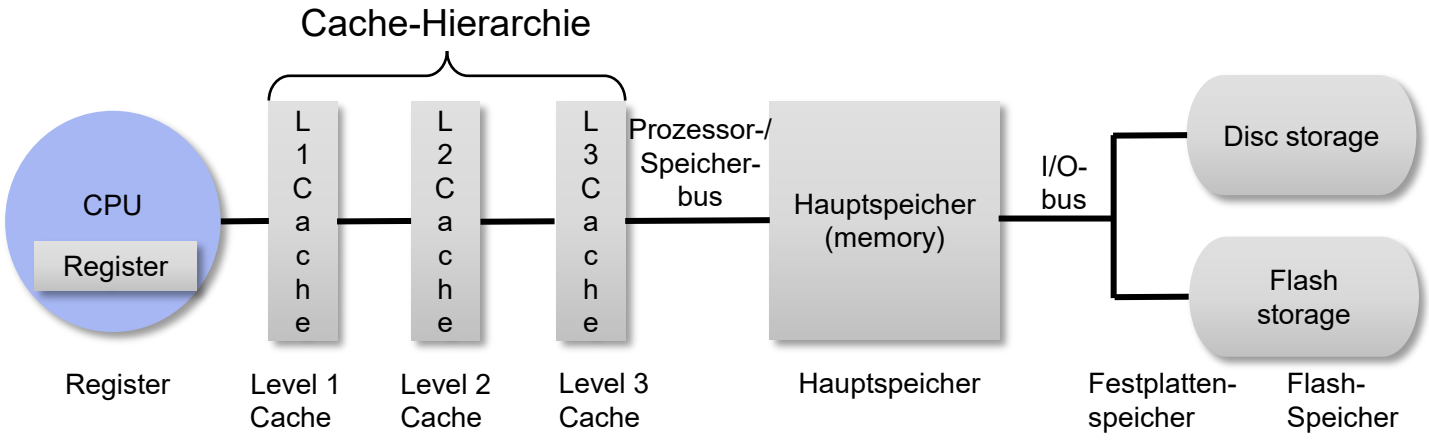
- **Speicherhierarchie**
- Systemaufbau mit Cache-Speicher
- Grundlegende Arbeitsweise
- Cache-Organisationsformen
- Grundlegende Fragen beim Entwurf
- Gültigkeitsproblem

Speicherhierarchie



Speicherhierarchie

- **Systemaufbau**
  - **Beispiel: Server**



Kapazität:	4000 bytes	64 KB	256 KB	16 -64 MB	32 -256 GB	16 -64 TB	1 -16 TB
Geschwindigkeit:	200 ps	1 ns	3 -10 ns	10 -20 ns	50 – 100 ns	5 – 10 ms	100 – 200 µs

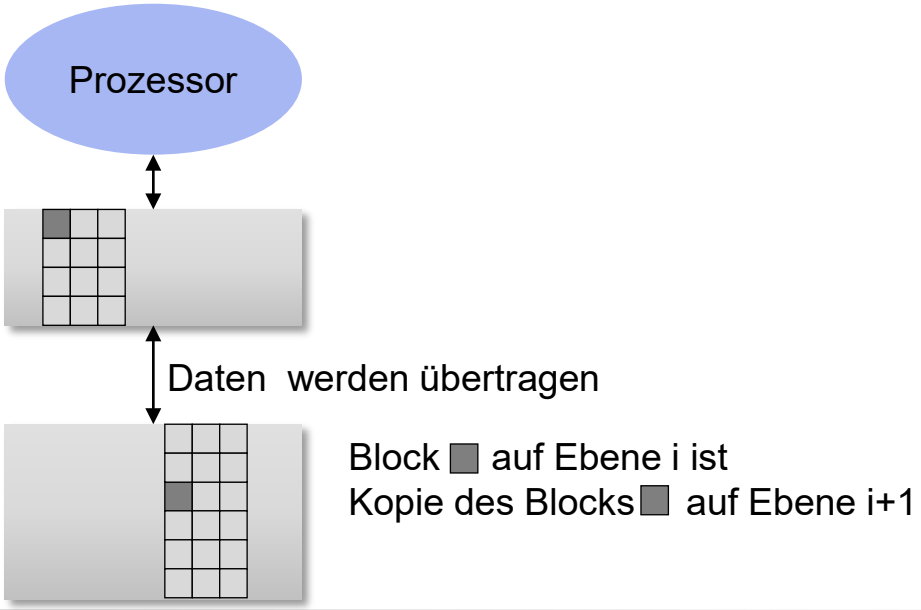
Quelle: Hennessy, J.; Patterson, D.: Computer Architecture A Quantitative Approach. Morgan Kaufman Publishers, 6th Edition, 2019

Speicherhierarchie

- **Zusammenspiel verschiedener Speicherkomponenten mit unterschiedlichen Technologien**
  - Die Speicherhierarchie wirkt für den Benutzer wie ein großer und schneller Speicher.
  - Daten werden nur jeweils gleichzeitig zwischen zwei benachbarten Ebenen übertragen.
  - Ein **Block** ist die kleinste Informationseinheit, die in der zweistufigen Hierarchie vorhanden oder nicht vorhanden sein kann.

Speicherhierarchie

- **Zusammenspiel verschiedener Speicherkomponenten mit unterschiedlichen Technologien**
  - Datentransfer zwischen zwei Ebenen der Speicherhierarchie



Speicherhierarchie

- **Datentransfer zwischen zwei Ebenen der Speicherhierarchie**
  - **Terminologie**
    - **Treffer (hit)**
      - Die vom Prozessor geforderten Daten sind in einem Block, der im Speicher auf der oberen Ebene vorhanden ist.
    - **Fehlzugriff (miss)**
      - Die vom Prozessor gefordeten Daten werden nicht im Speicher auf der oberen Ebene gefunden
      - Es erfolgt ein Zugriff auf den Speicher der untere Ebene in der Hierarchie, um den Block mit den angeforderten Objekten zu finden

Speicherhierarchie

- **Datentransfer zwischen zwei Ebenen der Speicherhierarchie**
  - **Terminologie**
    - **Trefferrate (hit rate)**
      - Anteil der Speicherzugriffe, bei denen der gesuchte Block im Speicher auf einer Ebene gefunden wird
    - **Fehlzugriffsrate (miss rate)**
      - Anteil der Speicherzugriffe, bei denen der gesuchte Block nicht im Speicher auf einer Ebene gefunden wird

Speicherhierarchie

- **Datentransfer zwischen zwei Ebenen der Speicherhierarchie**
  - **Terminologie**
    - **Zugriffszeit bei Treffer (hit time)**
      - Die Zeit für den Zugriff auf den Speicher der oberen Ebene und
      - die Zeit, die benötigt wird, um festzustellen, ob der Zugriff ein Treffer ist oder nicht.
    - **Fehlzugriffsaufwand (miss penalty)**
      - Die Zeit, die benötigt wird, um einen Block von der unteren Ebene in die obere Ebene der Speicherhierarchie zu laden;
        - Umfasst die Zeit für die Übertragung und das Einfügen des Blocks in den Speicher der höheren Ebene, auf der der Fehlzugriff stattgefunden hat.

Speicherhierarchie

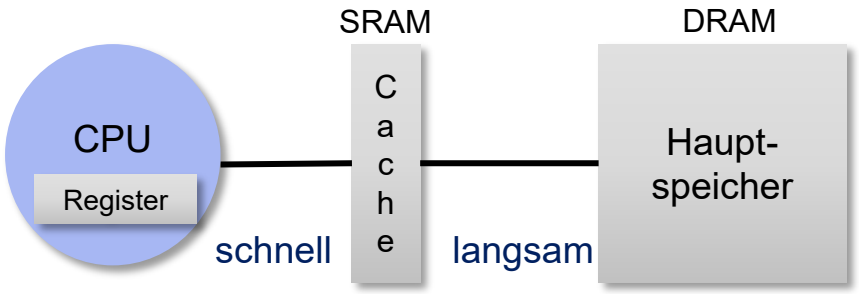
- **Ausnützen des Lokalitätsprinzips bei der Programmabarbeitung**
  - Programme greifen zu jedem Zeitpunkt nur auf einen kleinen Bereich ihres Adressraums zu;
  - **Zeitliche Lokalität (temporal locality)**
    - Tendenz des Prozessors, auf Objekte zuzugreifen, auf die er erst kürzlich zugegriffen hat
    - Beispiel: Schleifeniterationen: Prozessor greift wiederholt auf dieselbe Befehlsfolge zu;
    - Beispiel: häufig benutze Variablen;
  - **Räumliche Lokalität (spatial locality)**
    - Tendenz des Prozessors, dass, wenn er auf ein Objekt zugegriffen hat, er auch auf Objekte zugreift, deren Adressen nahe der des zugegriffenen Objekts sind;
    - Beispiel: sequentielle Programmausführung
    - Beispiel: Zugriff auf Datenstrukturen wie Felder

Speicherhierarchie

- **Ausnützen des Lokalitätsprinzips bei der Programmabarbeitung**
  - **Virtueller Speicher**
    - Vorspiegelung eines Hauptspeichers großer Kapazität
      - Kapazität des physikalischen Speichers ist begrenzt
      - Kapazitätserweiterung durch Hintergrundspeicher
    - Bereitstellen der zu einem Zeitpunkt benötigten Objekte im Hauptspeicher
      - Der Block mit den benötigten Objekten wird vom Hintergrundspeicher als Kopie in den Hauptspeicher geladen.
    - Stellt jeweils einen großen und einheitlichen Adressraum für die einzelnen Prozesse bereit;
    - Die Verwaltung ist Aufgabe des Betriebssystems und wird vom Prozessor unterstützt.

Speicherhierarchie

- **Ausnützen des Lokalitätsprinzips bei der Programmabarbeitung**
  - **Cache-Speicher**
    - Kleiner schneller Speicher, der als Puffer für einen langsameren, aber größeren Speicher dient;
      - Enthält Kopien der Objekte im Hauptspeicher;
    - Ist mit SRAM-Bausteinen aufgebaut



Speicherhierarchie

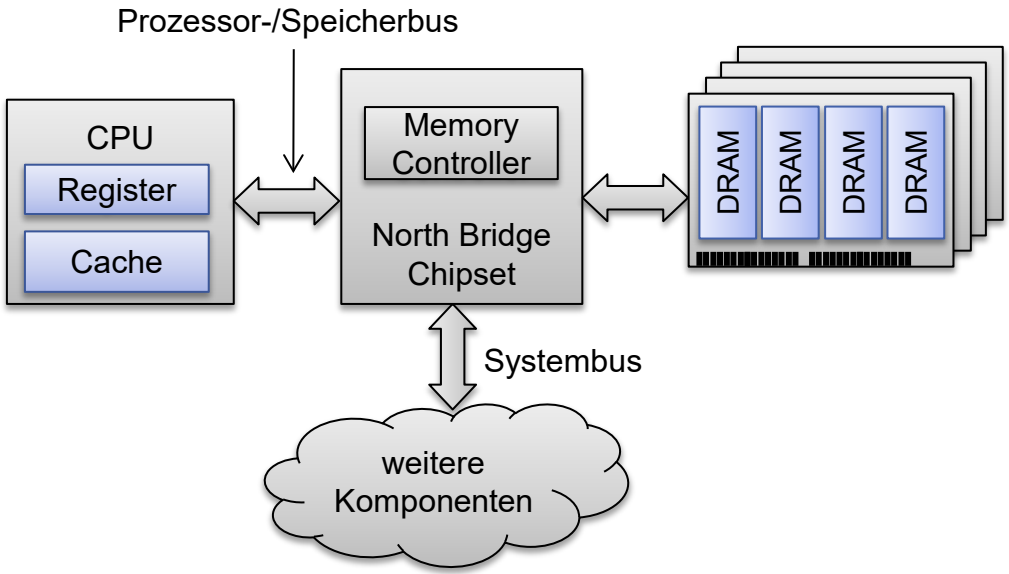
- **Zusammenspiel verschiedener Speicherkomponenten mit unterschiedlichen Technologien**
  - **Leistungsfähigkeit der Hierarchie ist bestimmt durch**
    - die Eigenschaften der Speicherkomponenten (Zugriffsart, Zugriffszeiten, Kapazität)
    - der Adressierung der Speicherplätze und
    - der Organisation des Betriebs
  - Konzepte für den Aufbau des Speichersystems wirken sich auf die Leistung des gesamten Systems aus.

Kapitel 8
Cache-Speicher

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

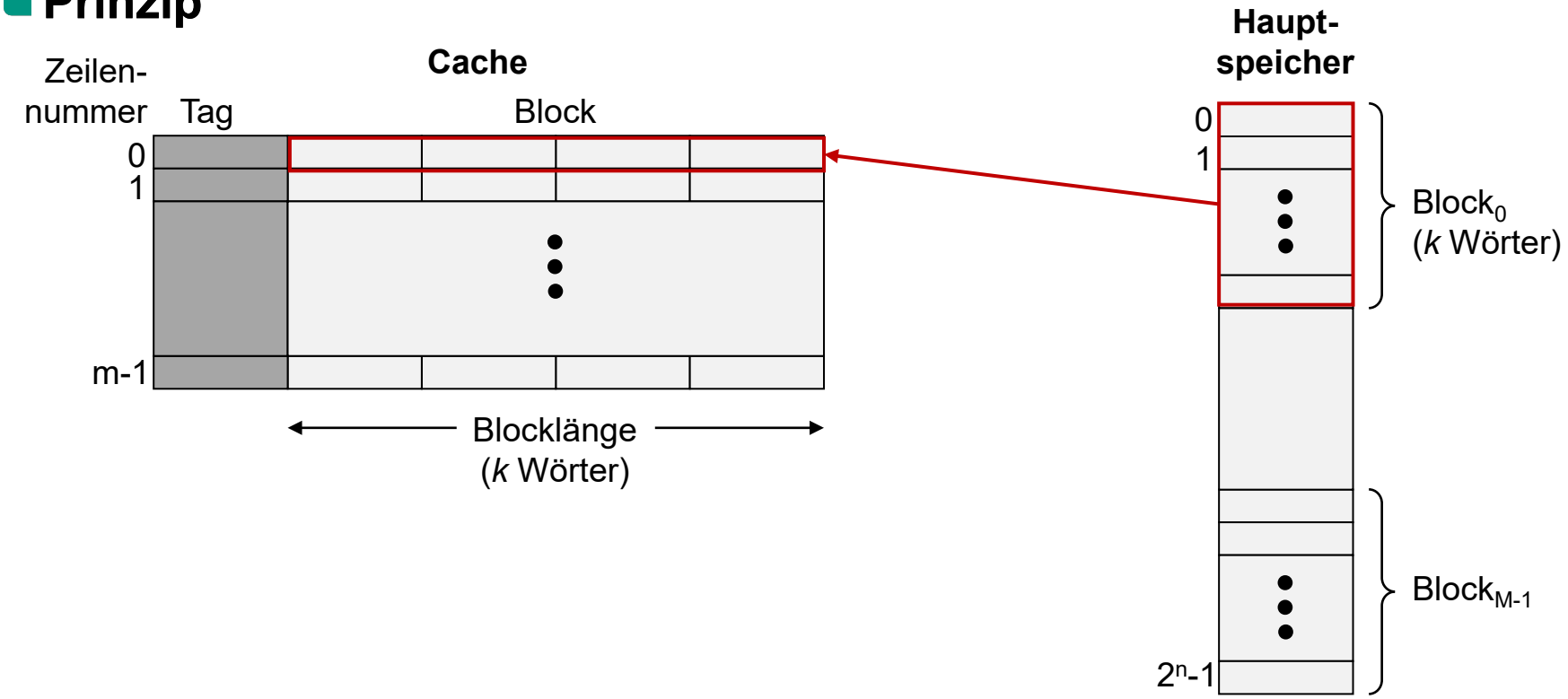
Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
  - **Beispiel: klassischer PC**



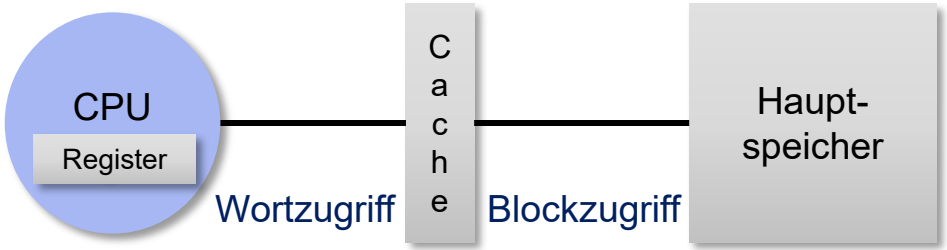
Cache-Speicher

Prinzip



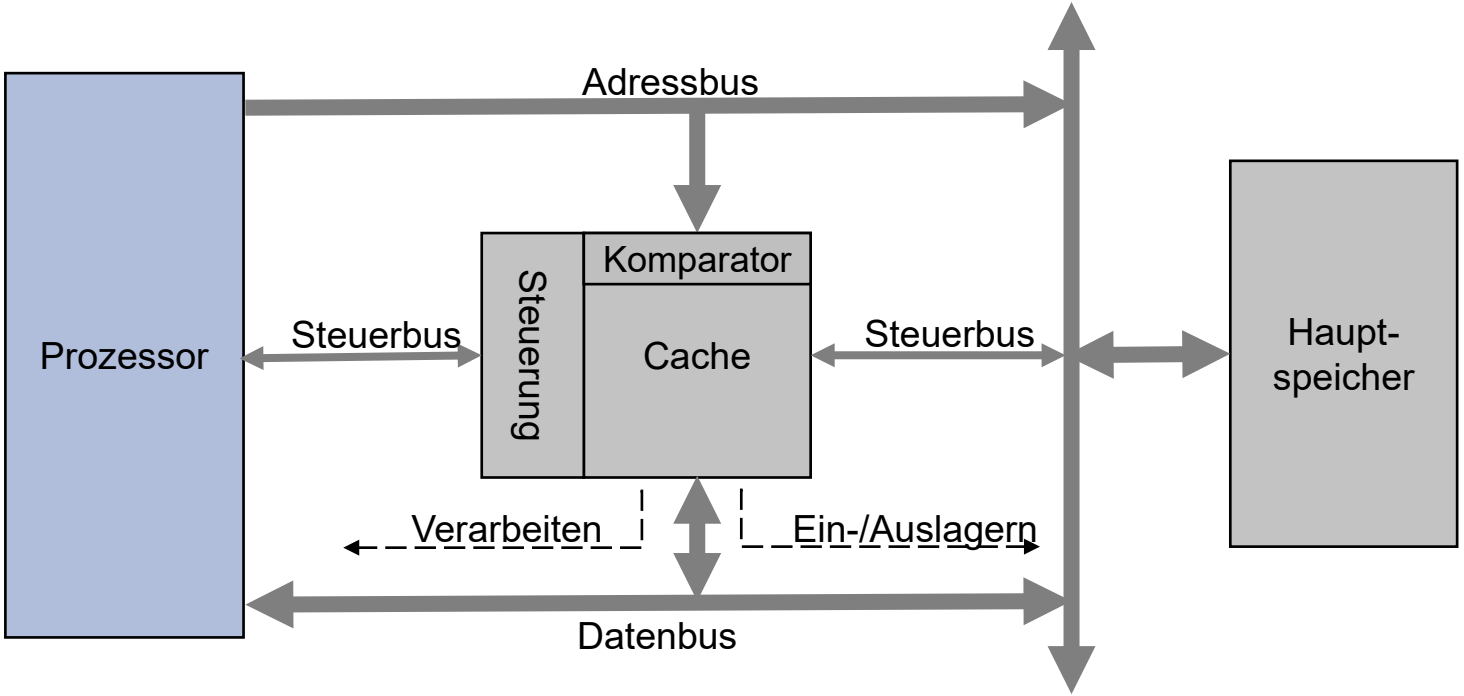
Cache-Speicher

- **Prinzip**
  - Der Cache enthält eine Teilmenge der Blöcke des Hauptspeicher als Kopien.
  - Wenn der Prozessor auf ein Speicherwort zugreifen will, findet eine Überprüfung statt, um zu bestimmen, ob das Wort im Cache ist:
    - Falls ja, wird das Wort an den Prozessor geliefert (Treffer);
    - Falls nein, wird ein Block, bestehend aus einer festen Anzahl von Wörtern aus dem Hauptspeicher geholt und in den Cache geladen, und dann wird das Wort an den Prozessor geliefert
  - Ausnützen des Lokalitätsprinzips



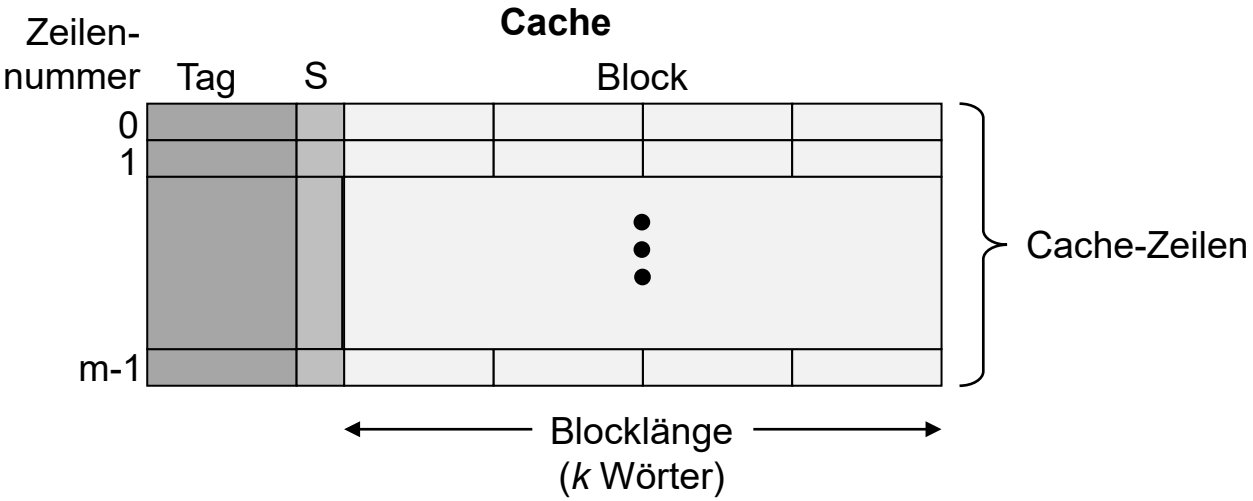
Cache-Speicher

Systemaufbau mit Cache-Speicher



Cache-Speicher

Aufbau eines Cache-Speichers



Cache-Speicher

- **Aufbau eines Cache-Speichers**
  - Ein Cache besteht aus *m* Zeilen
  - **Cache-Zeile**
    - **Datenteil:**
      - Enthält einen Block mit *k* Wörtern
    - **Adressteil:**
      - **Tag (Adressetikett):**
        - identifiziert, welcher Block in der Zeile enthalten ist
        - Teil der Hauptspeicheradresse des Speicherblocks, der für die im Block enthaltenen Wörter gleich ist (gemeinsamer Adressteil, MSB)
    - **Statusbits (S)**
      - Valid (V): zeigt an, ob die Zeile gültige Daten enthält
      - Dirty (D): zeigt an, ob ein oder mehrere Wörter der Zeile verändert worden sind

Cache-Speicher

- **Aufbau eines Cache-Speichers**
  - **Komparator**
    - Überprüft, ob die auf dem Adressbus liegende Adresse (und somit auch das dazugehörige Datum) im Cache vorhanden worden ist
    - Adressvergleich mit den Tags im Adressteil
      - Dieser Adressvergleich muss sehr schnell gehen (möglichst in einem Taktzyklus), da sonst der Cache-Speicher effektiv langsamer wäre als der Arbeitsspeicher.

Cache-Speicher

- **Aufbau eines Cache-Speichers**
  - **Cache-Steuerung /Cache-Controller**
    - Übernimmt die Steuerungsaufgaben für den Cache
      - Sorgt dafür, dass der Cache-Speicher in der Regel das Datum enthält, auf das der Prozessor als nächstes zugreift.
      - Weniger häufig benötigte Inhalte der Cache-Zeilen werden bei Bedarf nach verschiedenen Strategien aus dem Cache verdrängt.
    - Verschiedene Strategien für das Laden, Aktualisieren und Adressieren des Inhalts.

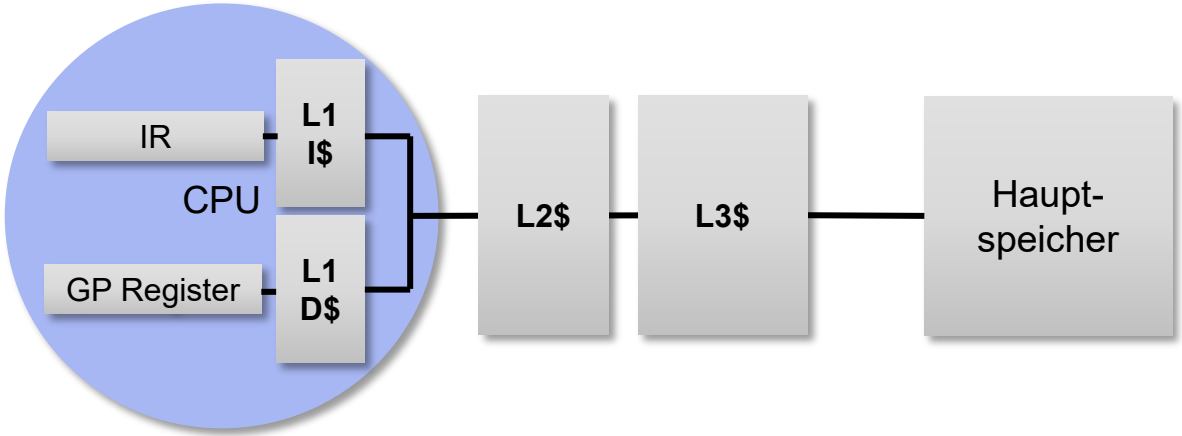
Cache-Speicher

- **Cache-Hierarchie**
  - Die Größe eines Cache bestimmt die Zugriffszeit:
    - L1 Cache klein, kann bei Treffer mit Taktrate des Prozessors mithalten
    - L2, L3 Caches können Zugriffe auf den Hauptspeicher auffangen

Kapazität:	64 KB	256 KB	16 -64 MB
Geschwindigkeit:	1 ns	3 -10 ns	10 -20 ns

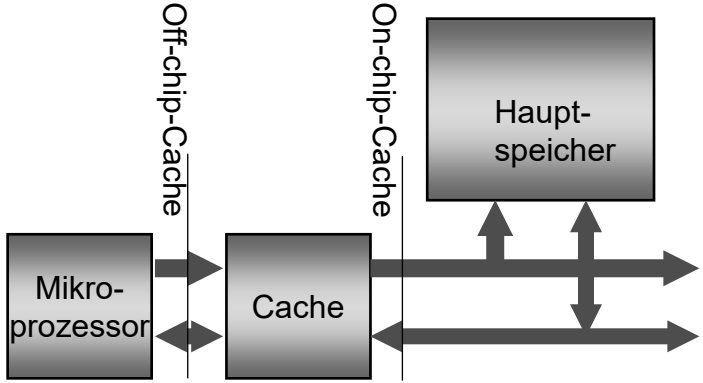
Cache-Speicher

- **Split-Caches**
  - Getrennte L1 Caches für Befehle (L1 I Cache, L1I$) und Daten (L1 D Cache, L1D$)
    - Unterschiedliches Lokalitätsverhalten von Befehlen und Daten
    - Paralleler Zugriff auf Befehle und Daten
  - Harvard-Konzept



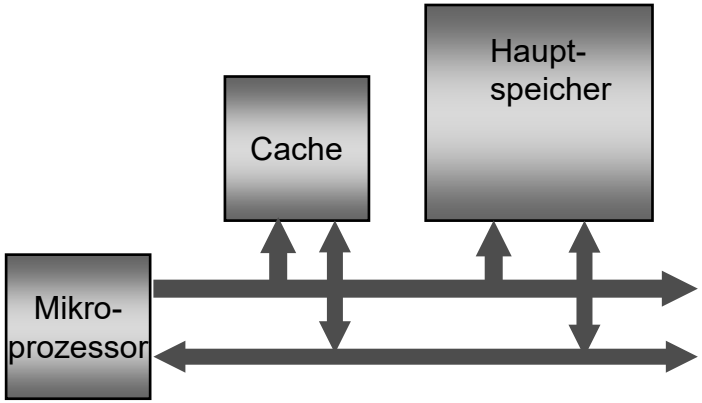
Cache-Speicher

- **Anbinding des Cache-Speichers**
  - **Look-through Cache**
    - Prozessor, Cache und Hauptspeicher sind in Reihe angeordnet.
    - Typisch für Rechensysteme, bei denen mehrere Prozessor-/Cache-Einheiten auf einen gemeinsamen Speicherbus zugreifen;
    - Zugriffsanforderungen der Prozessoren werden von den Cache-Speichern abgefangen und von der jeweiligen Cache-Steuereinheit nur dann an den Hauptspeicher weitergegeben, wenn diese nicht vom Cache beantwortet werden können.



Cache-Speicher

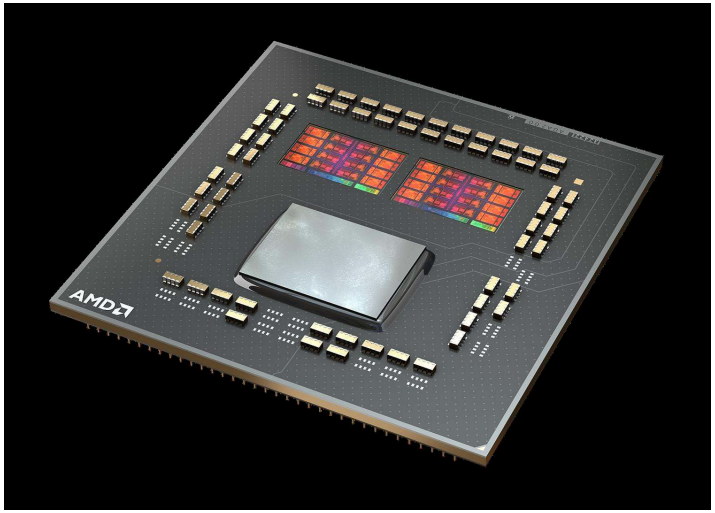
- **Anbinding des Cache-Speichers**
  - **Look-aside Cache**
    - Cache und Hauptspeicher werden parallel am Speicherbus betrieben.
    - Zugriffsanforderungen des Prozessors geht gleichzeitig an den Cache und an den Hauptspeicher
    - Kann die Anforderung vom Cache beantwortet werden, wird der Hauptspeicherzugriff gestoppt, wenn nicht, hat Hauptspeicherzugriff schon begonnen;



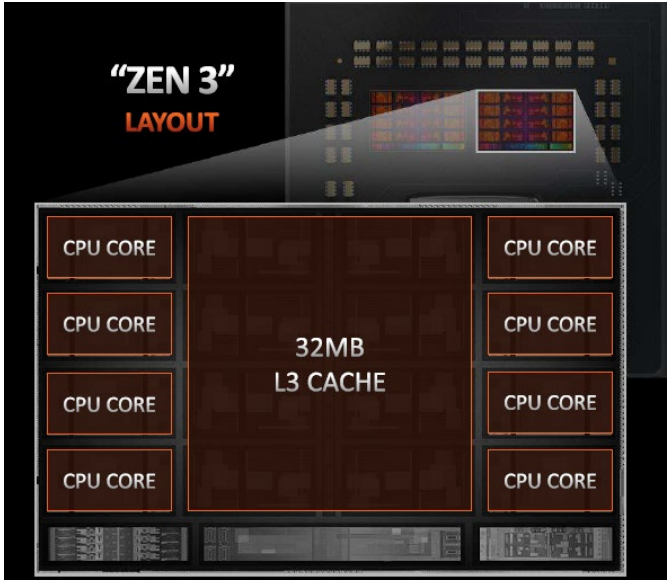
Cache-Speicher

- **Ausblick: Fallstudie AMD ZEN3 (2021)**

Multi Chip Module



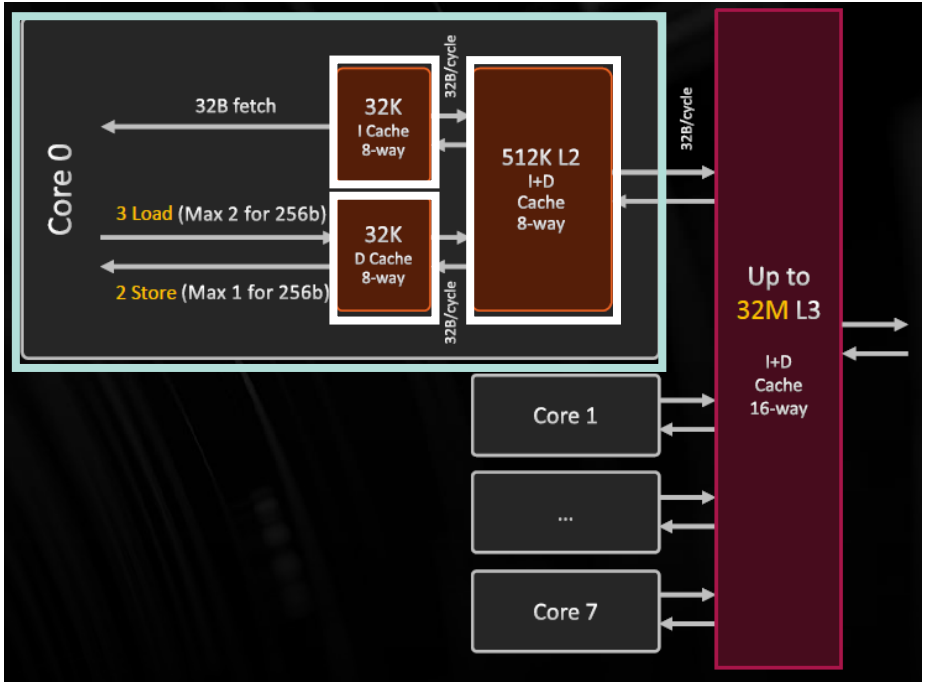
Prozessor Chip



Quelle: https://hc33.hotchips.org/assets/program/conference/day1/HC2021.C1.2%20AMD%20Mark%20Evers.pdf

Cache-Speicher

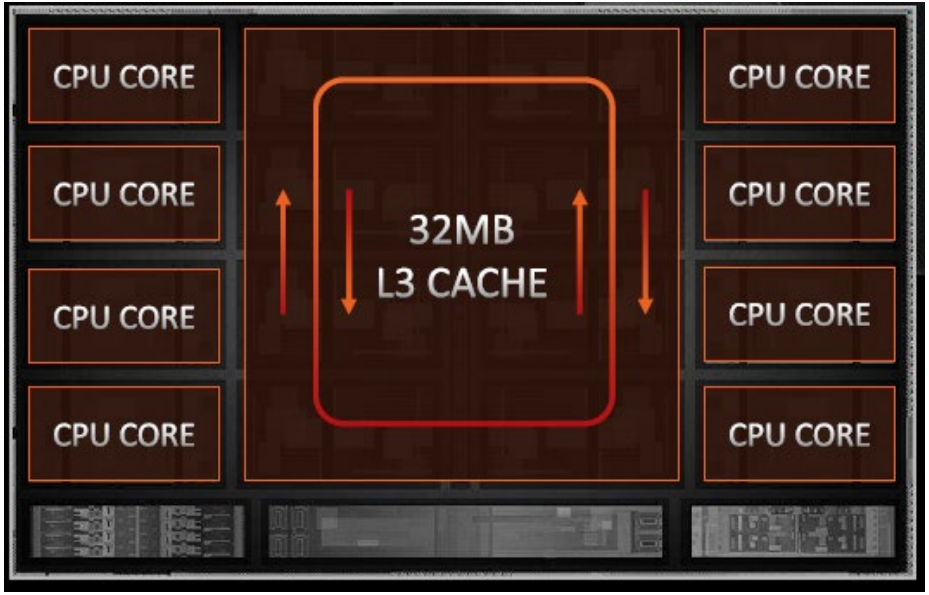
- **Ausblick: Fallstudie AMD ZEN3 (2021)**
  - **Cache-Hierarchie**
    - Pro Prozessorkern (Core)
      - Getrennte L1I$, L1D$
      - L2 Cache für Befehle und Daten



Quelle: https://hc33.hotchips.org/assets/program/conference/day1/HC2021.C1.2%20AMD%20Mark%20Evers.pdf

Cache-Speicher

- **Ausblick: Fallstudie AMD ZEN3 (2021)**
  - **Cache-Hierarchie**
    - Pro Prozessorkern (Core)
      - Getrennte L1I$, L1D$
      - L2 Cache für Befehle und Daten
    - Gemeinsamer L3 Cache



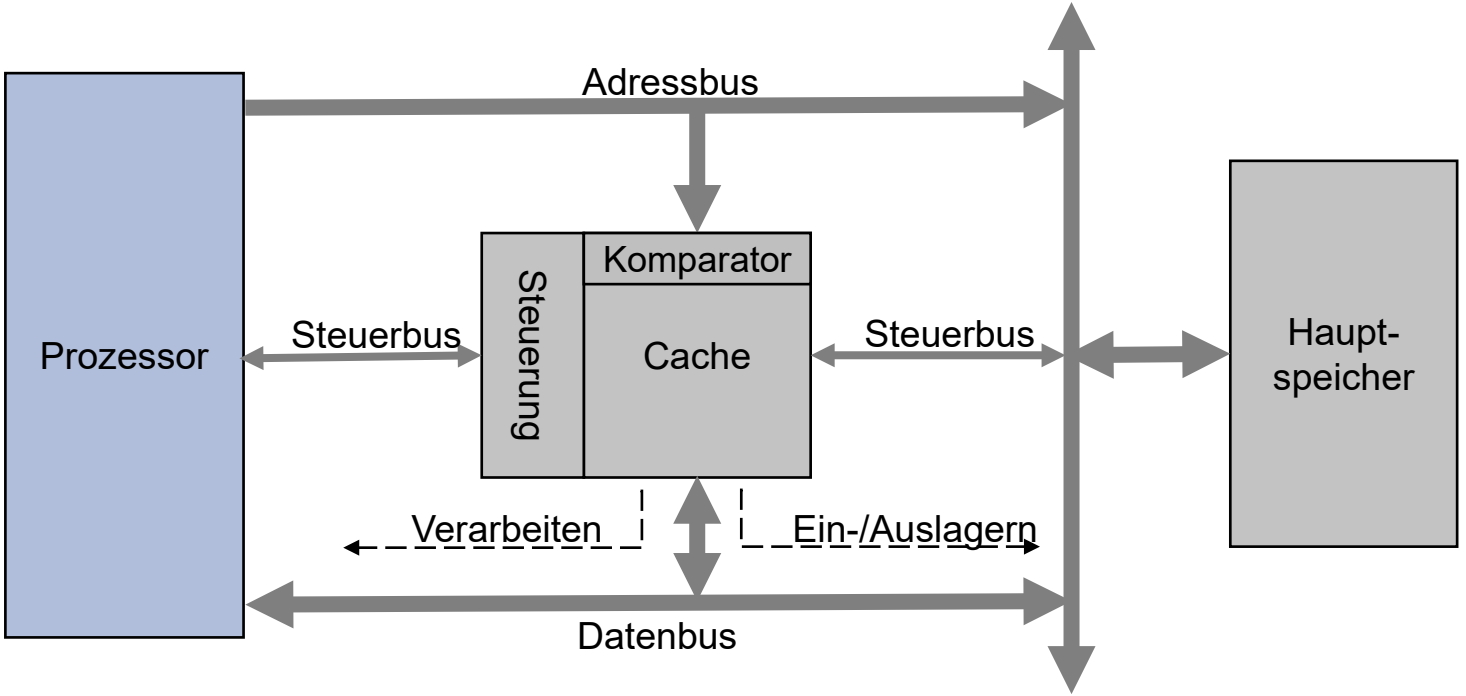
Quelle: https://hc33.hotchips.org/assets/program/conference/day1/HC2021.C1.2%20AMD%20Mark%20Evers.pdf

Kapitel 8
Cache-Speicher

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

Cache-Speicher

Systemaufbau mit Cache-Speicher

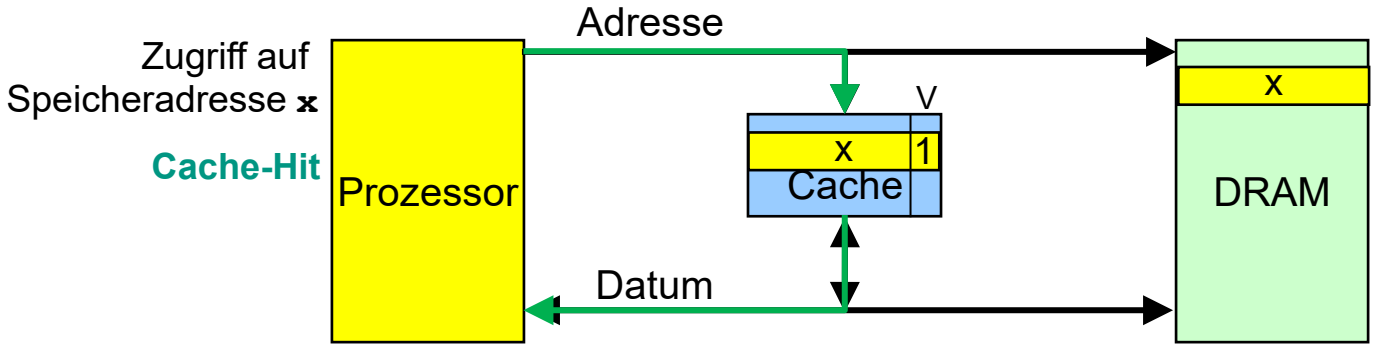


Cache-Speicher

- **Arbeitsweise**
  - **Zugriffe auf den Cache-Speicher**
    - Cache-Steuerung prüft bei Speicherzugriffen des Prozessors, ob
      - der zur Speicheradresse gehörende Hauptspeicherinhalt als Kopie im Cache steht (**Bedingung 1**) und
      - dieser Cache-Eintrag durch das Valid-Bit als gültig gekennzeichnet ist (**Bedingung 2**).

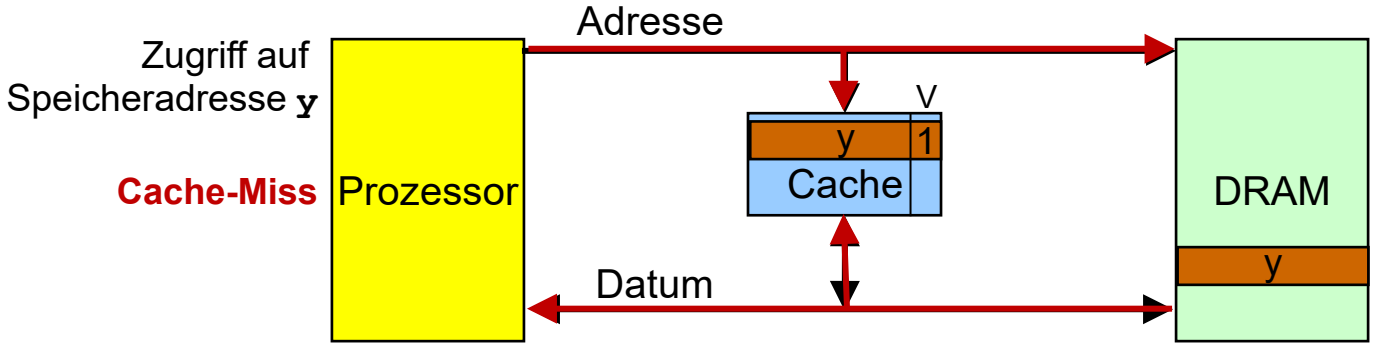
Cache-Speicher

- **Arbeitsweise**
  - **Zugriffe auf den Cache-Speicher**
    - **Treffer (Cache-Hit):**
      - Beide Bedingungen sind erfüllt.
      - Der Zugriff erfolgt auf den Cache.



Cache-Speicher

- **Arbeitsweise**
  - **Zugriffe auf den Cache-Speicher**
    - **Fehlzugriff (Cache-Miss):**
      - Eine der beiden oder beide Bedingungen sind **nicht** erfüllt.
      - Der Zugriff erfolgt auf Hauptspeicher (bzw. auf nächste Cache-Ebene).
      - Der Speicherblock mit dem Wort wird in den Cache geladen.
      - Das Wort wird an den Prozessor geliefert.



Cache-Speicher

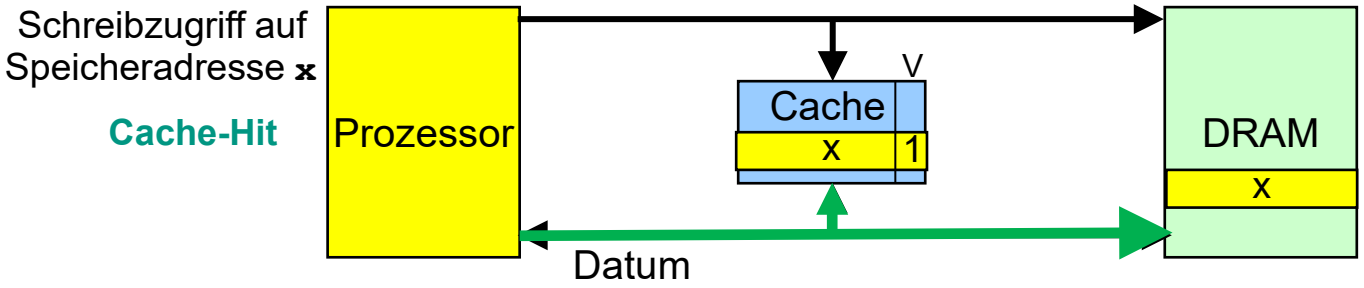
- **Arbeitsweise**
  - **Aktionen bei Fehlzugriffen**
    - **Lesezugriff (read-miss)**
      - Der Speicherblock mit dem Datum wird aus dem Hauptspeicher geholt und in eine Zeile des Cache-Speichers geladen.
      - Die Adressinformation wird im Adressteil der entsprechenden Zeile des Cache-Speichers geladen.
      - Der Eintrag der Zeile wird durch Setzen des Valid-Bits (V) als gültig gekennzeichnet.

Cache-Speicher

- **Arbeitsweise**
  - **Aktionen bei Fehlzugriffen**
    - **Schreibzugriff (write-miss):**
      - Die Aktualisierungsstrategie bestimmt, ob
        - der entsprechende Block in den Cache geladen und dann mit dem zu schreibenden Datum aktualisiert wird (write-allocate), oder ob
        - nur der Hauptspeicher aktualisiert wird und der Cache unverändert bleibt (write-no-allocate)

Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-through-Verfahren**
    - **Treffer:**
      - Der Cache und der Hauptspeicher werden aktualisiert

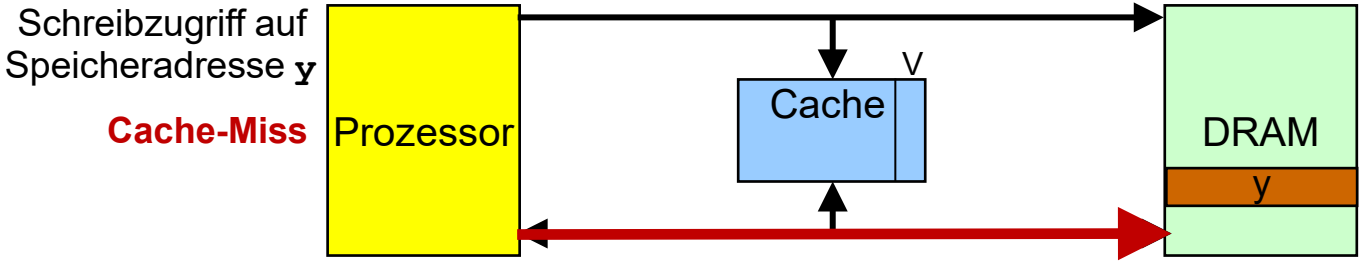


Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-through-Verfahren**
    - **Fehlzugriff (write-miss, write-no-allocate)**
      - Nur der Hauptspeicher wird aktualisiert
      - **Vorteil:** Cache- und Arbeitsspeicher sind aktuell (Cache Inhalt ist echte Teilmenge vom Arbeitsspeicher)
      - **Nachteil:** Schreibzugriffe benötigen immer die langsame Zykluszeit des Hauptspeichers und belasten den Systembus
    - Variante: **Buffered Write-through**
      - Die zu schreibenden Daten werden zuerst in einen kleinen Schreib-Puffer geschrieben und von dort später in den Hauptspeicher übertragen

Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-through-Verfahren**
    - **Fehlzugriff (write-miss, write-no-allocate)**
      - Nur der Hauptspeicher wird aktualisiert

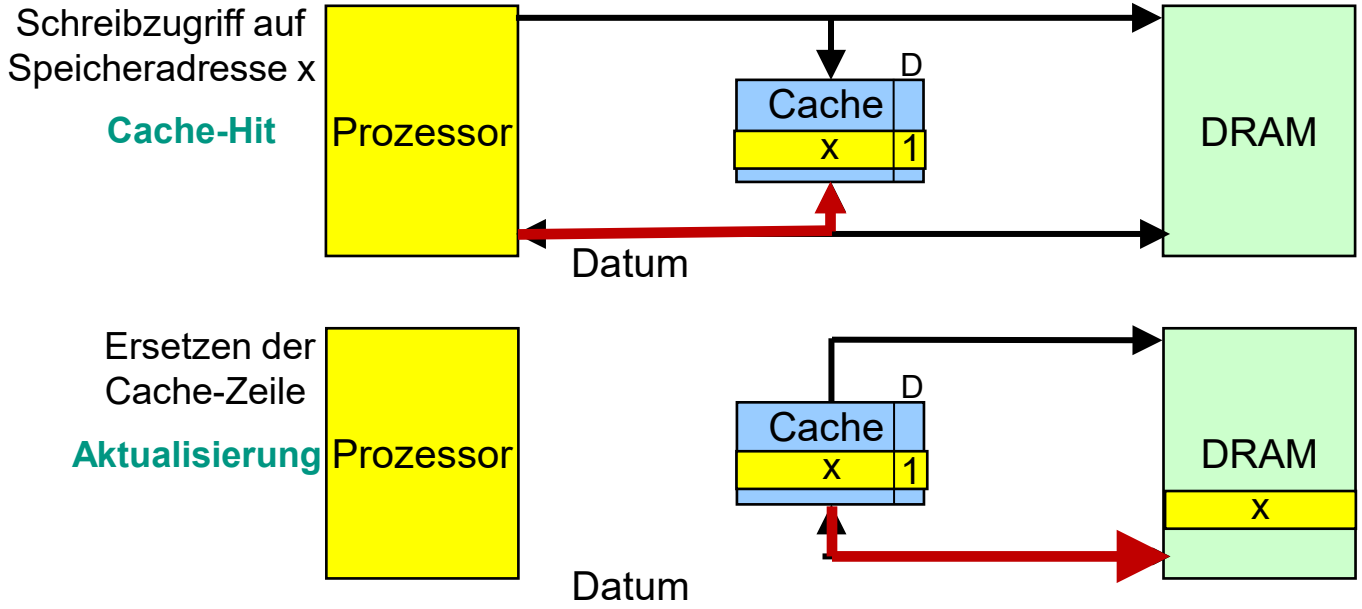


Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-back-Verfahren (copy-back)**
    - **Treffer (write-hit):**
      - Das Datum wird von der CPU nur in den Cachespeicher geschrieben und durch ein spezielles Bit (**Dirty-Bit**) gekennzeichnet.
      - Der Arbeitsspeicher wird erst dann aktualisiert, wenn eine als „dirty“ gekennzeichnete Cache-Zeile aus dem Cache verdrängt werden soll.
      - **Vorteil:**
        - Schreibzugriffe können mit der schnellen Cache-Zykluszeit abgewickelt werden.
        - Teilweise deutlich geringere Last auf dem Prozessor-/Speicherbus, da vom Programm mehrmals überschriebene Adressen nur einmal zum Hauptspeicher übertragen werden müssen.
      - **Nachteil:**
        - Hauptspeicher enthält nicht die aktuellen Daten.

Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-back-Verfahren (copy-back)**
    - **Treffer (write-hit):**

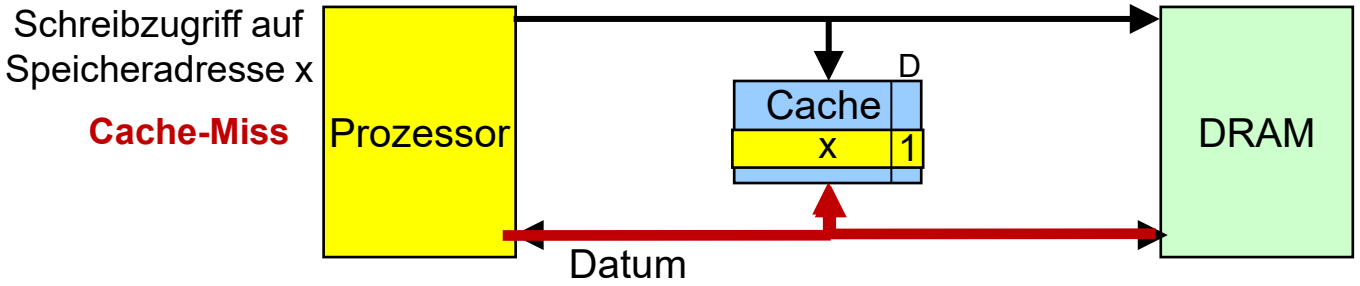


Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-back-Verfahren (copy-back)**
    - **Fehlzugriff (write-miss):**
      - Der Block mit dem zu verändernden Wort wird vom Hauptspeicher geholt und in eine Zeile des Cache-Speichers geladen.
        - Falls die zu ausgewählte Zeile mit einem Dirty-Bit gekennzeichnet ist, muss der Block dieser Zeile zuerst in den Hauptspeicher zurückgeschrieben werden.
      - Das zu schreibende Wort wird in die ausgewählte Cache-Zeile geschrieben.

Cache-Speicher

- **Aktualisierungsstrategien bei Schreibzugriffen**
  - **Write-back-Verfahren (copy-back)**
    - **Fehlzugriff (write-miss):**



Cache-Speicher

Aktualisierungsstrategien bei Schreibzugriffen

Cache-Zugriff	Write-Through	Write-Back
Read-Hit	Cache-Datum → CPU	Cache-Datum → CPU
Read-Miss	Ggf. muss im Cache eine Zeile verdrängt werden: einfach invalidieren; HS-Datenblock & Tag → Cache; HS-Datum → CPU; 1 → V	Ggf. muss eine Cache-Zeile verdrängt werden. Falls Dirty: Cache-Zeile → HS; HS-Datenblock & Tag → Cache; HS-Datum → CPU; 1 → V, 0 → D
Write-Hit	CPU-Datum → Cache & HS	CPU-Datum → Cache, 1 → D
Write-Miss	CPU-Datum → HS (ggf. auch in Cache)	Ggf. muss eine Cache-Zeile verdrängt werden. Falls Dirty: Cache-Zeile → HS; HS-Datenblock & Tag → Cache; 1 → V; CPU-Datum → Cache; 1 → D

Cache-Speicher

- **Metriken zur Bewertung der Cache-Leistung**
  - Die **Trefferrate (hit rate)** bezeichnet die Trefferquote im Cache:

$$\text{Trefferrate} = \frac{\text{Anzahl Treffer}}{\text{Anzahl Zugriffe}}$$

  - Die **Fehlzugriffsrate (miss rate)** bezeihnet den Anteil der Cache-Zugriffe, bei denen der Block mit dem adressierten Objekt nicht im Cache vorhanden ist.

$$\text{Fehlzugriffsrate} = 1 - \text{Trefferrate}$$

Cache-Speicher

- **Metriken zur Bewertung der Cache-Leistung**
  - Die **mittlere Zugriffszeit** t_{access} berechnet sich annähernd wie folgt:

$$t_{access} = \text{Trefferrate} * t_{hit} + (1 - \text{Trefferrate}) * t_{miss}$$

  - mit t_{hit}: Zugriffszeit auf den Cache
    t_{miss}: Zugriffszeit auf den Hauptspeicher
    (genauer: Zeit für die Zugriffe auf alles, was nach dem Cache kommt; ggf. erst eine weitere Cache-Ebene)