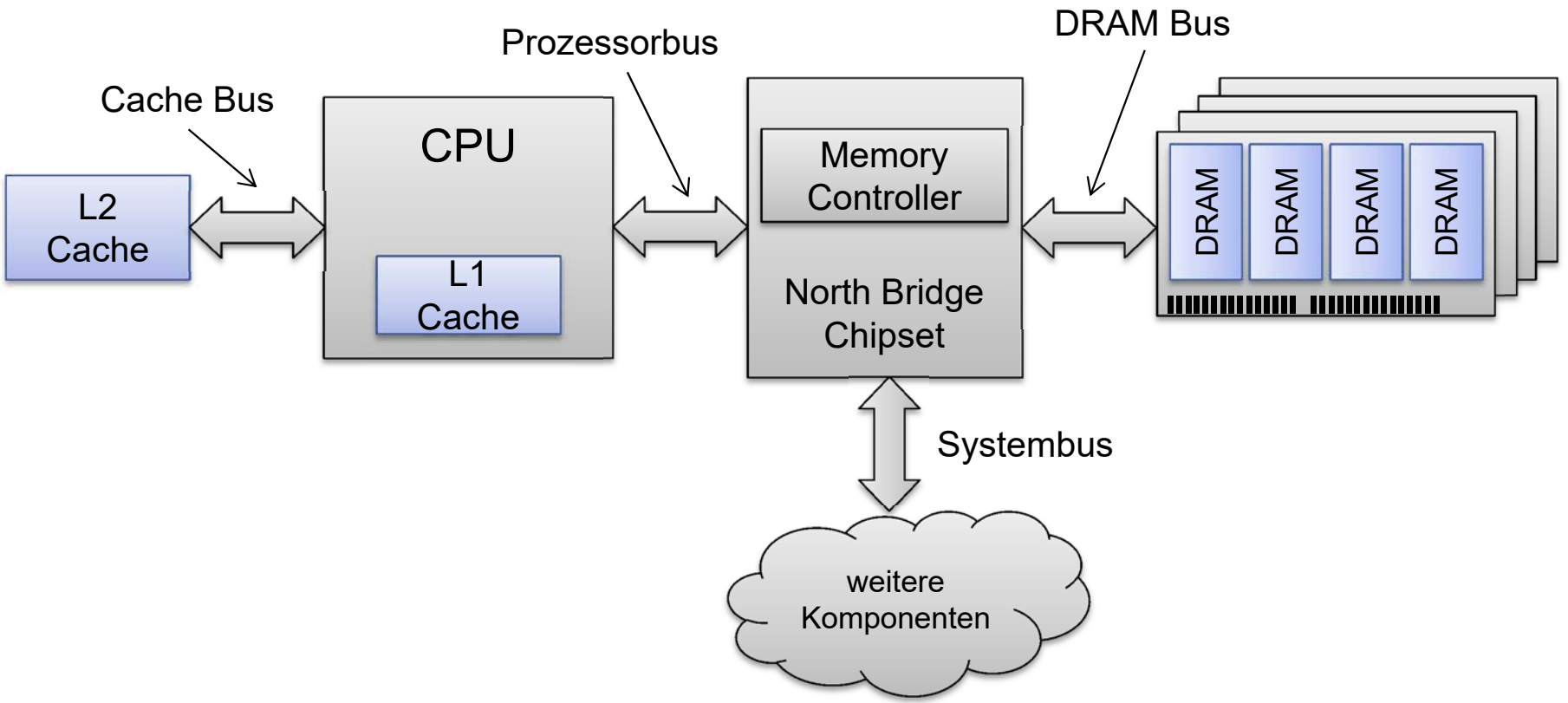
Kapitel 9
Cache-Speicher

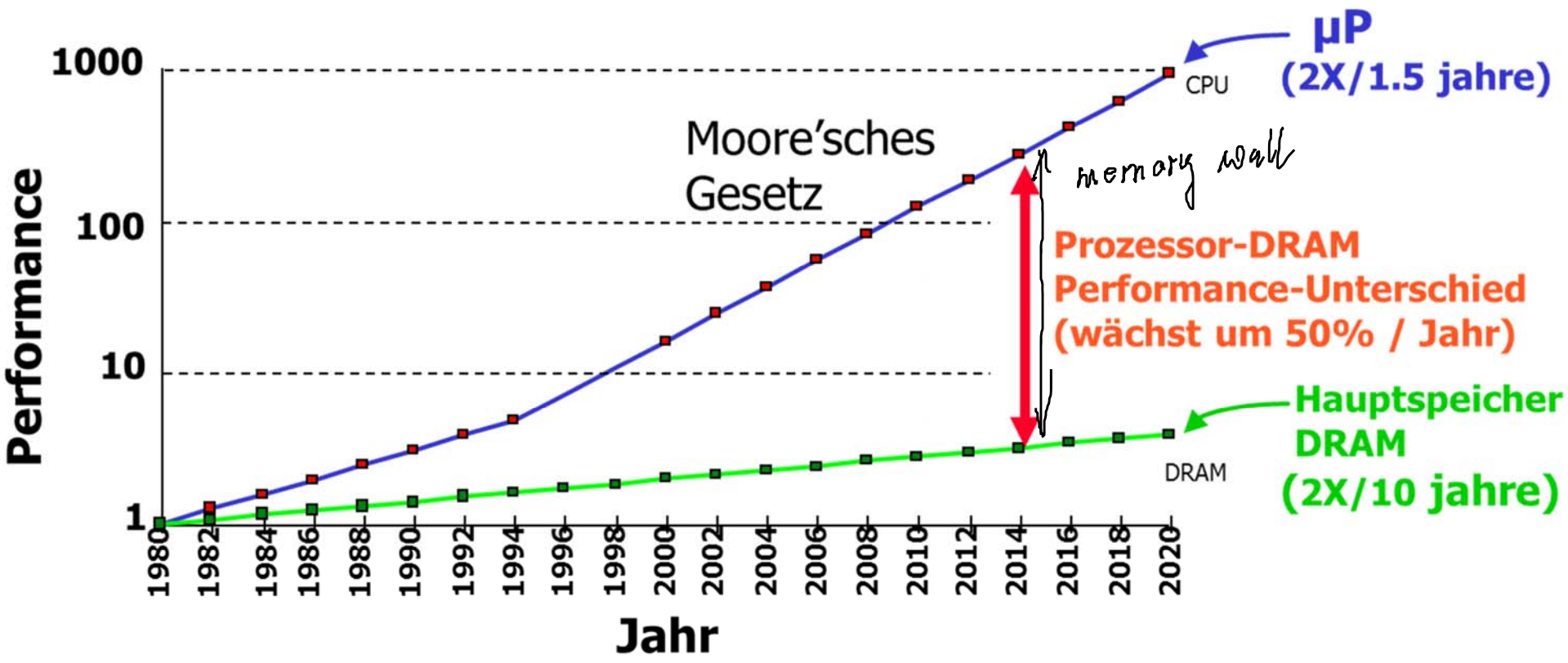
9.1 Speicherhierarchie

- **Beispiel: Aufbau klassischer PC**



9.1 Speicherhierarchie

- **Unterschied zwischen Prozessor- und Speichergeschwindigkeit**



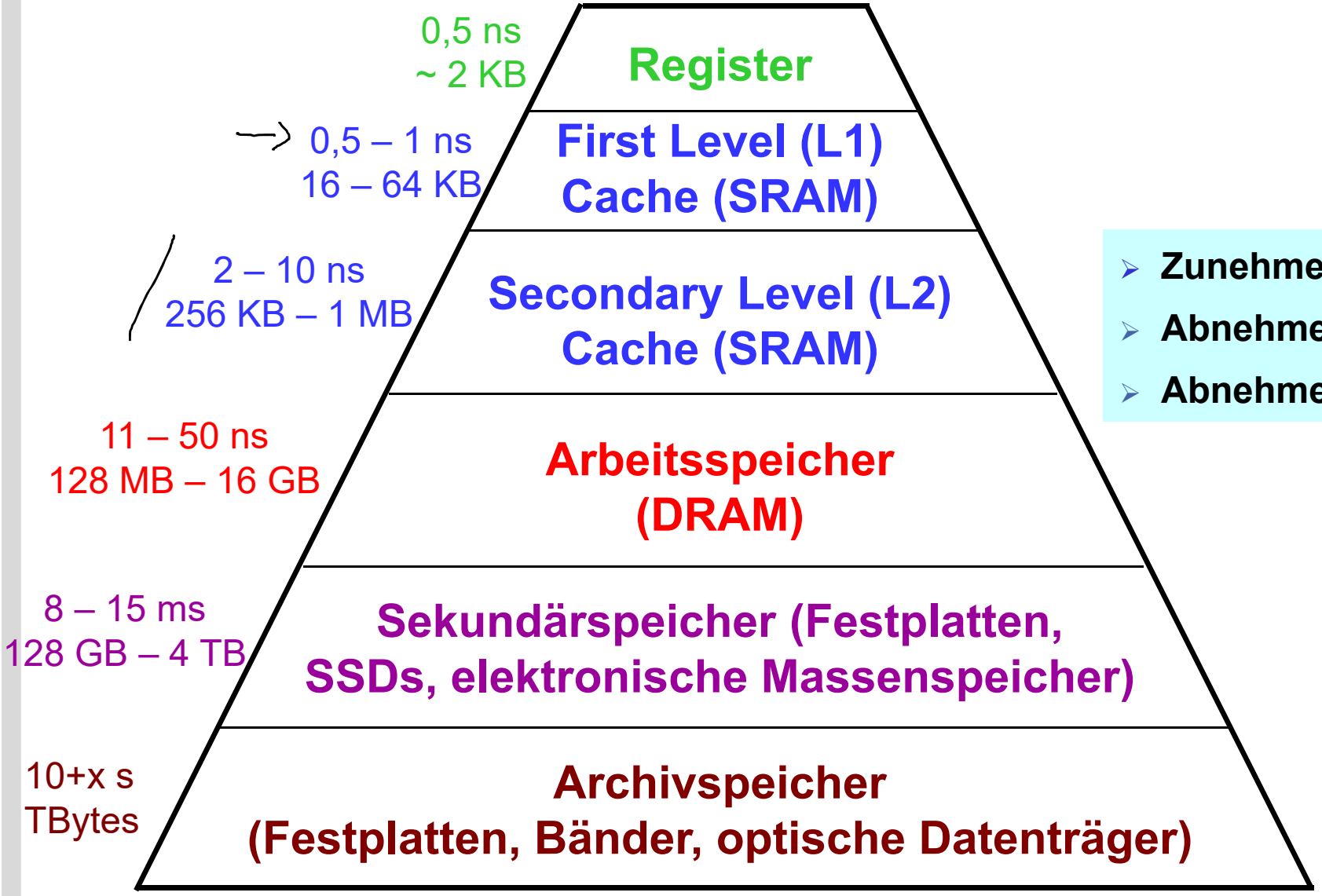
9.1 Speicherhierarchie

- **Unterschied zwischen Prozessor- und Speichergeschwindigkeit**
  - Immer größer werdende Lücke zwischen Verarbeitungsgeschwindigkeit von Prozessoren und Zugriffsgeschwindigkeit der DRAM-Speicherchips des Hauptspeichers
  - Ein technologisch einheitlicher Speicher mit kurzer Zugriffszeit und großer Kapazität ist aus Kostengründen i.A. nicht realisierbar
  - Lösung: Hierarchische Anordnung verschiedener Speicher und Verschiebung der Information zwischen den Schichten (Speicherhierarchie)

9.1 Speicherhierarchie

- **Unterschied zwischen Prozessor- und Speichergeschwindigkeit**
- **Speicherhierarchie**
  - zum Ausgleich der unterschiedlichen Zugriffszeiten der CPU und des Hauptspeichers
  - Zwei Strategien:
    - Cache-Speicher:
      - Kurze Zugriffszeiten → Beschleunigung des Prozessorzugriffs
    - Virtueller Speicher:
      - Vermeintliche Vergrößerung des tatsächlich vorhandenen Hauptspeichers
      - z.B. kann man bei gleichzeitiger Bearbeitung mehrerer Anwendungen jeder einzelnen Anwendung den Eindruck vermitteln, sie könnte über den kompletten Hauptspeicher (oder sogar mehr) verfügen

9.1 Speicherhierarchie



- Zunehmende Kosten/Byte
- Abnehmende Kapazität
- Abnehmende Zugriffszeit

9.1 Speicherhierarchie

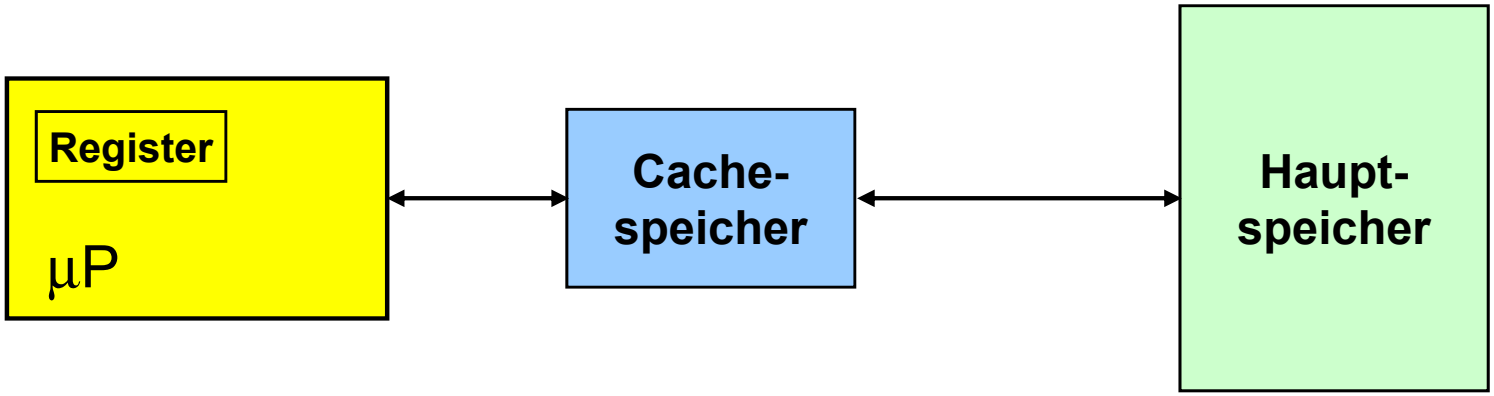
- **Speicherhierarchie:**
  - wirkt wie ein großer und schneller Speicher (widersprüchliche Anforderungen),
    - Ausnützen des Lokalitätsverhalten der Programmverarbeitung
    - Umlagerung der Information rechtzeitig (Umlagerungsstrategie, bzw. Ersetzungsstrategie)
    - Inhomogenität des Speichersystems für Benutzer nicht sichtbar (transparent)
  - Leistungsfähigkeit der Hierarchie ist bestimmt durch die Eigenschaften der Speichertechnologien (Zugriffsart, Zugriffszeiten, ...), Adressierung der Speicherplätze und Organisation des Betriebs

9.1 Speicherhierarchie

- **Unterschied zwischen Prozessor- und Speichergeschwindigkeit**
  - Problem:
    - die Buszykluszeit moderner Prozessoren ist deutlich kürzer als die Zykluszeit preiswerter, großer DRAM-Bausteine
      - dies zwingt zum Einfügen von Wartezyklen
    - SRAM-Speicher hingegen, der ohne Wartezyklen betrieben werden kann, ist klein und teuer
      - nur kleine Speicher können so aufgebaut werden

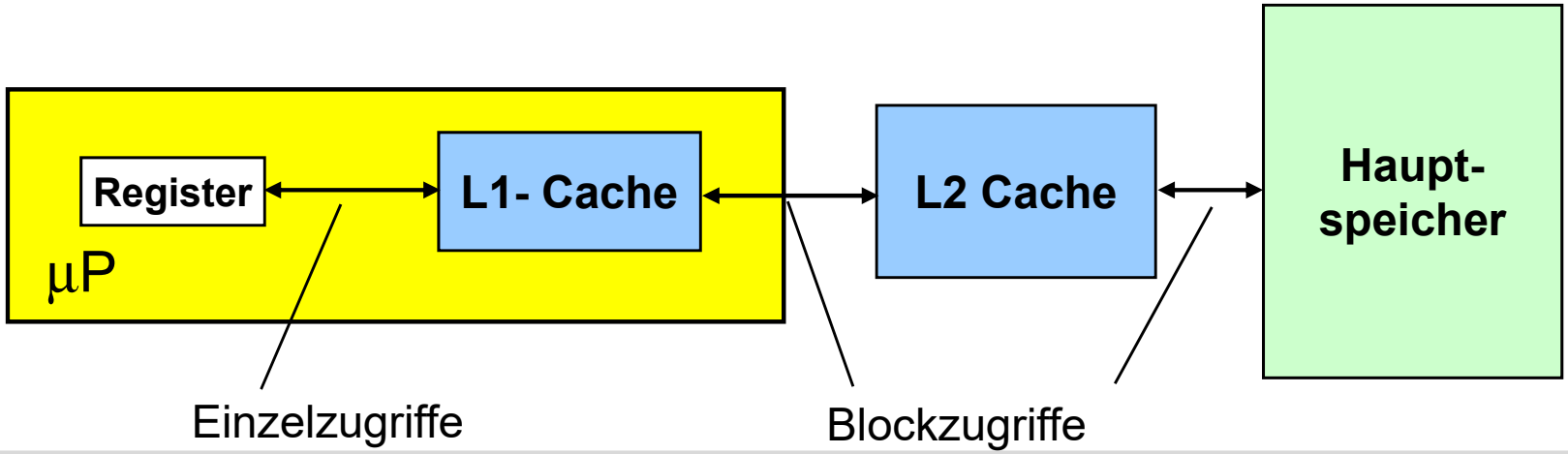
9.2 Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
  - Lösung des Problems:
    - zwischen den Prozessor und den relativ langsamen, aber billigen Hauptspeicher aus DRAM-Bausteinen legt man einen kleinen, schnellen Speicher aus SRAM-Bausteinen, den sogenannten Cache-Speicher
    - Auf den Cache-Speicher soll der Prozessor fast so schnell zugreifen können wie auf seine Register



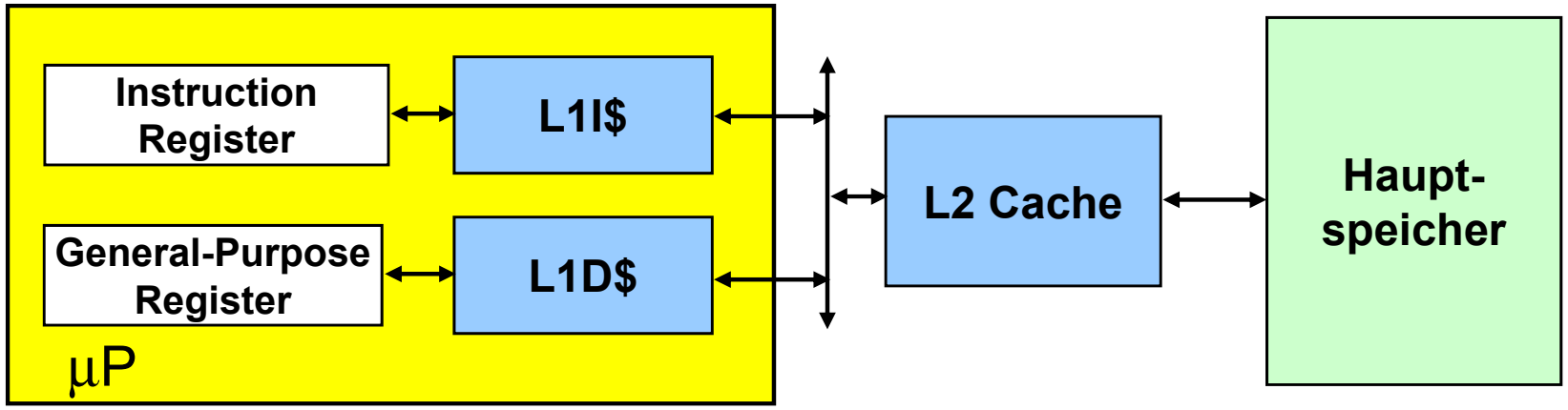
9.2 Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
  - Größe des Caches bestimmt die Zugriffszeit
    - Sehr kurze Zugriffszeiten (wie prozessorinterne Register) → sehr begrenzte Kapazität (wenige KBs)
  - Cache-Hierarchie
    - L1-Cache ist quasi Teil des Prozessors, um schnellen Zugriff sicherzustellen
    - L2-Cache ist größer, langsamer und weiter weg
    - L1-Cache holt sich größere Blöcke aus dem L2-Cache
    - ggf. teilen sich mehrere CPUs einen gemeinsamen L2 oder L3



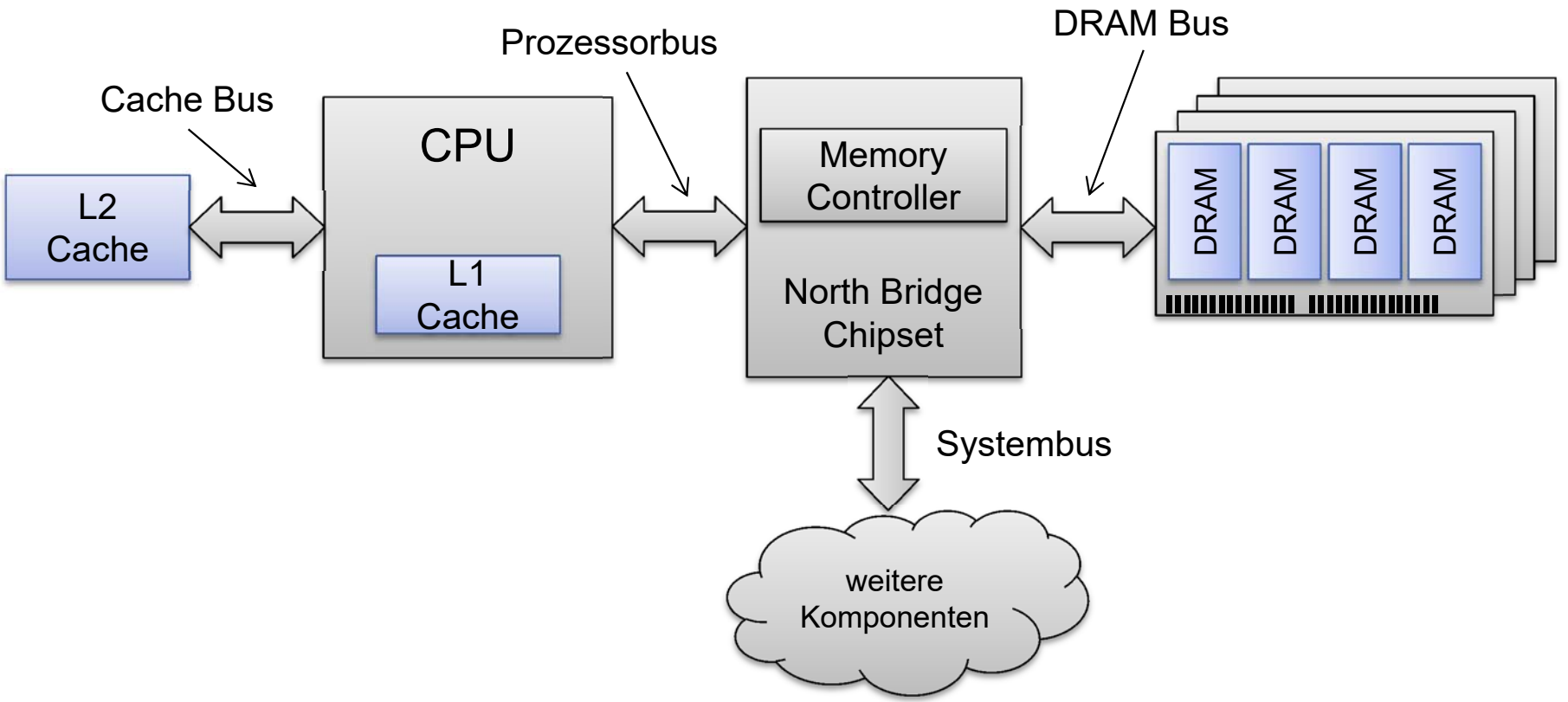
9.2 Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
  - Häufig zwei getrennte L1 Caches für Befehle (L1I$) und Daten (L1D$)
    - Erlauben parallelen Zugriff auf den nächsten Befehl und auf den Datenspeicher
    - Realisieren das Harvard Konzept



9.2 Cache-Speicher

- **Beispiel: Aufbau klassischer PC**

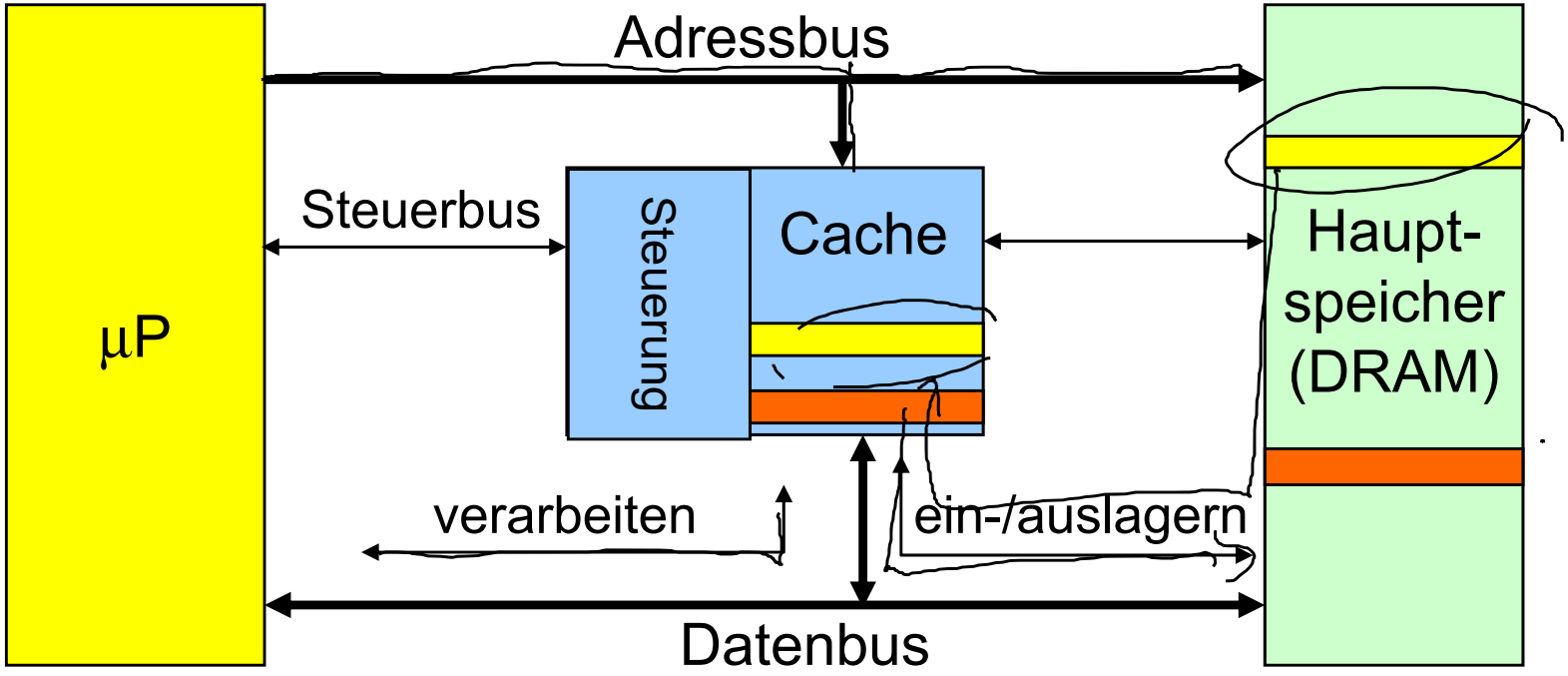


9.2 Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
  - Cache-Speicher
    - Pufferspeicher mit schnellem Zugriff
  - Wichtigste Anwendung:
    - Pufferspeicher zwischen Hauptspeicher und Prozessor
    - Stellt die während einer Programmausführung jeweils aktuellen Hauptspeicherinhalte für Prozessorzugriffe als Kopien möglichst schnell zur Verfügung.
  - Weiteres Beispiel:
    - Translation-Lookaside Buffer (TLB)
    - Verbesserung der Zugriffszeit von Plattenspeichern durch einen Cache (Plattencache)

9.2 Cache-Speicher

- **Systemaufbau mit Cache-Speicher**
- **Definition:**
  - **CPU-Cache-Speicher**
    - Pufferspeicher mit schnellem Zugriff
    - Bereithalten von Kopien derjenigen Teile des Hauptsspeichers, auf die mit hoher Wahrscheinlichkeit von der CPU als nächstes zugegriffen wird.

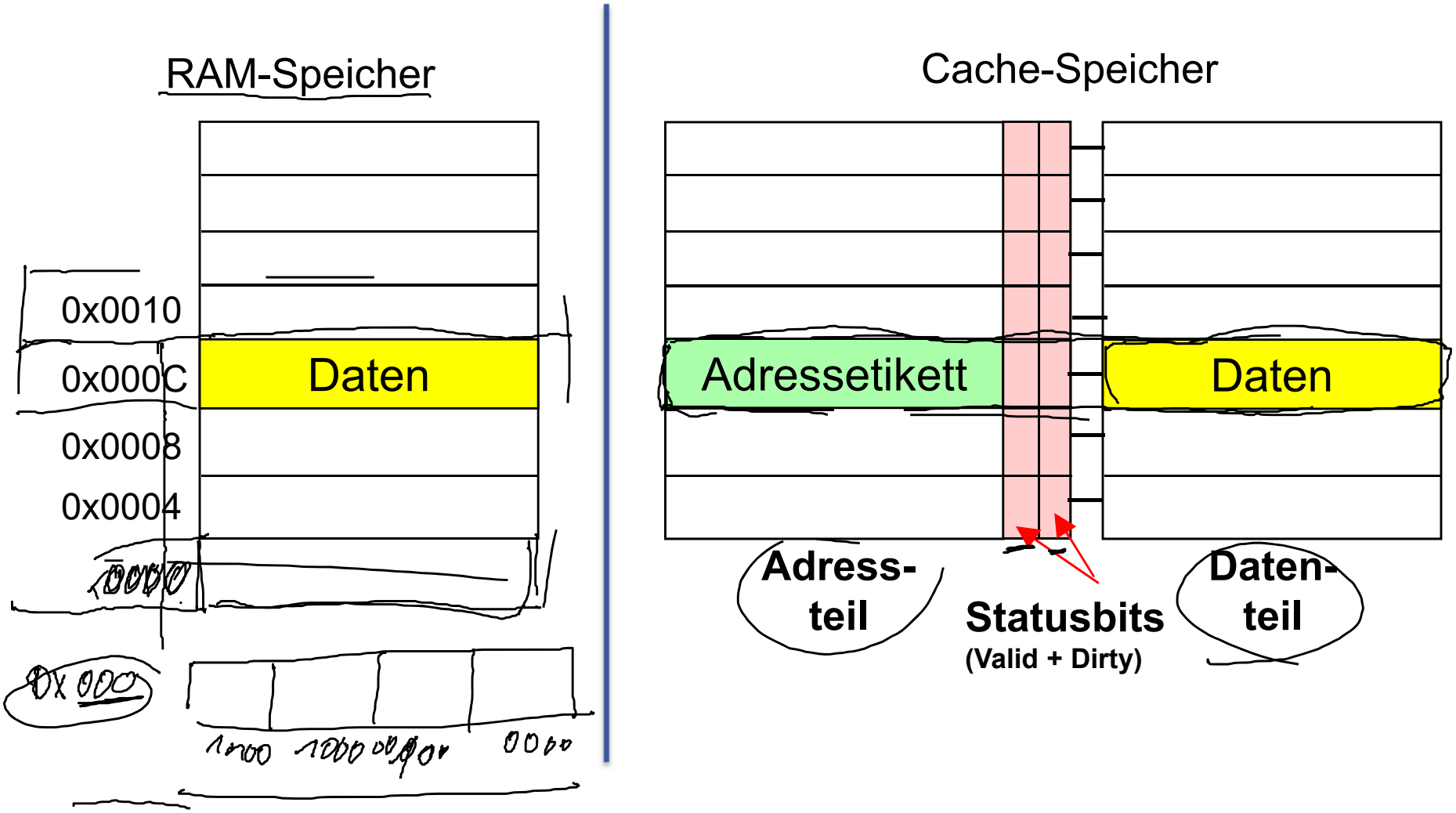


9.2 Cache-Speicher

- **CPU-Cache-Speicher**
  - Nützt die Lokalitätseigenschaften von Programmen aus:
    - **Zeitliche Lokalität:**
      - Die Information, die in naher Zukunft angesprochen wird, ist mit großer Wahrscheinlichkeit schon früher einmal angesprochen worden
      - Z.B. Befehle im Schleifenrumpf oder häufig benutzte Variablen
    - **Örtliche Lokalität**:
      - Ein zukünftiger Zugriff wird mit großer Wahrscheinlichkeit in der Nähe des bisherigen Zugriffs liegen
      - Z.B. Daten in einem Array oder die nächsten Befehle

9.2 Cache-Speicher

- **Grundlegender Aufbau**



9.2 Cache-Speicher

- **Merkmale**
  - **Blockrahmen, Zeile (Block-Frame, Cache-Line)**
  - **Datenteil**
    - Folge von Speicherwörtern im Cache-Speicher
    - Blocklänge in Bytes bestimmt die Länge der Zeile
    - Anzahl der Speicherplätze in einem Blockrahmen
  - **Adressetikett (Adress-Tag):**
    - Verbunden mit Blockrahmen
    - Enthält den gemeinsamen Adressteil der in einer Zeile gespeicherten Datenkopien.
  - **Statusbits**
    - Valid-Bit
      - Gültigkeitsbit zeigt an, ob Cache-Zeile gültige Kopien enthält
    - Dirty-Bit
      - Zeigt für Daten-Caches an, ob die Daten in der Cache-Zeile verändert worden sind

9.2 Cache-Speicher

- **Merkmale**
  - **Cache-Speicher-Steuerung /Cache-Controller**
    - Sorgt dafür, dass der Cache-Speicher in der Regel das Datum enthält, auf das der Prozessor als nächstes zugreift.
      - die CPU kann mit hoher Wahrscheinlichkeit das nächste Datum aus dem schnellen Cache und nicht aus dem langsamen Arbeitsspeicher holen
      - Kopiert alle Daten in den Cache, auf die der Prozessor zugreift
      - Weniger häufig benötigte Daten werden nach verschiedenen Strategien aus dem Cache verdrängt
    - Besondere Strategien für das
      - Laden
      - Aktualisieren und Adressieren des Inhalts

9.2 Cache-Speicher

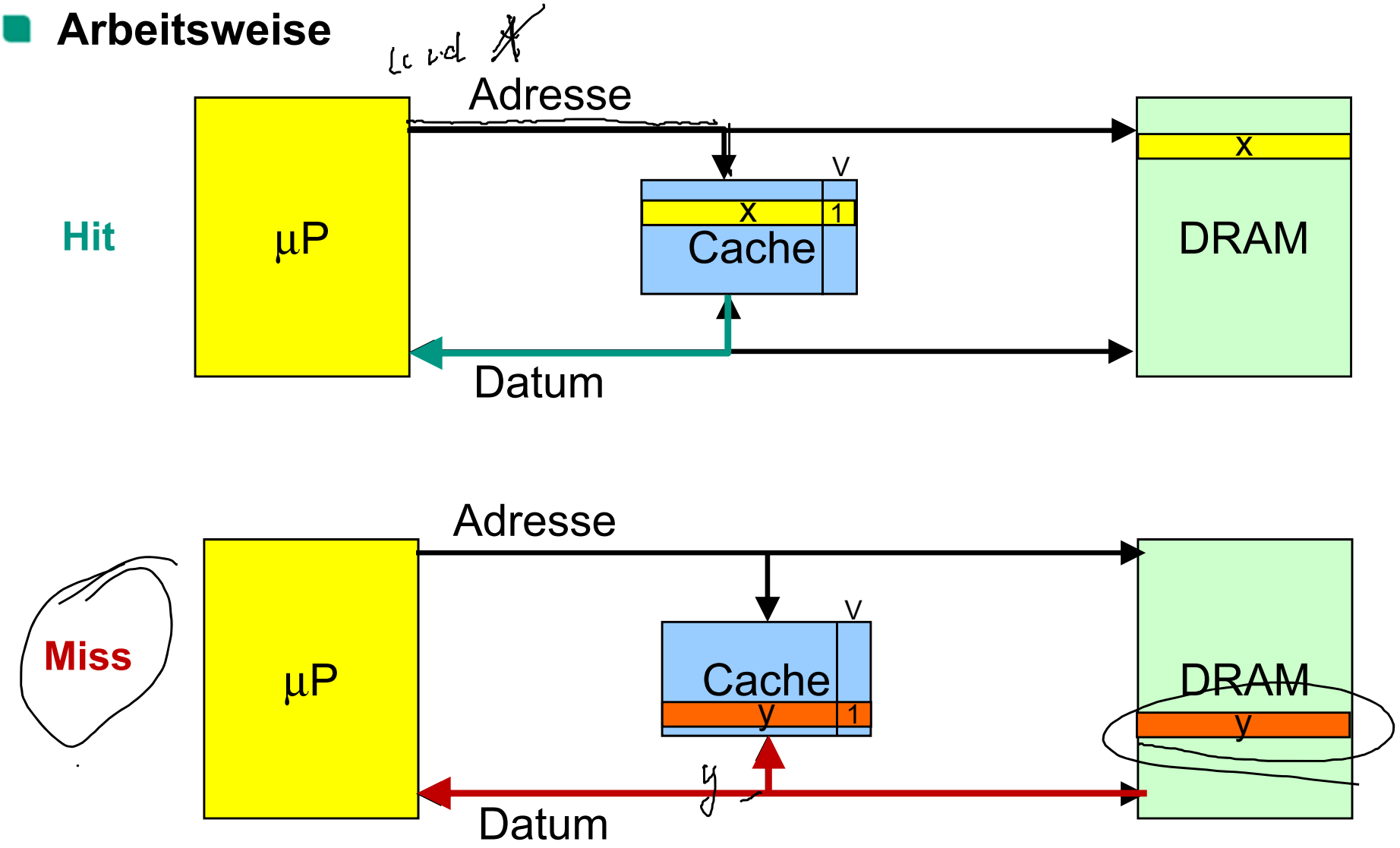
- **Arbeitsweise**
  - Cache-Steuerung prüft bei Speicherzugriffen des Mikroprozessors, ob
    - der zur Speicheradresse gehörende Hauptspeicherinhalt als Kopie im Cache steht (Bedingung 1) und
    - dieser Cache-Eintrag durch das Valid-Bit als gültig gekennzeichnet ist (Bedingung 2).
  - Treffer (Cache-Hit):
    - Beide Bedingungen sind erfüllt
    - Zugriff erfolgt auf Cache
  - Fehlzugriff (Cache-Miss):
    - Eine der beiden Bedingungen ist nicht erfüllt

9.2 Cache-Speicher

- **Arbeitsweise**
  - **Aktionen bei Lesezugriffen (read-miss)**
    - Lesen des Datums aus den niedrigeren Ebenen oder dem Hauptspeicher und Laden des Cache-Speichers
    - Kennzeichnen der Cache-Eintrages als gültig (Setzen des Valid-Bits)
    - Speichern der Adressinformation im Adressteil des Cache-Speichers

9.2 Cache-Speicher

- **Arbeitsweise**

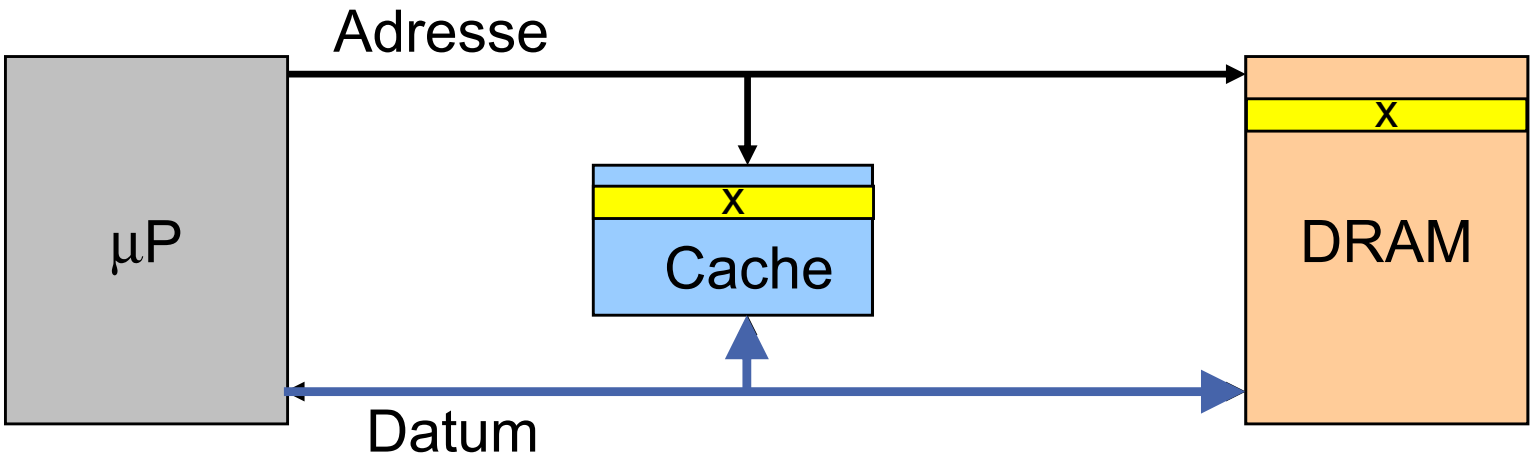


9.2 Cache-Speicher

- **Arbeitsweise**
  - **Aktionen bei Schreibzugriffen (write-miss):**
    - Aktualisierungsstrategie bestimmt, ob
      - der entsprechende Block in Cache geladen und dann mit dem zu schreibenden Datum aktualisiert wird, oder ob
      - nur der Hauptspeicher aktualisiert wird und der Cache unverändert bleibt.

9.2 Cache-Speicher

- **Arbeitsweise**
  - **Aktualisierungsstrategie Write-through**
    - Ein Datum wird von der CPU immer gleichzeitig in den Cache- und in den Arbeitsspeicher geschrieben



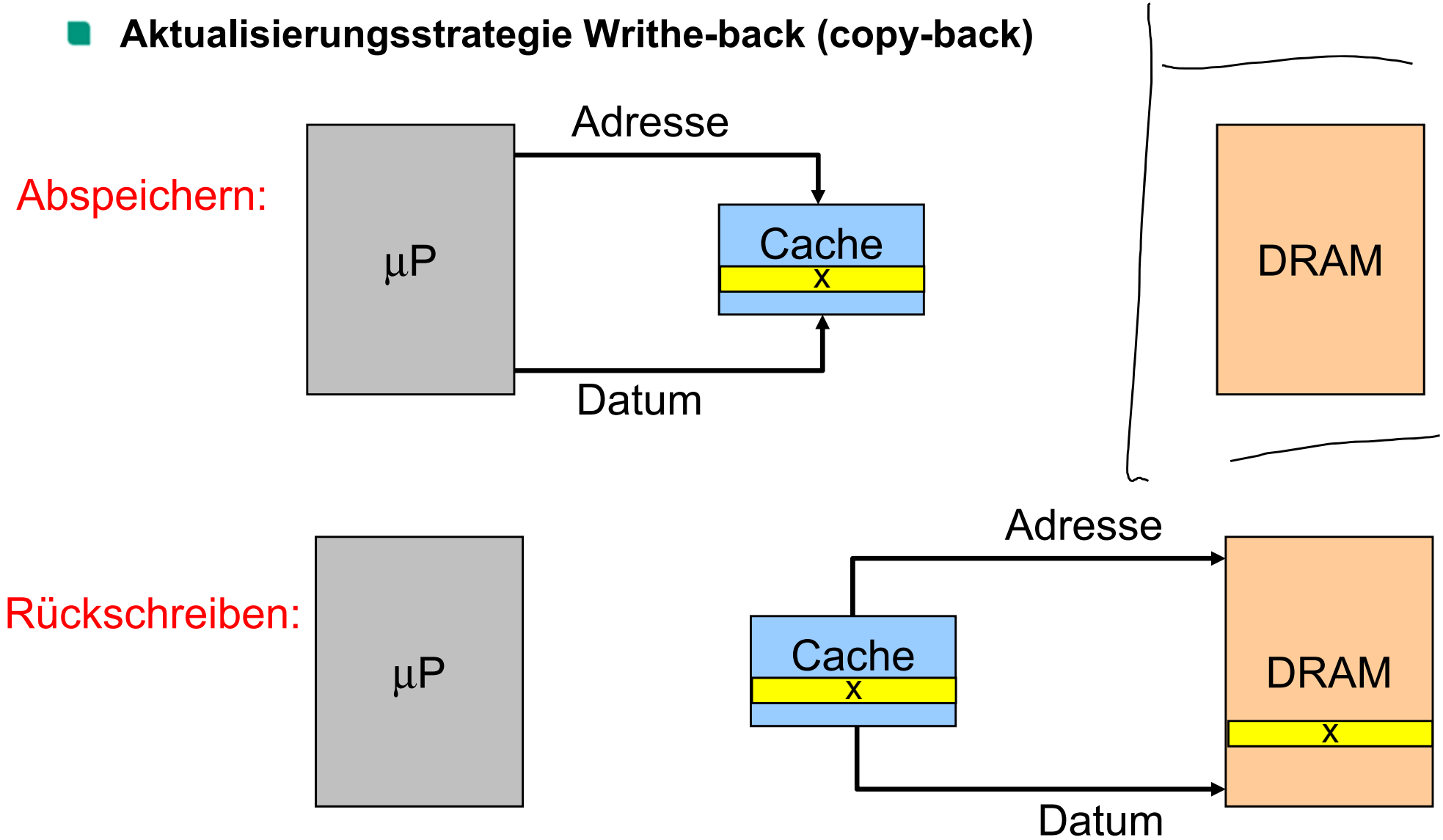
    - Vorteil: Cache- und Arbeitsspeicher sind aktuell (Cache Inhalt ist echte Teilmenge vom Arbeitsspeicher)
    - Nachteil: Schreibzugriffe benötigen immer die langsame Zykluszeit des Hauptspeichers und belasten den Systembus

9.2 Cache-Speicher

- **Arbeitsweise**
  - **Aktualisierungsstrategie Write-back (copy-back)**
    - Ein Datum wird von der CPU nur in den Cachespeicher geschrieben und durch ein spezielles Bit (dirty bit; ggf. auch altered bit oder modified bit genannt) gekennzeichnet
    - Der Arbeitsspeicher wird erst dann aktualisiert, wenn ein so gekennzeichnetes Datum aus dem Cache verdrängt wird
    - Vorteil:
      - auch Schreibzugriffe können mit der schnellen Cache-Zykluszeit abgewickelt werden (wie bei buffered write through)
      - Teilweise deutlich geringere Last auf dem Systembus (vom Programm mehrmals überschriebene Adressen müssen nur einmal zum Hauptspeicher übertragen werden)
    - Nachteil:
      - Hauptspeicher enthält nicht die aktuellen Daten

9.2 Cache-Speicher

- **Arbeitsweise**
  - **Aktualisierungsstrategie Writhe-back (copy-back)**



9.2 Cache-Speicher

Aktualisierungsstrategie

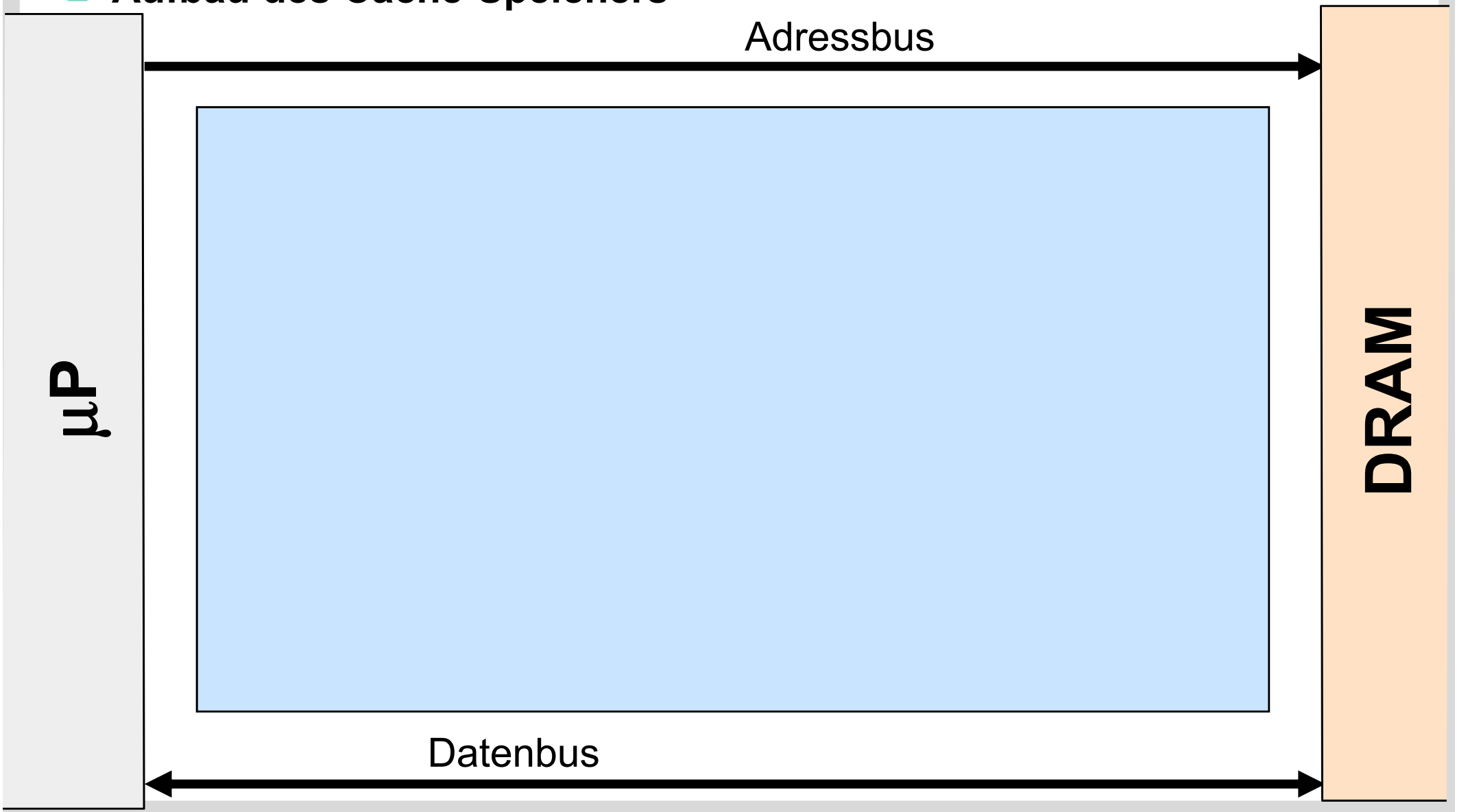
Cache-Zugriff	Write-Through	Copy-Back
Read-Hit	Cache-Datum --> CPU	Cache-Datum --> CPU
Read-Miss	HS-Block, Tag --> Cache HS-Datum --> CPU 1 --> V	Cache-Zeile --> HS HS-Block, Tag --> Cache HS-Datum --> CPU 1 --> V, 0 --> D
Write-Hit	CPU-Datum --> Cache,HS	CPU-Datum --> Cache 1 --> D
Write-Miss	CPU-Datum --> HS	Cache-Zeile --> HS HS-Block, Tag --> Cache 1 --> V CPU-Datum --> Cache 1 --> D

9.2 Cache-Speicher

- **Leistung**
  - Die Trefferrate (Hit-Rate) bezeichnet die Trefferquote im Cache:
    - Hit-Rate = Anzahl Treffer / Anzahl Zugriffe
  - Die mittlere Zugriffszeit berechnet sich annähernd wie folgt:
    - t_{Access} = (Hit-Rate) * t_{Hit} + (1 - Hit-Rate) * t_{Miss}
    - mit t_{Hit}: Zugriffszeit auf den Caches
      t_{Miss}: Zugriffszeit auf den Hauptspeichers
      (genauer: Zugriff auf alles, was nach dem Cache kommt; ggf. erst ein weiterer Cache Level)

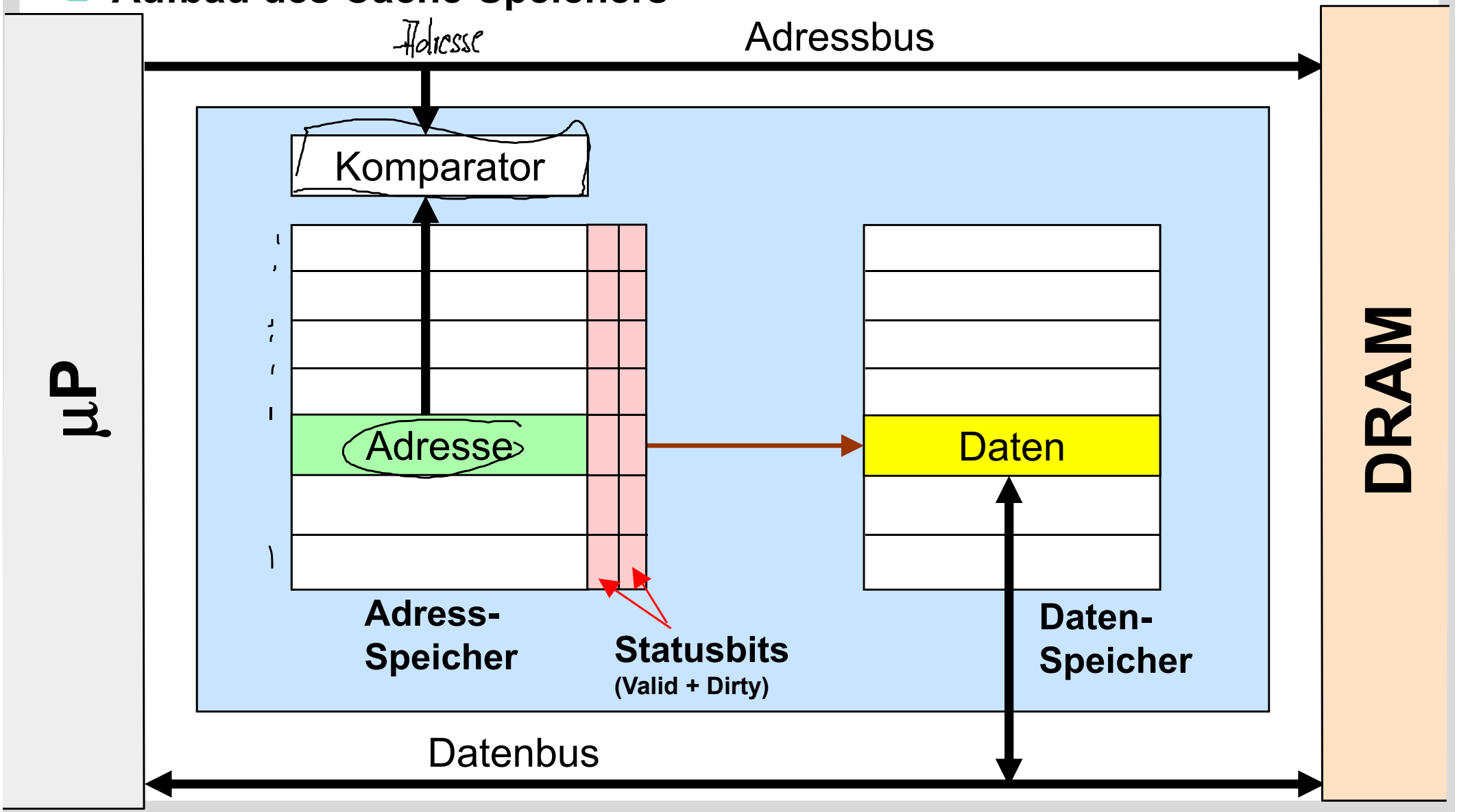
9.2 Cache-Speicher

- **Aufbau des Cache-Speichers**



9.2 Cache-Speicher

- **Aufbau des Cache-Speichers**



9.2 Cache-Speicher

- **Aufbau**
  - Ein Cache-Speicher besteht aus zwei Speicher-Einheiten:
    - **Datenteil**: enthält die im Cache abgelegten Daten
      - Jeder Dateneintrag besteht aus einem Datenblock
      - Mit jedem Datum, auf das der Prozessor zugreift, wird die Umgebung mit eingelagert ((aufeinanderfolgende Folge von Wörtern im Hauptspeicher)
        - Lokalitätsprinzip
        - Nutzt die Bursts des DRAM Hauptspeichers
    - **Adressteil**: enthält die Adressinformationen, wo diese Daten im Arbeitsspeicher stehen

9.2 Cache-Speicher

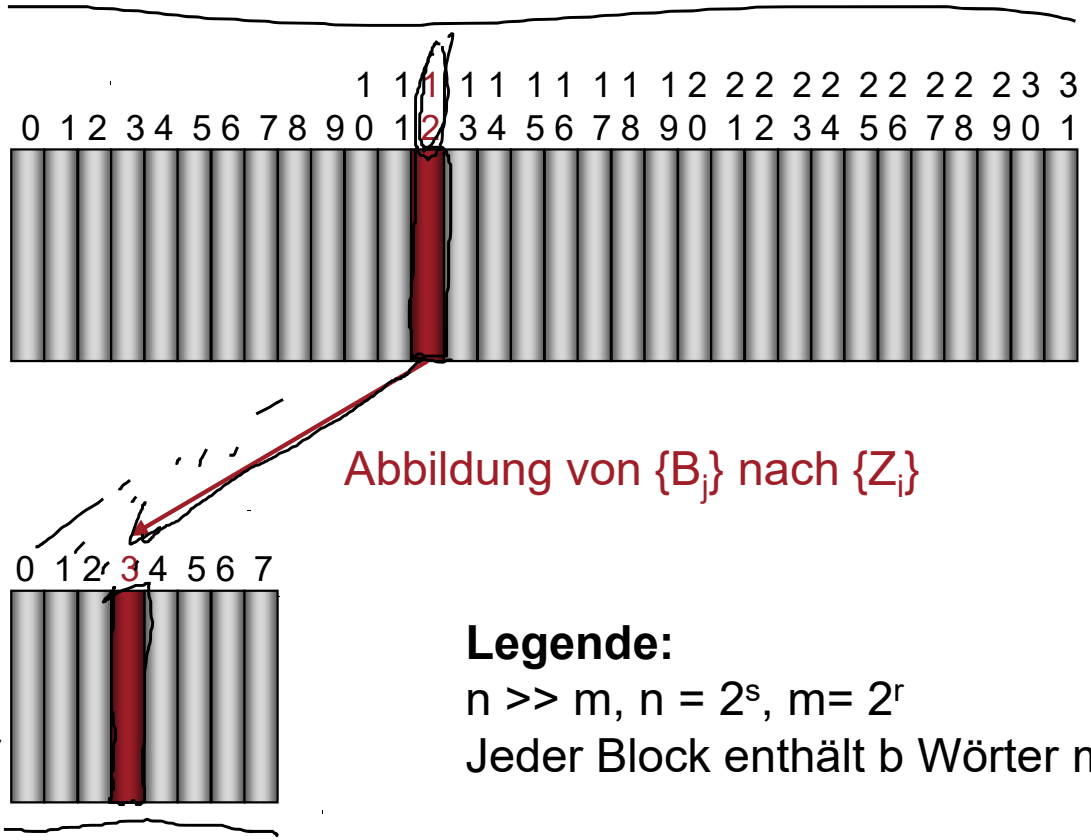
- **Aufbau**
  - **Blockrahmen, Zeile (Block-Frame, Cache-Line)**
    - **Datenteil**
      - Folge von Speicherwörtern im Cache-Speicher
      - Blocklänge in Bytes bestimmt die Länge der Zeile
      - Anzahl der Speicherplätze in einem Blockrahmen
    - **Adressetikett (Adress-Tag):**
      - Verbunden mit Blockrahmen
      - Enthält den gemeinsamen Adressteil der in einer Zeile gespeicherten Datenkopien.
    - **Statusbits**
      - Valid-Bit
        - Gültigkeitsbit zeigt an, ob Cache-Zeile gültige Kopien enthält
      - Dirty-Bit
        - Zeigt für Daten-Caches an, ob die Daten in der Cache-Zeile verändert worden sind

9.2 Cache-Speicher

- **Aufbau**
  - **Komparator**
    - ermittelt, ob das Datum, das zu einer auf dem Adressbus liegenden Adresse gehört, auch im Cache abgelegt worden ist
    - Adressvergleich mit den Adresstags im Adressteil
      - Dieser Adressvergleich muss sehr schnell gehen (möglichst in einem Taktzyklus), da sonst der Cachespeicher effektiv langsamer wäre als der Arbeitsspeicher

9.2 Cache-Speicher

- **Cache-Organisationsformen**
  - **Wohin wird ein Datenblock aus dem Hauptspeicher in den Cache geladen?**



Hauptspeicher
Block B_j , j= 0,1, …,n-1
Kapazität: n * b = 2^{s+w} Wörter

Cache
Zeile Z_i , i= 0,1, …,m-1
Kapazität: m * b = 2^{r+w} Wörter

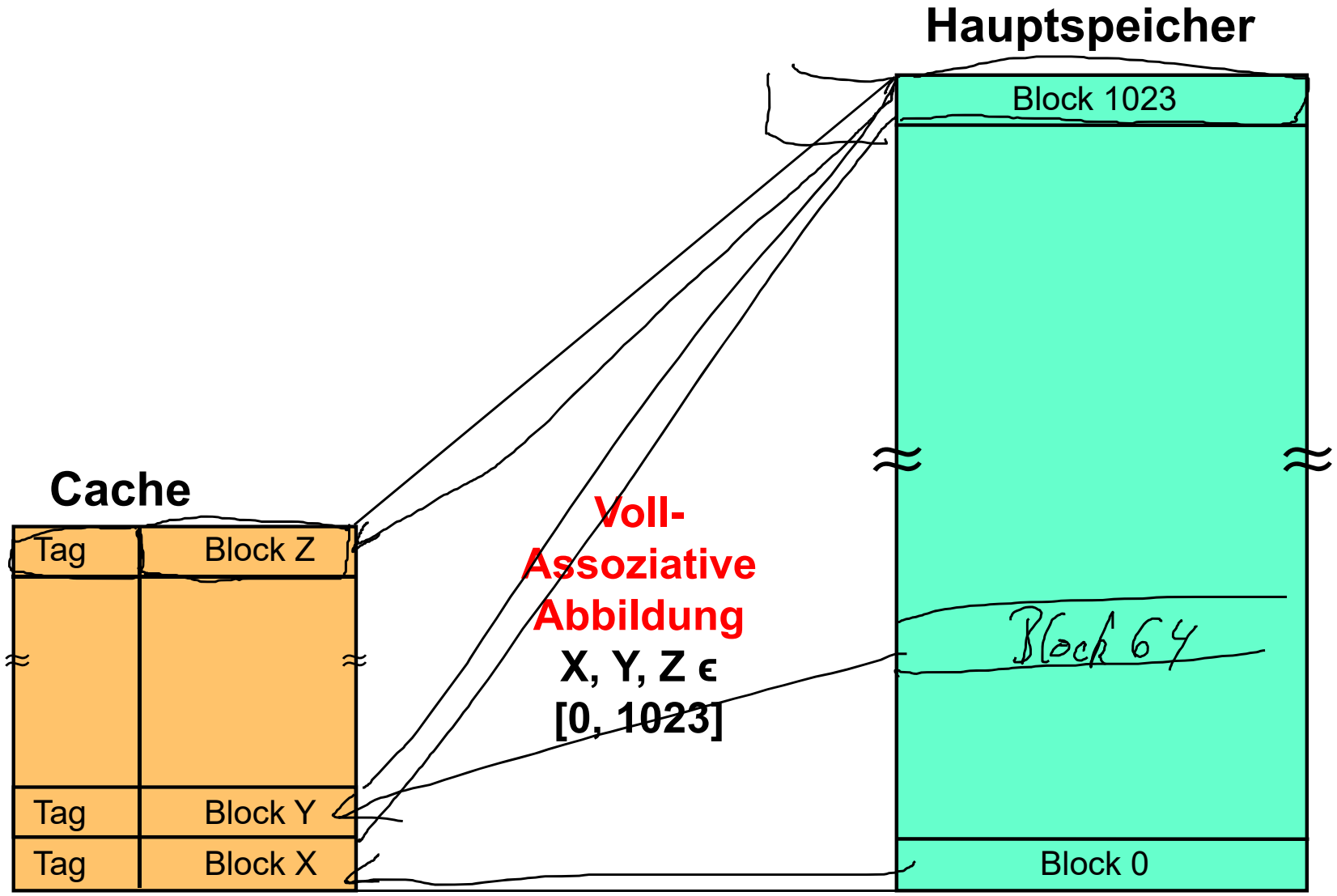
Legende:
n >> m, n = 2^s, m= 2^r
Jeder Block enthält b Wörter mit b= 2^w

9.2 Cache-Speicher

- **Cache-Organisationsformen**
  - 3 Techniken für den Adressvergleich → 3 Cache-Organisationsformen:
    - Voll-Assoziativer Cache
    - Direct Mapped Cache
    - N-fach satzassoziativer Cache (N-Wege mengenassoziativer Cache, n-way set-associativer Cache)

9.2 Cache-Speicher

- **Voll-Assoziativer Cache**

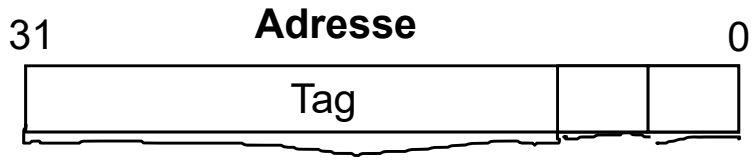


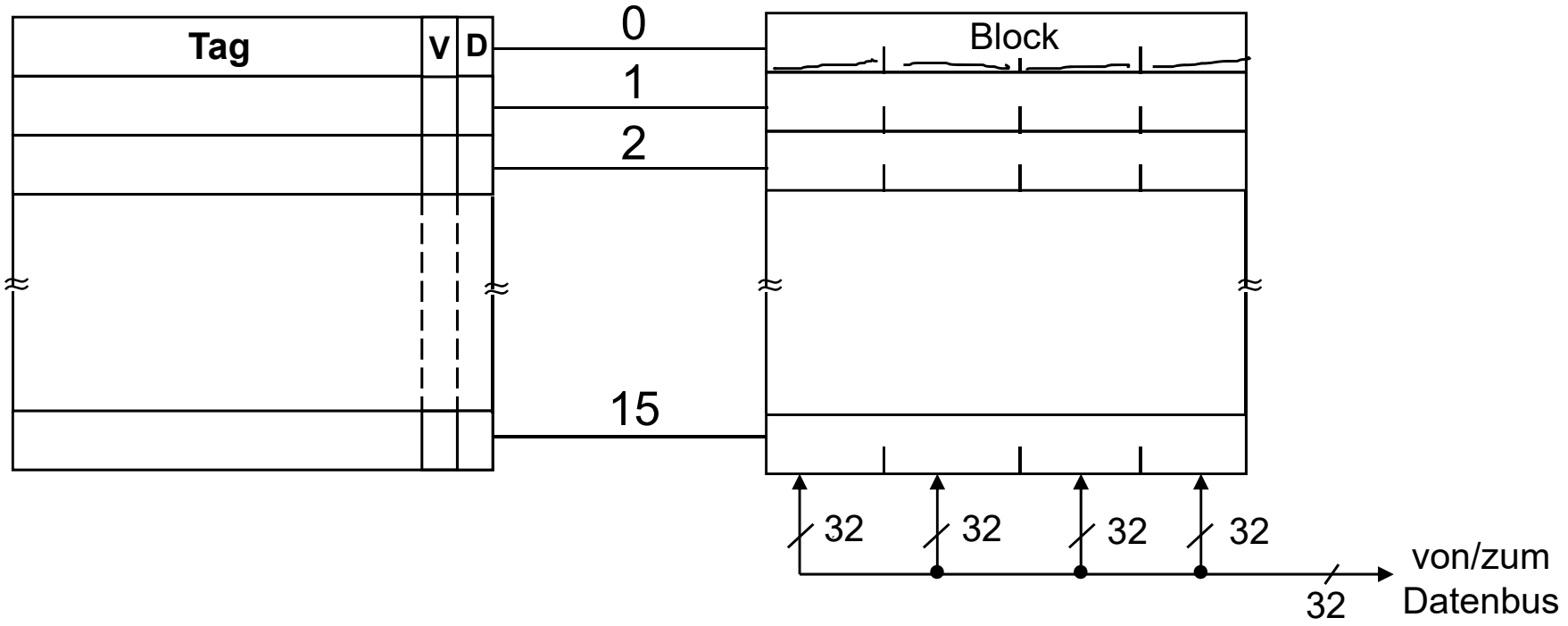
9.2 Cache-Speicher

Voll-Assoziativer Cache

Kapazität: 256 Byte

Block = 4 Wörter = 16 Byte



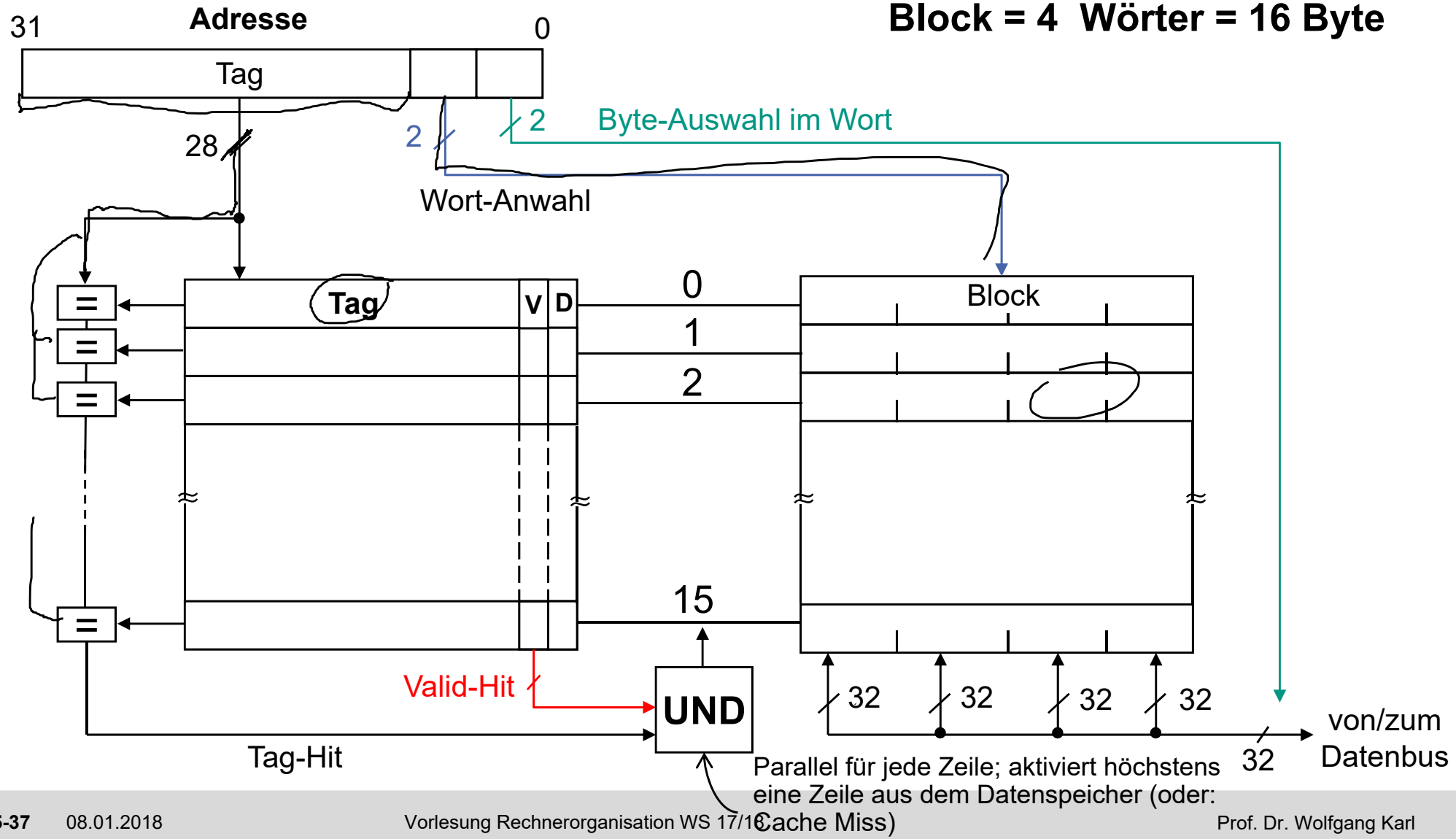


9.2 Cache-Speicher

Voll-Assoziativer Cache

Kapazität: 256 Byte

Block = 4 Wörter = 16 Byte



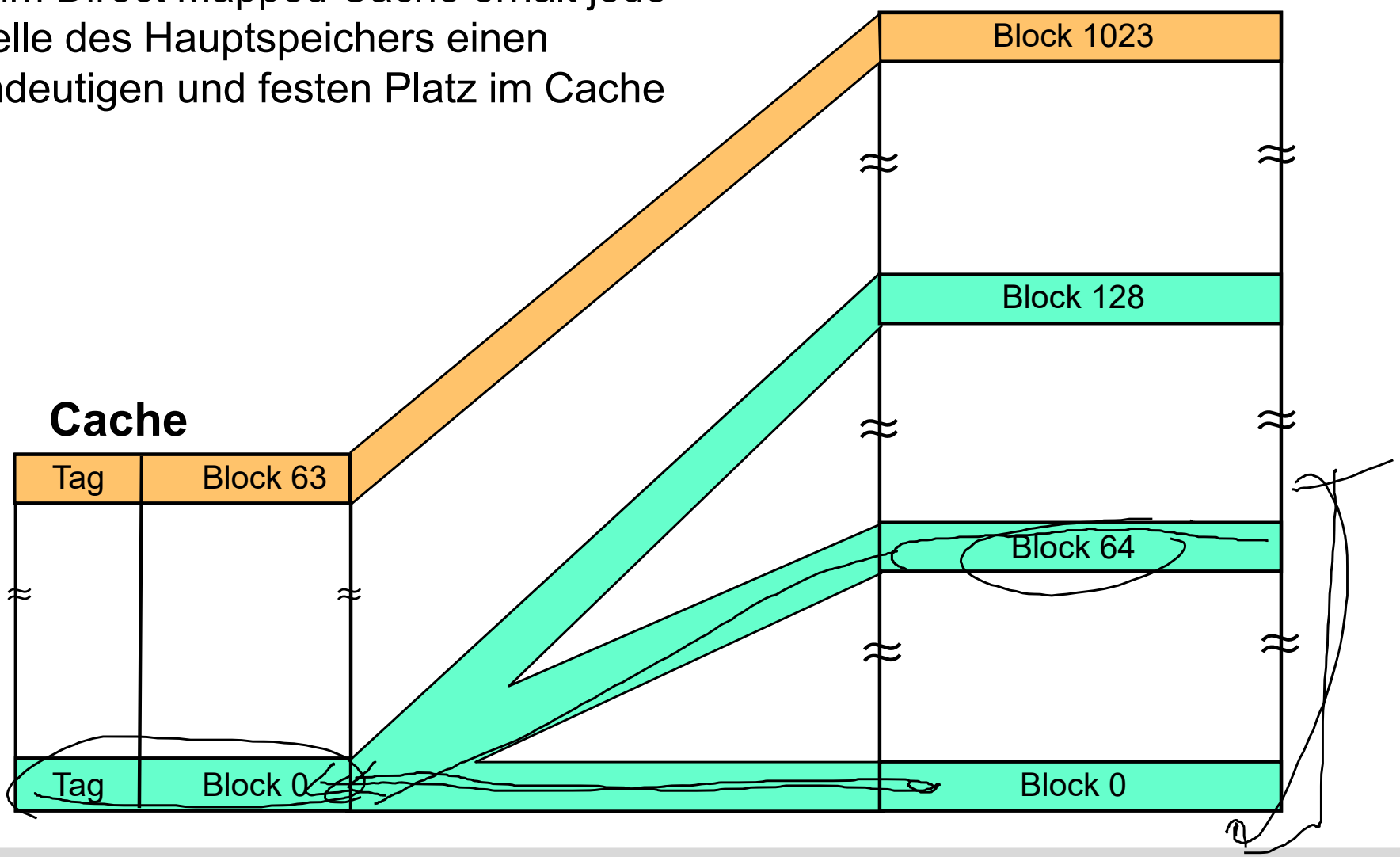
9.2 Cache-Speicher

- **Voll-Assoziativer Cache**
  - Vollparalleler Vergleich aller Adressen im Adressspeicher in einem einzigen Taktzyklus
  - Vorteile:
    - ein Datum kann an beliebiger Stelle im Cache abgelegt werden
    - Optimale Cache-Ausnutzung, völlig freie Wahl der Strategie bei Verdrängungen
  - Nachteile:
    - Hoher Hardwareaufwand (paralleler Zugriff auf alle Tags):
      - für jede Cache-Zeile ein Vergleicher
      - nur für sehr kleine Cachespeicher realisierbar
    - Die große Flexibilität der Abbildungsvorschrift erfordert eine weitere Hardware, welche die Ersetzungsstrategie (welcher Block soll überschrieben werden, wenn der Cache voll ist) realisiert

9.2 Cache-Speicher

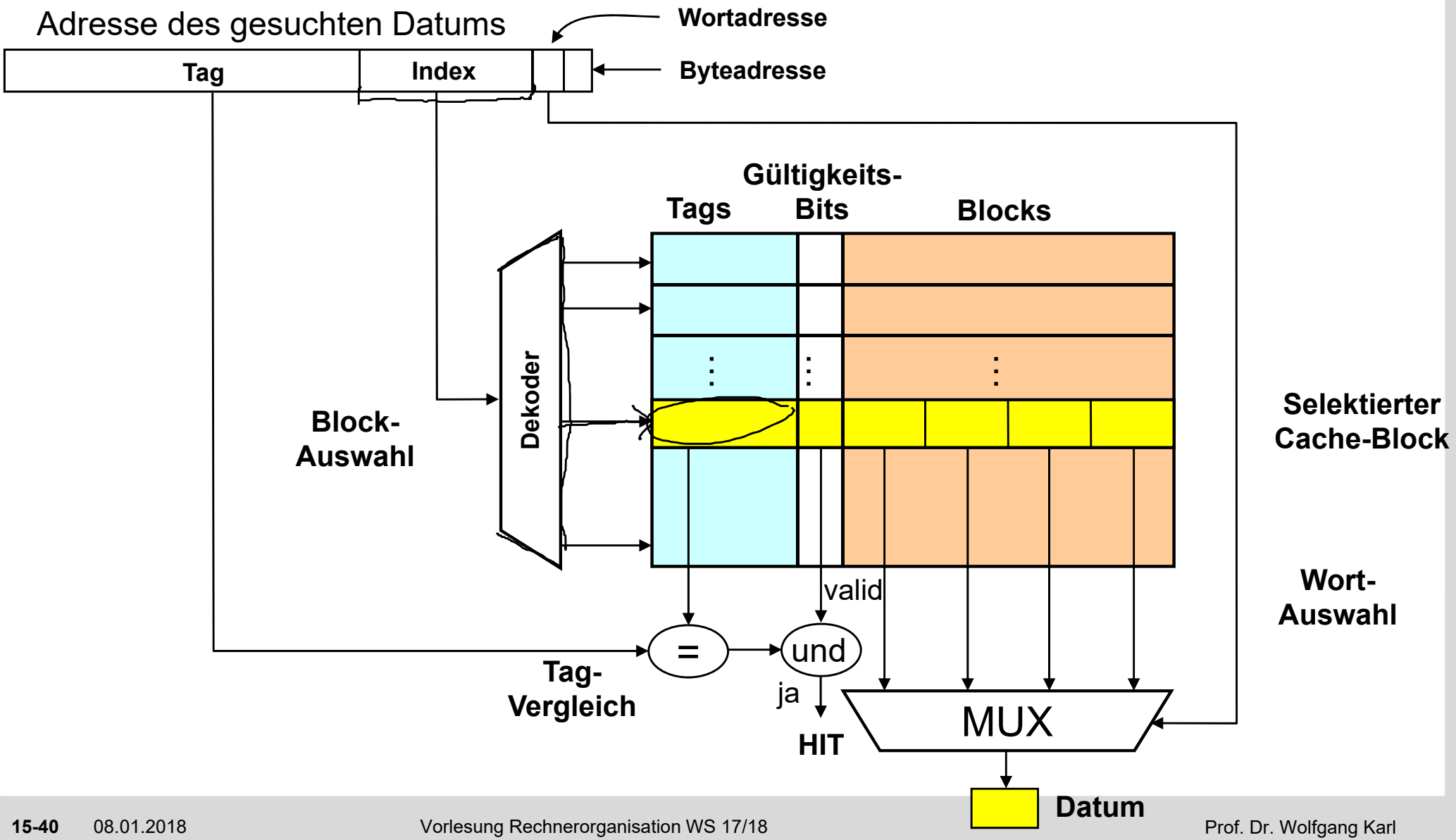
- **Direct-mapped Cache**

Beim Direct Mapped Cache erhält jede Stelle des Hauptspeichers einen eindeutigen und festen Platz im Cache



9.2 Cache-Speicher

Direct-mapped Cache



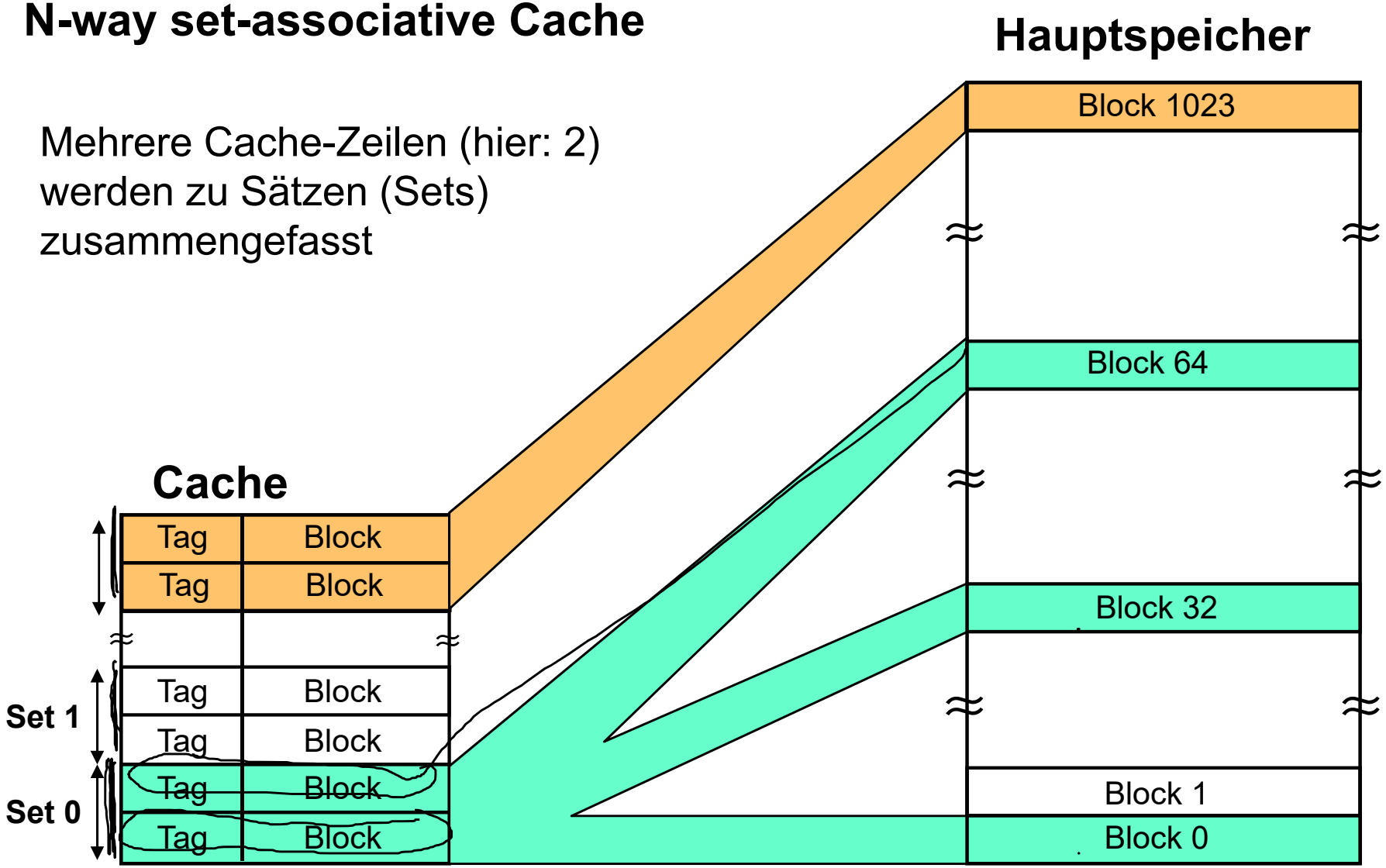
9.2 Cache-Speicher

- **Direct-mapped Cache**
  - Nachteile:
    - Ständige Konkurrenz/Kollision der Blöcke (z.B. 0, 64, 128, ...), 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; es kommt zum „Flattern“ (thrashing)
  - Vorteile:
    - Geringer Hardwareaufwand
      - nur 1 Tag muss zugegriffen werden
      - Kürzerer Tag
      - nur ein Vergleicher
    - Es ist keine Ersetzungsstrategie erforderlich, weil die direkte Zuordnung keine Alternativen zulässt
    - Der Zugriff erfolgt schnell, weil das Tag-Feld parallel mit dem zugehörigen Block gelesen werden kann

9.2 Cache-Speicher

- **N-way set-associative Cache**

Mehrere Cache-Zeilen (hier: 2) werden zu Sätzen (Sets) zusammengefasst



9.2 Cache-Speicher

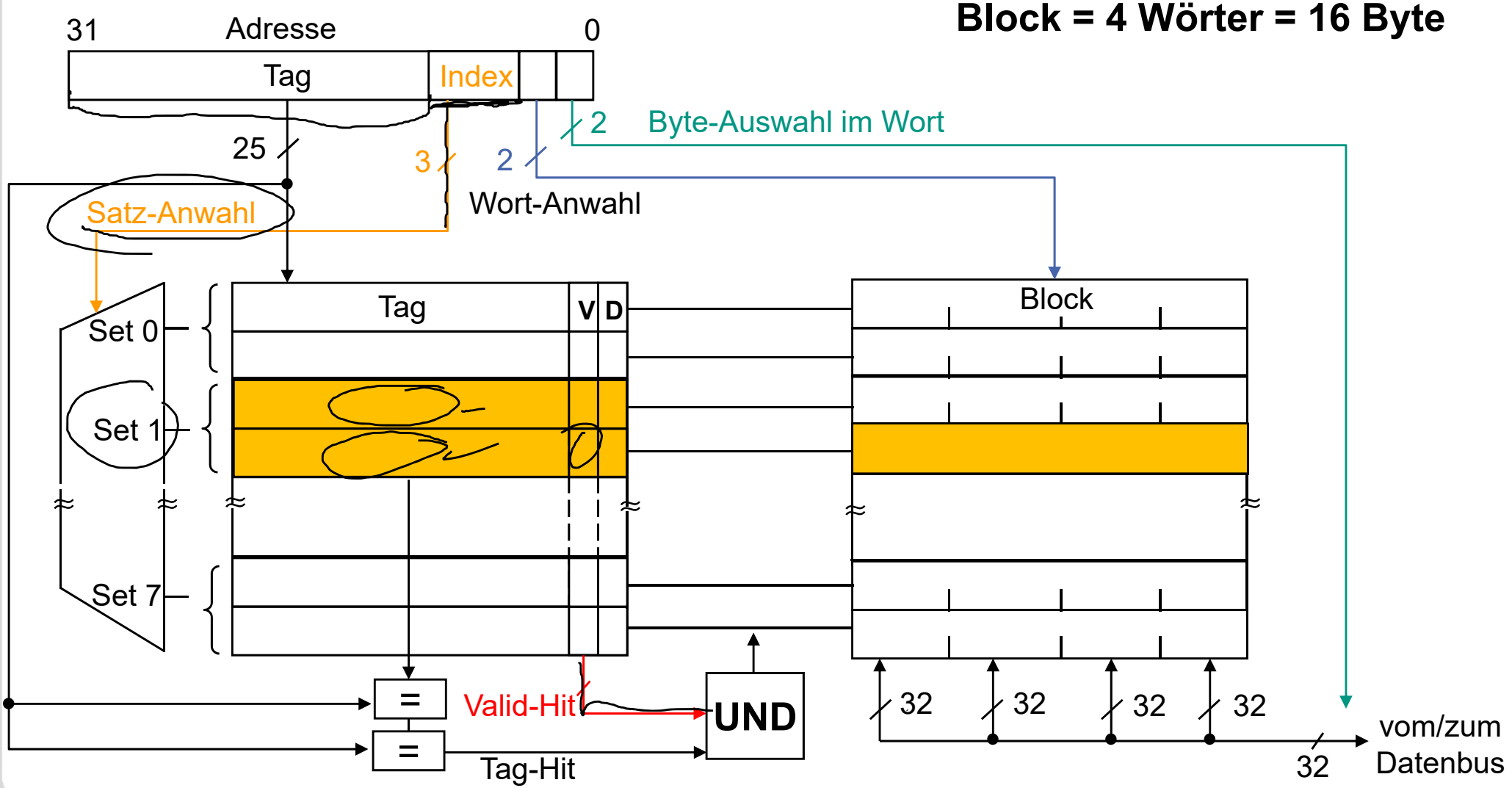
- **N-way set-associative Cache**
  - Kompromiss zwischen direct-mapped-Cache und vollassoziativem Cache
  - Verbesserte Trefferrate gegenüber direct-mapped Cache, da hier eine Auswahl möglich ist (der zu verdrängende Eintrag kann unter n Zeilen ausgewählt werden)
  - Ersetzungsstrategie:
    - Zyklisch (der zuerst eingelagerte Eintrag wird auch wieder verdrängt, FIFO-Strategie)
    - LRU-Strategie (least recently used) der am längsten nicht mehr benutzte Eintrag wird entfernt
    - Zufällig (durch Zufallsgenerator)
    - ...

9.2 Cache-Speicher

- **N-way set-associative Cache**

Kapazität: 256 Byte

Block = 4 Wörter = 16 Byte



9.2 Cache-Speicher

- **N-way set-associative Cache**
  - Zum Auffinden eines Datums werden die n Tags eines Sets mit demselben Index parallel verglichen werden
    - der Aufwand steigt mit der Zahl n ; für große n nähert sich der Aufwand den voll-assoziativen Caches
    - Kompromiss zwischen Direct Mapped Cache und voll-assoziativem Cache
  - Implementierung:
  - Alle Cache-Zeilen eines Satzes werden in einer Zeile der SRAM Arrays für Tags und Daten untergebracht
  - Vorteile: nur eine Zeile aus dem SRAM Array muss gelesen werden