Dr.-Ing. Oliver Knodel

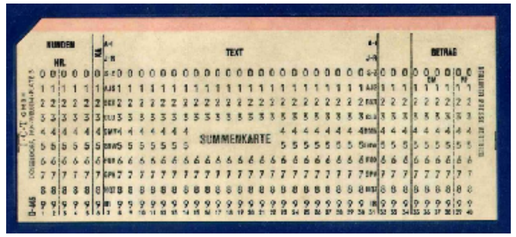
Speicherhierarchie und Cache

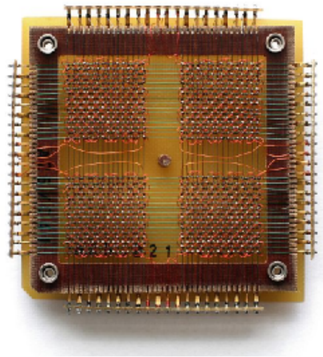
Dresden // Mai, 2024

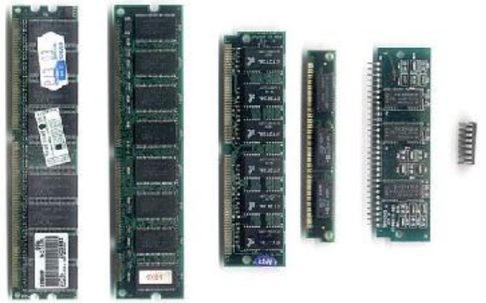
Übersicht Speicher

Speicher

mechanisch	optisch	magnetisch	elektronisch
Nockenwalze	Mikrofilm	Ferritkern	Halbleiter
Lochkarte	CD-ROM	Magnetblasen	Speicherprinzip:
Lochstreifen	DVD	Magnettrommel	- elektrostatisch
	MO-Disk	Magnetband	- elektrodynamisch
	Holographie	Magnetplatte	- ferromagnetisch
		Magnetkarte	- ferroelektrisch







Anforderungen an Speicher

Persistente Speicherung:

Die Speicherzustände sind zeitlich stabil und reproduzierbar.
Die Speicherzustände bleiben auch ohne Energieversorgung erhalten.

Veränderbarkeit und Lesbarkeit:

Der Speicherzustand ist zu jedem beliebigen Zeitpunkt lesbar und je nach Anwendung ein- oder mehrmalig oder beliebig oft veränderbar.

Auswählbarkeit:

Die Speicherzustände sind einzeln auswählbar, ansprechbar und eindeutig unterscheidbar.

Bündelung:

Mehrere Speicherzustände sind zu einer Einheit, einem Block zusammengefasst und nur gemeinsam ansprechbar, lesbar oder änderbar.

Eigenschaften von Speichern und Speichertypen

Zugriffsart:

sequentiell, strukturiert	SAM (Serial Adressable Memory)
inhaltsbezogen, assoziativ	CAM (Content Adressable Memory)
wahlfrei adressierbar	RAM (Random Access Memory)

Änderbarkeit:

fester Inhalt, nur lesbar	ROM (Read Only Memory)
einmalig programmierbar, nur lesbar	PROM (Programmable ROM)
meist lesbar, mehrmalig schreibbar	RMM (Read Mostly Memory)
mehrmalig programmierbar, löschbar, nur lesbar	EPROM (Erasable PROM)
beliebig oft les- und schreibbar	RWM (Read Write Memory)

Persistenz, Zeitverhalten:

statisch, flüchtig	SRAM (Static RAM)
dynamisch, flüchtig	DRAM (Dynamic RAM)
auch bei Spannungswegfall nicht flüchtig	NVRAM (Non-volatile RAM)

Komponenten von Speichern

Speicherelement:
realisiert (physikalisch) die Speicherzustände (meist binär).

Speicherzelle:
Zusammenfassung von Speicherelementen, kleinste ansprechbare Einheit.

Zugriffsbreite:
paralleler Zugriff auf mehrere Speicherzellen gleichzeitig (Bit, Byte, Wort, ...).

Speicherwort:
Feste Anzahl von Speicherzellen, die gleichzeitig als Block aus dem Speicher gelesen bzw. in ihn geschrieben werden kann (typisch 1 Wort).

Speicherkapazität:
Anzahl der effektiv nutzbaren Speicherzellen.

Ansteuerung:
taktsynchrone oder asynchrone Ansteuerung des Speichers.

Zeitverhalten von Speichern

Refreshzeit(Refresh Time):
Zeit zum Auffrischen des Speichers (nach einem Zugriff bzw. zyklisch)

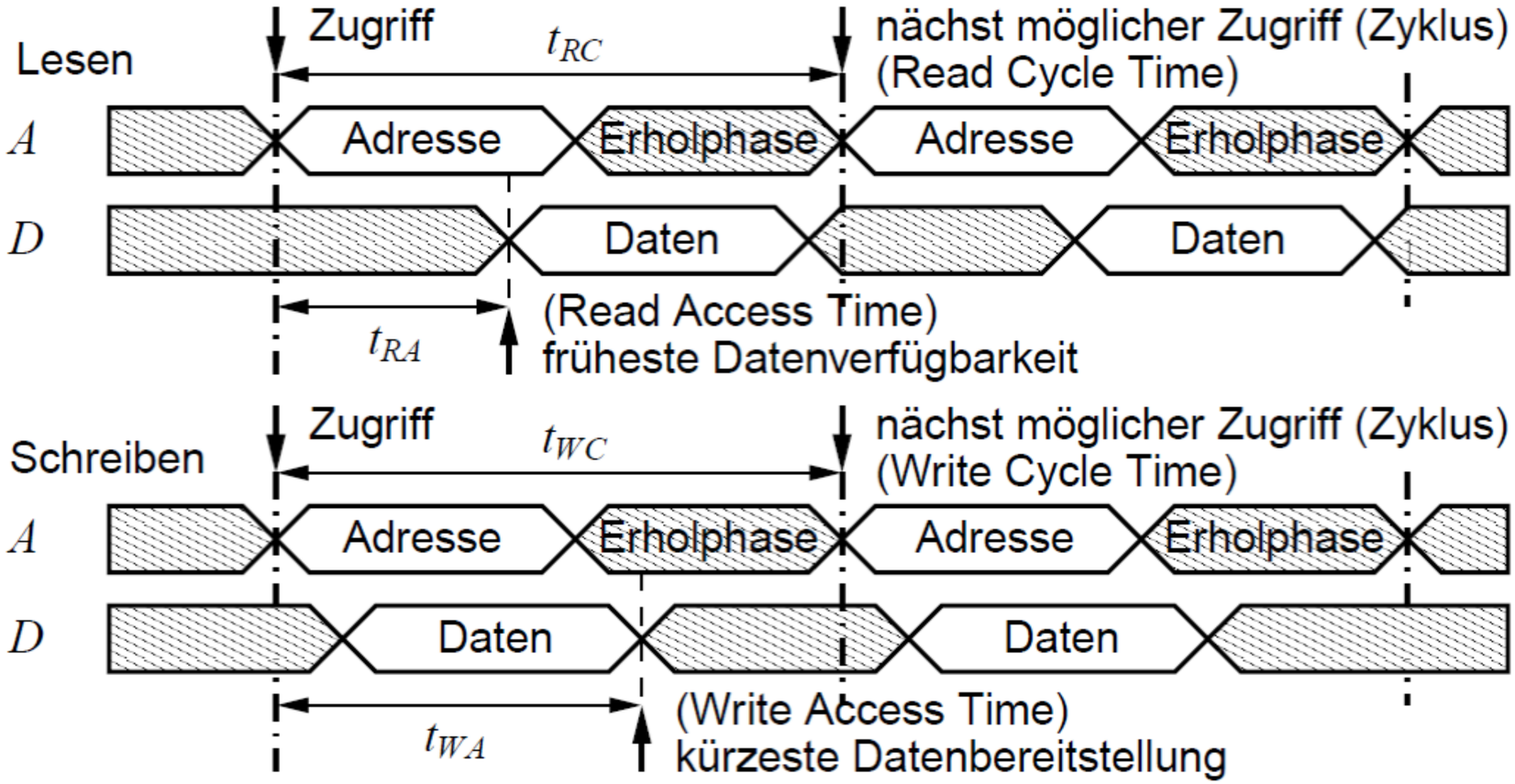
Zugriffszeit (Access Time):
minimale Zeit (Latenz) für einen Lese-/Schreibzugriff (Adresse $\rightarrow$ Datum)

Zykluszeit (Cycle Time):
minimale Zeit zwischen zwei aufeinanderfolgenden Lese-/Schreibzugriffen (einschließlich Auffrischen)

Auffrischen (Refresh):
Bei Speichern mit destruktivem Lesen bzw. bei dynamischen Speichern muss der Speicherinhalt nach dem Lesen bzw. zyklisch wieder aufgefrischt werden.

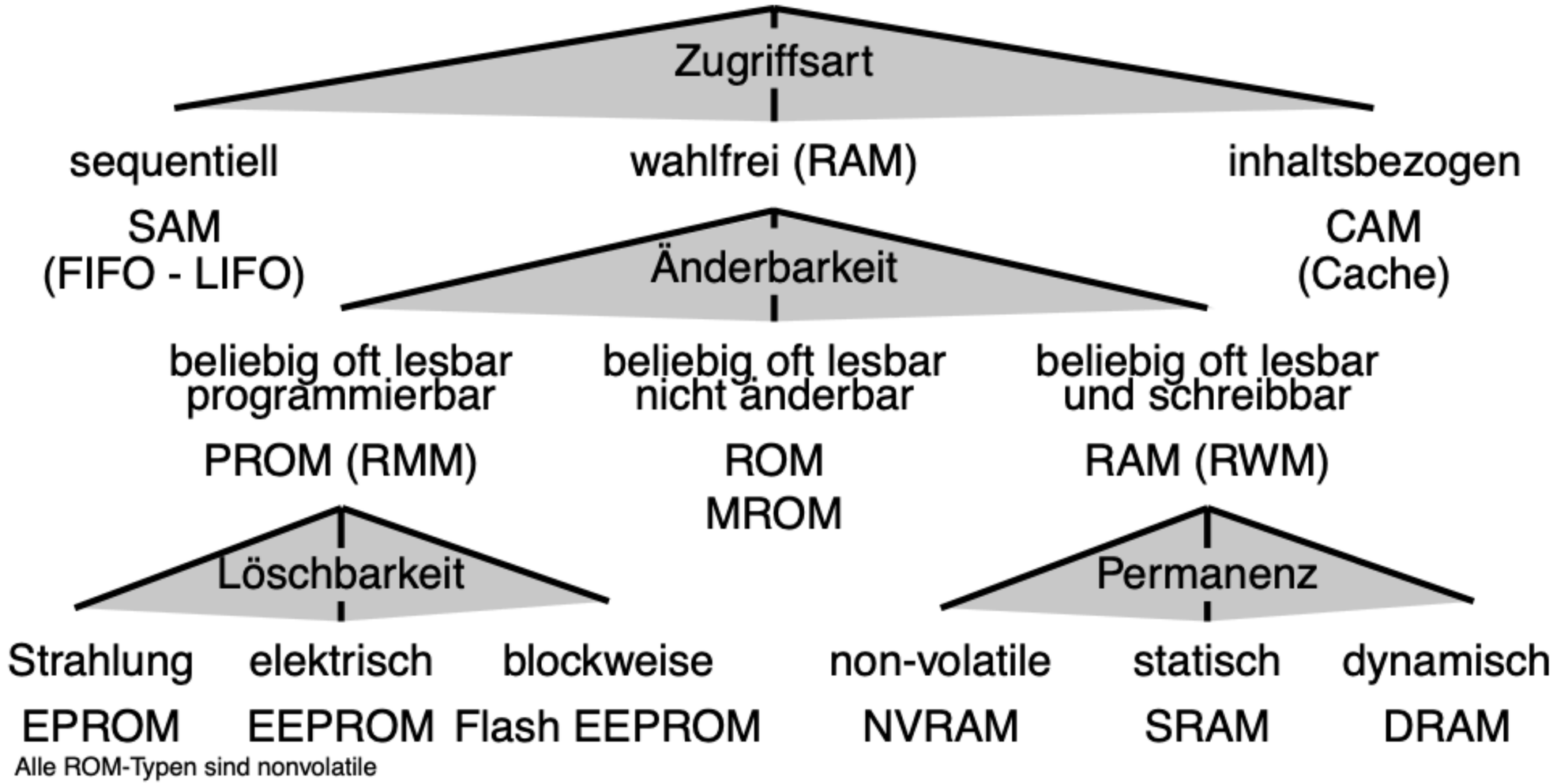
Speicherbandbreite:
maximale Datenmenge pro Zeiteinheit bzgl. der Lese-/Schreibzugriffe

Beispiel Zugriffszeiten

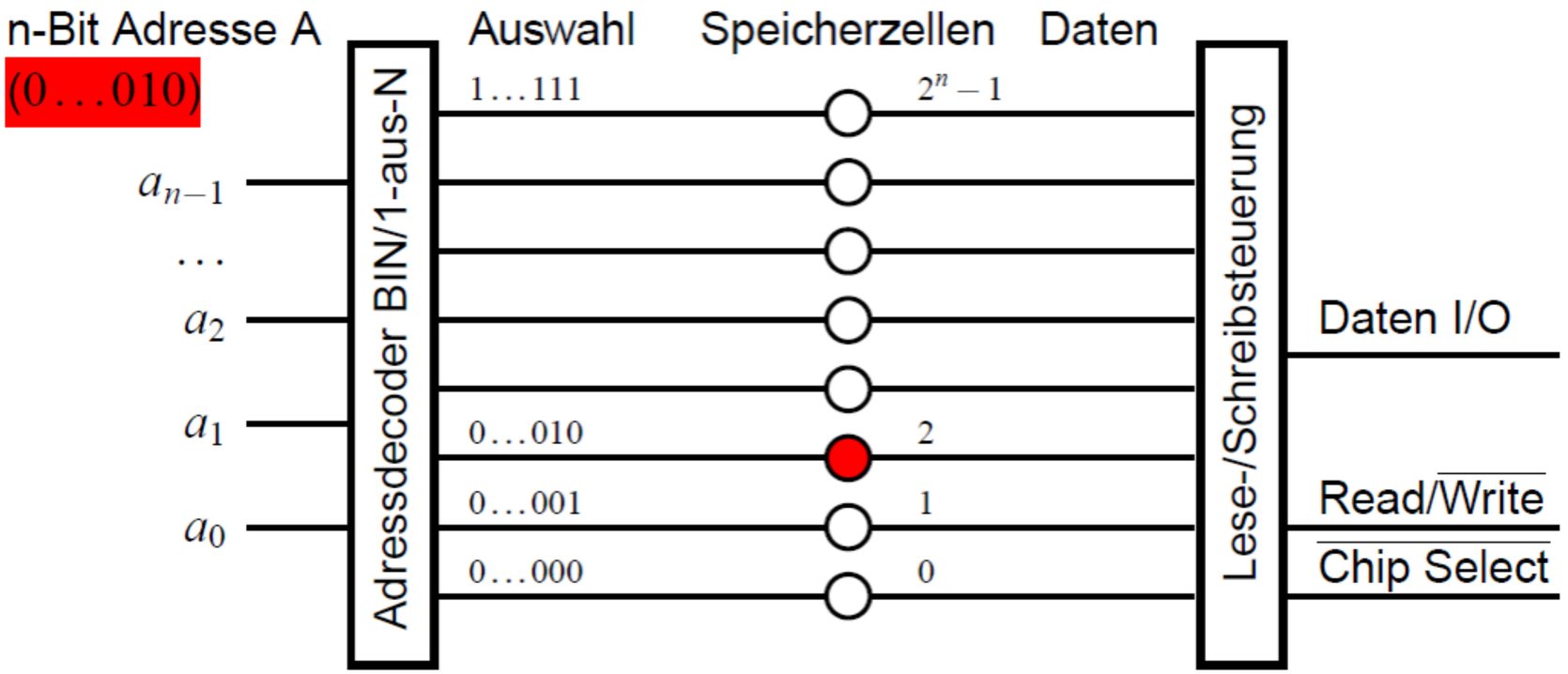


Aufbau und Funktion von Halbleiterspeichern

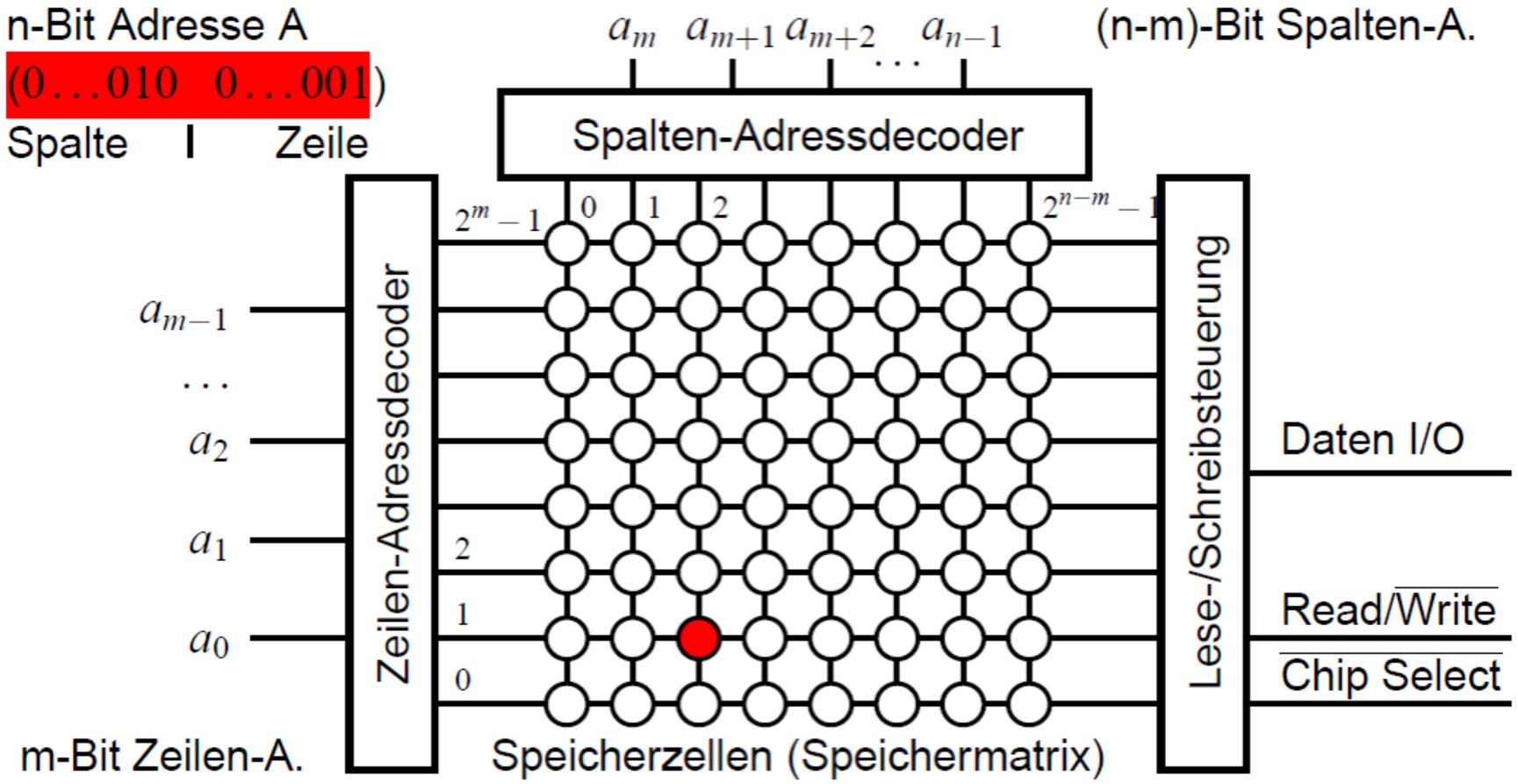
Klassifizierung Halbleiterspeicher



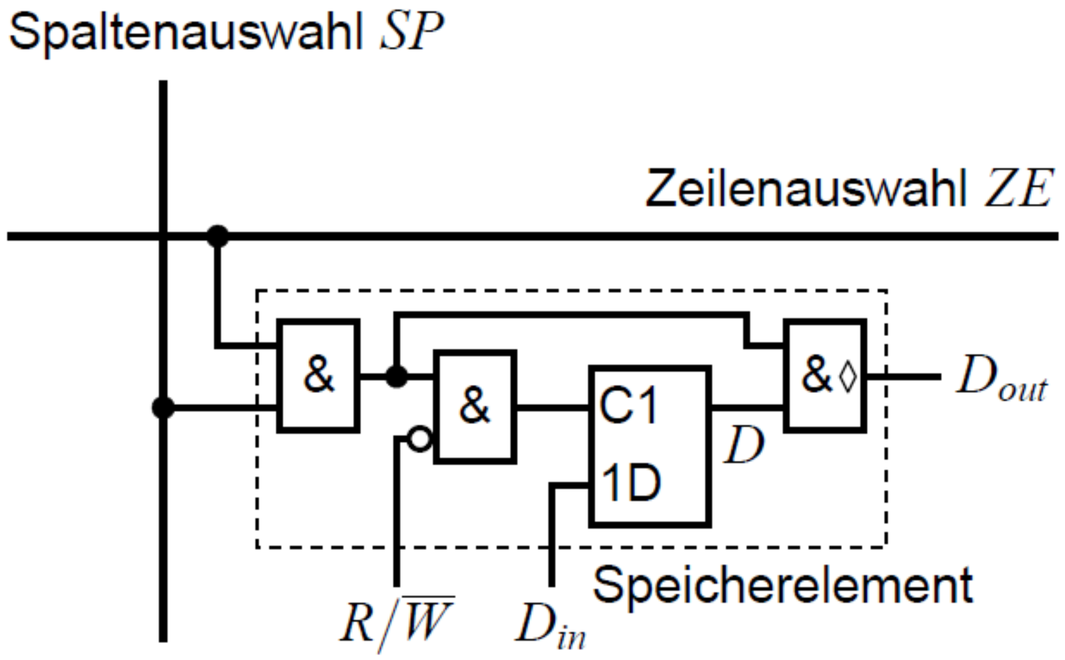
Speicherorganisation: Eindimensionale Adressierung



Speicherorganisation: Zweidimensionale Adressierung



Adressierung eines Speicherelementes



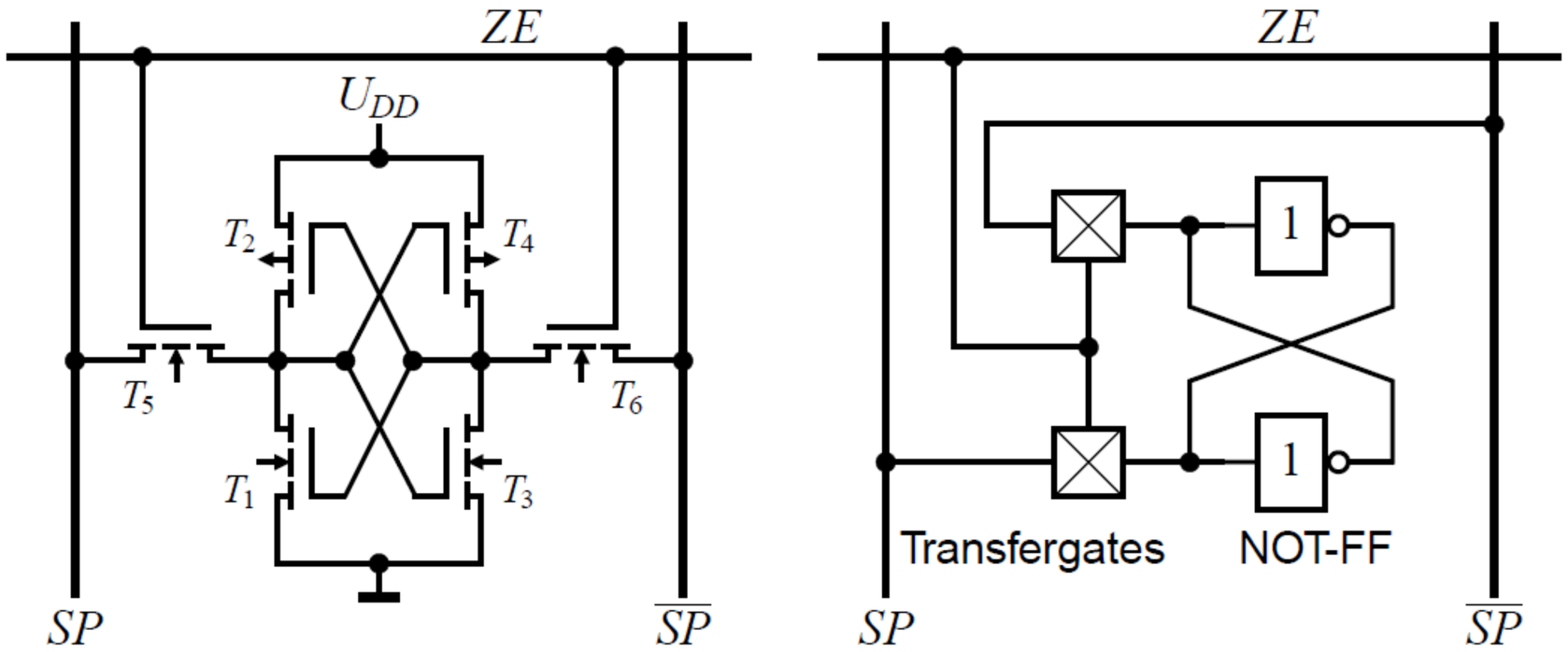
Speichertabelle

ZE	SP	$R/\overline{W}$	D^+	D_{out}
0	0	X	X	0
0	1	X	X	0
1	0	X	X	0
1	1	0	D_{in}	D_{in}
1	1	1	X	D

$R/\overline{W}$: 0 - schreiben
1 - lesen

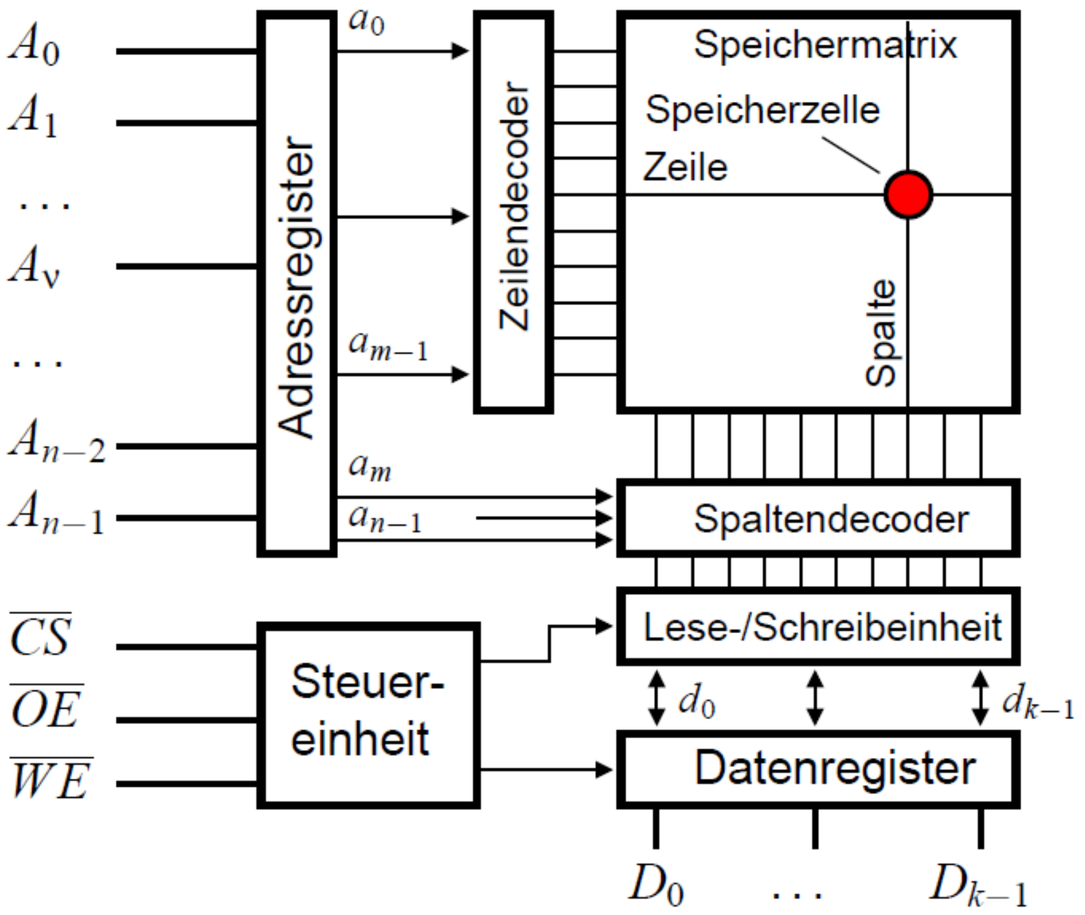
X - Don't-Care: 0 oder 1

Speichertechnologie: CMOS – SRAM



6-Transistor Speicherelement (T_1, T_2, T_3, T_4: SR-FF, T_5, T_6: Auswahltransistoren)
ZE: Zeilenanwahl, Adressleitung SP: Spaltenanwahl, Bitleitung

Aufbau eines SRAM



$\overline{CS}$ Chip Select

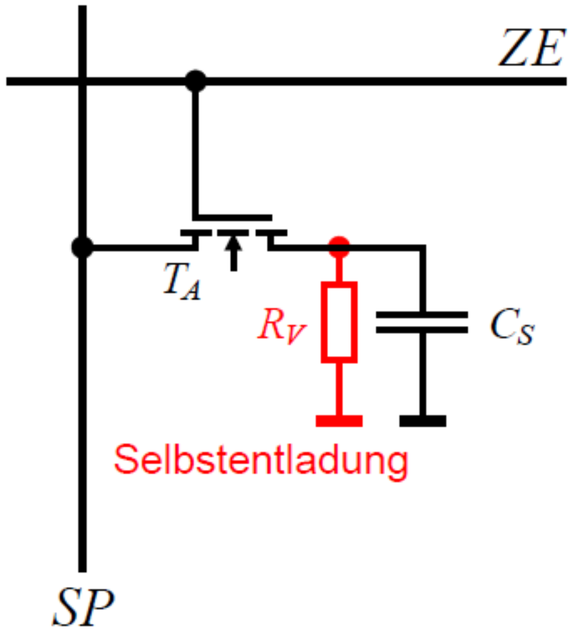
$\overline{OE}$ Output Enable

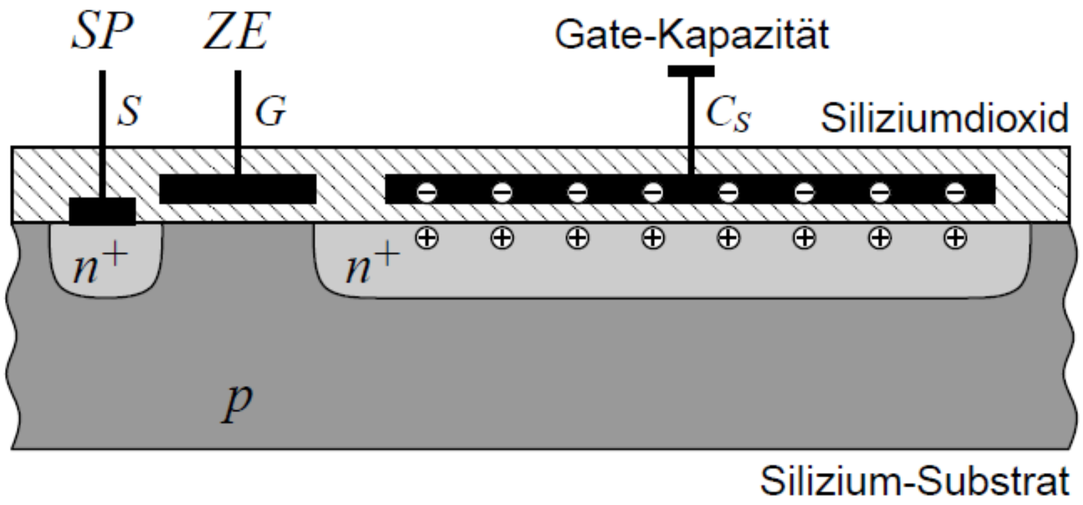
$\overline{WE}$ Write Enable

A Adresse

D Daten

Speichertechnologie: CMOS – DRAM

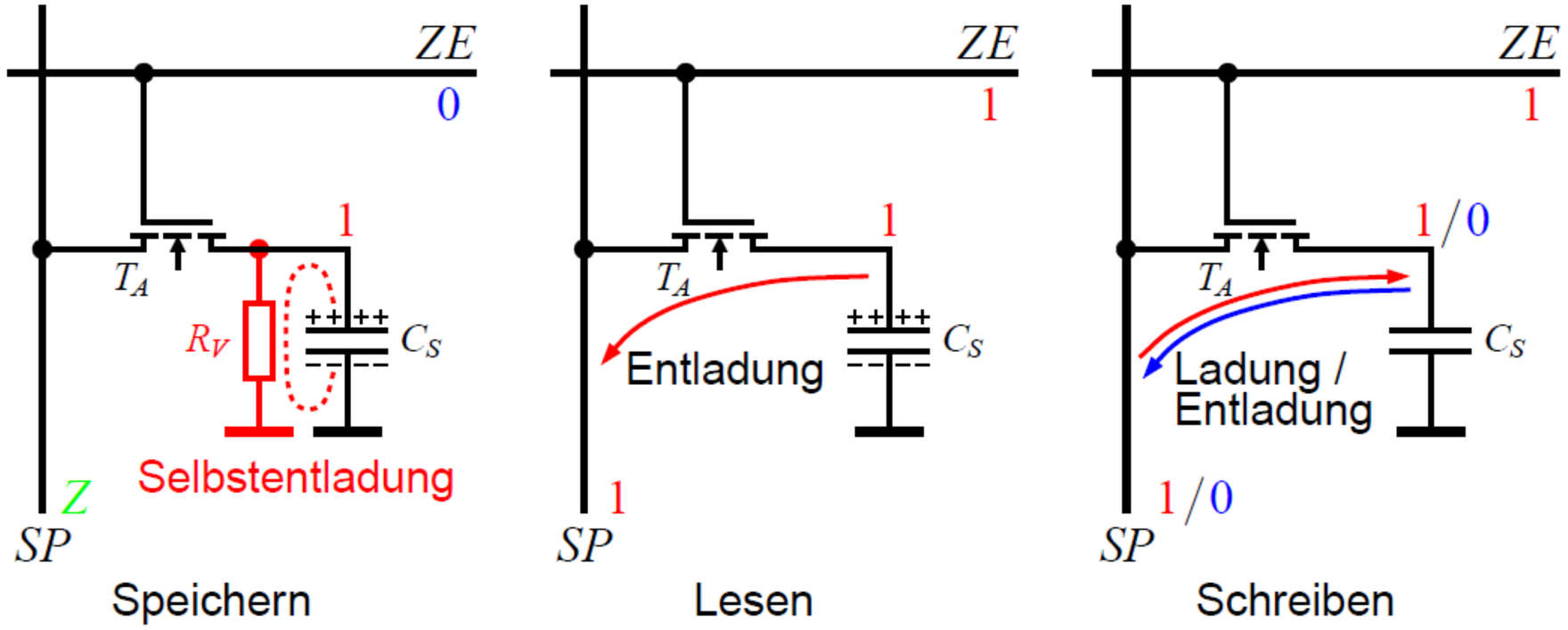




Schnittdarstellung planare DRAM-Speicherzelle

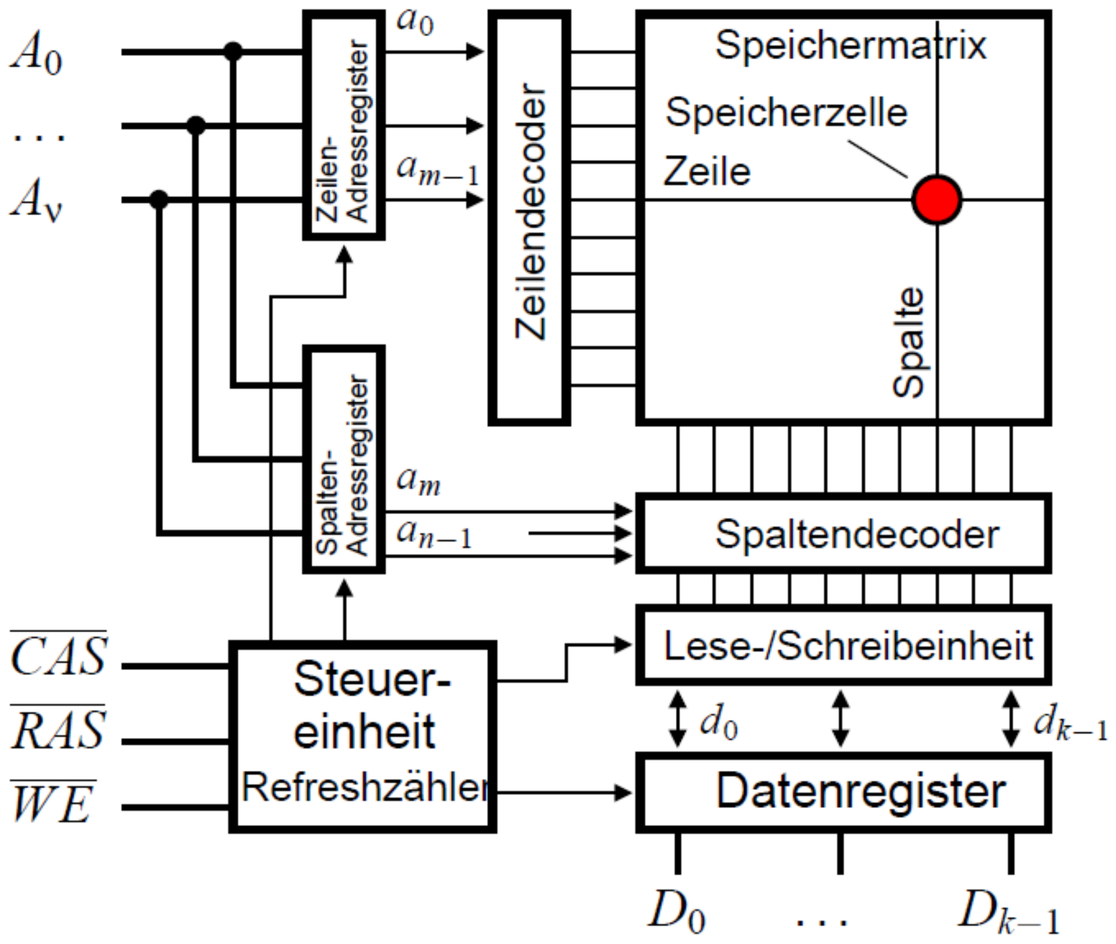
1-Transistor Speicherelement (C_S: Speicherkapazität, T_A: Auswahltransistor)
ZE: Zeilenanwahl, Adressleitung SP: Spaltenanwahl, Bitleitung

DRAM – Funktionsweise



1-Transistor Speicherelement (C_S: Speicherkapazität, T_A: Auswahltransistor)
ZE: Zeilenanwahl, Adressleitung SP: Spaltenanwahl, Bitleitung

Aufbau eines DRAM



$\overline{CAS}$ Column Address Strobe

$\overline{RAS}$ Row Address Strobe

$\overline{WE}$ Write Enable

A Adresse

D Daten

Auffrischen eines DRAM

Zielstellung: Ladungsregenerierung der Speicherkondensatoren wegen Selbstentladung (alle 1-16 ms notwendig),

RAS-only Refresh: Anlegen der Zeilenadresse ohne Spaltenadresse (Blindlesezyklus über DMA und Timer gesteuert – zeitaufwendig). Zeilen werden komplett gelesen und wieder zurückgeschrieben, ohne Datentransfer.

CAS-before-RAS Refresh: Refresh-Adresse wir von einem speicherinternen Adresszähler durchgezählt. Liegt CAS vor RAS an, wird mit den Refresh-Zyklen ab dem aktuellen Zählerstand begonnen.

Hidden-Refresh: DRAM realisiert intern (versteckt hinter dem Lesezyklus), durch einen speicherinternen Adresszähler gesteuert, selbstständig Refresh-Zyklen, ohne ein explizites externes Signal.

Vergrößerung eines RAM

Vergrößerung der Datenbreite bei konstantem Adressraum

$\rightarrow$ Vergrößerung der Datenbreite von 1-Bit auf m-Bit

- Parallelschaltung der Adressen $A_0 \ldots A_{n-1}$ und von $\overline{CS}$, $\overline{WE}$ der Speicher
- Datenleitungen der Speicher D als $D_0 \ldots D_{m-1}$ parallel herausführen

Vergrößerung des Adressraumes bei konstanter Datenbreite

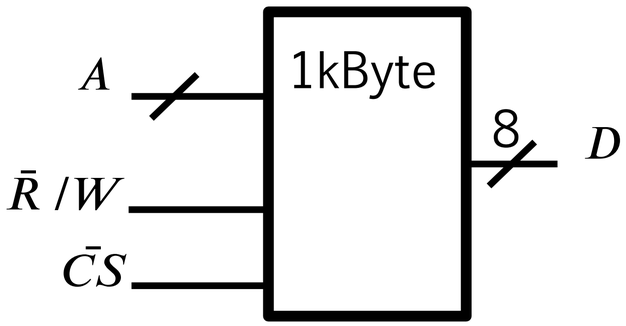
$\rightarrow$ Vergrößerung des Adressraumes von n-Bit auf m-Bit

- Parallelschaltung der Adressen $A_0 \ldots A_{n-1}$ und von $\overline{WE}$ der Speicher
- Schaltung der Adressen $A_n \ldots A_{m-1}$ und von $\overline{CS}$ auf einen Adressdecoder
- Verbindung von $\overline{CS}$ der Speicher mit den Ausgängen des Adressdecoders

$\Rightarrow$ Gleichzeitige Vergrößerung von Datenbreite und Adressraum analog

Aufgabe – Speicherdimensionierung

1. Folgende Speicherelemente mit der Größe von 1kByte sind gegeben. Der Datenbus (D) hat eine Breite von 8 Bit:



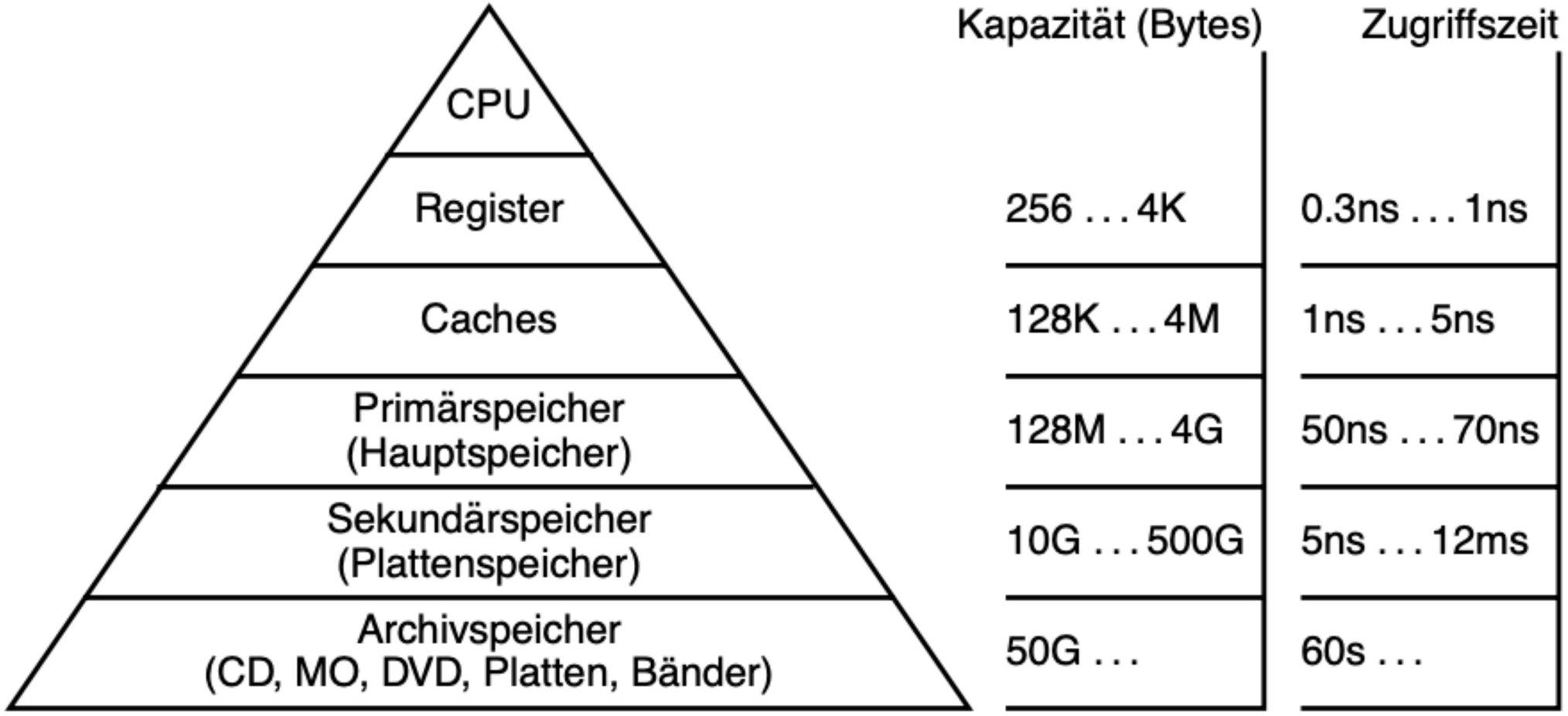
a) Wie viele Adressbits (A) hat ein solches Element?

b) Wie kann ein Speicher von 2 kByte mit einem Datenbus von 16 Bit Breite realisiert werden?

c) Wie kann der Adressraum des Speichers (unter Beibehaltung des Datenbusses von 8 Bit) auf 4 kByte erhöht werden?

Speicherhierarchie

Speicherhierarchie



Speicherkapazität ↔ Speicherzugriffszeit

Speicher mit kurzen Zugriffszeiten lassen sich nur mit relativ geringer Kapazität und mit hohem Kostenaufwand realisieren.

Dagegen lassen sich Speicher mit großer Kapazität nur mit relativ langen Zugriffszeiten kostengünstig realisieren.

Mit der Entfernung von der CPU wachsen die Zugriffszeit und die Kapazität der Speicher gleichermaßen, die Realisierungskosten pro Byte sinken dagegen.

Zielstellung:

- Erhöhung der Speicherbandbreite und Verringerung der Zugriffszeiten bei gleichgroßer oder größerer Speicherkapazität (Realisierungsfrage).
- Minimierung der Speicherkosten/Byte Speicherkapazität (Kostenfrage).

Lokalitätsprinzip des Speichers

Zeitliche Lokalität:

Nach einem Speicherzugriff ist die Wahrscheinlichkeit hoch, daß in einem der nächsten Befehle ein erneuter Zugriff den selben Speicherplatz erfolgt.

Örtliche Lokalität:

Nach einem Speicherzugriff ist die Wahrscheinlichkeit hoch, daß in einem der nächsten Befehle ein Zugriff auf einen benachbarten Speicherplatz erfolgt.

Ursachen zum Lokalitätsprinzip:

Befehle: Sequentieller Befehlsstrom, Programmschleifen, Unterprogrammen, Abspeicherung in Bibliotheken . . .

Daten: Zusammenhängende Datenstrukturen, Speicherzuteilung, Variablenanordnung durch Compiler . . .

90/10 Regel beim Speicher

90% aller Speicherzugriffe erfolgen auf nur 10% der Speicherplätze.

Konfliktlösung: Speicherkapazitaät ↔ Zugriffszeit

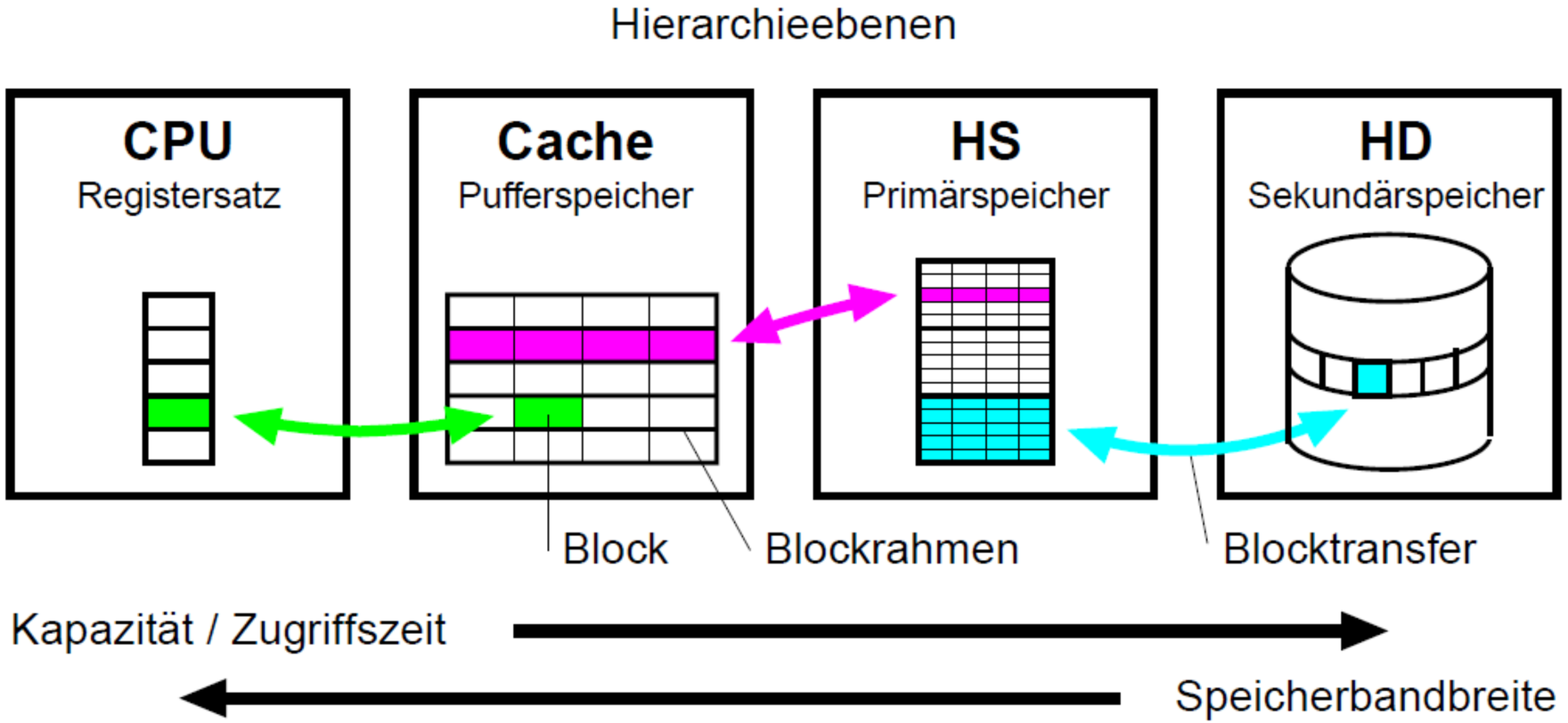
Lösung unter Ausnutzung des Lokalitätsprinzips:

- Häufig benötigte Daten werden im nahen kleinen und schnellen Speicher gehalten (als Kopien der Originaldaten).
- Seltener benötigte Daten werden im fernen großen, jedoch langsamen Speicher gehalten.
- Reicht die Kapazität des kleinen schnellen Speichers nicht mehr aus, so werden nicht mehr benötigte Daten in den großen langsamen Speicher ausgelagert und die neu benötigten Daten aus diesem nachgeladen.

⇒ **Aufbau einer Speicherhierarchie, Bildung von Hierarchieebenen**

Blockprinzip:
Der Austausch der Daten zwischen den einzelnen Hierachieebenen erfolgt nicht einzeln, sondern zusammengefaßt zu Blöcken (effektiverer Datentransfer).

Speicherhierarchie: Blockprinzip



Grundbegriffe der Speicherhierarchie

Block: Kleinster zusammenhängender Bereich, der zwischen den Hierarchieebenen transferiert, ausgetauscht wird (nicht notwendig gleich groß).

Blockrahmen (Frame): Bereich der höheren Hierarchieebene in den ein Block eingelagert, transferiert werden kann.

Blocktransfer: Blockaustausch zwischen zwei Hierarchieebenen (Burst, . . .).

Treffer (Hit): Die gesuchten Daten befinden sich in einem Blockrahmen der höheren Hierarchieebene und sind aktuell.

Trefferrate (Hit Rate): Relative Trefferanzahl bezogen auf die Gesamtzugriffe.

Fehlzugriff (Miss): Die gesuchten Daten wurden nicht in einem Blockrahmen der höheren Hierarchieebene gefunden bzw. sind nicht aktuell, kein Treffer.

Fehlerrate (Miss Rate): Relative Fehleranzahl (= 1 - Trefferrate).

Fehlerzuschlag (Miss Penalty): Zeit, die bei einem Fehlzugriff zusätzlich bis zur Erlangung der Daten benötigt wird.

Probleme der Speicherhierarchie

Abbildungsproblem:

Wie erfolgt die Abbildung der Blöcke einer Ebene auf die der nächst höheren?

Identifikationsproblem:

Wie werden die gesuchten Daten (Blöcke) lokalisiert und identifiziert?

Ersetzungsproblem:

Welcher Block wird beim Nachladen eines neuen ersetzt und wie?

Aktualisierungsproblem:

Wann und wie erfolgt die Aktualisierung der Blöcke in den einzelnen Ebenen bei der Veränderung von Daten in einem Block?

Konsistenzproblem (Kohärenzproblem):

Die Daten der Blöcke einer Hierarchieebene sind konsistent in den Blöcken aller niederen Ebenen enthalten (Datenkonsistenz über alle Ebenen – mit Ausnahme des Registersatzes) $\Rightarrow$ Aktualisierungsproblem.

Register und Registersatz

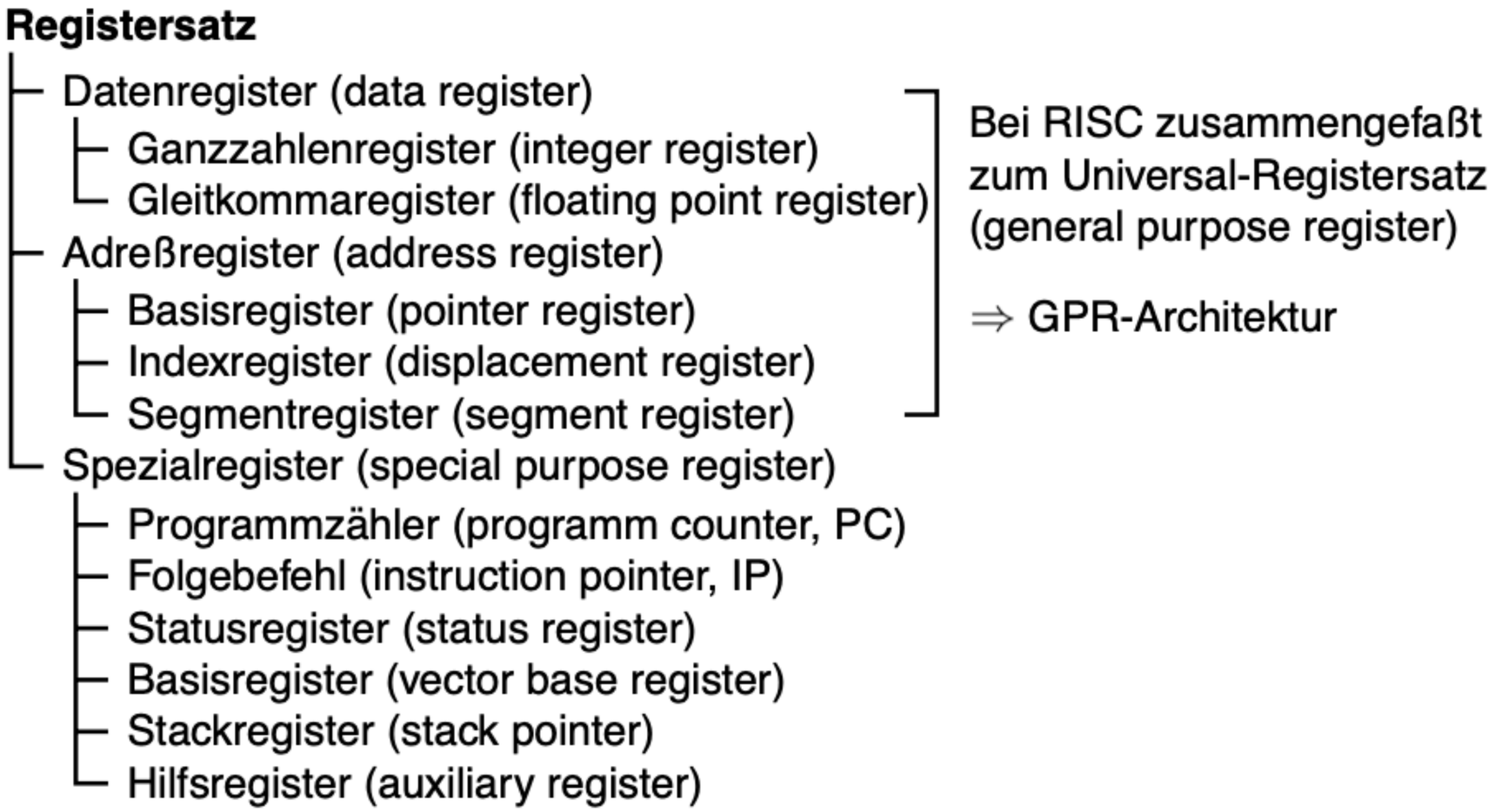
Register:

- SRAM-Speicher (Flipflop-Kette) innerhalb der CPU (mit CPU-Takt getaktet),
- Nutzung als Universalregister oder als Spezialregister (Registertypen, ...),
- Sondrefunktionen, auch verteilt innerhalb der CPU (Hilfsregister, ...),
- Realisierung in verschiedenen Datenformaten (Halbwort, Word, ...),
- Nutzung zur Rechnersteuerung (Befehlszähler, Statusregister, ...).

Registersatz:

- fest organisierter Satz von Registern innerhalb der CPU (8, 16, 32, ...),
- Operandenspeicherung, Registerspeicher der ALU (Daten, Adressen, ...),
- Register werden nicht adressiert, sondern direkt ausgewählt (Multiplexer),
- werden als Multiport-RAM ausgeführt (z.B. 2 Leseprots + 1 Schreibport, ...),
- aufwendige Hardware-Realisierung (Registerfenster, Schattenregister, ...).

Übersicht zu unterschiedliche Arten von Registern im Registersatz



Organisation des Cache Speichers

Cache Speicher

- CAM-Speicher (Content Addressable Memory) als Pufferspeicher zur Überbrückung bzw. Anpassung stark unterschiedlicher Zugriffszeiten (z.B. Prozessorregister – Hauptspeicher – Festplatte).
- Der Cache arbeitet inhaltsorientiert, ist nicht direkt adressierbar. Der Vergleich mit dem Inhalt kann auch maskiert erfolgen (Ausblenden einzelner Bits). Ein Cache-Zugriff ist nicht immer eindeutig.
- Im Cache werden nur Kopien der aktuellen Speicherinhalte der darunter liegenden Hierarchieebene abgebildet. Die aktuellen Daten einer Hierarchieebene befinden sich auch immer in allen darunter liegenden.
- Zur Überbrückung sehr großer Differenzen in der Zugriffszeit bzw. im Datendurchsatz können auch mehrere Caches hintereinander geschaltet werden (Primary Cache, Secondary Cache, . . .).
- Der Cache-Speicher kann sich mit auf dem CPU-Chip befinden (On-Chip Cache) oder extern (Off-Chip Cache).

Cache Begriffe

Adreßspeicher (Tag-RAM):

Hauptspeicheradresse bzw. Adreßteil des Datenblockes (Cache-Line).

Valid-Bit: Cache-Inhalt bzgl. der Adresse ist aktuell, gültig.

Dirty-Bit: Cache- und Hauptspeicherinhalt der Adresse sind nicht konsistent.

Datenspeicher (Data-RAM):

Datenblöcke (Cache-Line): Blöcke des Cache-Speichers, $\Rightarrow$ Blockprinzip.

Datenblockgröße (Cache-Line-Size): Blockgröße in Worten (Byte).

Cache-Treffer (Cache-Hit):

Daten / Befehle aktuell im Cache gefunden.

Trefferrate (Hit-Rate): Maß für die Effizienz des Caches.

Cache-Fehlzugriff (Cache-Miss):

Daten / Befehle nicht im Cache gefunden bzw. nicht aktuell.

Fehlerrate (Miss-Rate): 1 - Trefferrate.

Fehlerzuschlag (Miss-Penalty): Zugriffszeit nach Cache-Fehlzugriff.

Cache Eigenschaften

Struktur:

Ein Cache-Eintrag besteht aus dem Datenblock (Cache-Line, $\Rightarrow$ Blockprinzip) und dessen Adresse, Adressteil als Tag (Etikett) sowie Statusinformationen.

Sichtbarkeit:

Transparent (nicht sichtbar im Befehlssatz) oder nicht transparent (sichtbar, wird durch den Befehlssatz gesteuert, z.B. Laden, Invalidieren, . . .).

Adressraum:

Realer Adressraum (realer Cache) oder virtueller (virtueller Cache).

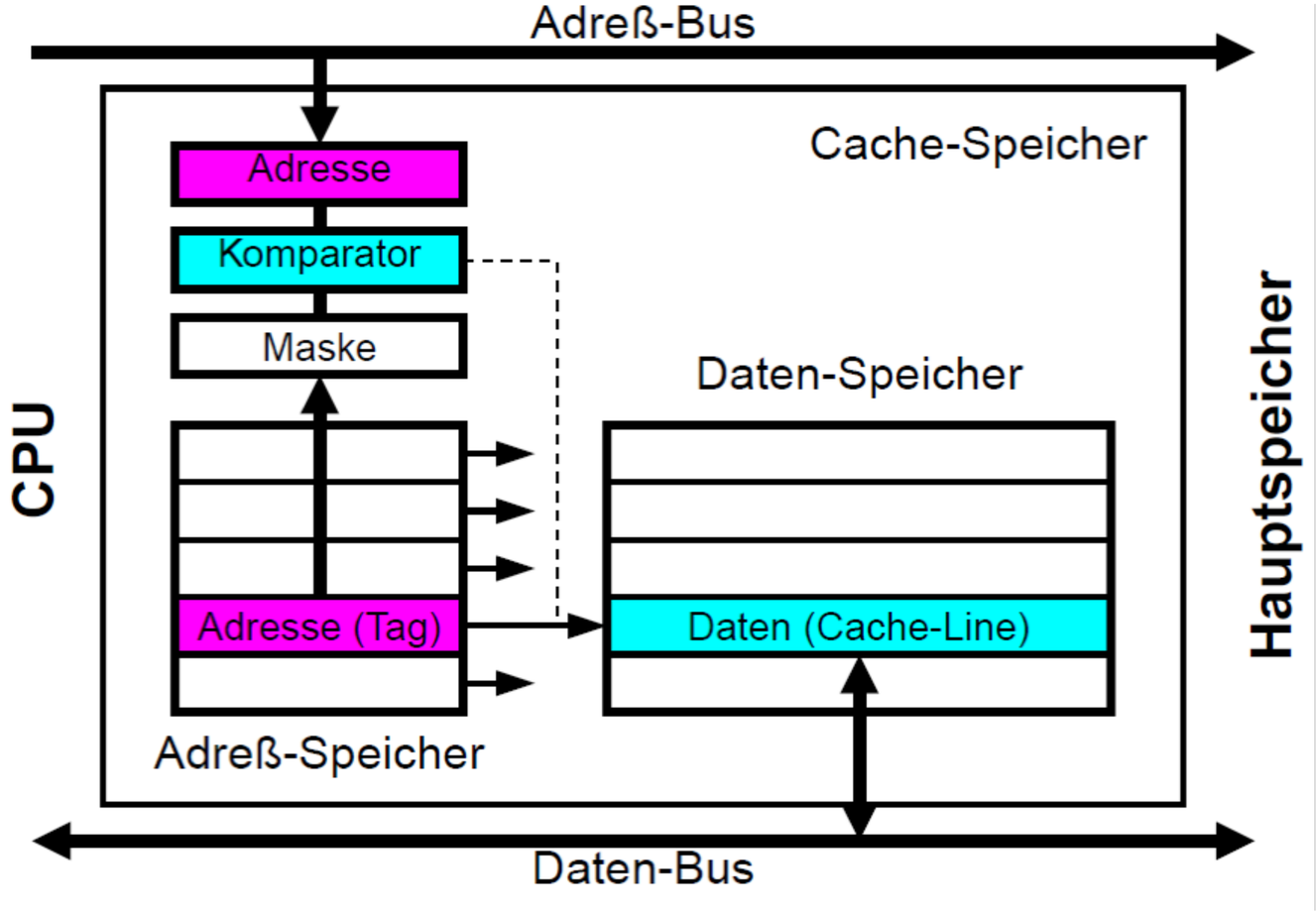
Architektur:

Befehle und Daten gemeinsam (Unified Cache) oder getrennt (Split Cache).

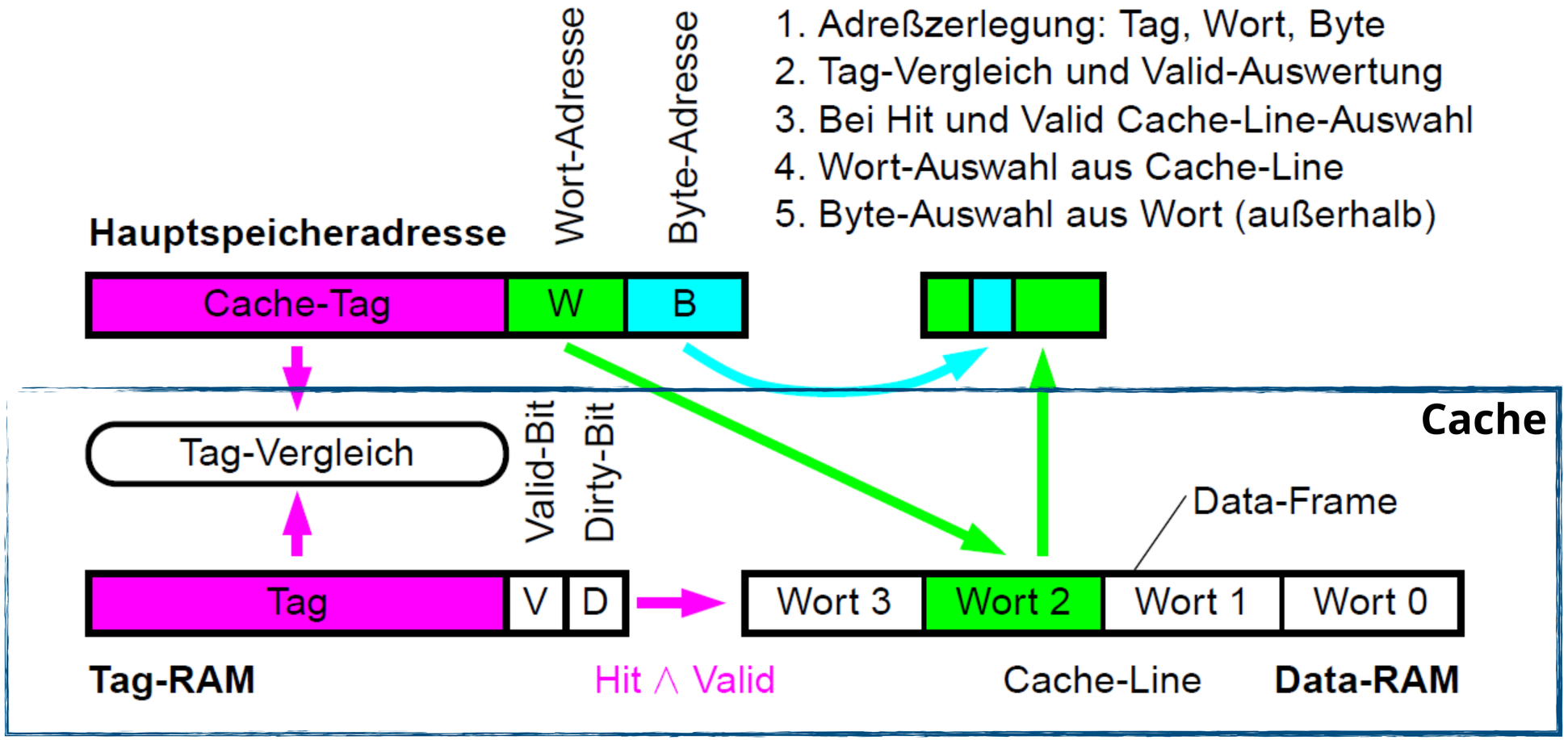
Organisation:

Lösung der Probleme der Speicherhierarchie: Abbildung, Identifikation, Ersetzung, Aktualisierung, Konsistenz, Kohärenz.

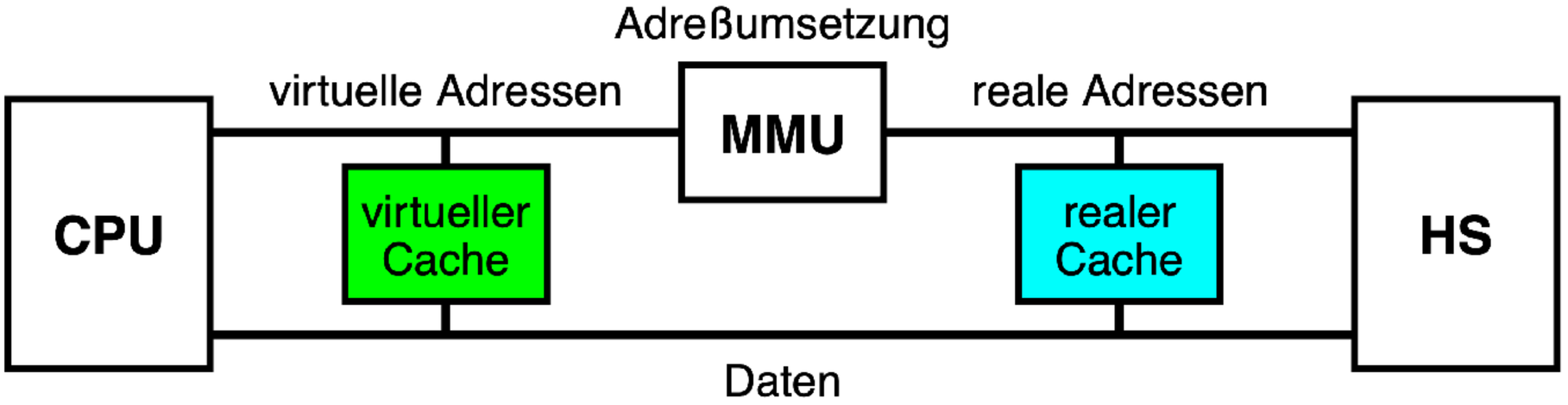
Aufbau eines Cache Speichers



Struktur eines Caches

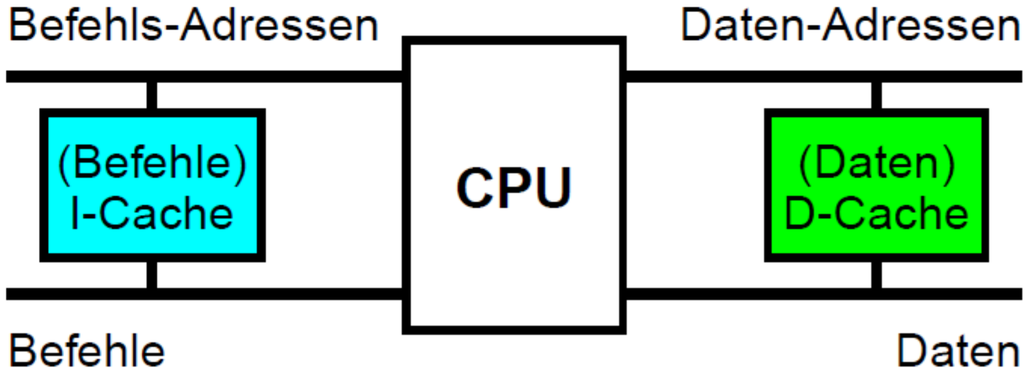


Cache-Adressraum



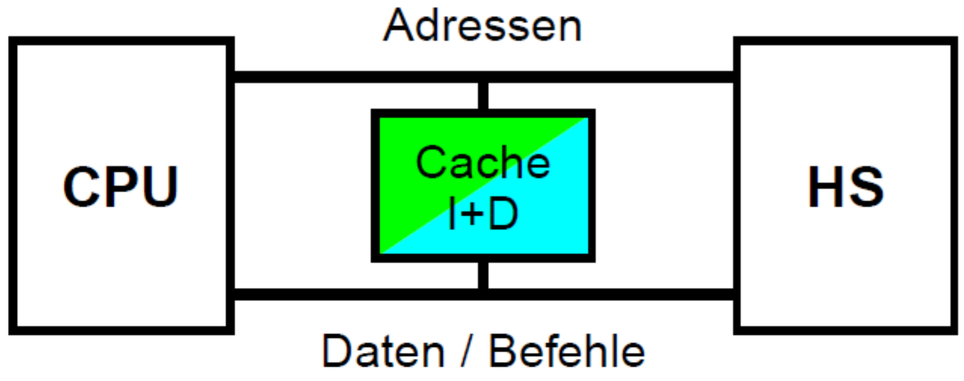
Virtueller Cache	Realer Cache
- schnell, Cache und MMU nebenläufig	- langsam, Adressen durch MMU
- Invalidierung bei Prozeßwechsel, notwendig, da gleiche Adressen möglich	- direkter I/O-Cache-Transfer möglich, da reale Adressen
- besondere Eignung als Befehls-Cache, da Rückschreiben entfällt	- für Snooping geeignet, da 1:1 Adreß-Daten-Abbildung

Cache-Architektur auf innerster Prozessorebene



Harvard - Architektur

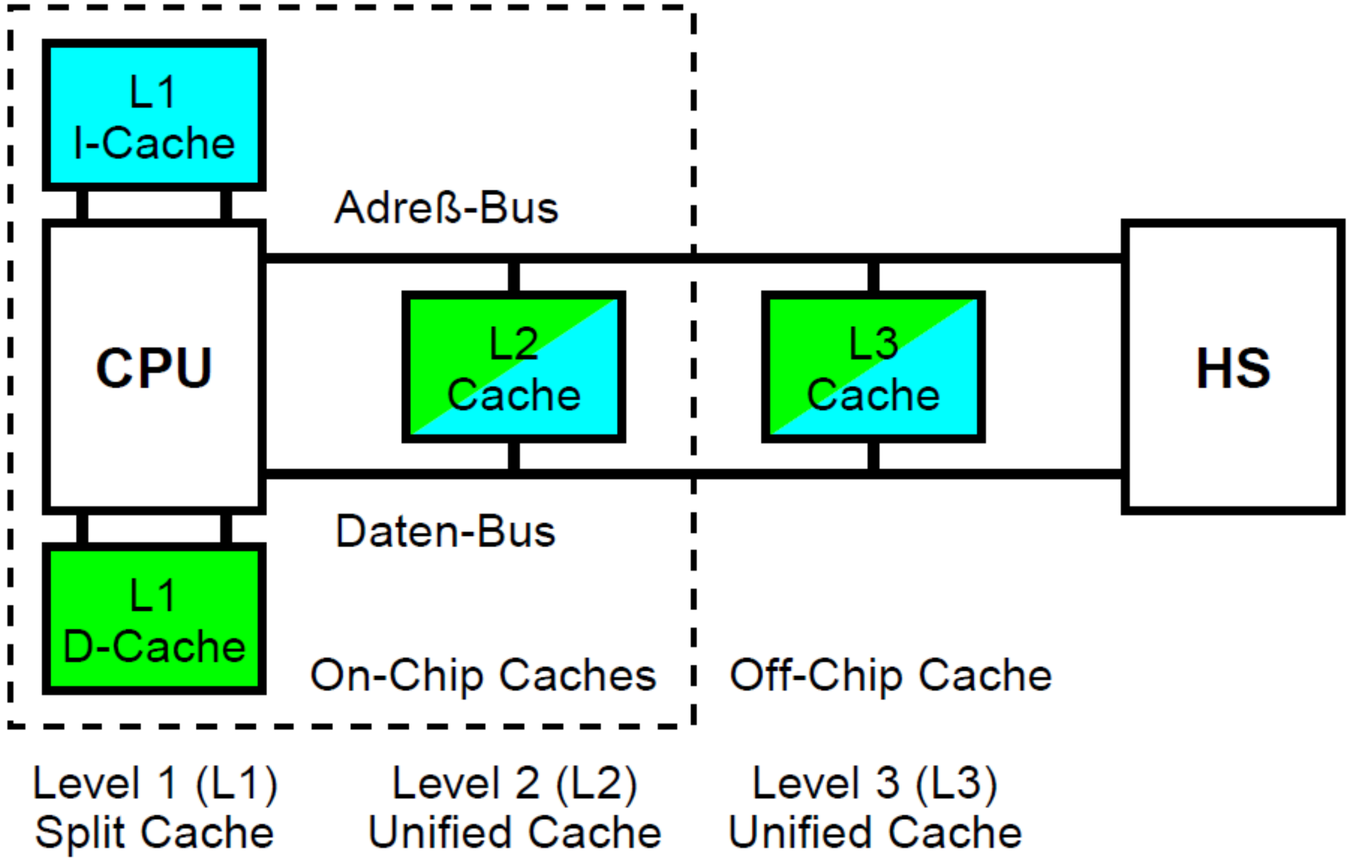
Split Cache, getrennt
für Daten und Befehle
(separate Lokalitäten)



Princeton - Architektur

Unified Cache, gemeinsam
für Daten und Befehle
(überlagerte Lokalitäten)

Moderne Cache-Architektur mit drei Ebenen



Cache-Assoziativität

Voll Assoziativer Cache (Fully-Associative Cache)

Ein Speicherblock kann sich in einem beliebigen Cache-Frame befinden (k Cache-Frame (Cache-Line) ⇒ k-fach assoziativ).
⇒ sehr gute Trefferrate, komplizierte Hardware-Realisierung, langsam (k voll-parallele Komparatoren für die gesamte Tag-Breite).

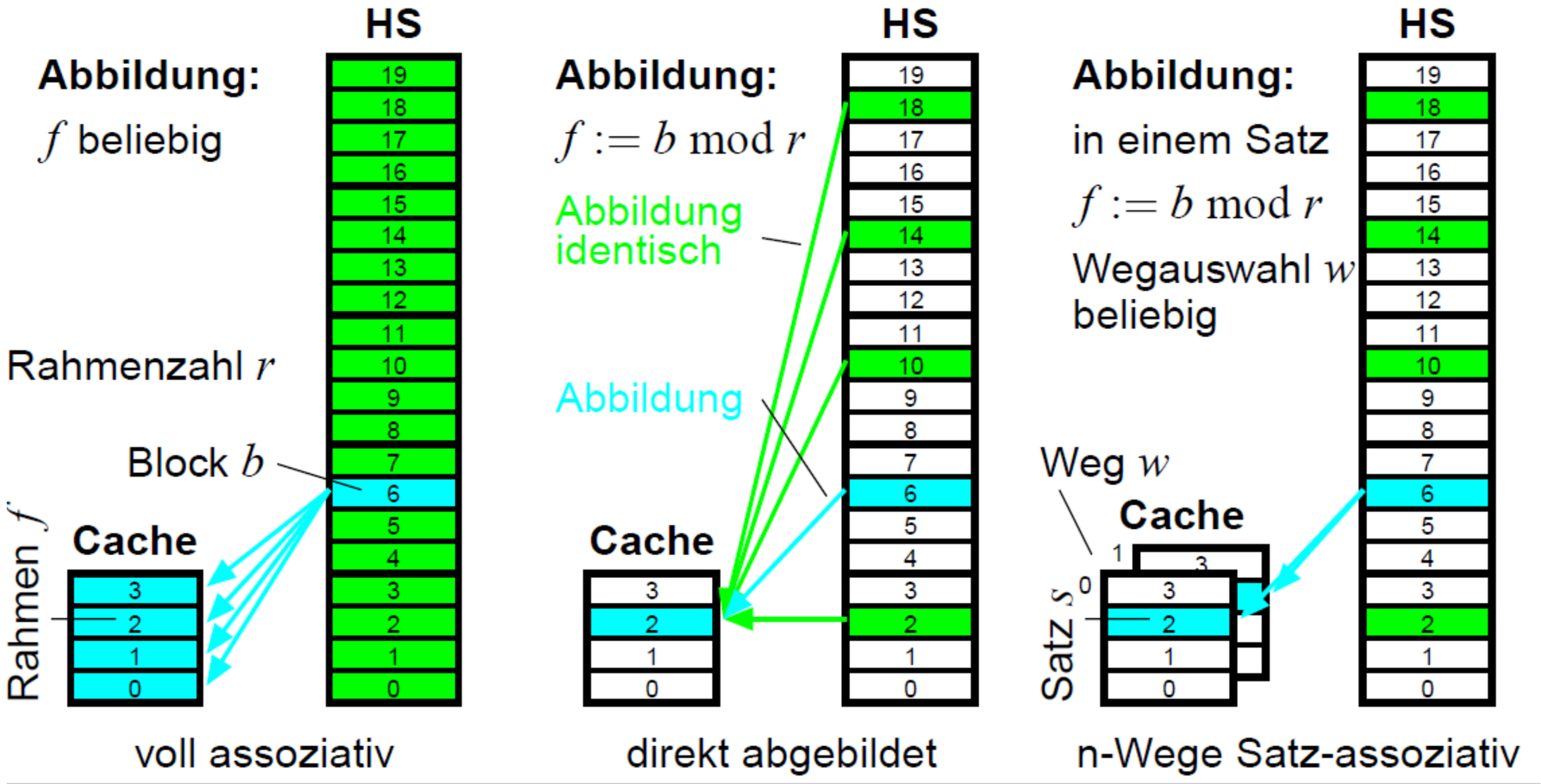
Direkt Abgebildeter Cache (Direct-Mapped Cache)

Ein Speicherblock kann sich nur in einem bestimmten Cache-Frame befinden, durch den Index-Teil der Adresse festgelegt (⇒ einfach assoziativ).
⇒ schlechte Trefferrate, einfache Hardware-Realisierung, schnell (Ping Pong Effekt (Cache Trashing) - ständiges Daten-Ein- und -Auslagern).

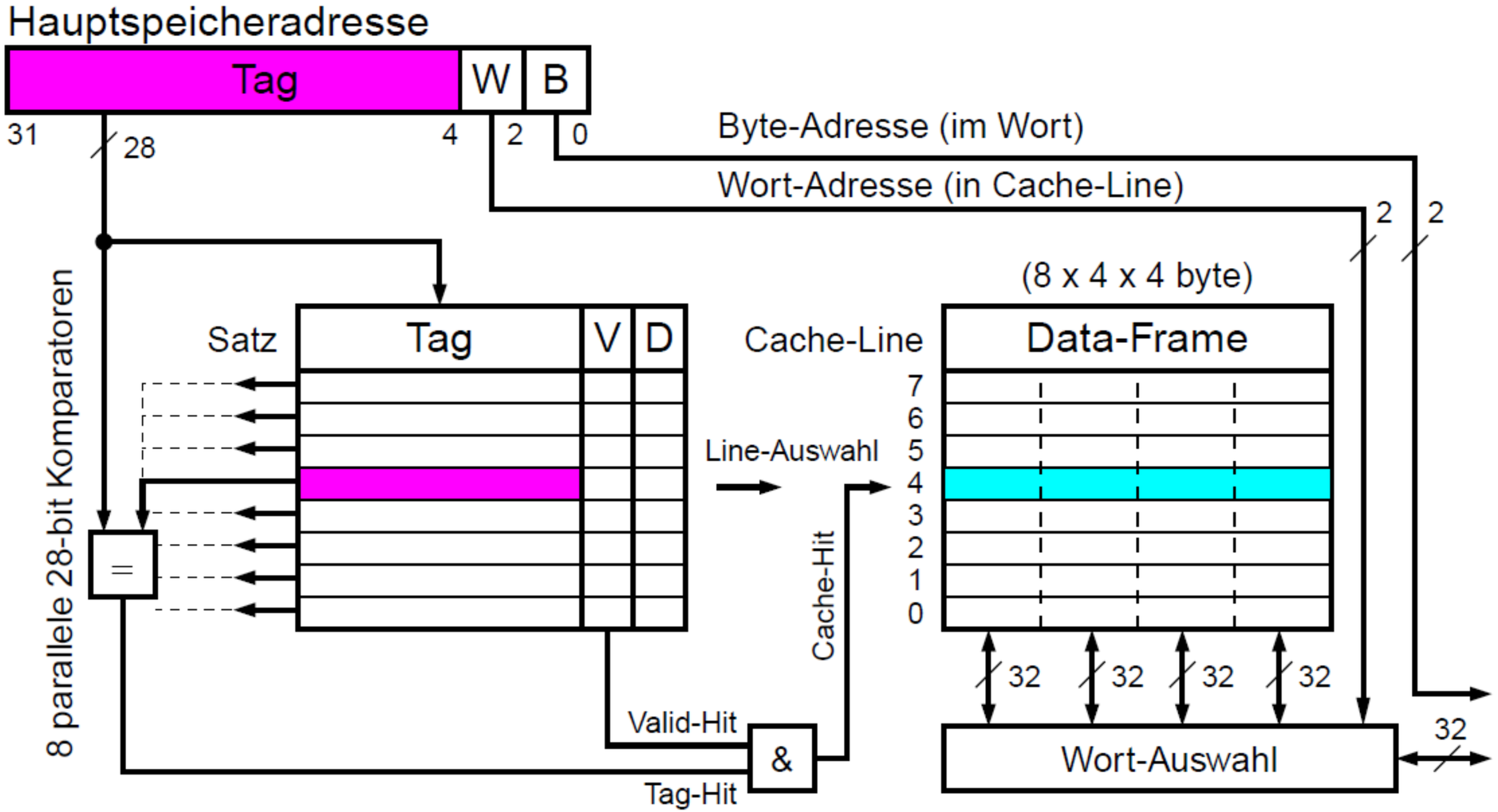
k-Wege Satz-Assoziativer Cache (k-Way Set-Associative)

Kann als parallele Anordnung von k direkt abgebildeten Caches verstanden werden, Sätze durch Index-Teil der Adresse festgelegt (⇒ k-fach assoziativ).
⇒ Kompromis aus direkt abgebildeten und voll assoziativen Cache.

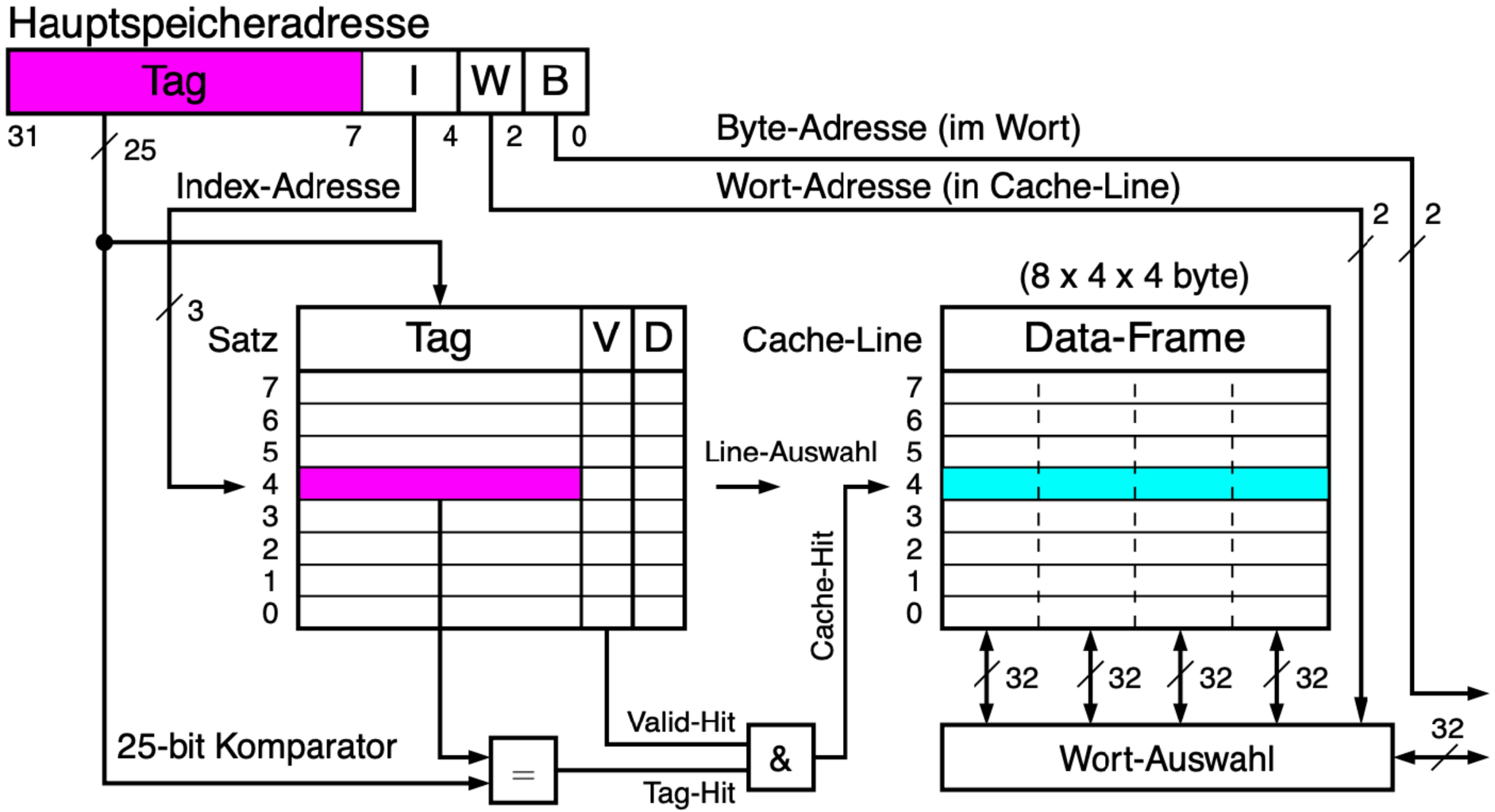
Abbildungsproblem innerhalb der Cache-Organisation



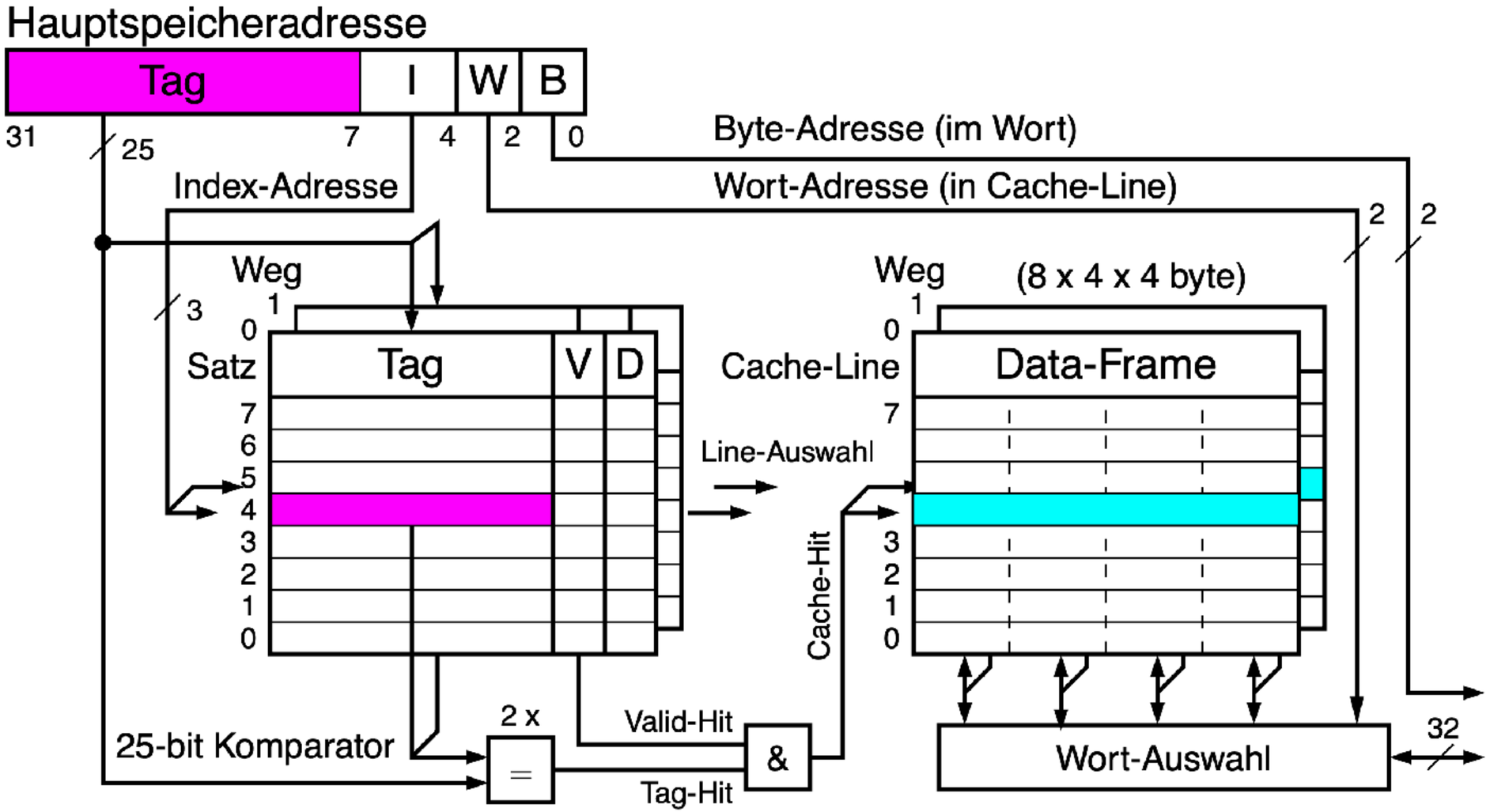
Voll-Assoziativer Cache (8-fach assoziativ)



Direkt Abgebildeter Cache (einfach assoziativ)

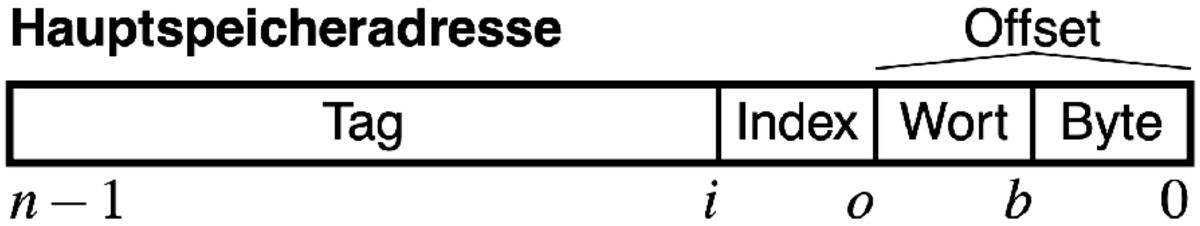


2-Wege Satz-Assoziativer Cache (2-fach assoziativ)



Unterteilung der Hauptspeicheradresse für Cache-Zugriffe

Aufteilung des Hauptspeicheradresse in Tag, Index und Offset (Wort und Byte):



HS-Adresse : n bit Index : $i-o$ bit Byte : b bit

Tag : $n-i$ bit Wort : $o-b$ bit Offset : o bit

Tag: Adreßteil zum assoziativen Vergleich im Cache-Tag-RAM.

Index: Satzauswahl (Set) im Cache (Vollassoziativer Cache $\Rightarrow$ kein Index).

Offset: Bestimmt die Blockgröße im Cache-Data-RAM (Cache-Line-Size).

Wort: Wort-Adresse innerhalb eines Datenblockes (Cache-Line).

Byte: Byte-Adresse innerhalb eines Wortes.

Cache-Organisation – Ersetzungsploblem

Welche Cache-Line wird beim Nachladen eines mehrfach assoziativen Caches verdrängt, ersetzt (beim direkt abgebildeten Cache kein Ersetzungsproblem)?

LRU (Least Recently Used):
Ersetze am längsten nicht mehr benutzte Cache-Line . Für die Entscheidung ist eine Referenzstatistik erforderlich (UC - Utilization Counter).

Zufallsprinzip (RANDOM):
Ersetze eine Cache-Line nach dem Zufallsprinzip (Pseudo-Zufallszahlen).

FIFO (First In First Out):
Ersetze älteste geladene Cache-Line (erfordert Ladestatistik).

LFU (Least Frequently Used):
Ersetze am wenigsten benutzte Cache-Line (erfordert Benutzungsstatistik).

Round Robbin : Ersetze Cache-Line in vorher festgelegter Reihenfolge.

Optimal (Perfect): Ersetze in Zukunft nicht mehr benötigte Cache-Line.

Aktualisierungsproblem — Datenkonsistenz und Datenkohärenz

Konsistenz:

Sowohl der Cache als auch der Hauptspeicher enthalten stets die identischen aktuellen Daten. Alle untergeordneten Hierarchieebenen enthalten stets auch die Daten aller übergeordneten Ebenen in identischer Form.

Kohärenz:

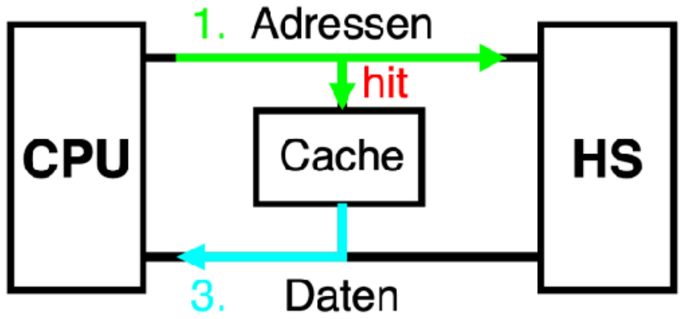
Der CPU werden immer die aktuellen Daten bereitgestellt, unabhängig davon, ob sie im Cache oder im Hauptspeicher stehen. Auf veraltete Daten (Stale Data) darf nicht zugegriffen werden ($\Rightarrow$ Konsistenzabschwächung).

Problemfälle:

- Auslagern, Rückschreiben in untergeordnete Hierarchieebenen (Castoff).
- Mehrere Master im System (CPU und DMAC, Direct Memory Access).

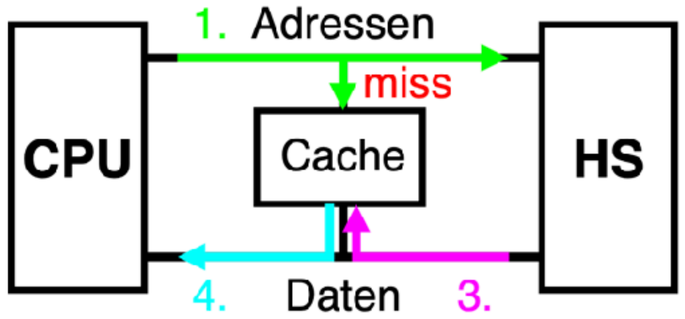
Cache Lese-Strategie (Read Strategy)

Treffer (Read-Hit)



1. Cache und HS adressieren
2. Cache-Hit, HS-Zugriff abbrechen
3. Lesen der Daten aus dem Cache

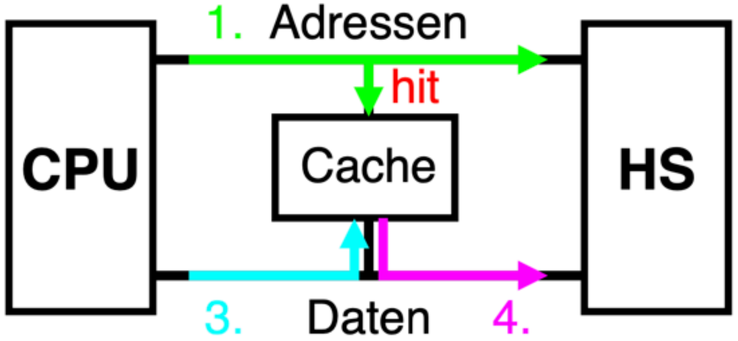
Fehlzugriff (Read-Miss)



1. Cache und HS adressieren
2. Cache-Miss, HS-Zugriff fortsetzen
3. Verdrängen einer Cache-Line (Ersetzungsproblem, Castoff)
4. Cache-Line aus HS in Cache laden
5. Lesen der Daten aus dem Cache

Treffer (Write-Hit) Cache Schreib-Strategie (Write Strategy)

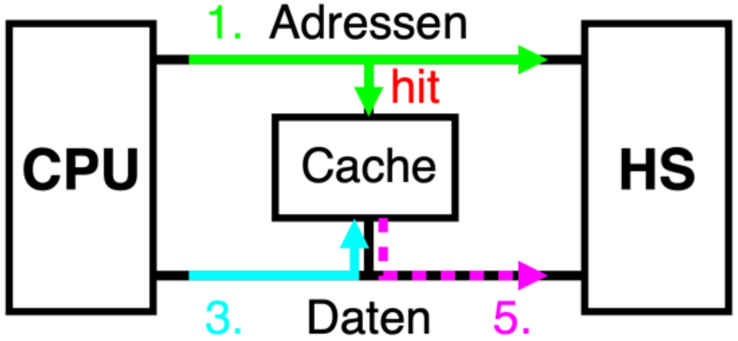
Write-Through (WT)



1. Cache und HS adressieren
2. Cache-Hit
3. Schreiben der Daten in den Cache
4. HS unverzüglich aktualisieren
   (Schreibpuffer, Write-Buffer)

⇒ Zeitaufwendig aber konsistent

Write-Back, Copy-Back (WB)

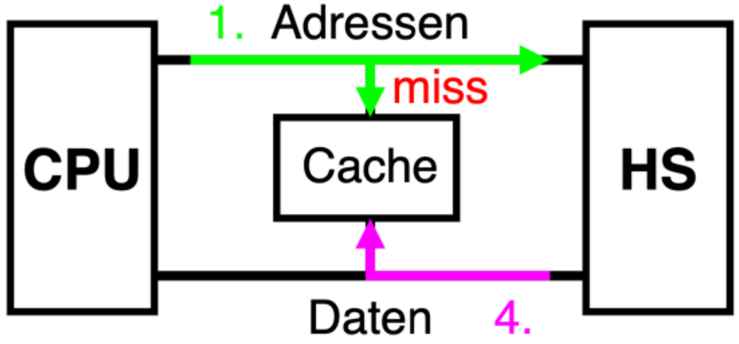


1. Cache und HS adressieren
2. Cache-Hit
3. Schreiben der Daten in den Cache
4. Setzen des Dirty-Bit
5. HS bei Verdrängung aktualisieren

⇒ Probleme mit Daten-Konsistenz

Fehler (Write-Miss) Cache Schreib-Strategie (Write Strategy)

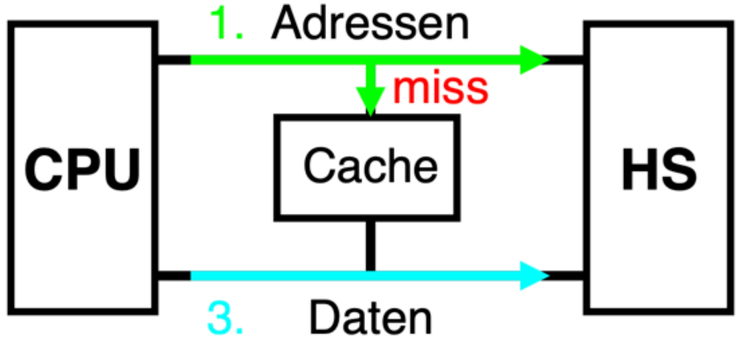
Write-Allocate



1. Cache und HS adressieren
2. Cache-Miss, HS-Zugriff fortsetzen
3. Verdrängen einer Cache-Line (Ersetzungsproblem, Castoff)
4. Cache-Line aus HS in Cache laden

Weiter analog Treffer (Write-Hit), typisch:

Write-Around (No-Write-Allocate)



1. Cache und HS adressieren
2. Cache-Miss, HS-Zugriff fortsetzen
3. Schreiben der Daten in den HS (Cache wird nicht geladen)

Write-Allocate mit Write-Back
Write-Around mit Write-Through

Weiterführende Konzepte

Virtueller Speicher als weitere Ebene

Zielstellung:

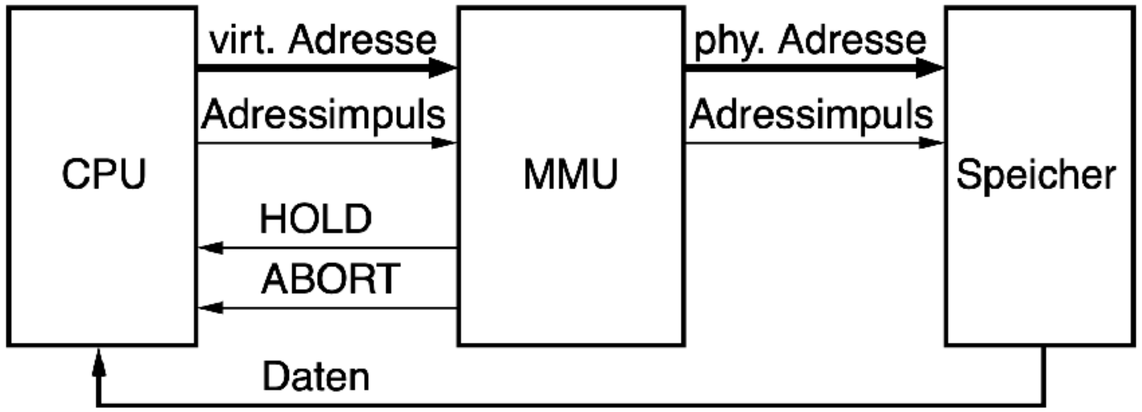
- Vergrösserung des nutzbaren Adressraumes weit über den des technisch verfügbaren Hauptspeichers hinaus
- Unterstützung mehrerer getrennter Adressräume
- Realisierung ”beliebig“ teilbarer und nutzbarer Adressräume
- Trennung von einzelnen Prozessen bzgl. Zugriffsrechte, Schutzmechanismen
- gemeinsame Nutzung von Prozessen bzw. Speicherbereichen
- Effektive Einbeziehung des Sekundärspeichers (Festplatte), kurze Lade- und Speicherzeiten
- Entlastung des Programmieren von aufwendiger Adressrechnung, Overlayprozess, Partitionierung, Verwaltungsaufgaben

Grundkonzept des Virtuellen Speichers

- Trennung von Adressraum (address space) und Speicheradressen (physikalischer Speicher)
- Einführung eines oder mehrerer beliebig grosser virtueller Adressräume
- Unterteilung des virtuellen und des physikalischen Adressraumes in Blöcke gleicher oder variabler Grösse
- Adressumsetzung (address translation) zwischen virtuellen und physikalischen Adressen durch Tabellen
- Nutzung des Sekundärspeichers für Ein- und Auslagerungen
- Automatische Realisierung der gesamten Prozessaufteilung, Speicherzuordnung, Adresstransformation, Umlagerungen, durch die Hardware und das Betriebssystem, für den Nutzer völlig transparent

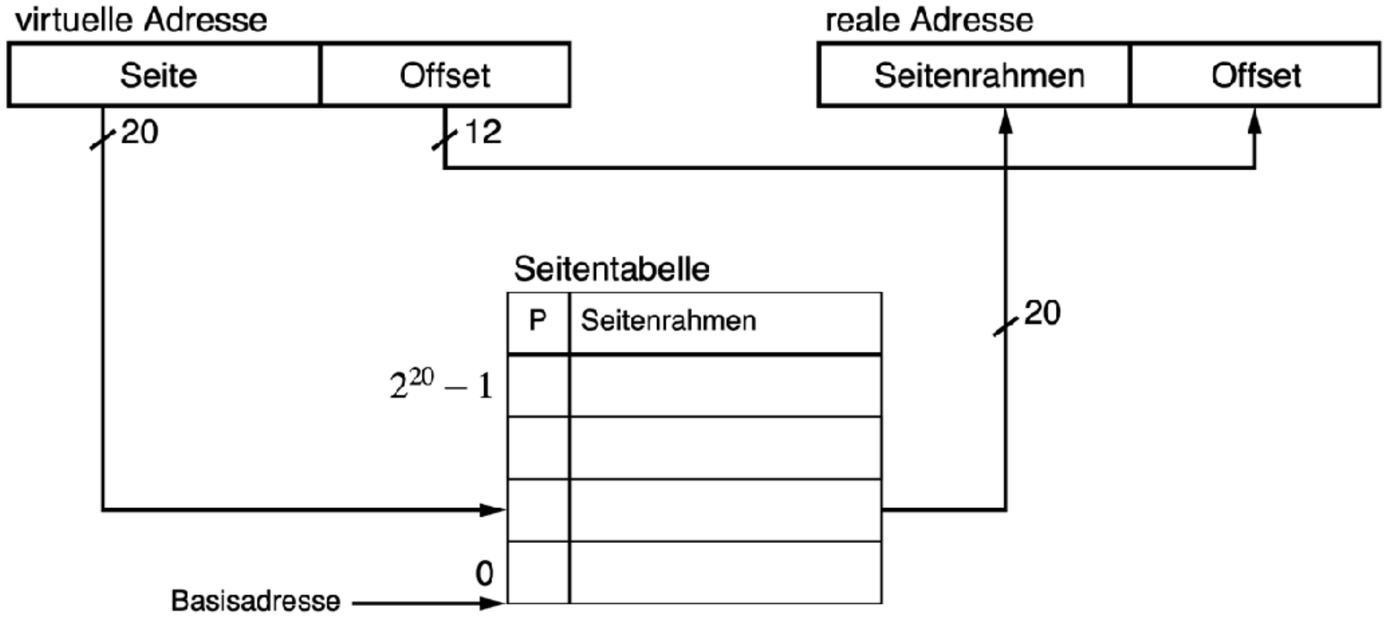
MMU (Memory Management Unit) zur Adressübersetzung

- Unterteilung der virtuellen und physikalischen Adressen in Seitennummer (Pagenumber) bzw. Seitenrahmennummer und Seitenoffset (Page Offset)
- Realisierung der Abbildung virtuelle $\rightarrow$ physikalische Adresse (Address Translation) durch eine Seitentabelle (Page Table)
- Die Adressabbildung virtuell $\rightarrow$ physikalisch erfolgt hardwaremässig durch die Memory Managemant Unit (MMU) und wird durch das Betriebssystem unterstützt



Adressbildung beim Paging

- Unterteilung des virtuellen Adressraumes in Blöcke gleicher Grösse (Seiten, Pages)
- Die Seitengrösse entspricht immer einer Potenz von 2 (z.B. 4 Kbyte, 4 Mbyte)

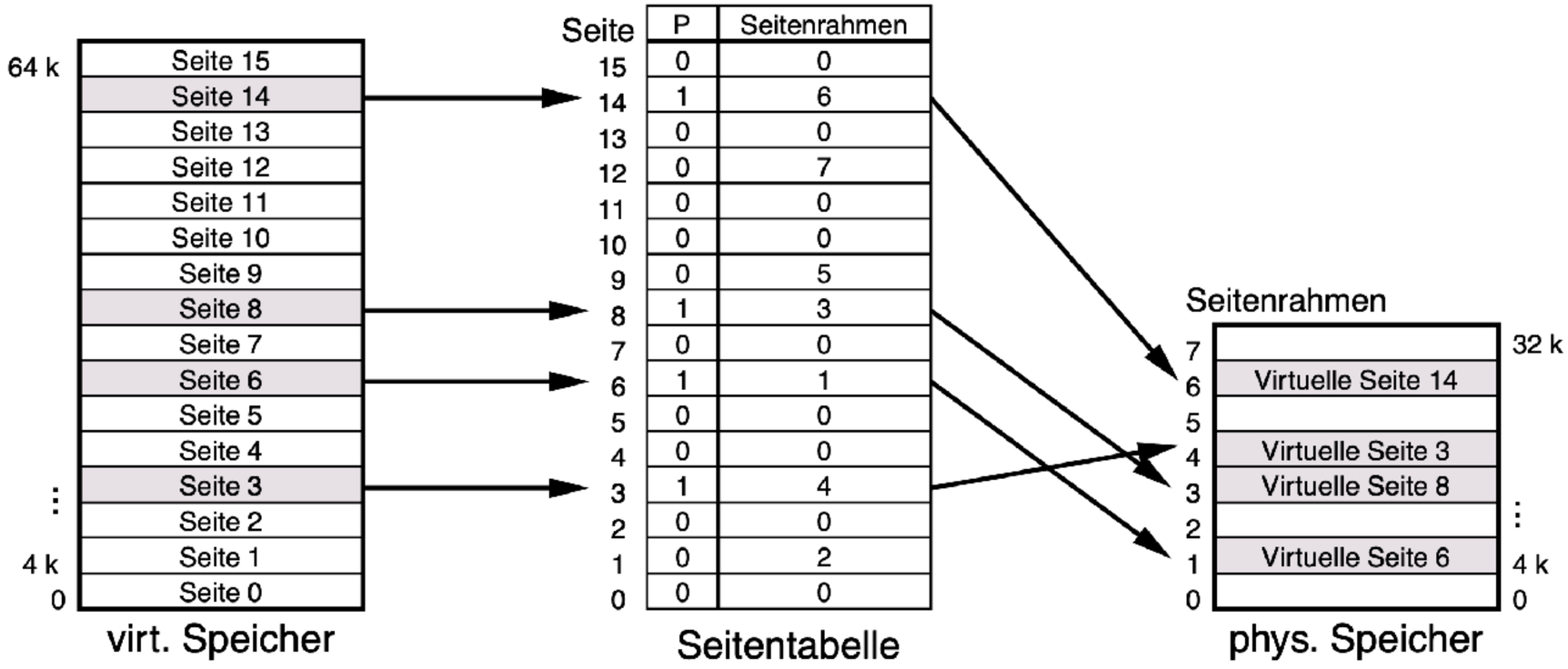


Seitentabelle enthält die Seitenrähmen der Seiten die sich im phys. Speicher befinden (Seite → Seitenrahmen).
P: Present/Absent-Bit (1 = Vorhanden im Hauptspeicher / 0 = Fehlt im Hauptspeicher)

- Seitenrahmennummer (physikalischer Speicher)
- Verfügbarkeit (Present/Absent-Bit)
- Schutz (Privileg)
- Verändert (Dirty-Bit)
- Referenziert (Accessed)
- Caching

Beispiel einer Seitentabelle

Mögliche Abbildung von einen Virtuellen Speicher mit 16 Seiten auf einen Arbeitsspeicher mit 8 Seitenrahmen.



Mehrprogrammbetrieb (multiprogramming)

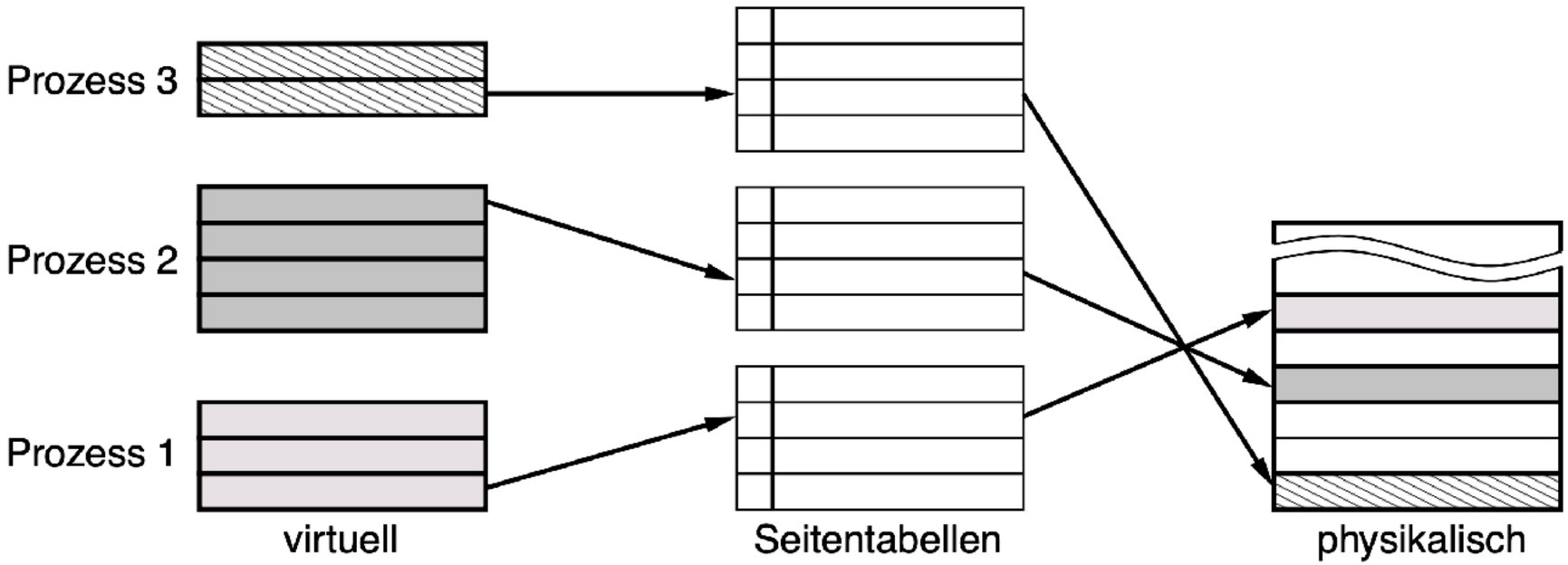
Realisierung von Mehrprogrammsystemen

- Unterteilung der Abarbeitungszeit in Zeitscheiben (timesharing-Systeme)
- Unterteilung des Hauptspeichers in Bereiche (Partitionierung)

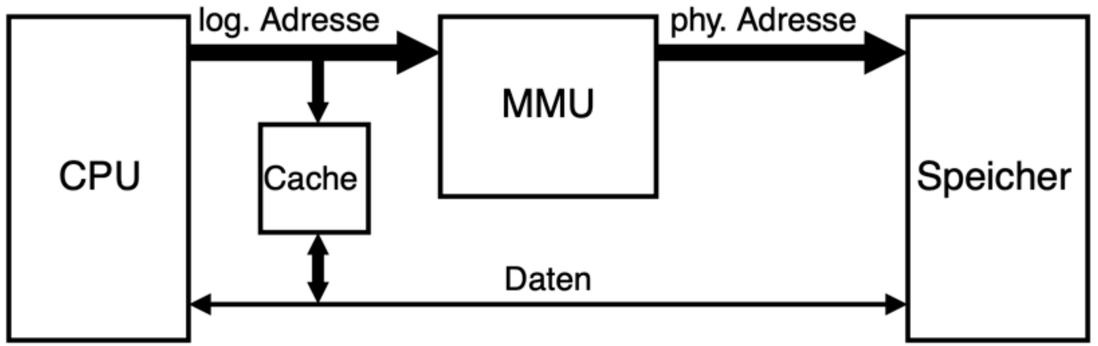
→ Abarbeitung mehrerer Programme (Prozesse) gleichzeitig, parallel (innerhalb einer Zeitscheibe nur je ein Prozess)

- Einbenutzersysteme (single user mode) ein Nutzer mit mehreren Programmen
- Mehrbenutzersysteme (multi user mode) mehrere Nutzer mit vielen Programmen
- zu den Nutzerprozessen parallele Abarbeitung von Betriebssystemprozessen, Treibern, E/A ...

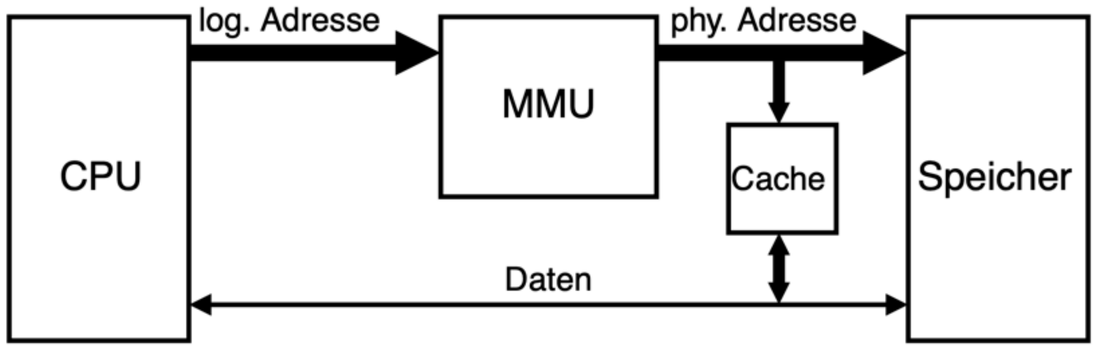
Mehrprogramm- und Benutzerbetrieb



Virtueller Cache



Physikalischer Cache



Swapping

Unter Swapping (Ein-/Auslagerung) wird bei Timesharing-Systemen das Verschieben ganzer Prozesse (Programme) zwischen primären und sekundären Speicher (Hauptspeicher-Festplatte) verstanden.

- Definition eines Swap-Bereiches auf der Festplatte
- Allokation von Swap-Partitionen im Swap-Bereich (analog Hauptspeicherpartitionierung)
- Ein- und Auslagerung von Prozessen als Transfer zwischen Hauptspeicher- und Swap-Partitionen

Ursachen, Notwendigkeit für Swapping:

- es gibt mehr Prozesse als mögliche Speicherpartitionen (zu kleiner Hauptspeicher)
- Prozesse mit langen Wartezeiten (z.B. auf E/A-Geräte)
- inaktive Prozesse

Aufgaben – Cache I

1. Stellen Sie eine moderne Cache-Architektur mit drei Ebenen und Unterteilung der L1-Ebene in Harvard mittels PMS dar.
2. Das Speichersubsystem ist in modernen Rechnen hierarchisch aufgebaut.
   a) In welchen markanten Eigenschaften unterscheiden sich die Komponenten dieser Hierarchie?
   b) Welche typische Eigenschaft des Verlaufs von Speicherzugriffen wird von der Speicherhierarchie zur Leistungssteigerung ausgenutzt? Benennen Sie deren beide Erscheinungsformen und beschreiben Sie diese kurz.

Aufgaben – Cache II

3. Für eine byteorientierte Speicherhierarchie mit 32-Bit-Adressen werde ein vollassoziativer Cache mit einer Kapazität von 16 KiByte angenommen. Seine Cachezeilen sind 64 Byte lang und in Worte von je 4 Byte untergliedert.
   a) Wie wird eine Speicheradresse zum Zweck eines Cache-Lookups untergliedert?
   b) Wie viele Zeilen enthält der Cache?
   c) Welche Kapazität hat der Tag-Speicher (ohne etwaige Statusbits)?
   d) Was muss für einen Cache-Hit neben einem erfolgreichen Tag-Abgleich noch überprüft werden?

Aufgaben – Cache III

4. Wie ändern sich die Parameter, wenn der Cache aus 2. als Direkt-Abgebildeter Cache vorliegt?
5. Ein byteorientiertes Speichersystem mit einem physischen 16-Bit Adressraum enthält einen 2-fach satzassoziativen Cache mit 16 Cachezeilen über je 16 Byte. Die Ersetzungsstrategie ist LRU (Least-recently-used).
   a) Geben Sie die Adressaufteilung an.
   b) Über den leeren Cache laufen nun folgende Speicherzugriffe:

Nr..	R/W	Adresse	Index	Tag	Hit	Ersetzt Nr.
1 .	RD	0xA31B	0x1	0x146	–	–
2 .	RD	0xA39F				
3 .	WR	0xA3D0				
4 .	RD	0xA3D4				
5 .	RD	0x079C				
6 .	RD	0xA315				
7 .	WR	0xA3D7				

Aufgaben – Cache IV

6) In einem Rechner mit einem Hauptspeicher von 4 GByte wird eine Programmschleife durchlaufen, welche auf dem Adressbereich von `0xA010 0000` bis `0xA010 01FF` liegt. Die Befehle liegen im 32-Bit Format vor und der Prozessor besitzt als Befehlscache einen Direkt-Abgebildeten Cache, bei dem jede Cacheline aus 4 Byte besteht und der Index 6 Bit breit ist.
   a) Welche Adressbits bilden den Tag?
   b) Wieviel Zeilen hat der Cache?
   c) Wie groß ist der Cache?
   d) Wie viele Bytes umfasst die Programmschleife?
   e) Wie hoch ist die Trefferquote des Cache?


END