Rechnerorganisation – Kapitel 8

Prof. Sarah Harris
Fachgebiet Eingebettete Systeme und ihre Anwendungen (ESA)
Fachbereich Informatik

TECHNISCHE UNIVERSITÄT DARMSTADT

SoSe 2016

Kapitel 8: Themen

TECHNISCHE UNIVERSITÄT DARMSTADT

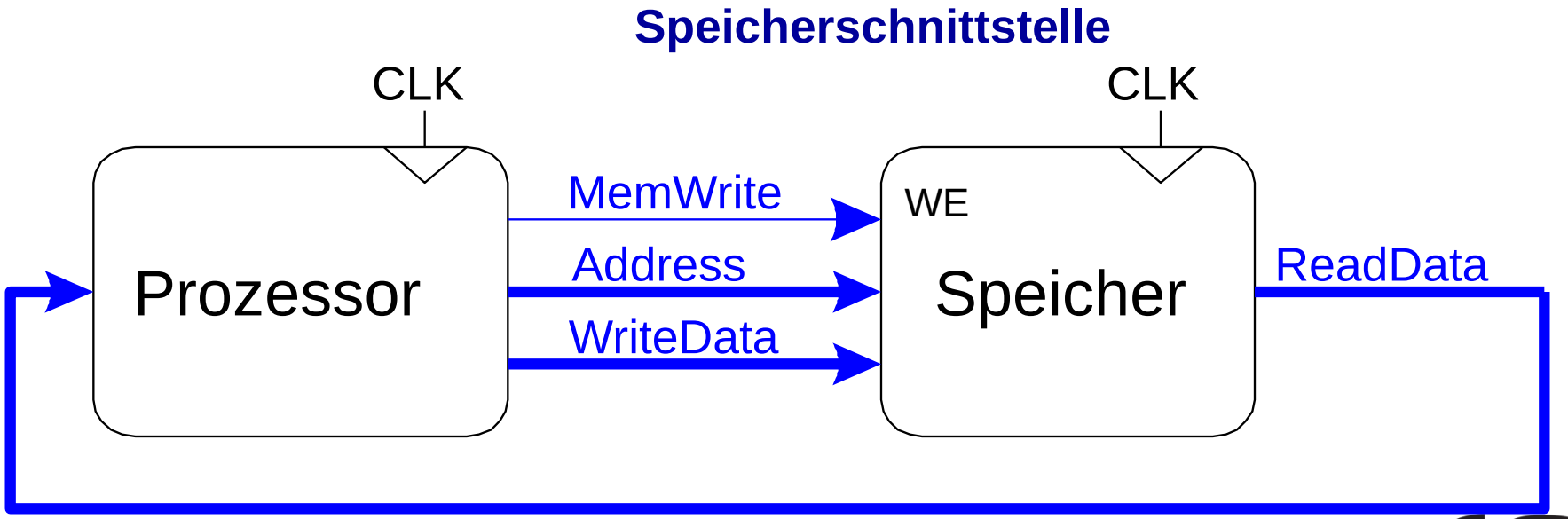
- Einleitung
- Leistungsvergleich von Speichersystemen
- Caches
- Virtueller Speicher
- Speichereinblendung von Ein-/Ausgabegeräten
- Zusammenfassung

Einleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

Rechenleistung hängt ab von:

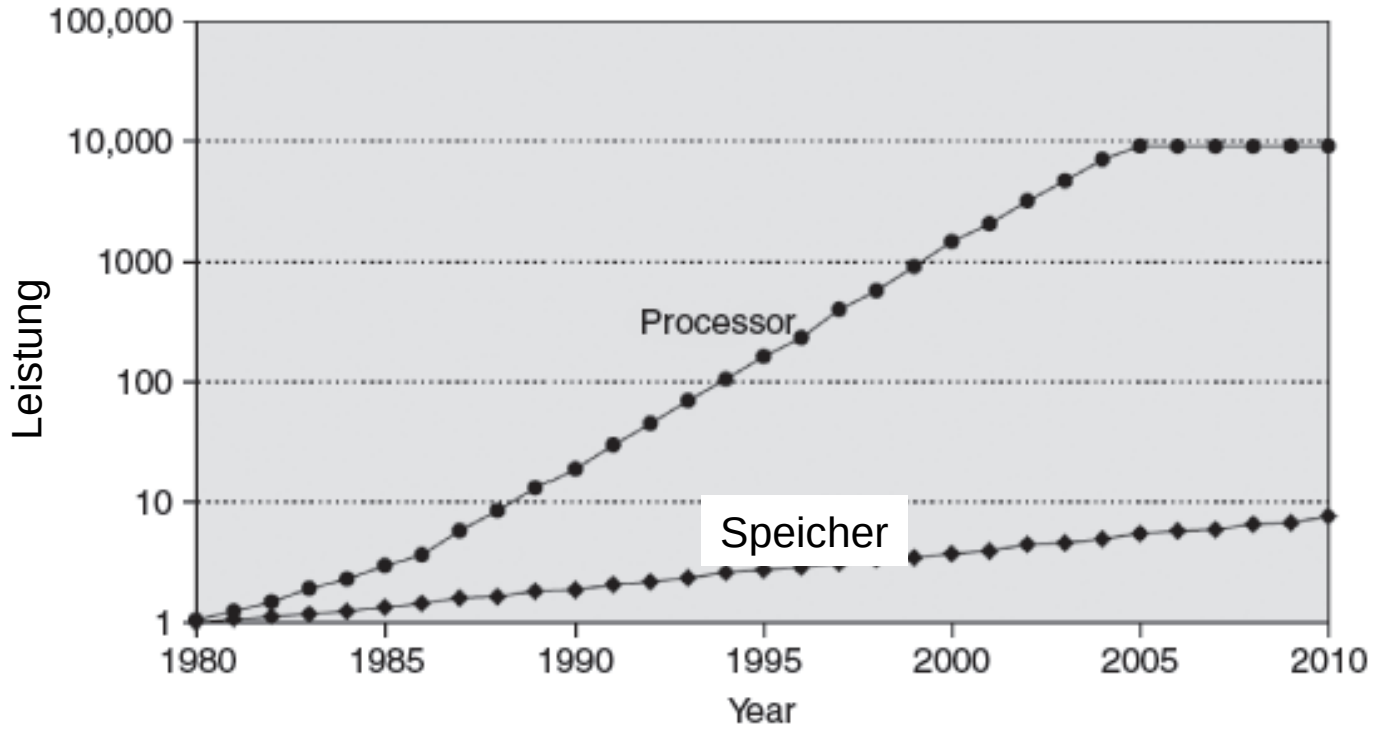
- Prozessorleistung
- Leistung des Speichersystems



Embedded Systems & Applications

Einleitung

- Annahme bisher in der Vorlesung: Speicherzugriffe dauern 1 Takt
- Ist aber seit den 1980‘er Jahren nicht mehr wahr



Herausforderungen beim Entwurf von Speichersystemen

TECHNISCHE UNIVERSITÄT DARMSTADT

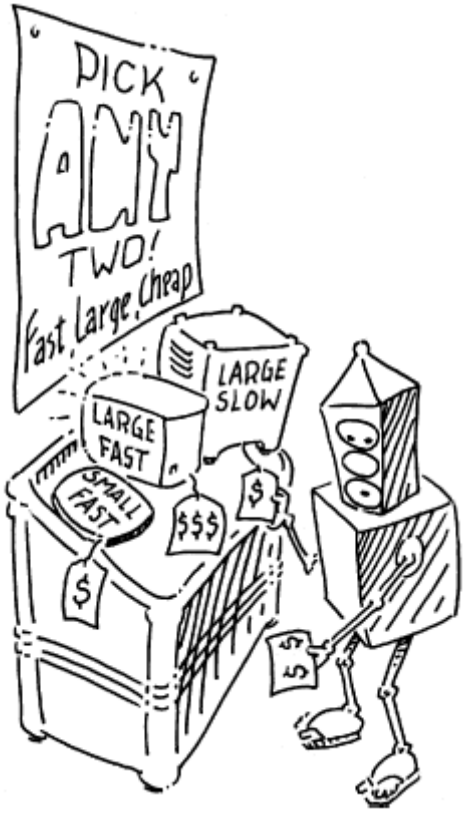
Speichersystem soll so schnell sein wie Prozessor

- Zumindest dem Anschein nach

Idealer Speicher:

- Schnell
- Billig
- Hohes Fassungsvermögen

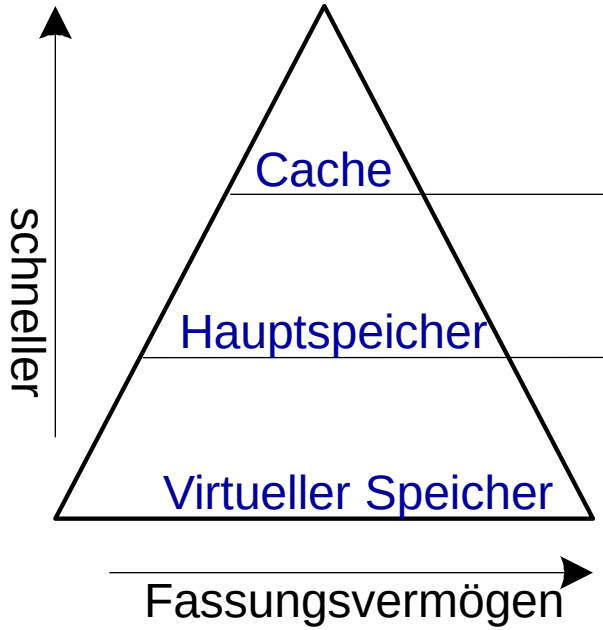
Praktisch: Nur zwei von drei Eigenschaften realisierbar!



Verwende Hierarchie von Speichern

Embedded Systems & Applications

Speicherhierarchie: Beispiele für Ebenen



Technologie	Kosten/GB	Zugriffszeit
SRAM	~ $10,000	~ 1 ns
DRAM	~ $10	~ 10-50 ns
Festplatte	~ $0.1	~ 10,000,000 ns
SSD	~ $1	~ 100,000 ns

Lokalität

TECHNISCHE UNIVERSITÄT DARMSTADT

Nutze Lokalität zur Beschleunigung von Speicherzugriffen aus

Zeitliche Lokalität:

- Ein gerade benutztes Datum wird wahrscheinlich bald wieder gebraucht
- **Ausnutzung:** gerade benutzte Daten in höheren Ebenen der Speicherhierarchie halten

Räumliche Lokalität:

- Um ein benutztes Datum herum liegende Daten werden wahrscheinlich auch bald gebraucht
- **Ausnutzung:** Beim Zugriff auf ein Datum auch benachbarte Daten in höhere Ebenen der Speicherhierarchie bringen

Leistung eines Speichersystems

Treffer (*hit*): Datum wird auf dieser Ebene der Speicherhierachie gefunden

Verfehlt (*miss*): nicht gefunden (suche auf tieferer Ebene)

Hit-Rate = # hits / # Speicherzugriffe
= 1 – Miss Rate

Miss-Rate (*MR*) = # misses / # Speicherzugriffe
= 1 – Hit Rate

Durchschnittliche Speicherzugriffszeit (*average memory access time, AMAT*): Durchschnittliche Zeit, die der Prozessor braucht, um auf ein Datum zuzugreifen

AMAT = $t_{\text{cache}} + MR_{\text{cache}} (t_{MM} + MR_{MM} t_{VM})$

MM = main memory, Hauptspeicher
VM = virtual memory, Virtueller Speicher

Beispiel 1: Leistung eines Speichersystems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Programm hat 2000 Load- und Store-Befehle
- 1250 der Daten werden im Cache vorgefunden
- Der Rest kommt aus anderen Ebenen der Speicherhierarchie

- **Was sind die Hit- und Miss-Rates des Caches?**

Embedded Systems & Applications

Beispiel 2: Leistung eines Speichersystems

TECHNISCHE UNIVERSITÄT DARMSTADT

Annahme: Prozessor hat zwei Hierarchieebenen:

- Cache
- Hauptspeicher (Main Memory)

t_{cache} = 1 Takt

t_{MM} = 100 Takte

Wie lang ist die AMAT des Programmes aus Beispiel 1?

AMAT =

Embedded Systems & Applications

Gene Amdahl, 1922 -

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Amdahls Gesetz:** Die Optimierung eines Subsystems bringt nur dann etwas, wenn das Subsystem tatsächlich großen Einfluss auf die Gesamtrechenleistung hat
- Gründete drei Firmen, darunter auch die Amdahl Corporation in 1970
  - IBM-kompatible Großrechner
    - I.d.R. schneller und/oder billiger

Embedded Systems & Applications

Cache

TECHNISCHE UNIVERSITÄT DARMSTADT

„Verstecktes Lager“

- Höchste Ebene der Speicherhierarchie
- Schnell (oft Zugriffszeit ~ 1 Takt)
- Stellt dem Prozessor idealerweise die meisten benötigten Daten zur Verfügung
- Speichert (i.d.R.) die zuletzt benutzten Daten

Embedded Systems & Applications

Aufbau von Caches: Entwurfsentscheidungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Welche Daten werden im Cache gehalten?
- Wie werden die Daten gefunden?
- Wie werden Daten ersetzt?

- Schwerpunkt hier auf Loads
- Stores werden aber ähnlich gehandhabt

Embedded Systems & Applications

Welche Daten werden im Cache gehalten?

Aufbau von Caches: Entwurfsentscheidungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Welche Daten werden im Cache gehalten?
- Wie werden die Daten gefunden?
- Wie werden Daten ersetzt?

Embedded Systems & Applications

Welche Daten werden im Cache gehalten?

TECHNISCHE UNIVERSITÄT DARMSTADT

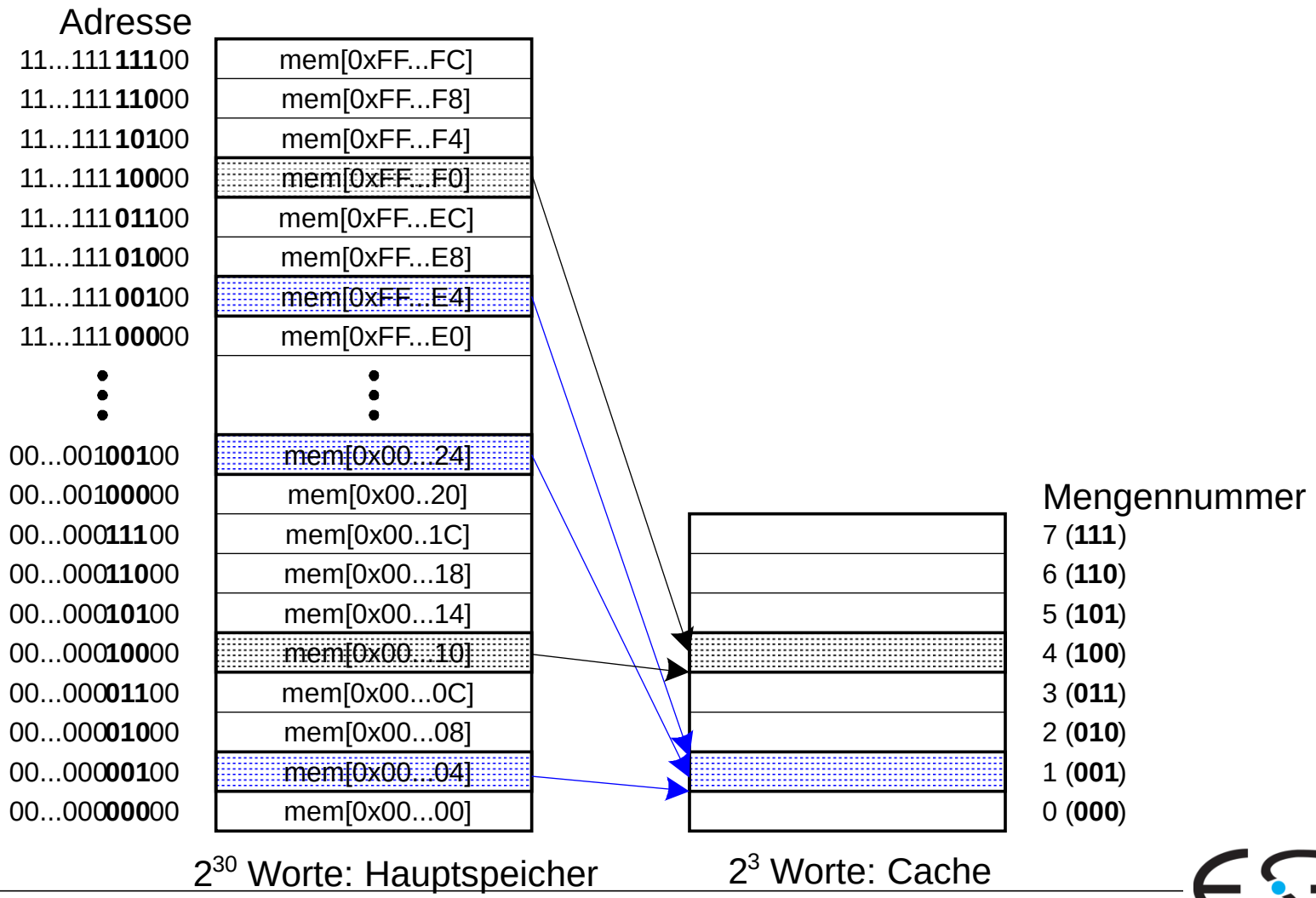
Annahme:

Wir speichern nur 8 Wörter im Cache

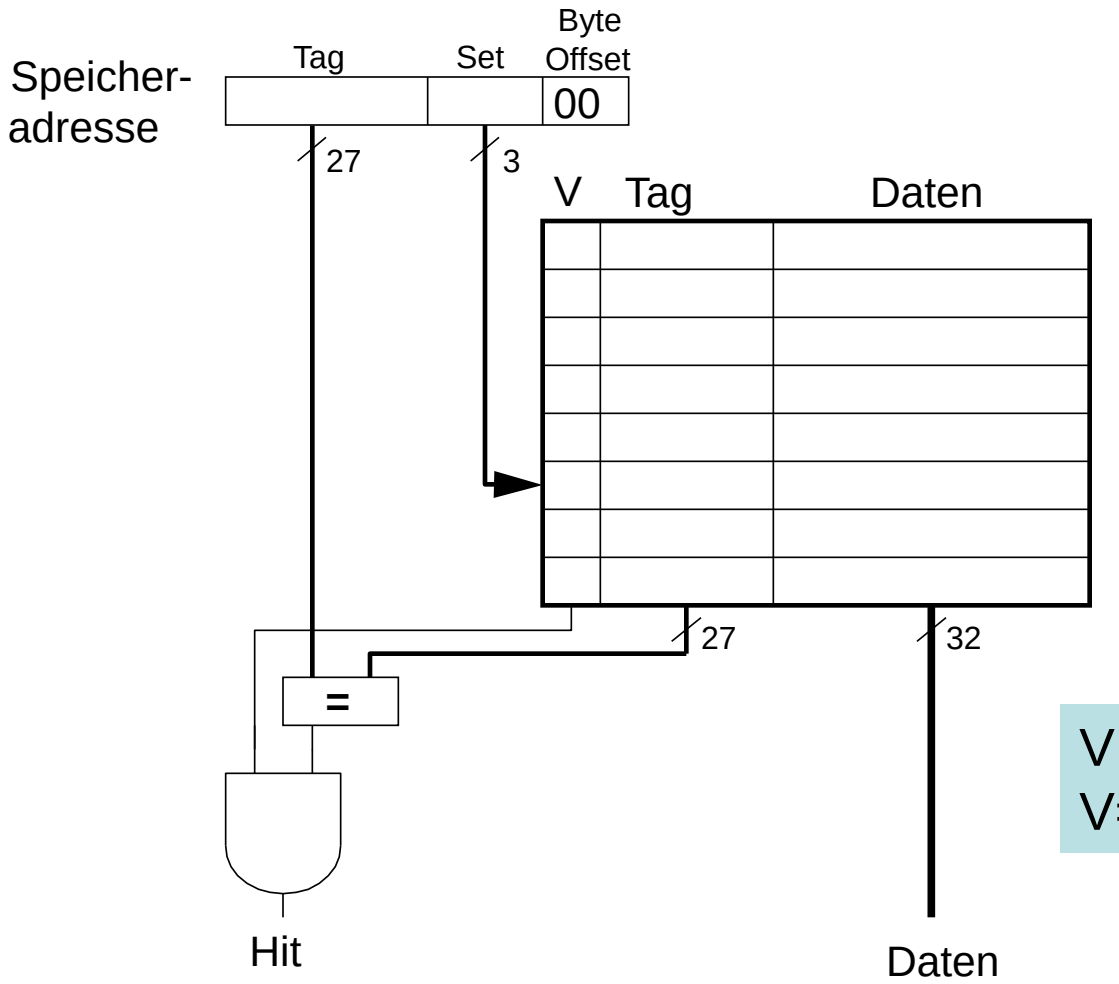
(**Kapazität** = 8 Wörter oder 32 Bytes)

Embedded Systems & Applications

Direktabgebildeter Cache

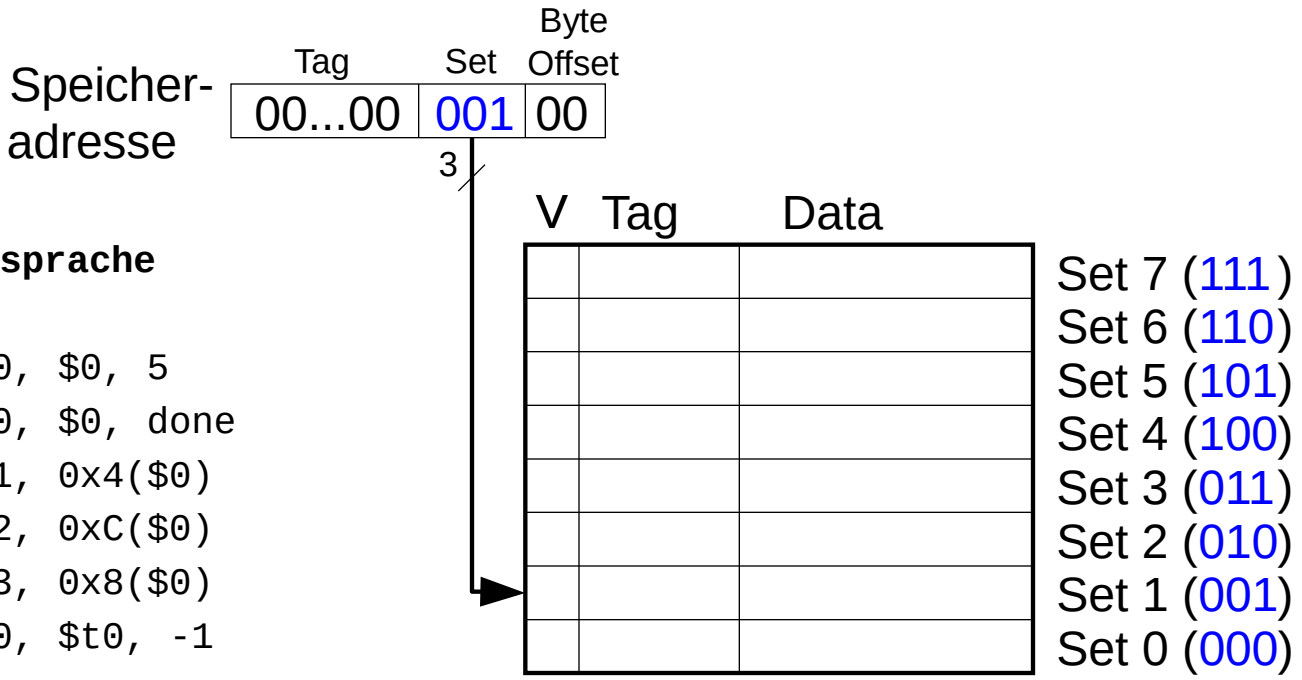


Direktabgebildeter Cache: Hardware



V steht für „Valid“
V=1 -> gültiger Eintrag

Leistung eines direktabgebildeten Caches

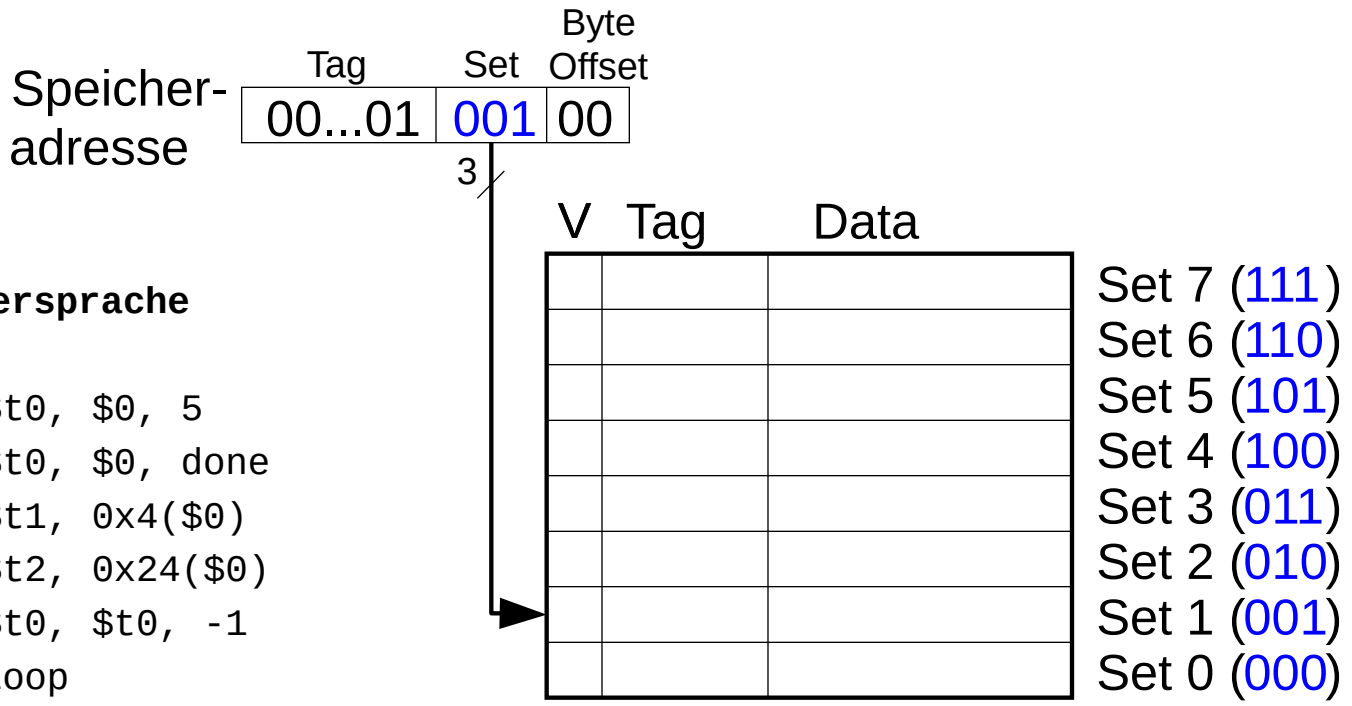



```
# MIPS Assemblersprache

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate =

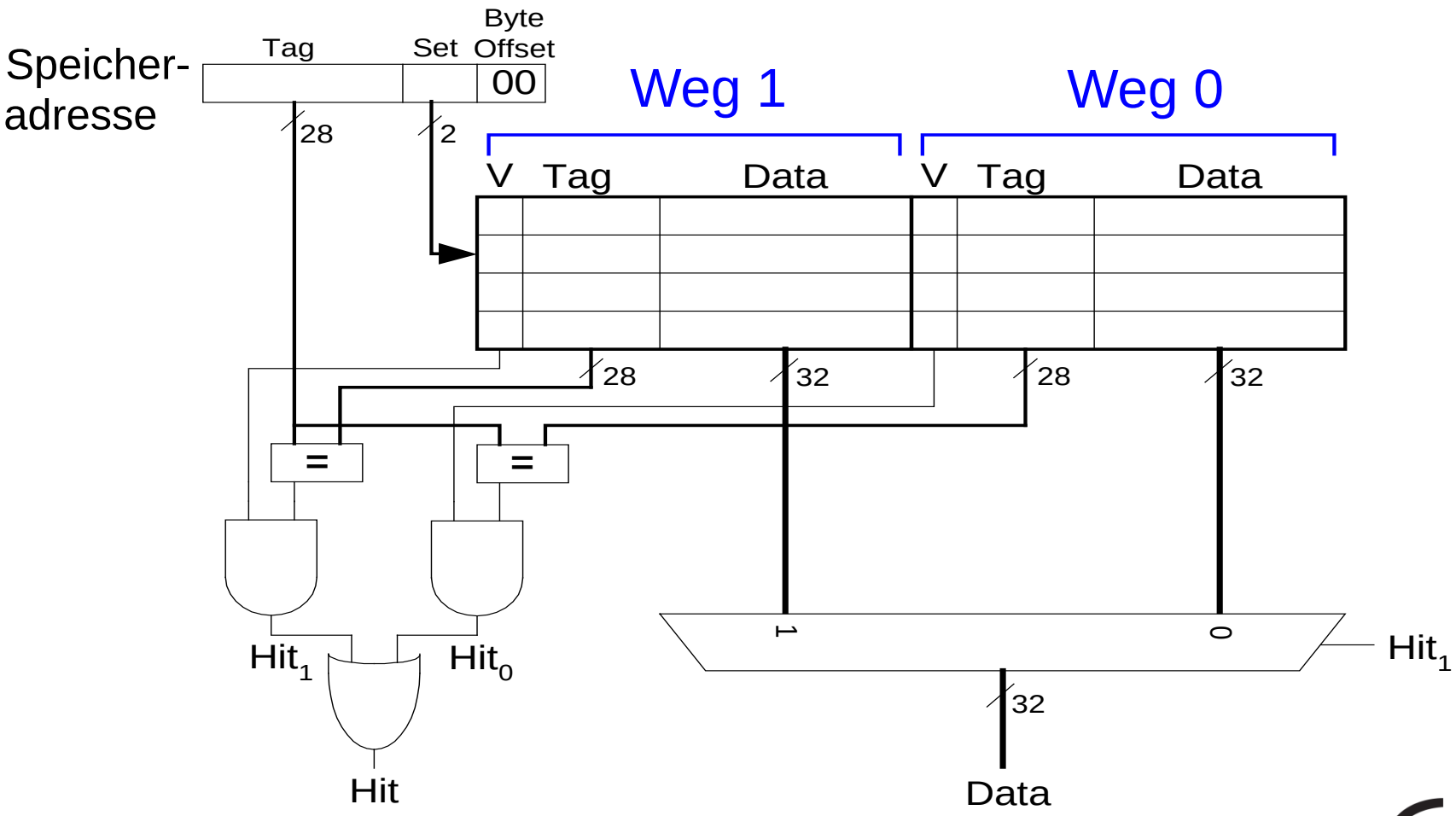
Konflikte bei direktabgebildeten Caches




```
# MIPS Assemblersprache

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```

N-Wege Assoziativer Cache

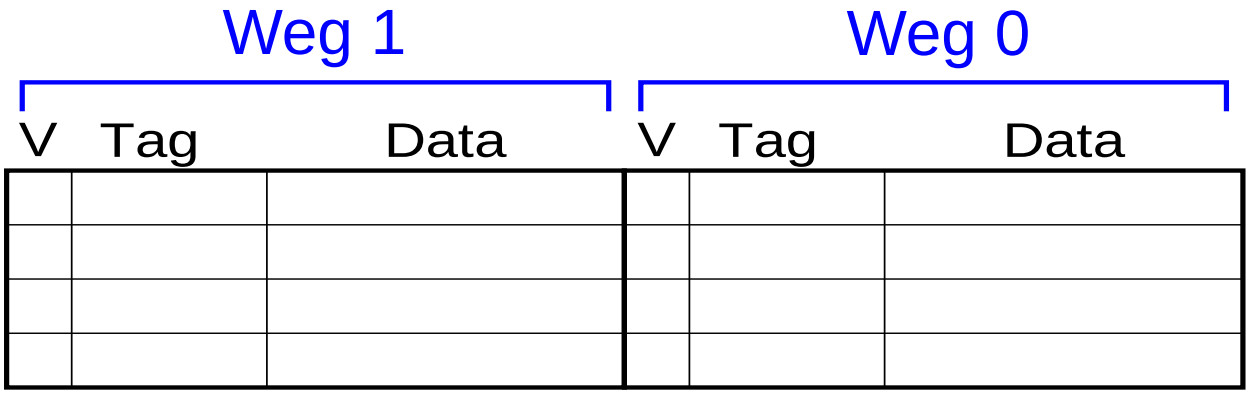


Leistung eines *N*-Wege assoziativen Caches

TECHNISCHE UNIVERSITÄT DARMSTADT

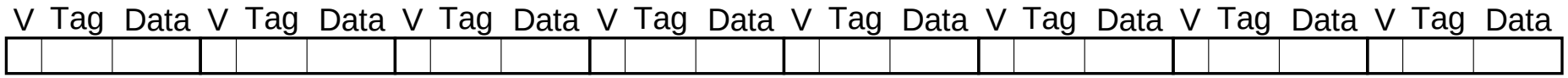
```
# MIPS Assemblersprache

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0x24($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Embedded Systems & Applications

Vollassoziativer Cache



- **Keine Konflikt-Misses**
- **Kostet viel Hardware**

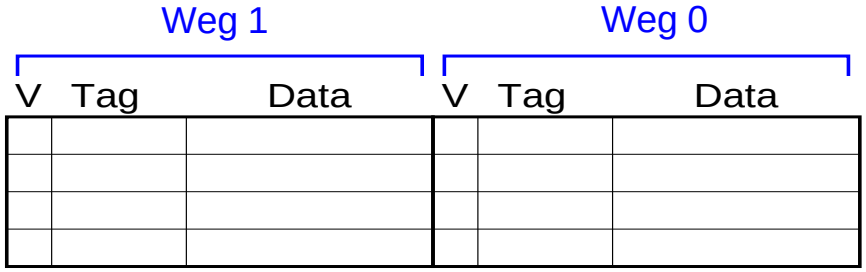
Wie werden Daten gefunden?

TECHNISCHE UNIVERSITÄT DARMSTADT

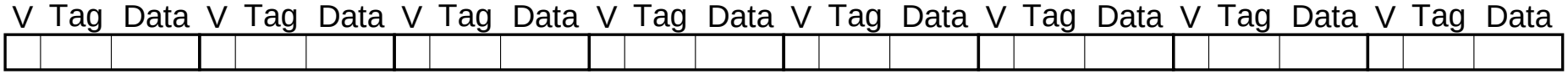
Caches werden klassifiziert nach Anzahl von Blöcken in einer Menge:

- **Direktabgebildet (*direct mapped*):** Ein Block pro Menge

- ***N*-Wege Mengenassoziativ (*N-way set associative*):** *N* Blöcke pro Menge



- **Vollassoziativ (*fully associative*):** Alle Cache Blöcke in einer Menge



Embedded Systems & Applications

Wie werden Daten gefunden?

TECHNISCHE UNIVERSITÄT DARMSTADT

Caches werden klassifiziert nach Anzahl von Blöcken in einer Menge:

- **Direktabgebildet (*direct mapped*):** Ein Block pro Menge
- ***N*-Wege Mengenassoziativ (*N-way set associative*):** *N* Blöcke pro Menge
- **Vollassoziativ (*fully associative*):** Alle Cache Blöcke in einer Menge

Cache ist organisiert als *S* Mengen (*sets*)

- Jede Speicheradresse wird genau auf eine Menge abgebildet

Embedded Systems & Applications

Caches: Begriffe

TECHNISCHE UNIVERSITÄT DARMSTADT

Kapazität (*capacity, C*):

- Anzahl der im Cache speicherbaren Bytes

Blockgröße (*block size, b*):

- Anzahl der auf einen Satz in den Cache geladenen Bytes

Blockanzahl (*B = C/b*):

- Anzahl von Blöcken im Cache: *B = C/b*

Assoziativitätsgrad (*degree of associativity, N*):

- Anzahl von Blöcken in einer Assoziativitätsmenge (kurz: *Menge)*

Anzahl von Mengen (*S = B/N*):

- Jede Speicheradresse wird auf genau eine Menge abgebildet

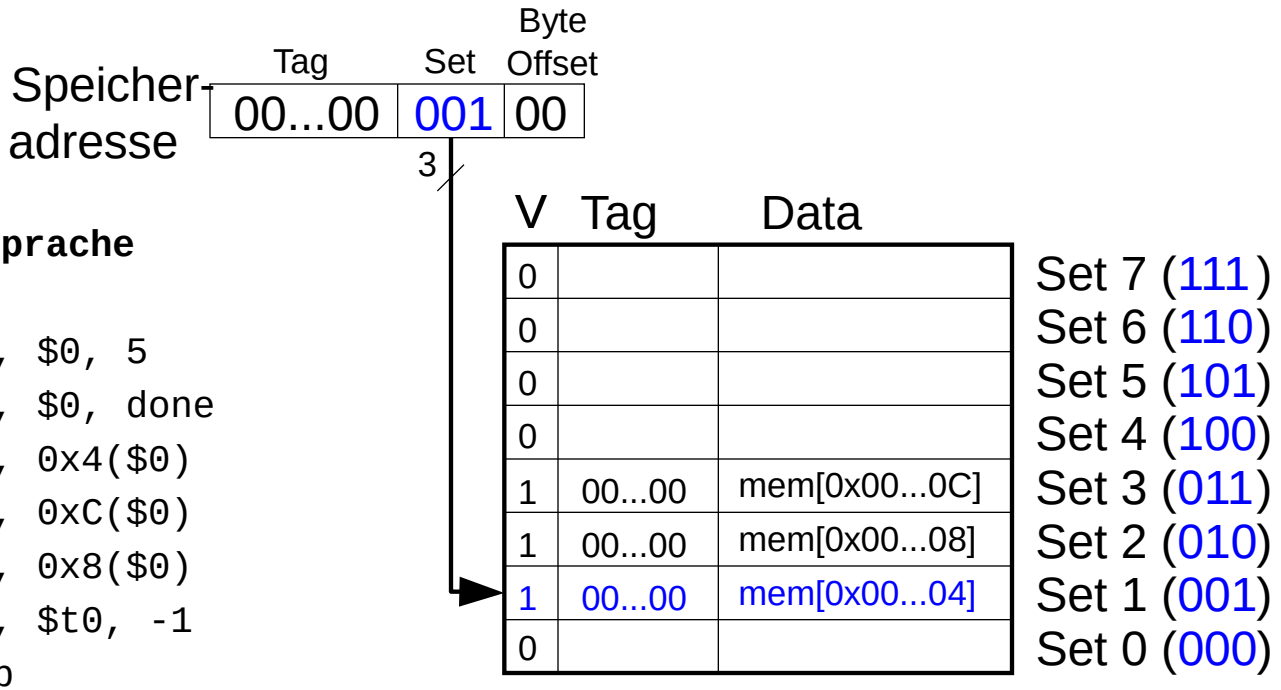
Welche Daten werden im Cache gehalten?

TECHNISCHE UNIVERSITÄT DARMSTADT

Basiere Vorhersagen auf bisherigem Verhalten

- **Zeitliche Lokalität**: kopiere gerade benutzte Daten in den Cache. Bei nächster Verwendung werden die Daten im Cache gefunden (*Cache Hit*).
- **Räumliche Lokalität**: kopiere benachbarte Daten auch in den Cache
  - Blockgröße: Anzahl von Bytes, die immer zusammen in den Cache kopiert werden

Wiederholung: Leistung eines direktabgebildeten Caches




```
# MIPS Assemblersprache

        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```

Miss Rate = 3 / 15

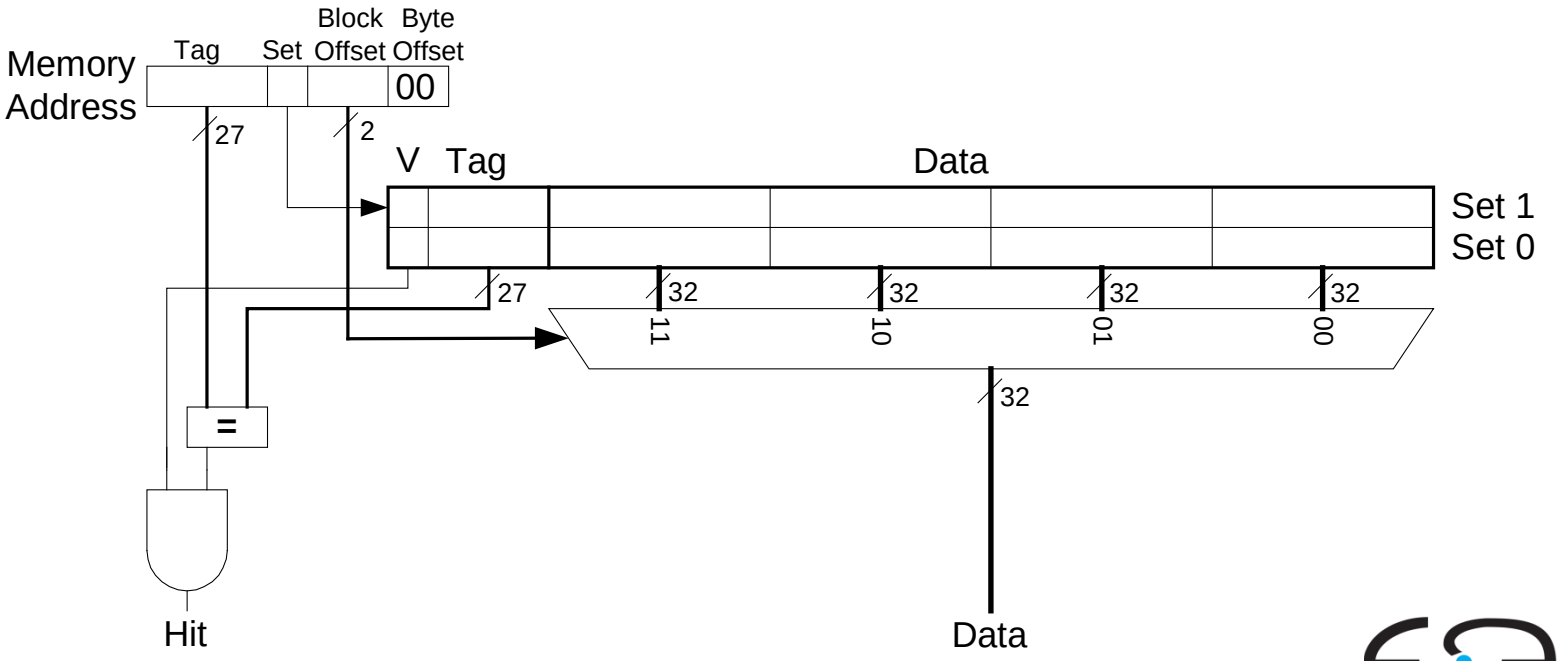
= 20%

Räumliche Lokalität

TECHNISCHE UNIVERSITÄT DARMSTADT

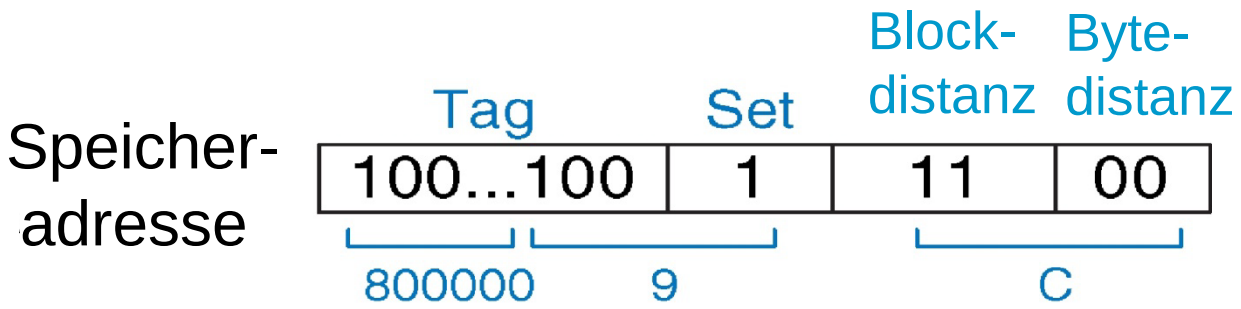
Blockgröße erhöhen:

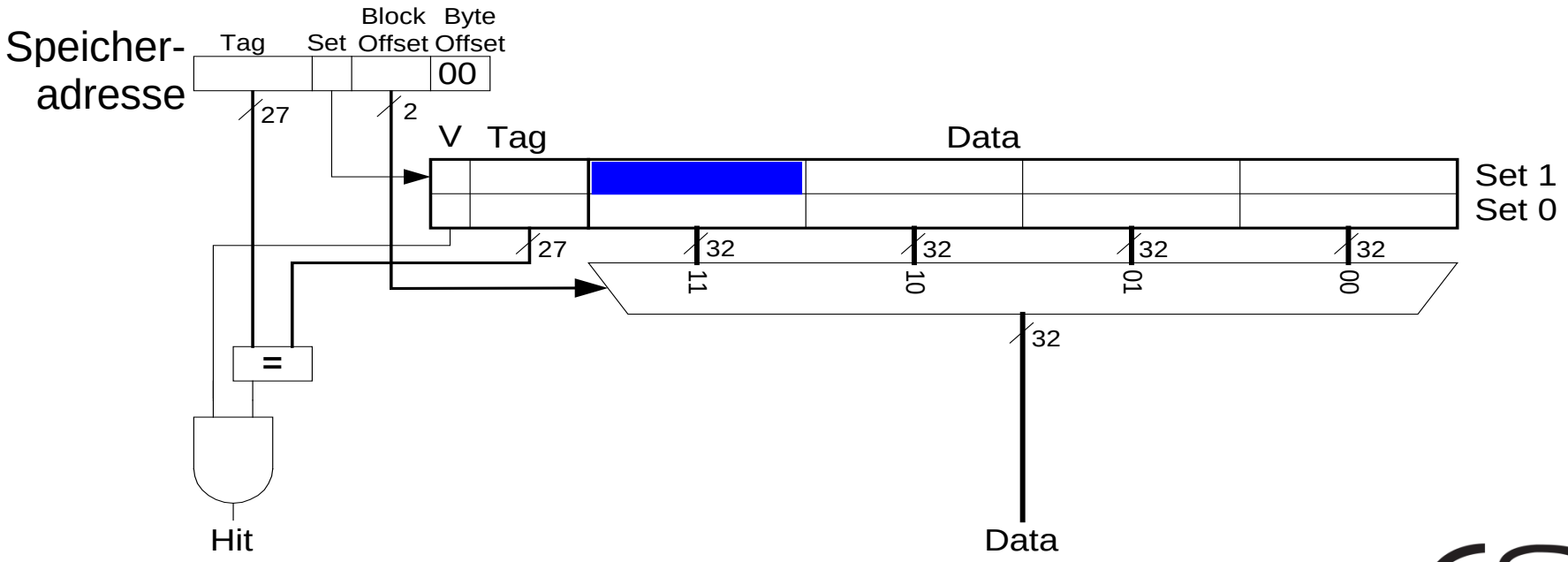
- Blockgröße, ***b* = 4** Wörterss
- ***C*** = 8 Wörter
- **Direktabgebildeter** Cache (1 Block pro Menge)
- Blockanzahl, ***B*** = *C*/*b* = 8/4 = 2



Embedded Systems & Applications

Cache mit erhöhter Blockgröße

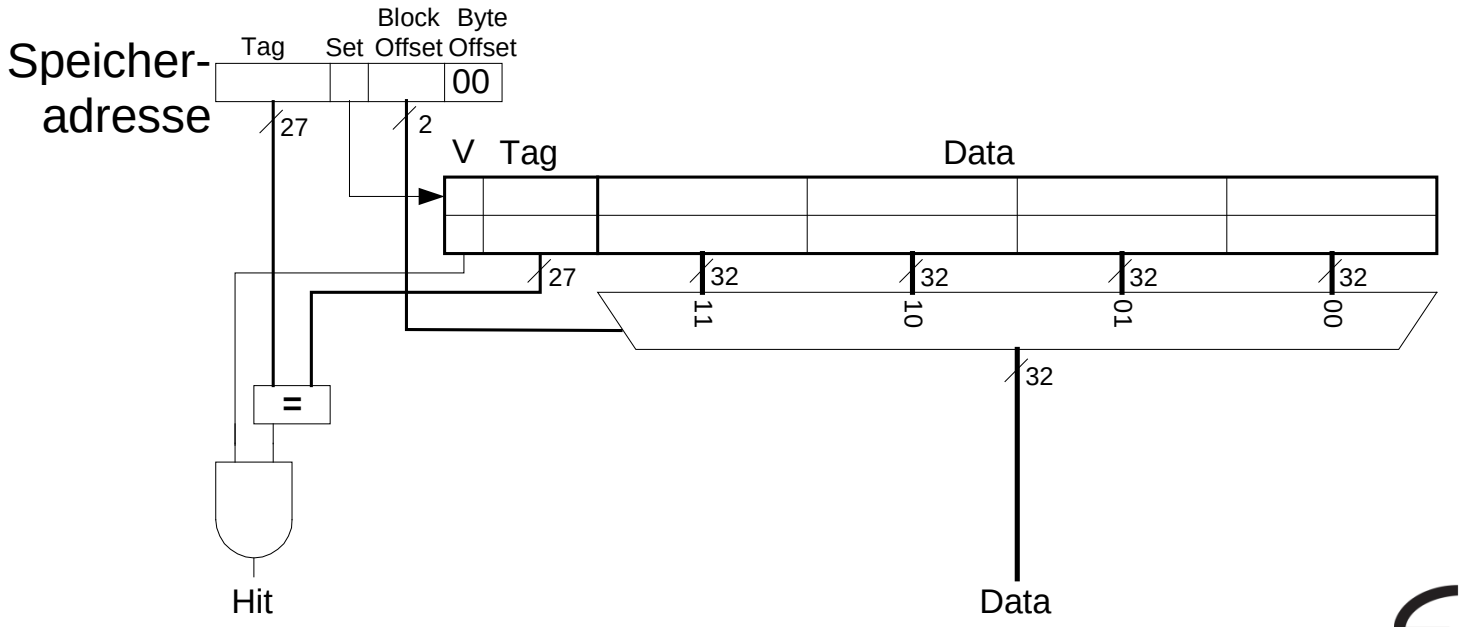




Leistung eines direktabgebildten Caches mit erhöhter Blockgröße

TECHNISCHE UNIVERSITÄT DARMSTADT

```
        addi $t0, $0, 5
loop:   beq  $t0, $0, done
        lw   $t1, 0x4($0)
        lw   $t2, 0xC($0)
        lw   $t3, 0x8($0)
        addi $t0, $t0, -1
        j    loop
done:
```



Embedded Systems & Applications

Zusammenfassung: Aufbau von Caches

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kapazität: ***C***
- Blockgröße: ***b***
- Blockanzahl: ***B*** = *C*/*b*
- *Assoziativitätsgrad*: ***N***
- Anzahl von Mengen (Sets): ***S*** = *B*/*N*

Name	Assoziativitätsgrad (*N*)	Anzahl von Mengen (*S* = *B*/*N*)
Direktabgebildete	1	*B*
N-Wege Mengenassoziativ	1 < *N* < *B*	*B* / *N*
Vollassoziativ	*B*	1

Embedded Systems & Applications

Kapazitäts-Misses

TECHNISCHE UNIVERSITÄT DARMSTADT

Ein Cache könnte zu klein sein, alle benötigten Daten auf einmal zu halten

- Wenn der Cache schon voll ist, müssen Daten entfernt werden
  - z.B. Datum *Y* könnte entfernt werden bei Zugriff auf neues Datum *X*
- Ein ***Kapazitäts-Miss*** tritt auf, wenn das Programm dann wieder auf Datum *Y* zugreift

In assoziativen Caches muss entschieden werden, welches Datum entfernt werden soll

- **Least Recently Used (LRU) Ersatz**: den Block entfernen, den wir am längsten nicht benutzt haben

Embedded Systems & Applications

Ersetzen mit LRU-Schema

TECHNISCHE UNIVERSITÄT DARMSTADT

```
# MIPS Assemblersprache

lw $t0, 0x04($0)
lw $t1, 0x24($0)
lw $t2, 0x54($0)
```

(a)

V	U	Tag	Data	V	Tag	Data	Mengennummer
							3 (**11**)
							2 (**10**)
							1 (**01**)
							0 (**00**)

(b)

V	U	Tag	Data	V	Tag	Data	Mengennummer
							3 (**11**)
							2 (**10**)
							1 (**01**)
							0 (**00**)

Embedded Systems & Applications

Arten von Cache Misses

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Unvermeidbare (Compulsory):** beim ersten Zugriff auf Daten
- **Kapazität:** der Cache ist zu klein, um alle benötigten Daten zu halten
- **Konflikt:** mehrere Daten müssten sich am gleichen Ort in Cache befinden

Miss-Kosten (Miss penalty): die benötigte Zeit, um die Daten aus unteren Ebenen der Speicher-Hierarchie zu holen

Zusammenfassung von Caches

TECHNISCHE UNIVERSITÄT DARMSTADT

Welche Daten werden im Cache gehalten?

- Kürzlich zuvor gebrauchte Daten (zeitliche Lokalität)
- Benachbart liegende Daten (räumliche Lokalität, Verwendung von größeren Blöcken)

Wie werden die Daten gefunden?

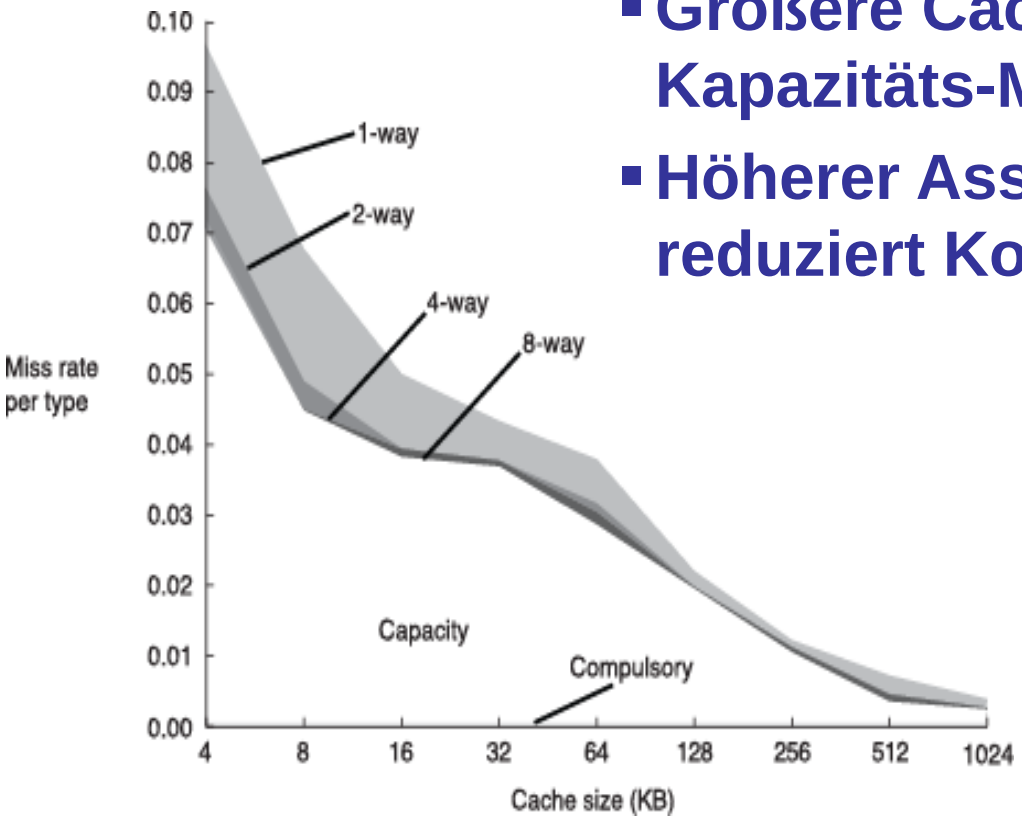
- Datenadresse bestimmt:
  - Menge
  - Wort innerhalb eines Blocks
- In assoziativen Caches, Daten können in einem von mehreren Wegen sein

Wie werden Daten ersetzt?

- Der älteste nicht gebrauchte Weg einer Menge wird ersetzt

Embedded Systems & Applications

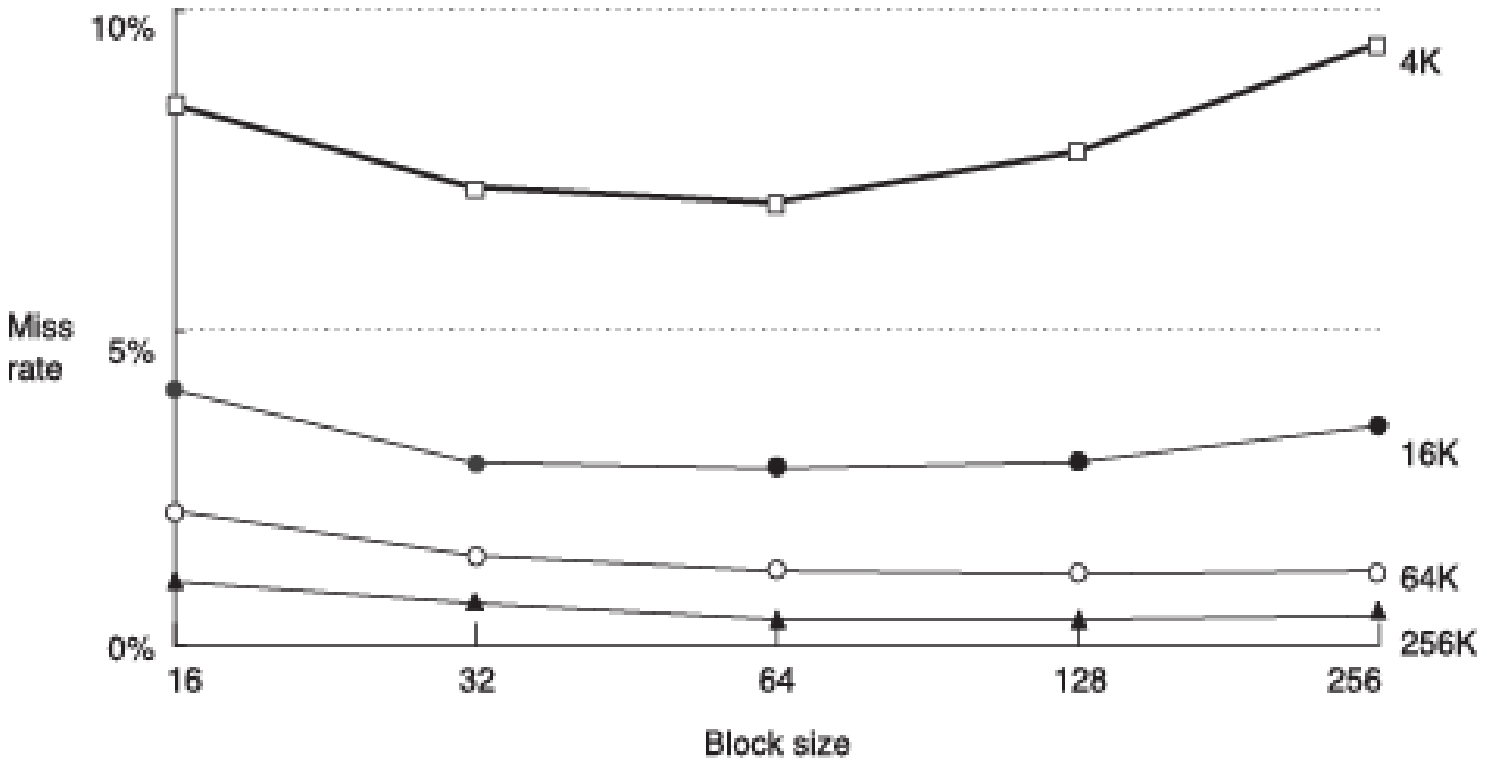
Miss Rate von Daten



- **Größere Caches reduzieren Kapazitäts-Misses**
- **Höherer Assoziativitätsgrad reduziert Konflikt-Misses**

adaptiert vom Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*

Miss Rate von Daten



- **Größere Blöcke reduzieren unvermeidbare (compulsory) Misses**
- **Größere Blöcke steigern Konflikt-Misses**

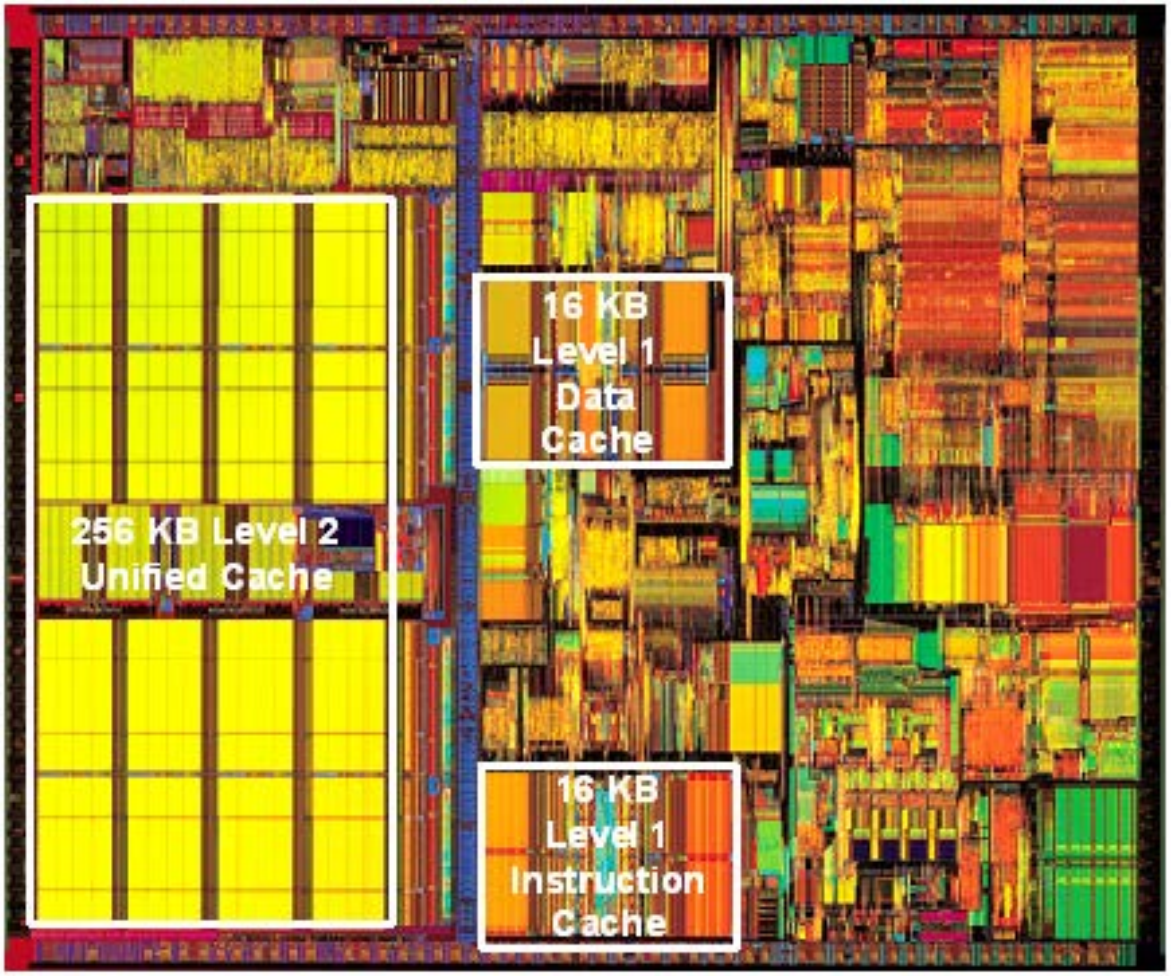
Multi-Level Caches

TECHNISCHE UNIVERSITÄT DARMSTADT

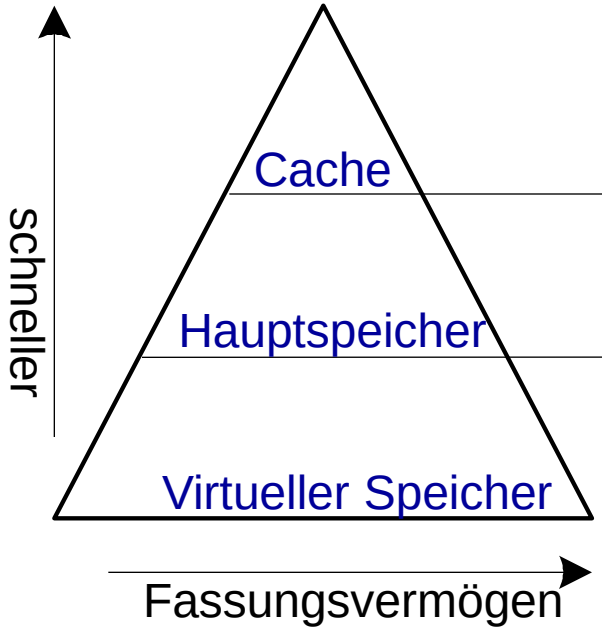
Größere Caches erreichen niedrigere Miss Rates, aber haben längere Zugriffszeiten

- Wir können die Cachehierarchie erweitern:
  - Level 1 (L1): klein aber schnell
    - z.B. 16 KB, 1 Takt
  - Level 2 (L2): größer aber langsamer
    - z.B. 256 KB, 2-6 Takte
  - ...wären auch weitere Ebenen möglich
    - Heute: bis zu L4 im praktischen Einsatz

Intel Pentium III Chip



Speicherhierarchie



Technologie	Kosten/GB	Zugriffszeit
SRAM	~ $10,000	~ 1 ns
DRAM	~ $10	~ 10-50 ns
Festplatte	~ $0.1	~ 10,000,000 ns
SSD	~ $1	~ 100,000 ns

- **Physikalischer Speicher:** DRAM (Hauptspeicher)
- **Virtueller Speicher:** Festplatte
  - Langsam, gross, günstig

Virtueller Speicher: Festplatte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bietet die Illusion, dass der Speicher größer ist
  - … ohne die Kosten von DRAM
- Hauptspeicher (DRAM) wirkt als *Cache* für die Festplatte

Embedded Systems & Applications

Die Festplatte

TECHNISCHE UNIVERSITÄT DARMSTADT

Magnet-platten

Lese/Schreib-zeiger

Es dauert Millisekunden um den Zeiger zu bewegen

Embedded Systems & Applications

Virtueller Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jedes Programm benutzt **Virtuelle Adressen**
- Alle Daten stehen im virtuellen Speicher
- Hauptspeicher (physikalischer Speicher) wirkt als Cache für virtuellen Speicher
  - Eine Untermenge der Daten des virtuellen Speichers steht im Hauptspeicher
  - CPU versucht, die Daten erstmals im Hauptspeicher zu finden
  - Wenn dort nicht gefunden, werden die Daten vom virtuellen Speicher (Festplatte) geholt

Vergleich der Terminologie: Cache / Virtueller Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

Hauptspeicher (physikalischer Speicher) wirkt als Cache für virtuellen Speicher

Cache	Virtueller Speicher
Block	Seite (Page)
Blockgröße	Seitengröße
Blockdistanz (Block Offset)	Seitendistanz (Page Offset)
Miss	Seitenfehler (Page Miss)
Tag	Virtuelle Seitennummer

Embedded Systems & Applications

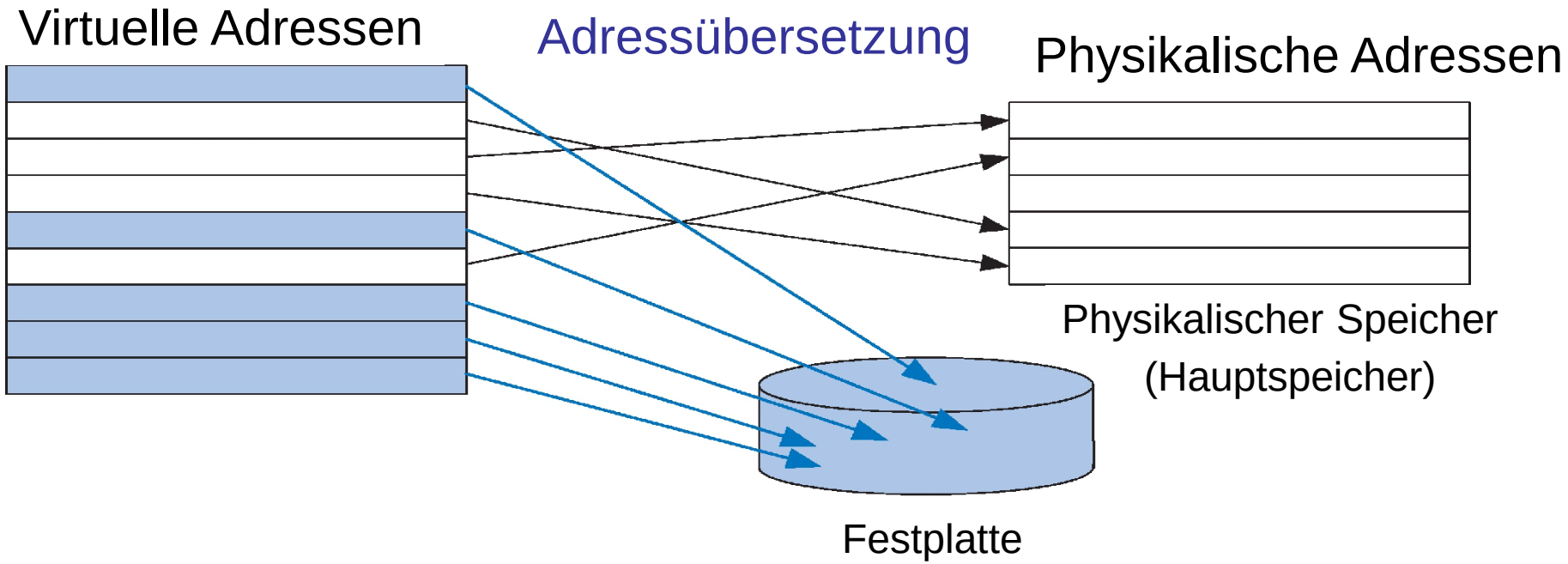
Virtueller Speicher: Begriffe

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Seitengröße:** Anzahl der Bytes, die auf ein Mal von Festplatte zum Hauptspeicher geliefert wird
- **Adressübersetzung:** die Hauptspeicheradresse (physikalische Adresse, PA) eines Datums wird aus der virtuellen Adresse (VA) bestimmt
- **Seitentabelle (Page Table):** Tabelle für die Adressübersetzung

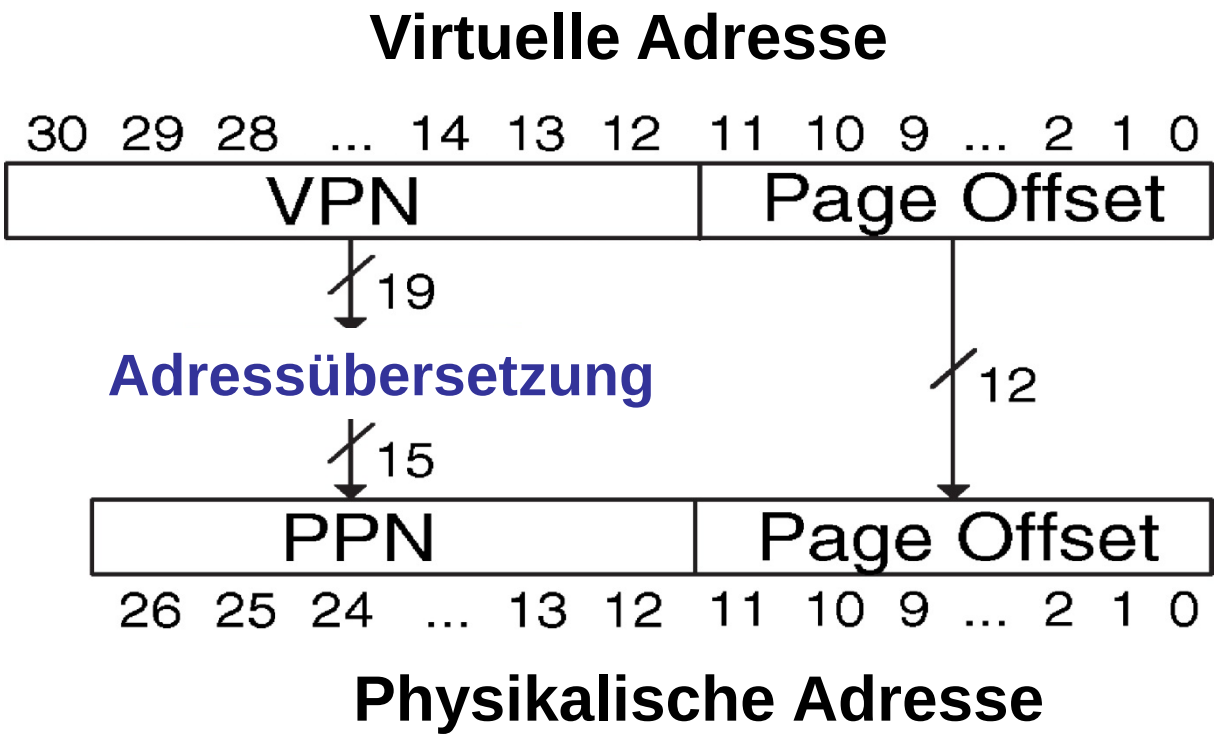
Embedded Systems & Applications

Virtuelle und Physikalische Adressen



- Die meisten Zugriffe werden **im Hauptspeicher** (i.d.R. GBs an physikalischem Speicher) gefunden werden
- Aber Programme haben die große Kapazität der Festplatte (i.d.R. TBs) als virtuellem Speicher zur Verfügung

Adressübersetzung



Beispiel: Virtueller Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

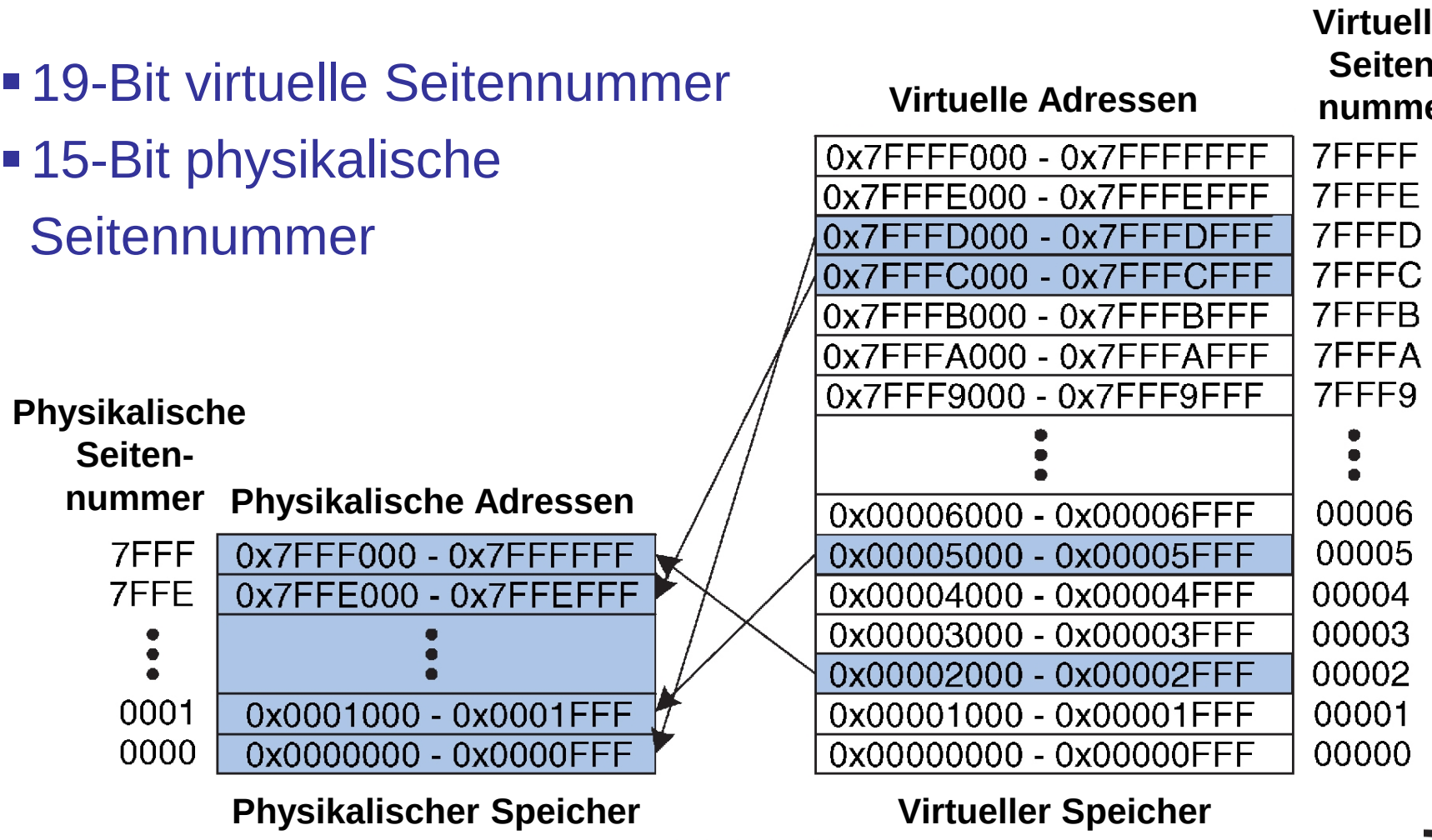
- **System:**
  - Größe des virtuellen Speichers: 2 GB = 2^{31} Bytes
  - Größe des physikalischen Speichers: 128 MB = 2^{27} Bytes
  - Seitengröße (Page Size): 4 KB = 2^{12} Bytes
- **Organisation:**
  - Virtuelle Adresse:
  - Physikalische Adresse:
  - Seitendistanz (Page Offset):
  - # virtuelle Seiten =
  - # physikalische Seiten =

Embedded Systems & Applications

Beispiel: Virtueller Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

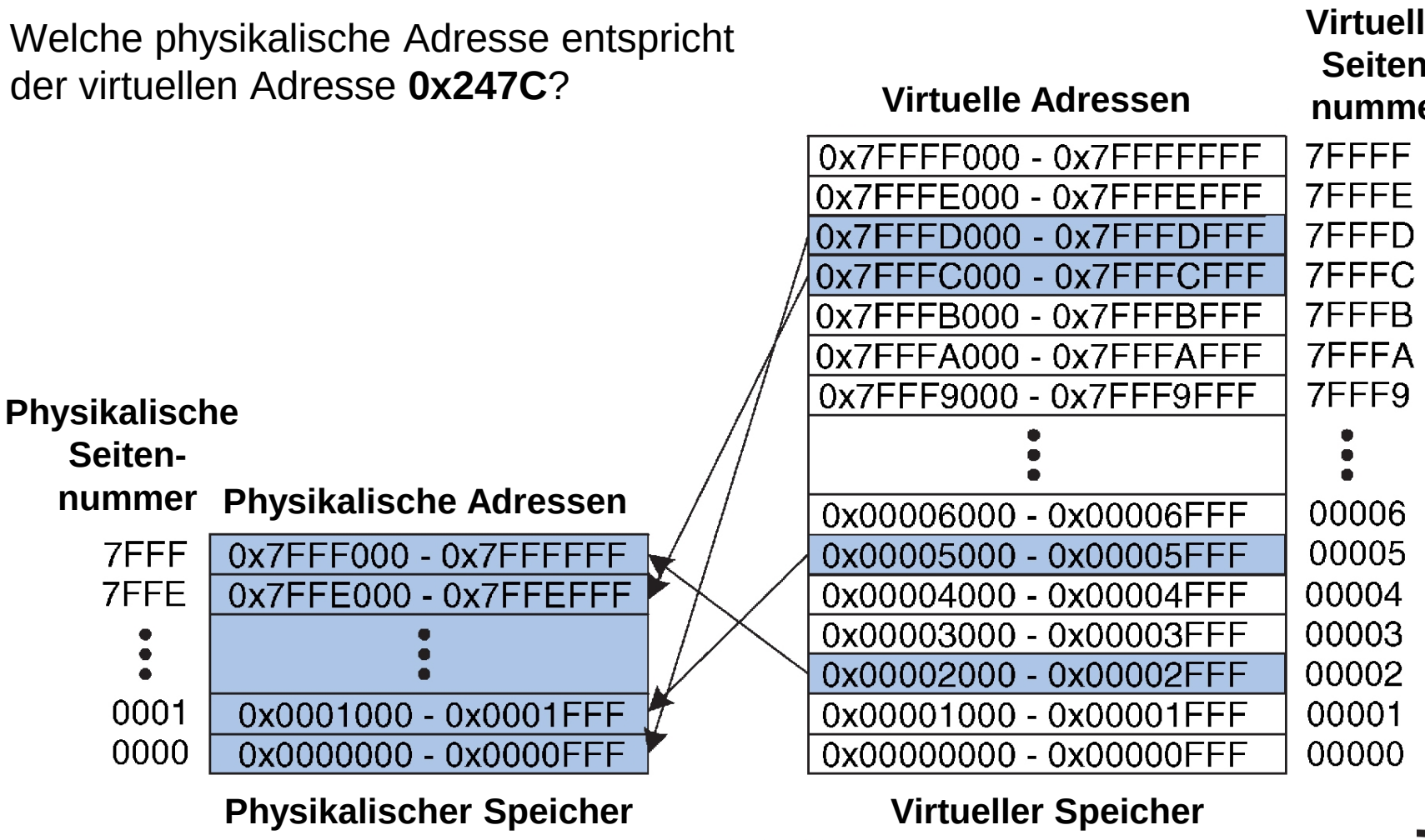
- 19-Bit virtuelle Seitennummer
- 15-Bit physikalische Seitennummer



Embedded Systems & Applications

Beispiel: Virtueller Speicher

Welche physikalische Adresse entspricht der virtuellen Adresse **0x247C**?



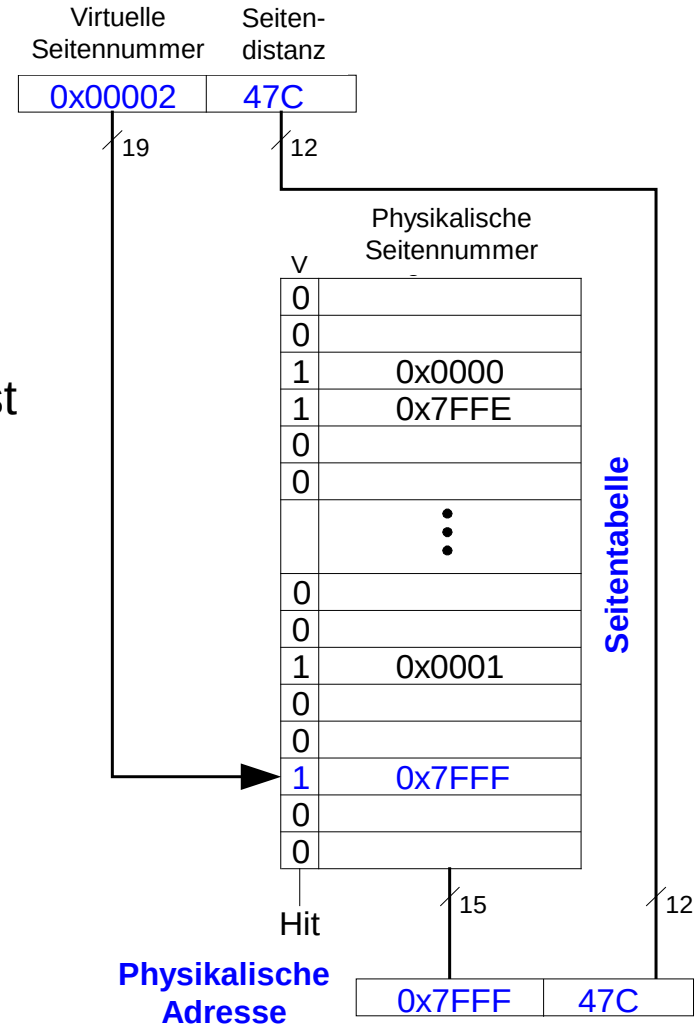
Wie übersetzen wir Adressen?

TECHNISCHE UNIVERSITÄT DARMSTADT

Seitentabelle (Page table)

- Jede virtuelle Speicherseite hat einen Eintrag in der Tabelle
- Jeder Eintrag besteht aus:
  - **Gültigkeits-Bit (Valid bit):** 1 wenn die virtuelle Seite im physikalischen Speicher steht, sonst 0 (= sie müsste von der Festplatte geholt werden)
  - **Physikalische Seitennummer:** wo im Hauptspeicher die Seite steht

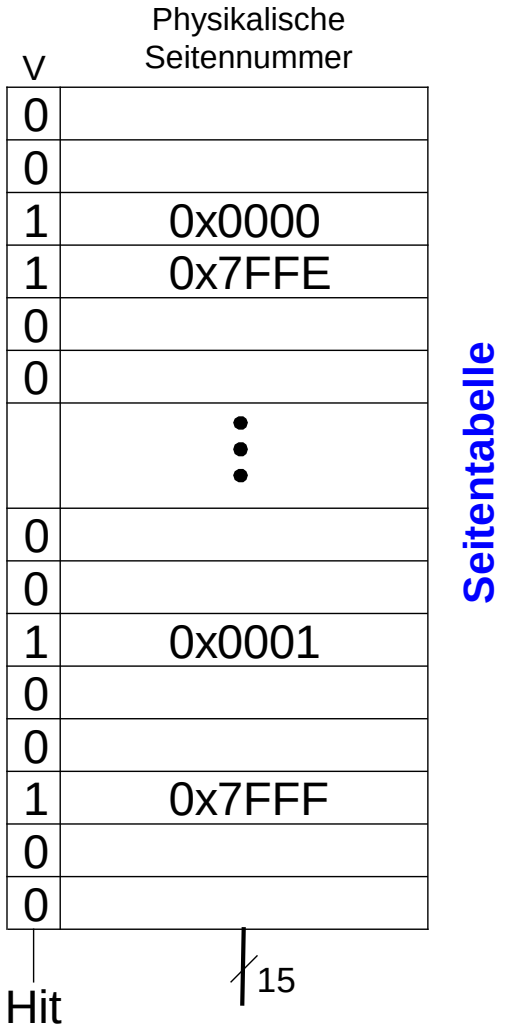
Beispiel: Seitentabelle (Page Table)



VSN (Virtuelle Seitennummer) ist der **Index** in der Seitentabelle

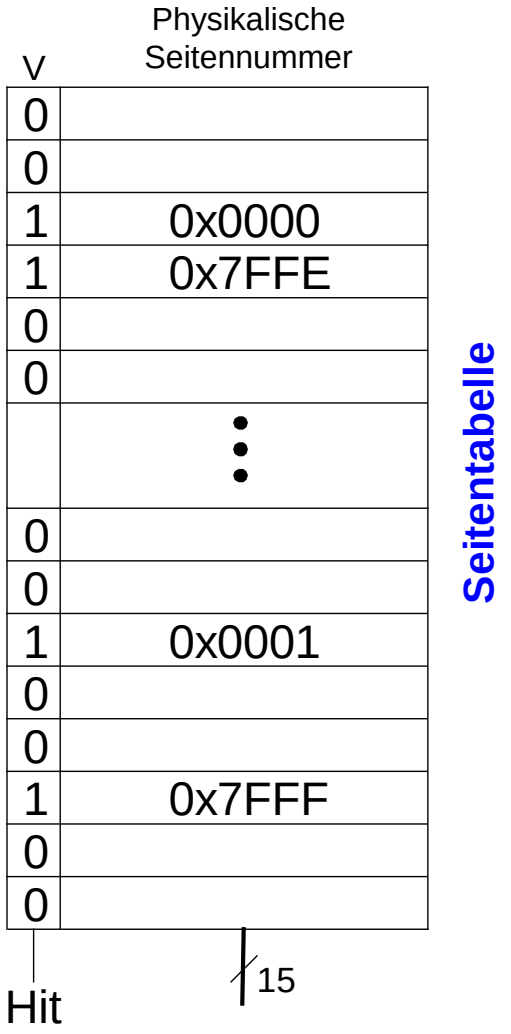
Beispiel 1: Seitentabelle

Welche physikalische Adresse entspricht der virtuellen Adresse **0x5F20**?



Beispiel 2: Seitentabelle

Welche physikalische Adresse entspricht der virtuellen Adresse **0x73E0**?



Herausforderungen beim Aufbau von Seitentabellen

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Die Seitentabelle ist gross**
  - steht häufig auch im physikalischen Speicher (und braucht dort Platz!)
- Alle Lese-/Schreibbefehle fordern **2 Hauptspeicherzugriffe**
  - einmal auf die Übersetzung (Zugriff auf Seitentabelle)
  - einmal auf die eigentlichen Daten (nach der Übersetzung)
- **Halbiert den Speicherdurchsatz**
  - *Kann man aber schlauer anstellen …*

Translation Lookaside Buffer (TLB)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein kleiner Cache für die neuesten Übersetzungen
- Die meisten Speicherzugriffe werden jetzt nur einen Hauptspeicherzugriff erfordern

Translation Lookaside Buffer (TLB)

TECHNISCHE UNIVERSITÄT DARMSTADT

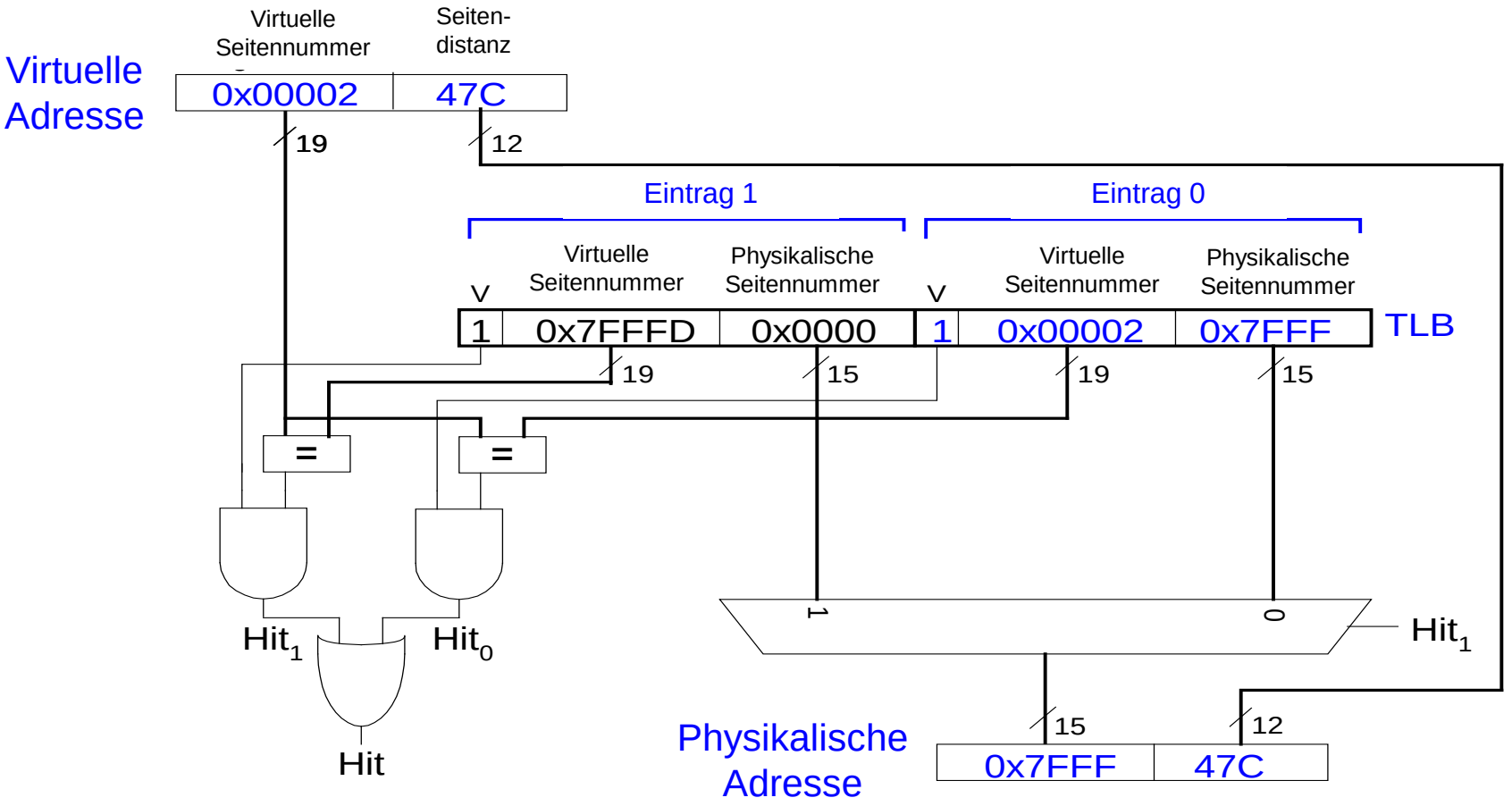
Seitentabellenzugriffe haben hohe zeitliche Lokalität, da:

- Ursprüngliche Datenzugriffe haben selber hohe zeitliche und räumliche Lokalität
- Die Seitengröße ist (relativ) groß, aufeinanderfolgende Lese-/Schreibbefehle werden oftmals auf die gleiche Seite zugreifen

TLB

- Klein:
  - Zugriff <= 1 Takt
  - Typischerweise 16 - 512 Einträge
  - Vollassoziativ
  - Typischerweise > 99 % Hit Rates
- Die meisten der Lese-/Schreibbefehle erfordern damit nur einen Hauptspeicherzugriff

Beispiel: TLB mit zwei Einträgen



Virtueller Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

Jedes Programm erzeugt **virtuelle Adressen**

- Der ganze Bereich der virtuellen Adressen steht auf der Festplatte
- Eine Untermenge der virtuellen Adressen steht im Hauptspeicher
  - DRAM / physikalischen Speicher
- Der Prozessor erzeugt virtuelle Adressen und muss diese in physikalische Adressen übersetzen
- Daten, die im DRAM nicht gefunden werden, müssen von der Festplatte geholt werden

Embedded Systems & Applications

Speicherschutz (Memory Protection)

TECHNISCHE UNIVERSITÄT DARMSTADT

Hier vereinfacht angenommen: Ein Programm ist ein *Prozess*

Jeder Prozess hat seine **eigene Seitentabelle**, damit gilt:

- Jeder Prozess kann **den ganzen Adressraum** benutzen – ohne sich Sorgen machen zu müssen welche Adressen die anderen Prozesse benutzen
  - Zwei Prozesse können die *gleiche virtuelle* Adresse für zwei *verschiedene* Variablen verwenden
  - Ein Prozess muss nicht die Adressvergabe eines anderen Prozesses berücksichtigen
- Ein Prozess kann **nur auf die physikalischen Seiten zugreifen**, die in seiner **eigenen Seitentabelle** stehen
  - Damit kann der Speicher anderer Prozessen nicht überschrieben werden
  - Ein Prozess (z.B. Virus) kann den Speicher eines anderen Prozesses nicht korrumpieren

Zusammenfassung vom virtuellem Speicher

TECHNISCHE UNIVERSITÄT DARMSTADT

- Virtueller Speicher erhöht die zugängliche **Speicherkapazität**
- Eine **Untermenge** der virtuellen Seiten wird im physikalischen Speicher gehalten
- Eine **Seitentabelle** zeigt an, wo die virtuellen Seiten im physikalischen Speicher stehen – und erlaubt so die **Adressübersetzung**
- Ein **TLB** beschleunigt die Adressübersetzung
- Da Prozesse ihre eigenen Seitentabellen haben, erlaubt virtueller Speicher auch **Speicherschutz** (**memory protection**)

Speichereinblendung von Ein-/Ausgabegeräten

TECHNISCHE UNIVERSITÄT DARMSTADT

Prozessoren greifen auf **Ein-/Ausgabegeräte** (z.B. Tastaturen, Monitore, und Drucker) zu genau **wie auf Daten im Speicher**

- Ein oder mehrere Adressen sind einem Gerät zugewiesen

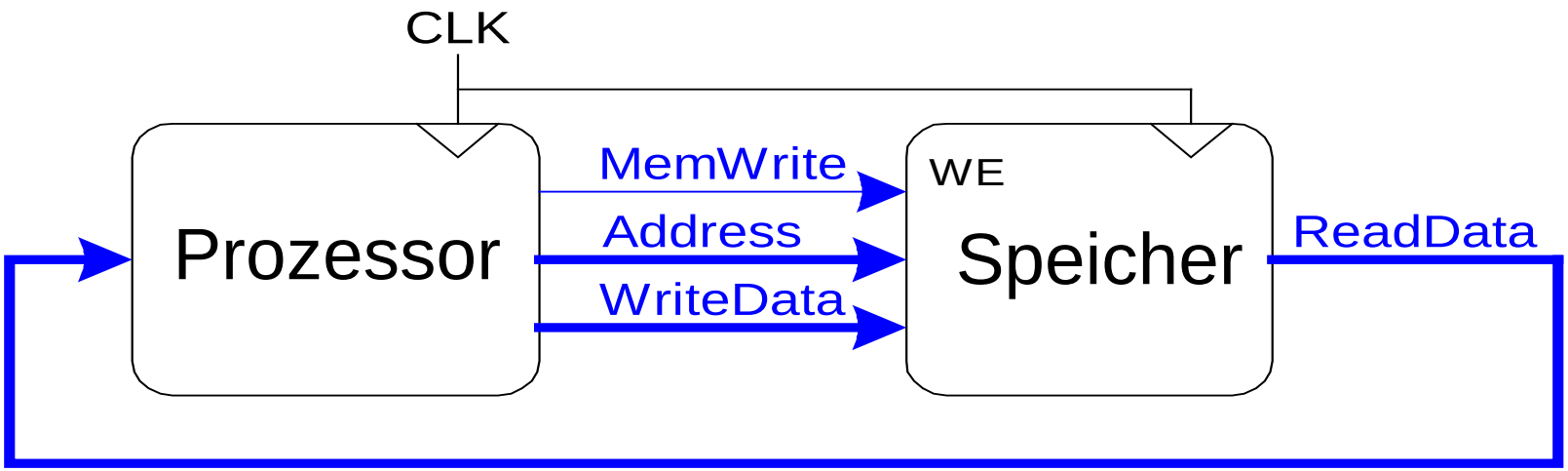
- Wenn auf diese Adressen zugegriffen wird, werden die Daten vom **Gerät** gelesen oder geschrieben (statt vom Speicher!)

- Eine Untermenge der Adressen sind den Ein-/Ausgabegeräten zugewiesen,
  - z.B., Adressen 0xFFFF0000 bis 0xFFFFFFFF könnten dafür reserviert sein
  - Konkrete Zuordnung hängt von Rechnerarchitektur und Betriebssystem ab
    - Wird oftmals beim Starten des Betriebssystems ausgegeben:

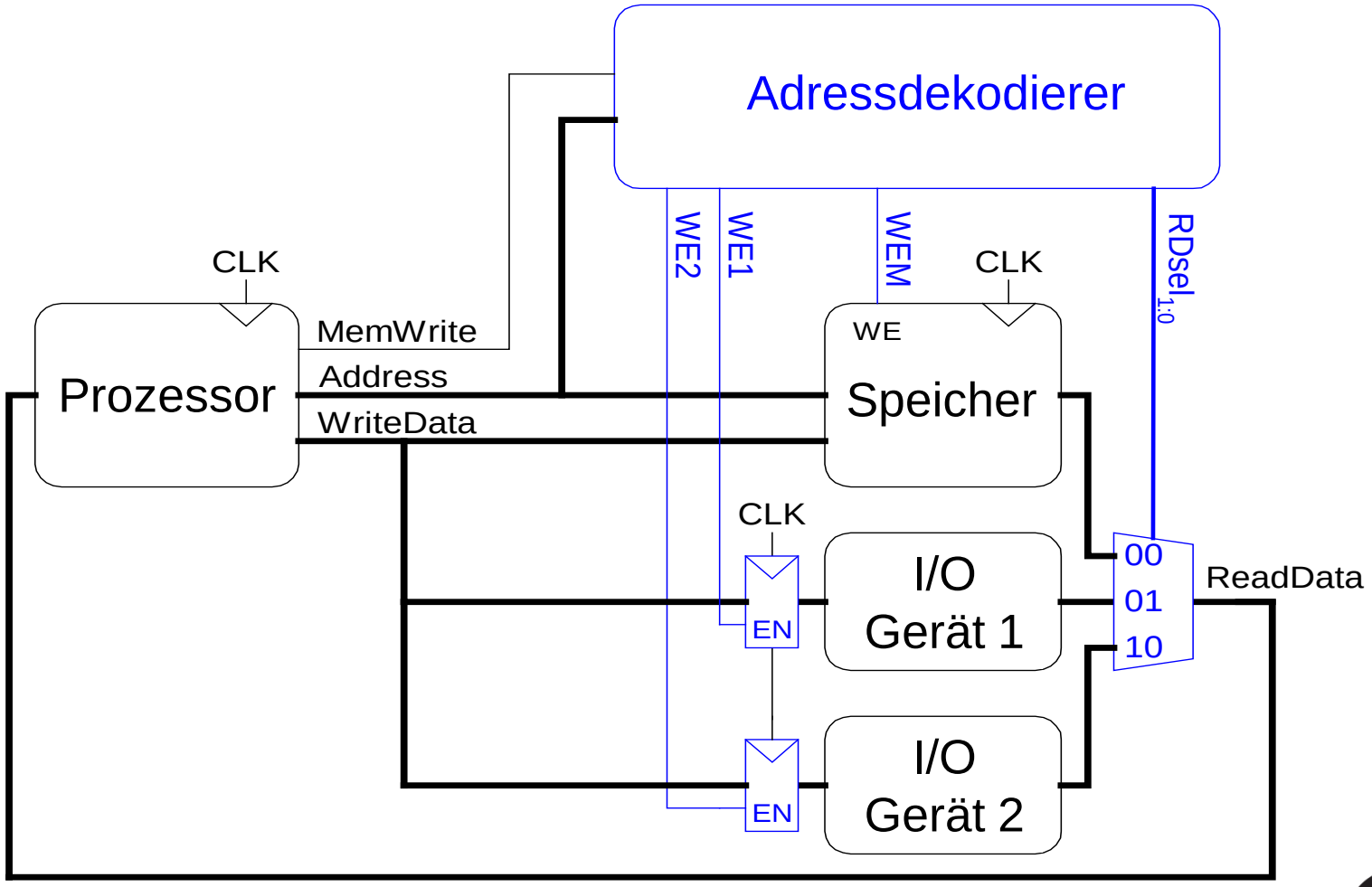
```
…
uhci_hcd 0000:00:1d.1: UHCI Host Controller
uhci_hcd 0000:00:1d.1: new USB bus registered, assigned bus number 7
uhci_hcd 0000:00:1d.1: irq 19, io base 0x0000d400
usb usb7: New USB device found, idVendor=1d6b, idProduct=0001
…
```


Embedded Systems & Applications

Speicherschnittstelle



Erweiterung der Hardware für Ein-/Ausgabegeräte



Hardware für Ein-/Ausgabegeräte

TECHNISCHE UNIVERSITÄT DARMSTADT

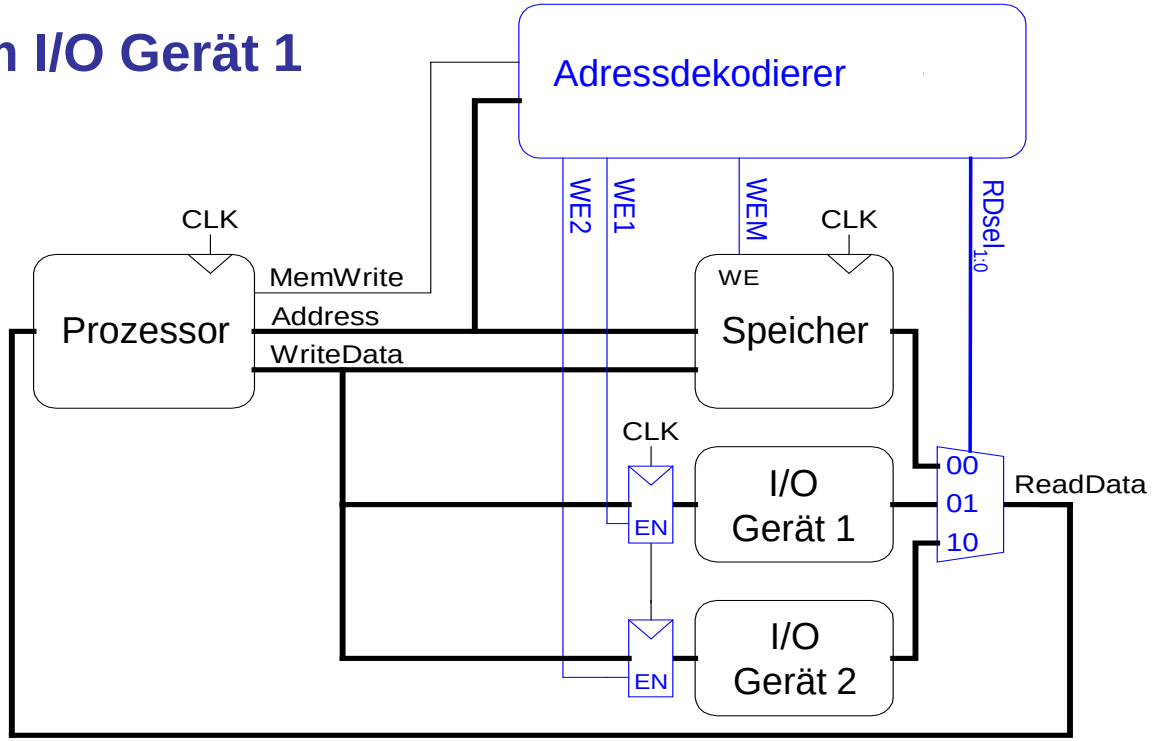
- **Adressdekodierer:**
  - Dekodiert die Adresse um festzustellen, auf welches Gerät (E/A-Gerät oder Speicher) der Prozessor zugreifen möchte
- **Ein-/Ausgaberegister:**
  - Beim **Schreiben** auf ein Gerät: Enthält Werte, die der Prozessor zu den Ein-/Ausgabegeräten schreibt
- **Multiplexer:**
  - Beim **Lesen** von einem Gerät: Wählt zwischen Geräten und Speicher eine Quelle für die Daten aus, die dem Prozessor geschickt werden

Beispiel: Zugriff auf Ein-/Ausgabegeräte

TECHNISCHE UNIVERSITÄT DARMSTADT

Annahme: dem Ein-/Ausgabegerät 1 ist die Adresse 0xFFFFFFF4 zugewiesen

- **Übertragen Sie den Wert 42 auf das I/O Gerät 1**
- **Lesen Sie den Wert vom I/O Gerät 1 in Register `$t3`**



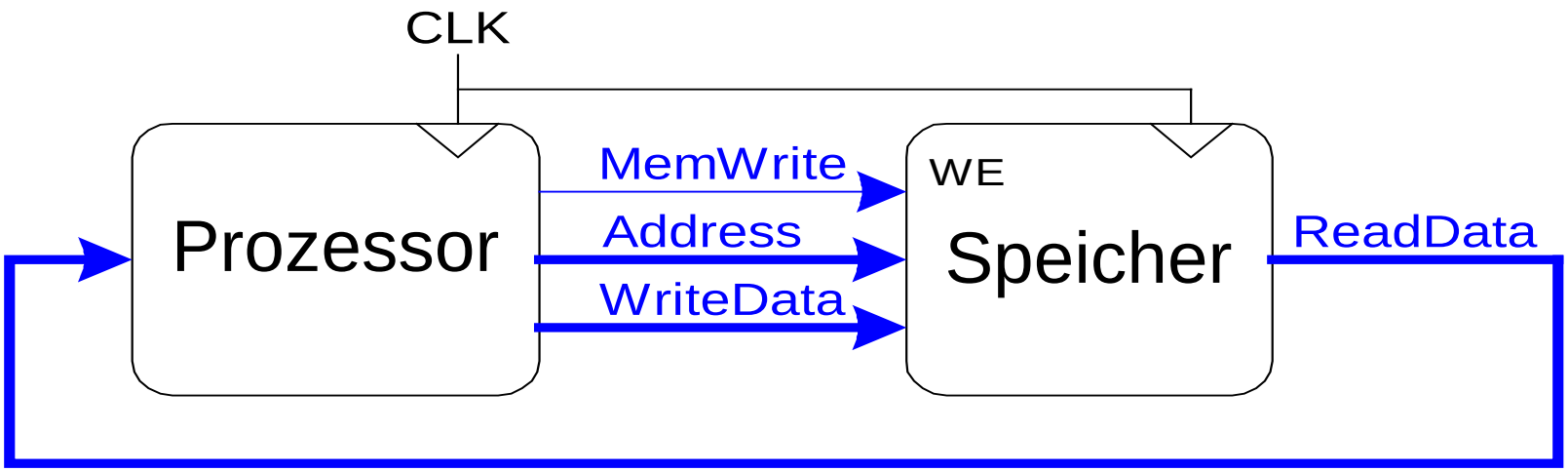
Embedded Systems & Applications

Wiederholung: Speichersysteme

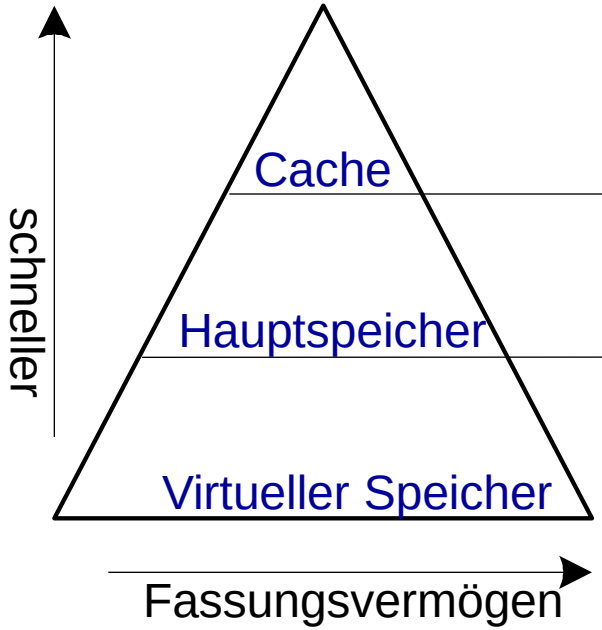
TECHNISCHE UNIVERSITÄT DARMSTADT

- Speicherschnittstelle
- Speicherhierarchie
- Ein-/Ausgabe von Daten durch Prozessor

Wiederholung: Speicherschnittstelle



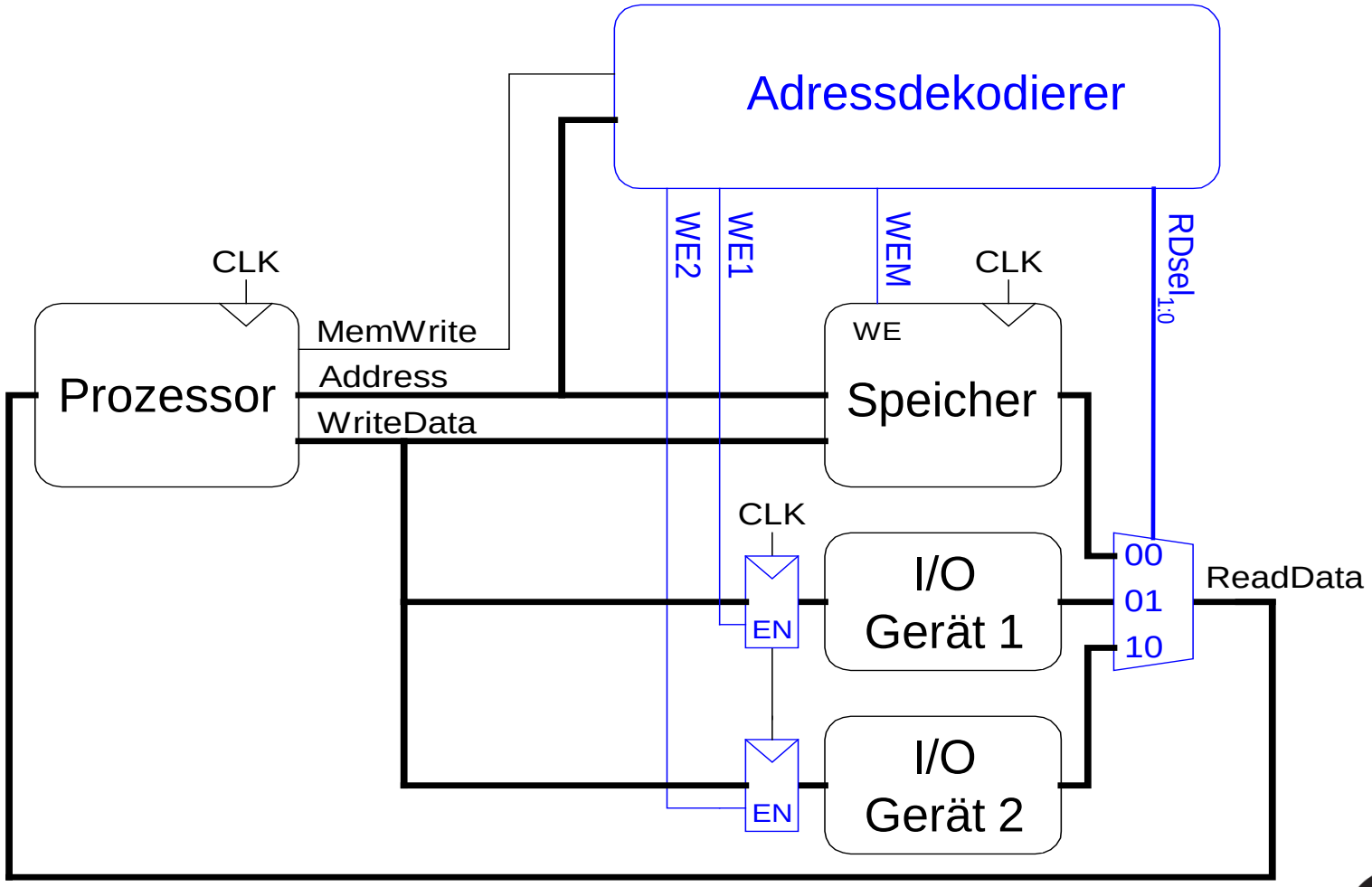
Wiederholung: Speicherhierarchie



Technologie	Kosten/GB	Zugriffszeit
SRAM	~ $10,000	~ 1 ns
DRAM	~ $10	~ 10-50 ns
Festplatte	~ $0.1	~ 100,000 ns
SSD	~ $1	~ 10,000,000 ns

Durch Hierarchie, emuliert das Speichersystem Speicher der **schnell, gross, und günstig** ist.

Speichereinblendung von Ein-/Ausgabegeräten



Zusammenfassung der Veranstaltung

TECHNISCHE UNIVERSITÄT DARMSTADT

Lehrstoff:

- Rechnerarchitektur – Assemblersprache
- Mikroarchitektur – Aufbau eines Prozessors
- Speichersysteme

- Digitale Geräte werden immer mehr benutzt
- Jetzt haben Sie die Werkzeuge, diese auch zu *verstehen*

  … und auch mit zu entwerfen!
- Macht’s gut!

Embedded Systems & Applications