Technische Grundlagen der Informatik – Kapitel 8

Prof. Dr. Andreas Koch
Fachbereich Informatik
TU Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

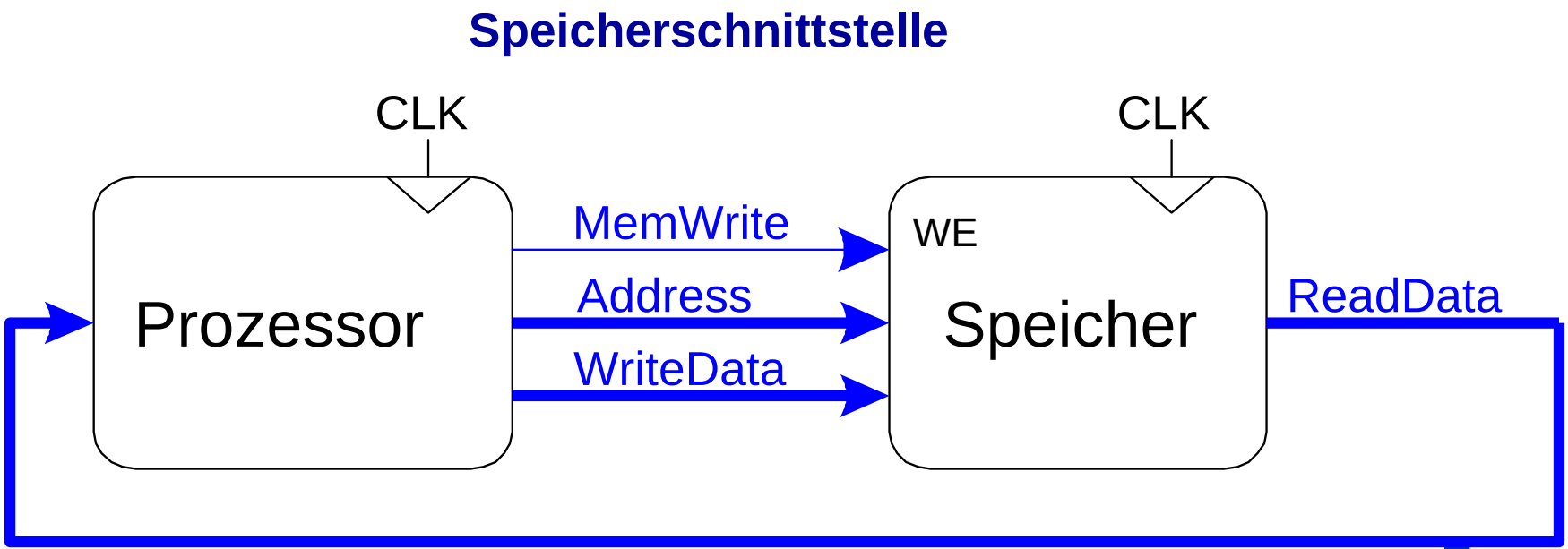
Kapitel 8: Themen

TECHNISCHE UNIVERSITÄT DARMSTADT

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

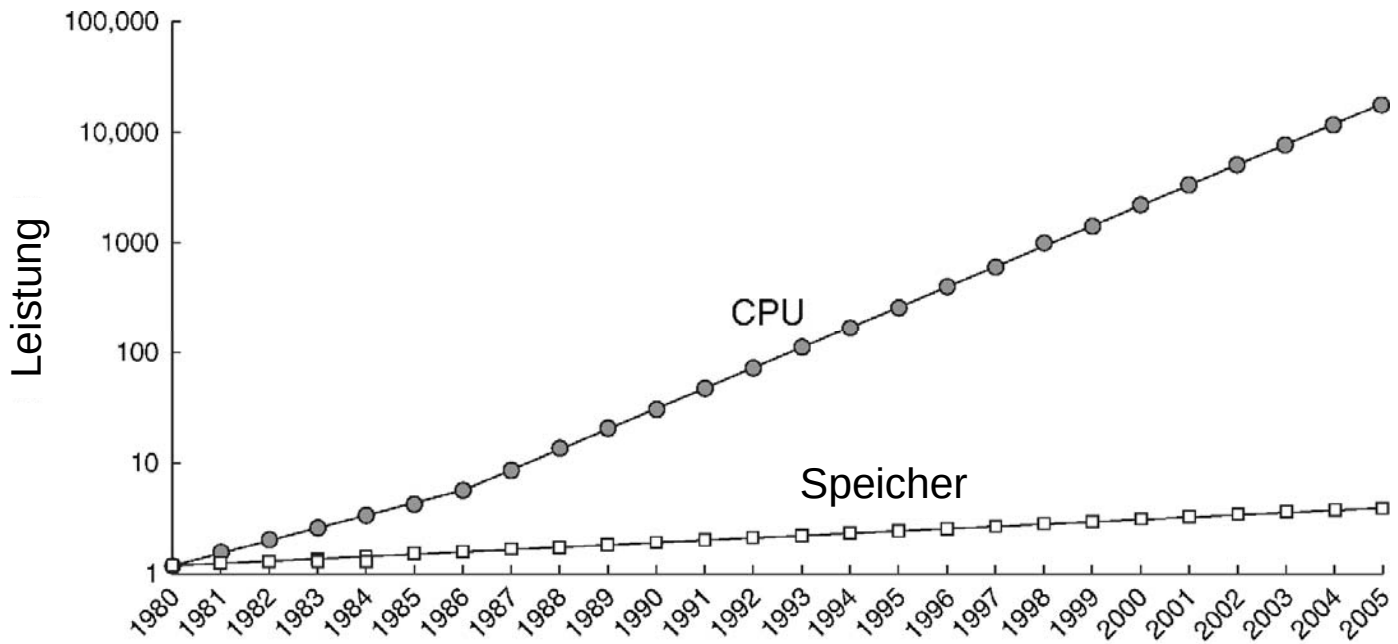
Einleitung

- Rechenleistung hängt ab von:
  - Prozessorleistung
  - Leistung des Speichersystems



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

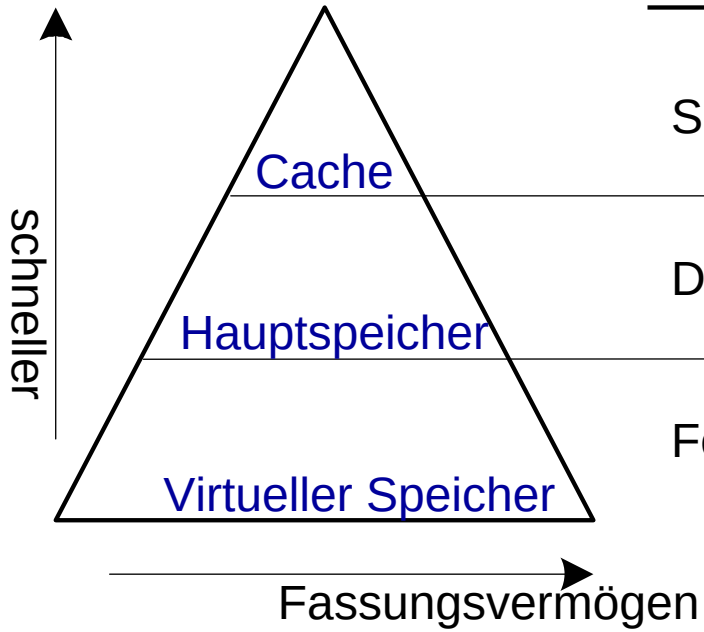
TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

Speicherhierarchie: Beispiele für Ebenen



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

TECHNISCHE UNIVERSITÄT DARMSTADT

- **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

embedded systems + applications

Beispiel 1: Leistung eines Speichersystems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Programm hat 2.000 Load- und Store-Befehle
- 1.250 der Daten werden im Cache vorgefunden
- Der Rest kommt aus anderen Ebenen der Speicherhierarchie

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

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?**

Hit Rate = 1250/2000 **= 0,625**

Miss Rate = 750/2000 = **0,375** = 1 – Hit Rate

Beispiel 2: Leistung eines Speichersystems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Annahme: Prozessor hat zwei Hierarchieebenen:
  - Cache
  - Hauptspeicher
- t_{cache} = 1 Takt
- t_{MM} = 100 Takt

- **Wie lang ist die AMAT des Programmes aus Beispiel 1?**

Beispiel 2: Leistung eines Speichersystems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Annahme: Prozessor hat zwei Hierarchieebenen:
  - Cache
  - Hauptspeicher
- t_{cache} = 1 Takt
- t_{MM} = 100 Takte

- **Wie lang ist die AMAT des Programmes aus Beispiel 1?**

$$\begin{aligned} \textbf{AMAT} &= t_{cache} + MR_{cache}\, t_{MM} \\ &= [1 + 0{,}375 * 100] \text{ Takte} \\ &= \textbf{38,5 Takte} \end{aligned}$$

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

Geheimes 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

Aufbau von Caches: Entwurfsentscheidungen

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

Schwerpunkt hier auf Loads, Stores werden aber ähnlich gehandhabt

Welche Daten werden im Cache gehalten?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Idealfall: Cache “ahnt” im Voraus, welche Daten der Prozessor benötigen wird und hält diese bereit
- Praxis: Prophezeiungen in der Regel ungenau
- 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

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 Assoziativitätsmengen ($S = B/N$):
  - Jede Speicheradresse wird auf genau eine Assoziativitätsmenge abgebildet

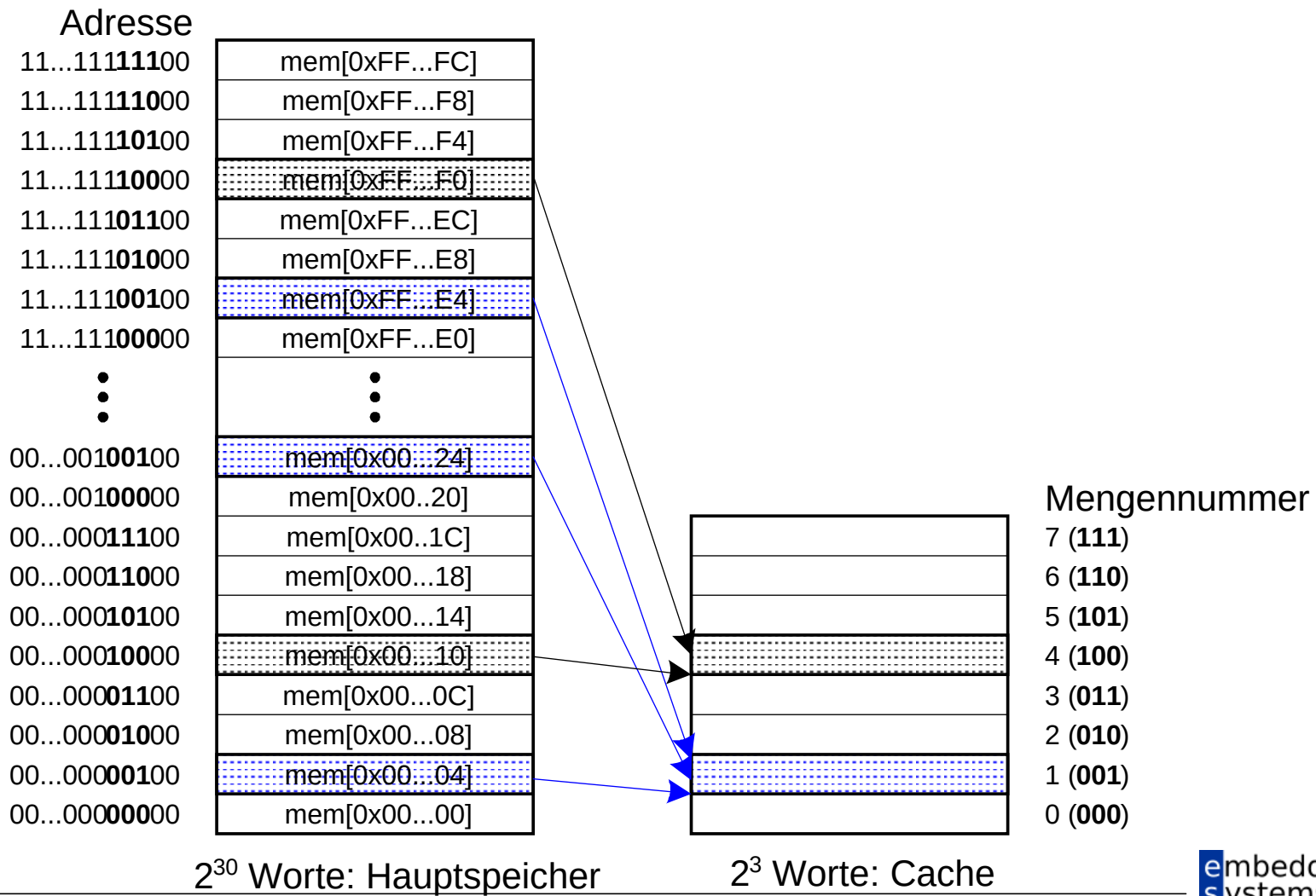
Wie werden Daten gefunden?

TECHNISCHE UNIVERSITÄT DARMSTADT

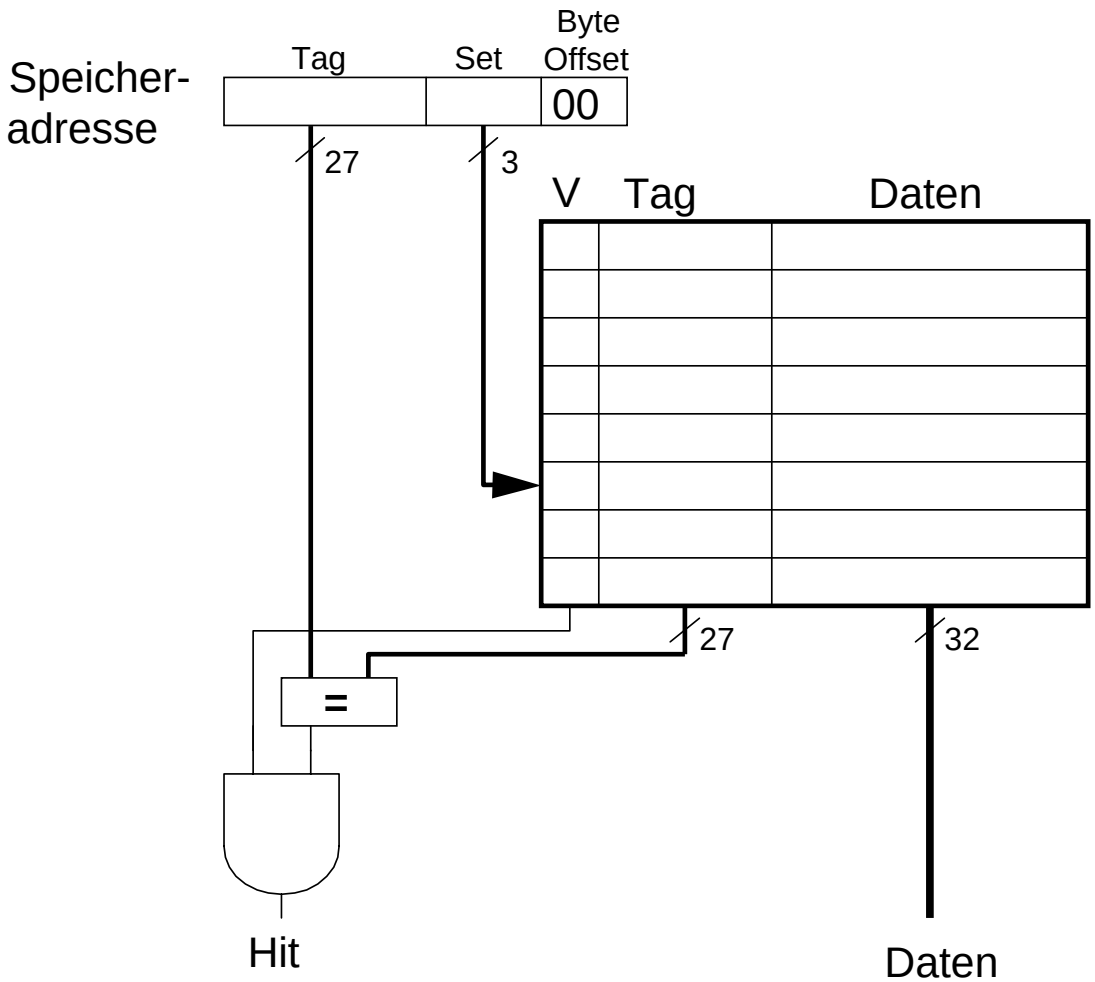
- Cache ist organisiert als *S* Mengen (*sets)*
- Jede Speicheradresse wird genau auf eine Menge abgebildet

- Caches werden klassifiziert nach Anzahl von Blöcken in einer Menge:
  - **Direktabgebildet (*direct mapped*):** Ein Block pro Menge
  - ***N*-fach Mengenassoziativ (*N-way set associative*):** *N* Blöcke pro Menge
  - **Vollassizuativ (*fully associative)*:** Alle Cache Blocks in einer Menge
- Beispielorganisationen für einen Cache mit:
  - Kapazität *C* = 8 Worte
  - Blockgröße *b* = 1 Wort
  - Damit ist Blockanzahl *B* = 8

Direktabgebildeter Cache



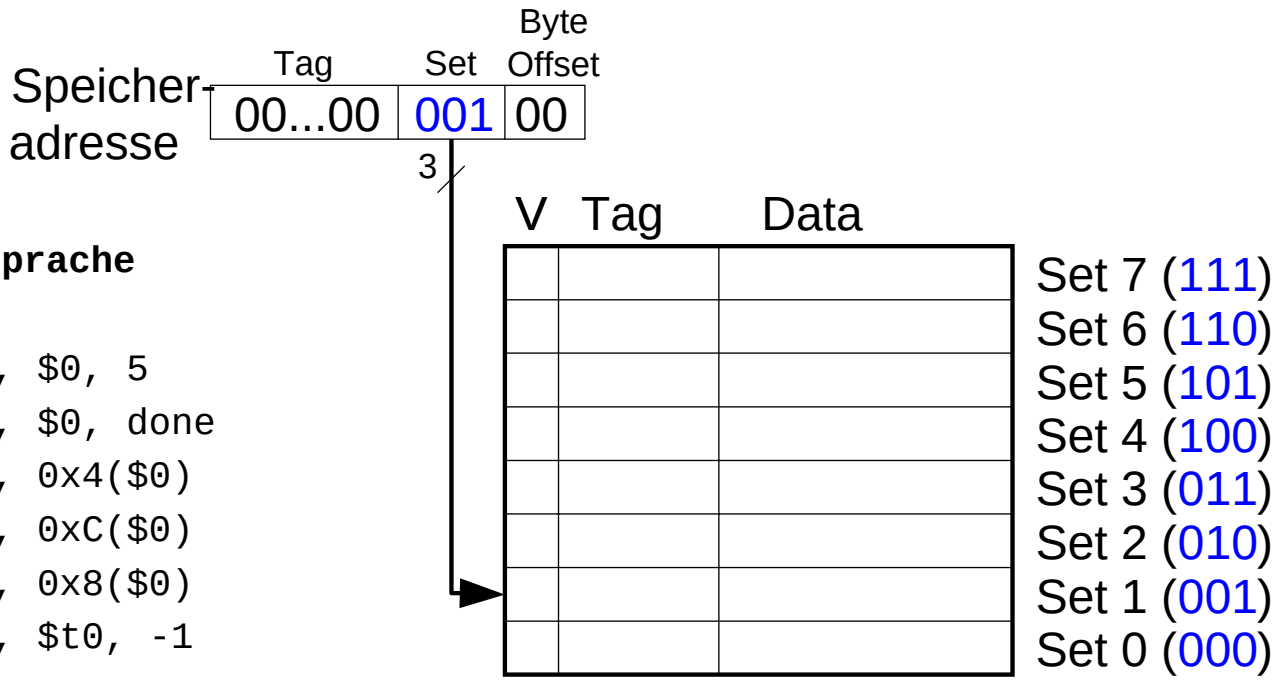
Direktabgebildeter Cache: Hardware



8 Einträge x
(1+27+32)-Bit
SRAM

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

Leistung eines direktabgebildeten Caches

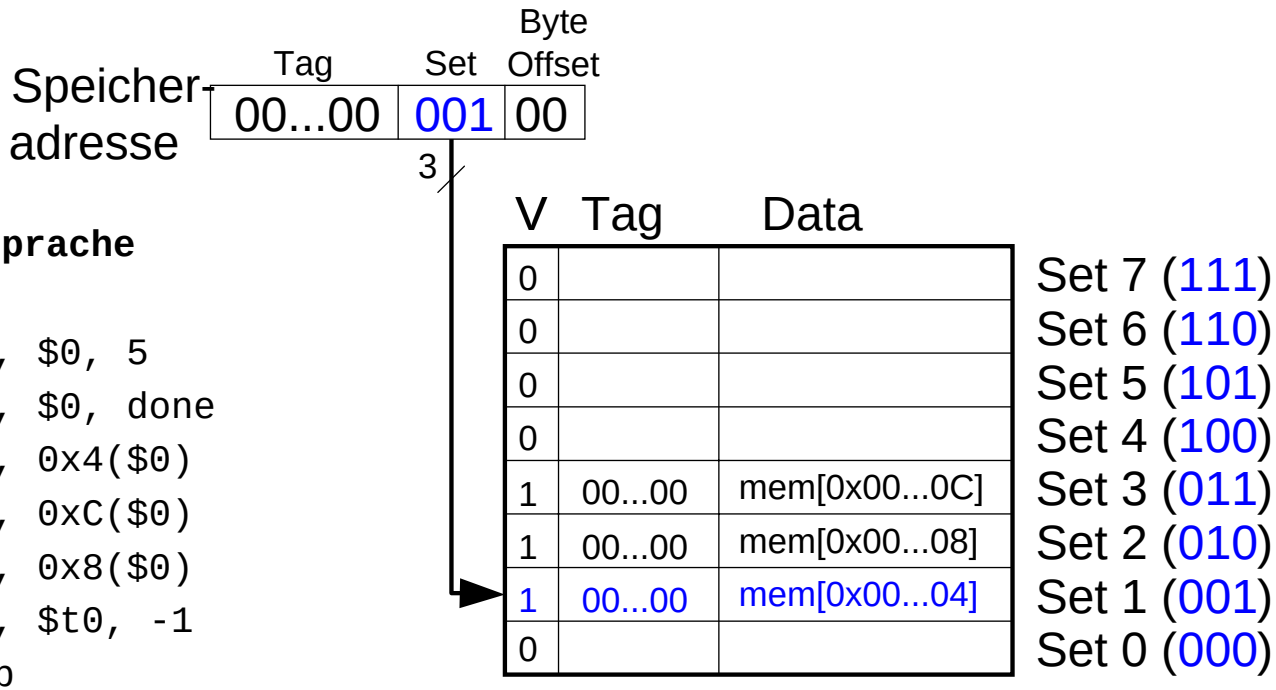



```
# MIPS Assemblersprache

        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 =

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:
```



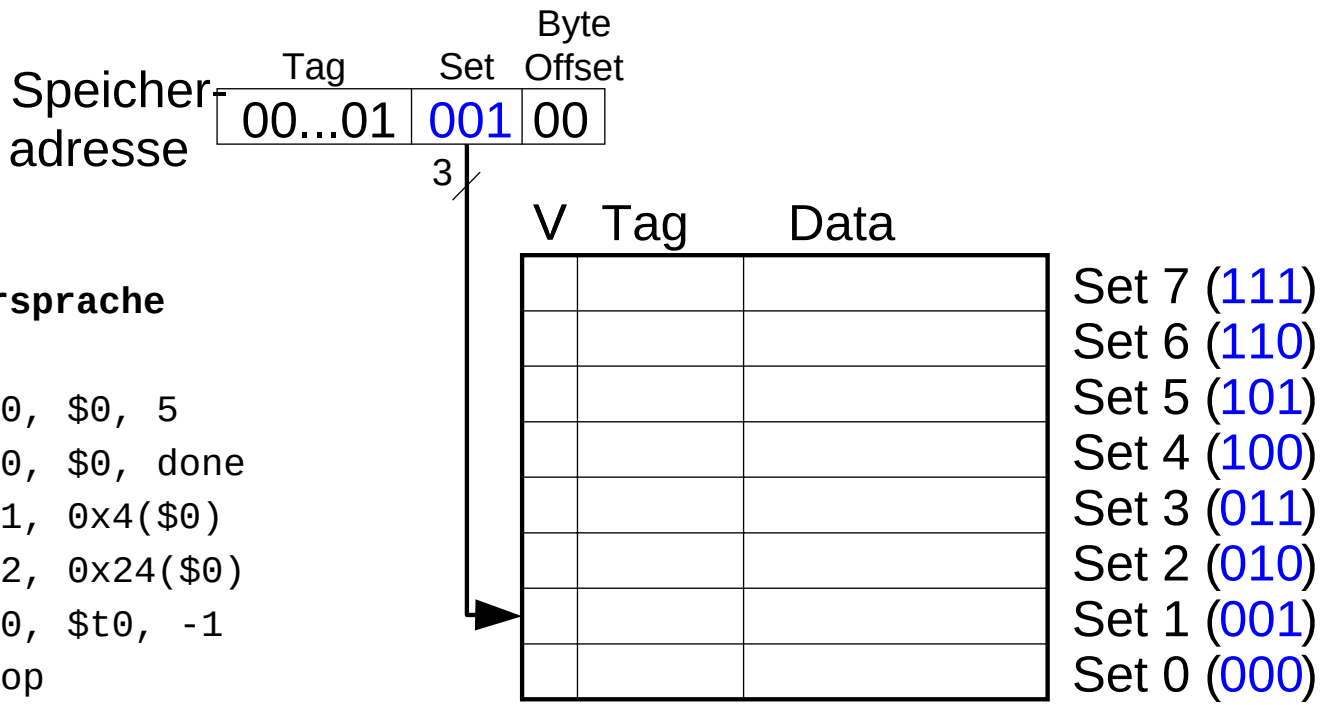
Miss Rate = 3 / 15

= 20%

Temporale Lokalität

Unvermeidbare (*compulsory*) Misses

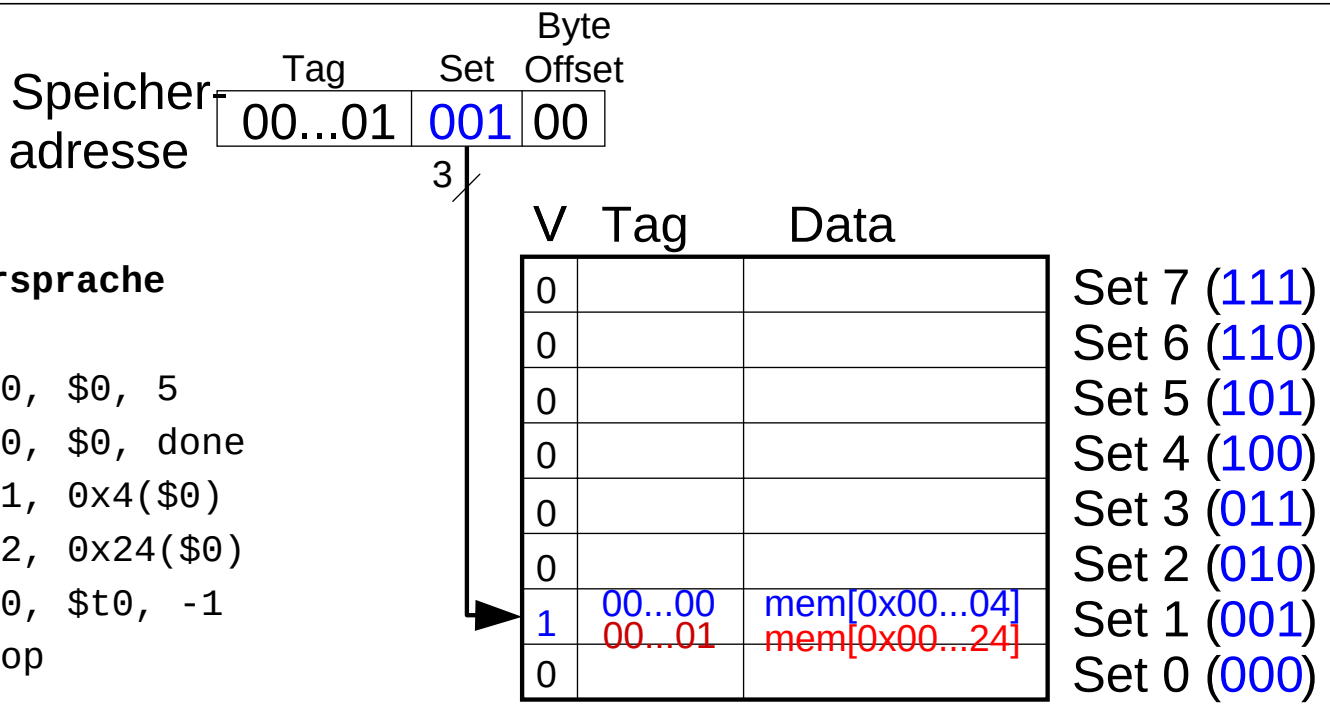
Konflikte bei direktabgebildeten Caches




```
# 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:
```

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:
```



Miss Rate = 10/10

= 100%

Conflict Misses