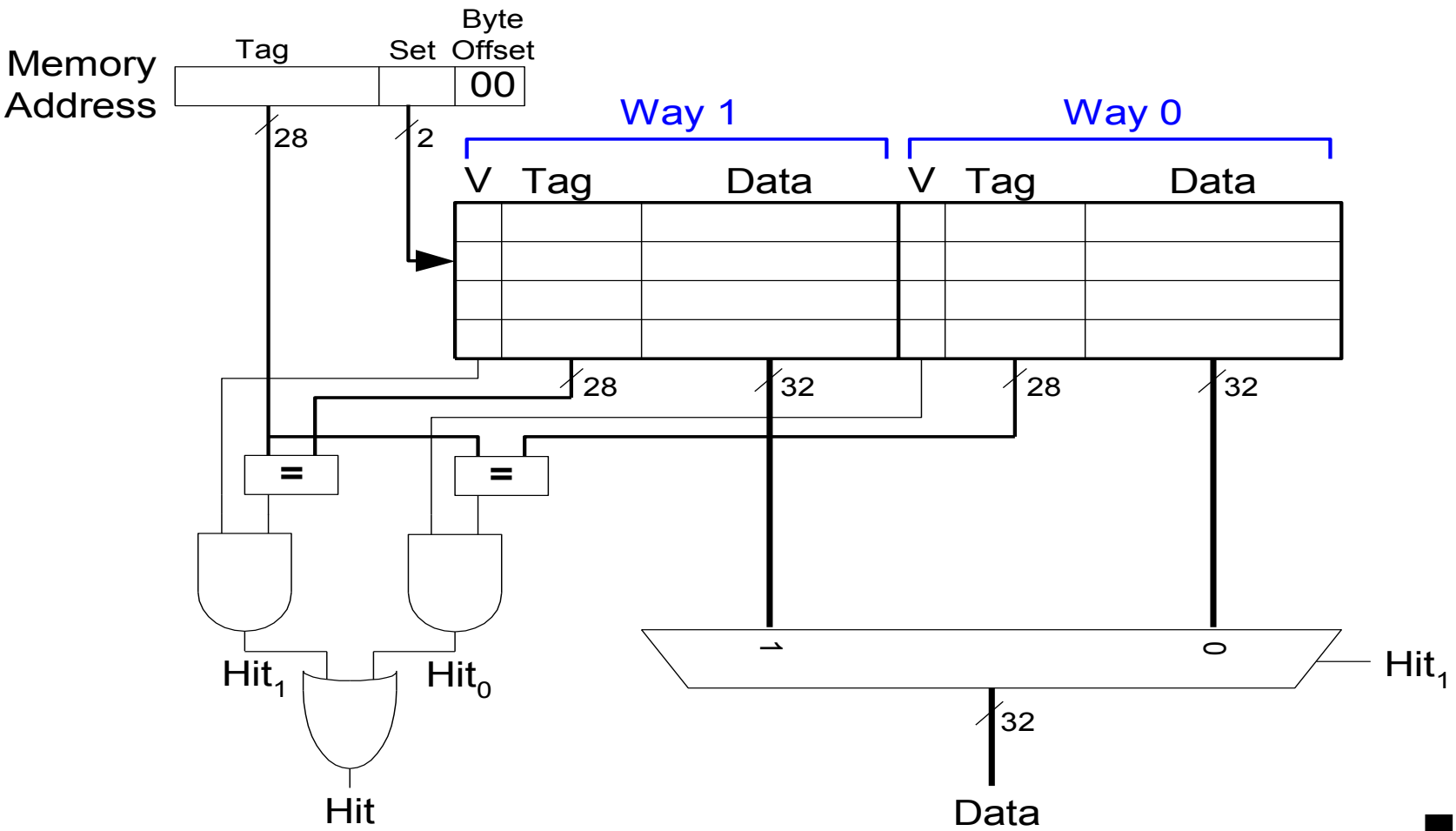
Meisenantworten

- *Lautstärke des Microphons zu niedrig*! Aye, aye, Sir!
- *Wie funktioniert binäre Division*? Bitte 5.2.7 (S.253) im Textbuch lesen.
- *Welche anderen Veranstaltungen bietet IAS an*?
  - Immer: Robot Learning (WiSe), Machine Learning: Statistische Ansätze (SoSe)
  - Manchmal: Computational Engineering (WiSe)
- *Lob, Erklärungen zu Evaluation*. Danke!
- *Radiobuttons in Lehrewiki*. Gute Idee, verlangt aber ein anderes Back-End. Mal sehen…

N-fach Mengenassoziativ



N-fach Mengenassoziativ

```
# MIPS assembly code

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

Way 1			Way 0		
V	Tag	Data	V	Tag	Data

N-fach Mengenassoziativ

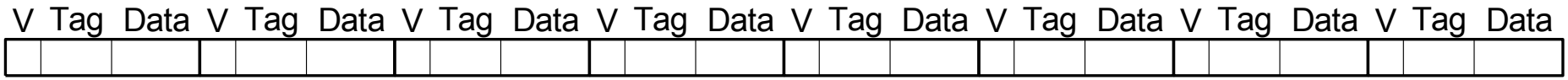
```
# MIPS assembly code

        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 = 2/10
= 20%

"Associativity" reduziert die Conflict Miss Rate

Way 1			Way 0			
V	Tag	Data	V	Tag	Data	
0			0			Set 3
0			0			Set 2
1	00...10	mem[0x00...24]	1	00...00	mem[0x00...04]	Set 1
0			0			Set 0

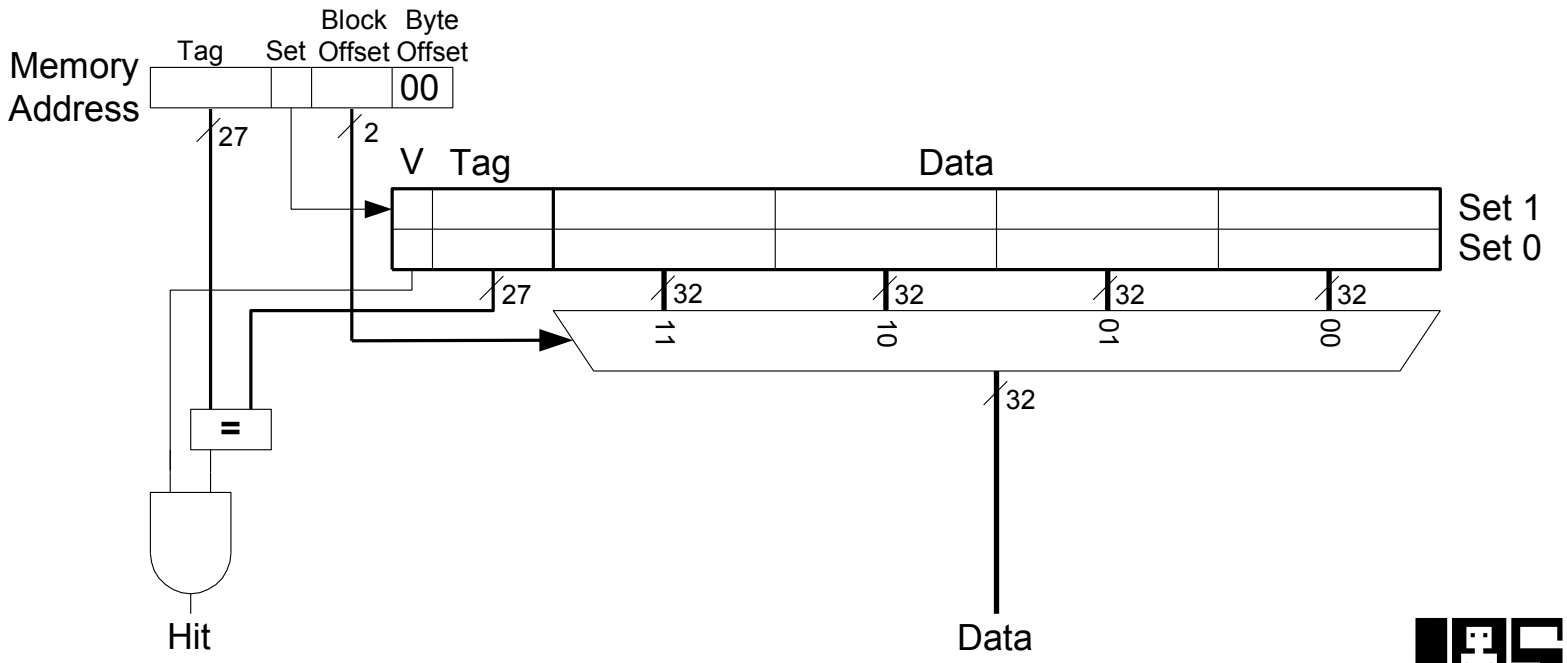
Vollassoziativ



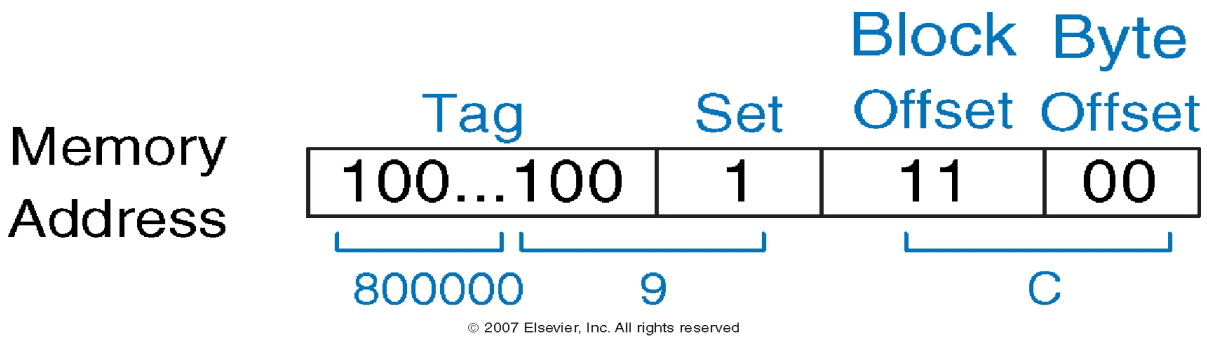
Keine “Conflict Misses” – aber superteuer zu bauen!

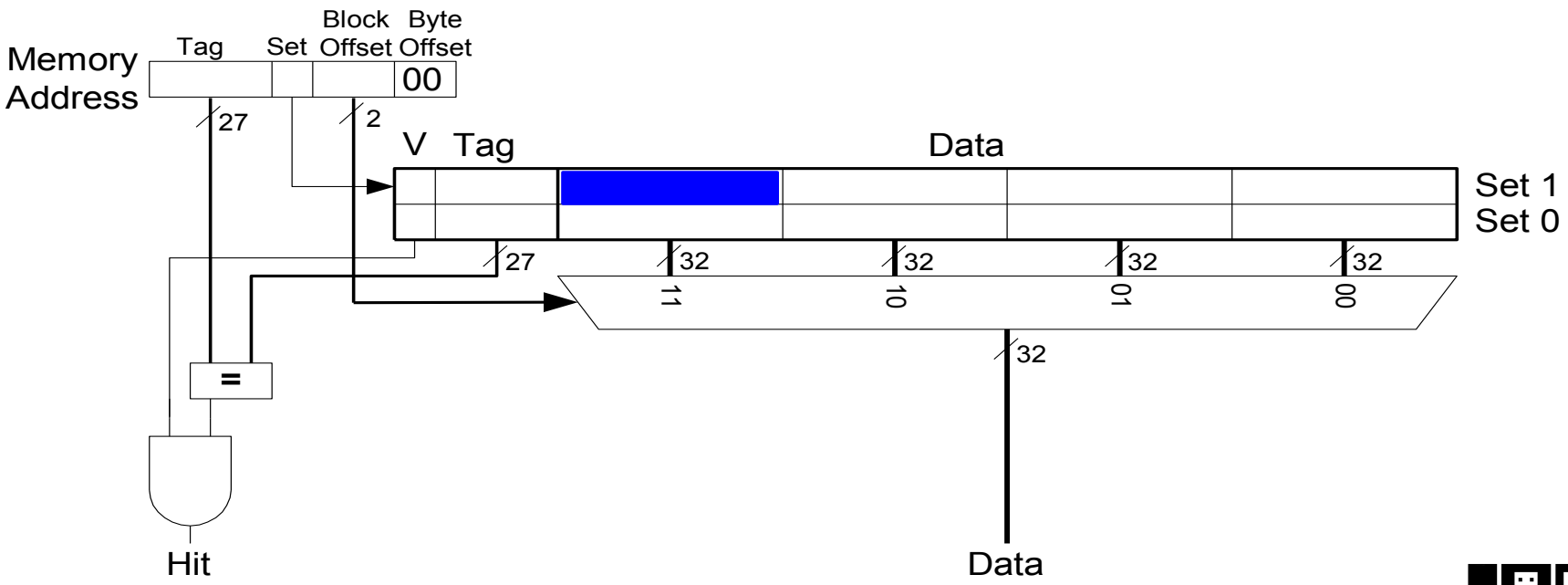
Räumliche Lokalität?

- Blockgröße erhöhen!
  - Blockgröße , ***b* = 4** Worte
  - *C* = 8 Worte
  - Direktabgebildeter Cache (1 Block pro Assoziativitätsmenge)
  - Blockanzahl, *B* = *C*/*b* = 8/4 = 2



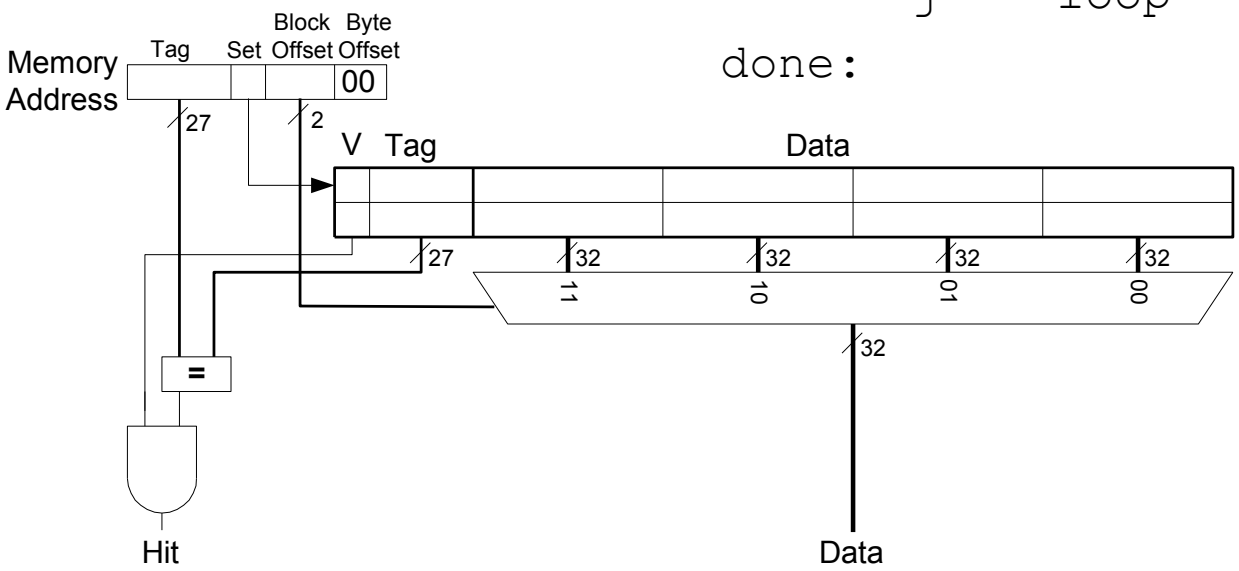
Cache mit höherer Blockgröße





Direktabbildung mit höherer Blockgröße

```
        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:
```



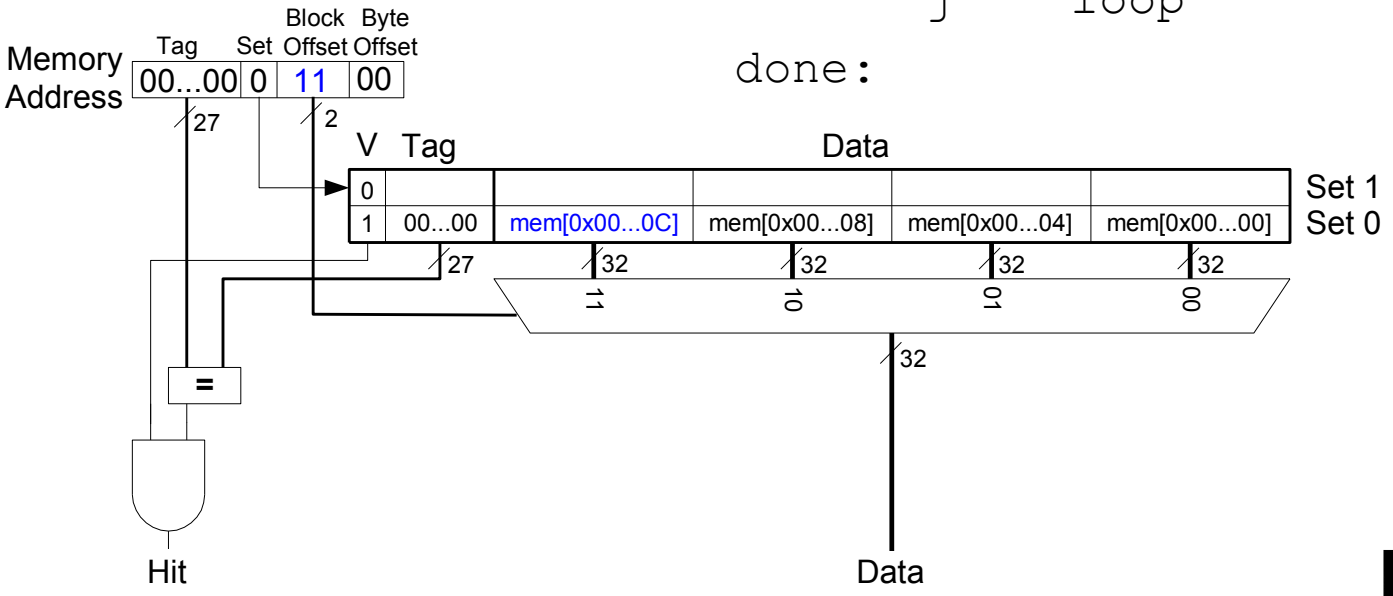
Direktabbildung mit höherer Blockgröße

TECHNISCHE UNIVERSITÄT DARMSTADT

Miss Rate = 1/15
= 6.67%

Größerer Blöcke reduzieren die Anzahl der unvermeidbaren (compulsory) Misses durch räumliche Lokalität

```
        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:
```



Cache Organisation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kapazität: C
- Blockgröße b
- Anzahl von Blöcken im Cache: $B = C/b$
- Assoziativitätsgrad, d.h., Anzahl von Blöcken in einer Assoziativitätsmenge: N
- Anzahl von Assoziativitätsmengen: $S = B/N$

Organisation	Assoziativitätsgrad (N)	Anzahl von Assoziativitäts mengen S
Direktabgebildet	1	B
N-fach Mengenassoziativ	$1 < N < B$	B / N
Vollassoziativ	B	1

Sorten von Misses

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Compulsory:** das erste Mal das wir einen Datenpunkt erwischen
- **Capacity:** Cache reicht nicht aus um alle notwendigen Datenpunkte zu speichern.
- **Conflict:** Mehrere Datenpunkte liegen am gleichen Punkt im Cache

Miss penalty: Zeitliche Verzögerungen die durch das Holen des Blocks aus dem Speicher entstehen.

Capacity Misses

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder sinnvolle Cache ist zu klein um alle benötigten Daten zu speichern.
- Wenn der Cache voll ist und ein neuer Datenpunkt X vom Programm angefordert wird, muss ein anderer Datenpunkt Y aus dem Cache rausgeworfen werden.
- Ein ***Capacity Miss*** passiert wenn Datenpunkt Y danach doch wieder gebraucht wird.

- **Wie können wir in einem 2-fach Mengenassoziativen Cache die Wahrscheinlichkeit minimieren das Y umsonst weggeschmissen wird?**

Capacity Misses

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder sinnvolle Cache ist zu klein um alle benötigten Daten zu speichern.
- Wenn der Cache voll ist und ein neuer Datenpunkt X vom Programm angefordert wird, muss ein anderer Datenpunkt Y aus dem Cache rausgeworfen werden.
- Ein ***Capacity Miss*** passiert wenn Datenpunkt Y danach doch wieder gebraucht wird.

- **Wie können wir in einem 2-fach Mengenassoziativen Cache die Wahrscheinlichkeit minimieren das Y umsonst weggeschmissen wird?**
  - **Least recently used (LRU) replacement**: Ersatz des am längsten ungenutzen Eintrags.

“Last Recently Used Replacement” Ersatz des am längsten ungenutzen Eintrags

TECHNISCHE UNIVERSITÄT DARMSTADT

```
# MIPS assembly

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

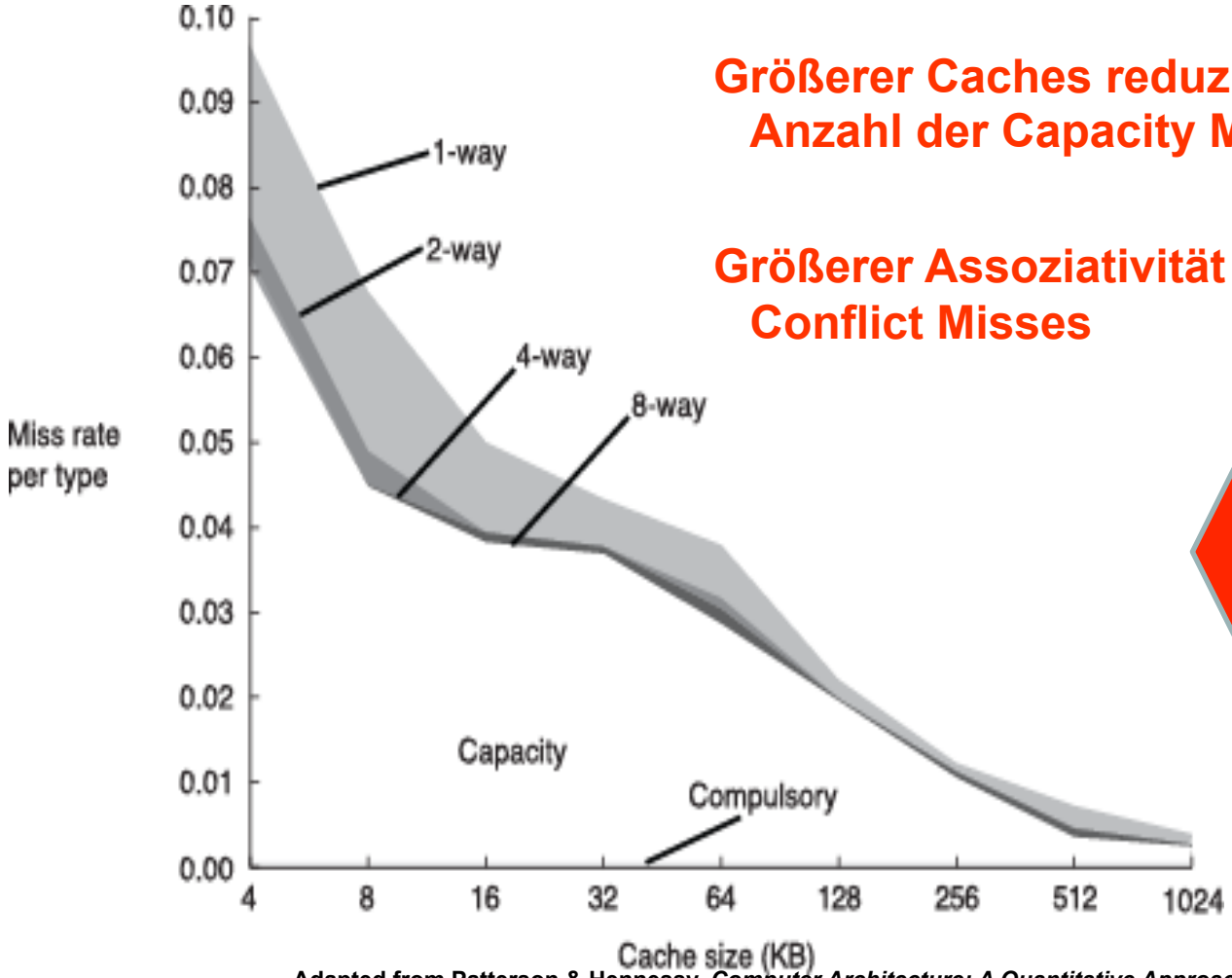
	Way 1				Way 0			
	V	U	Tag	Data	V	Tag	Data	
	0	0			0			Set 3 (11)
	0	0			0			Set 2 (10)
	1	0	00...010	mem[0x00...24]	1	00...000	mem[0x00...04]	Set 1 (01)
	0	0			0			Set 0 (00)

(a)

	Way 1				Way 0			
	V	U	Tag	Data	V	Tag	Data	
	0	0			0			Set 3 (11)
	0	0			0			Set 2 (10)
	1	1	00...010	mem[0x00...24]	1	00...101	mem[0x00...54]	Set 1 (01)
	0	0			0			Set 0 (00)

(b)

Miss Rate



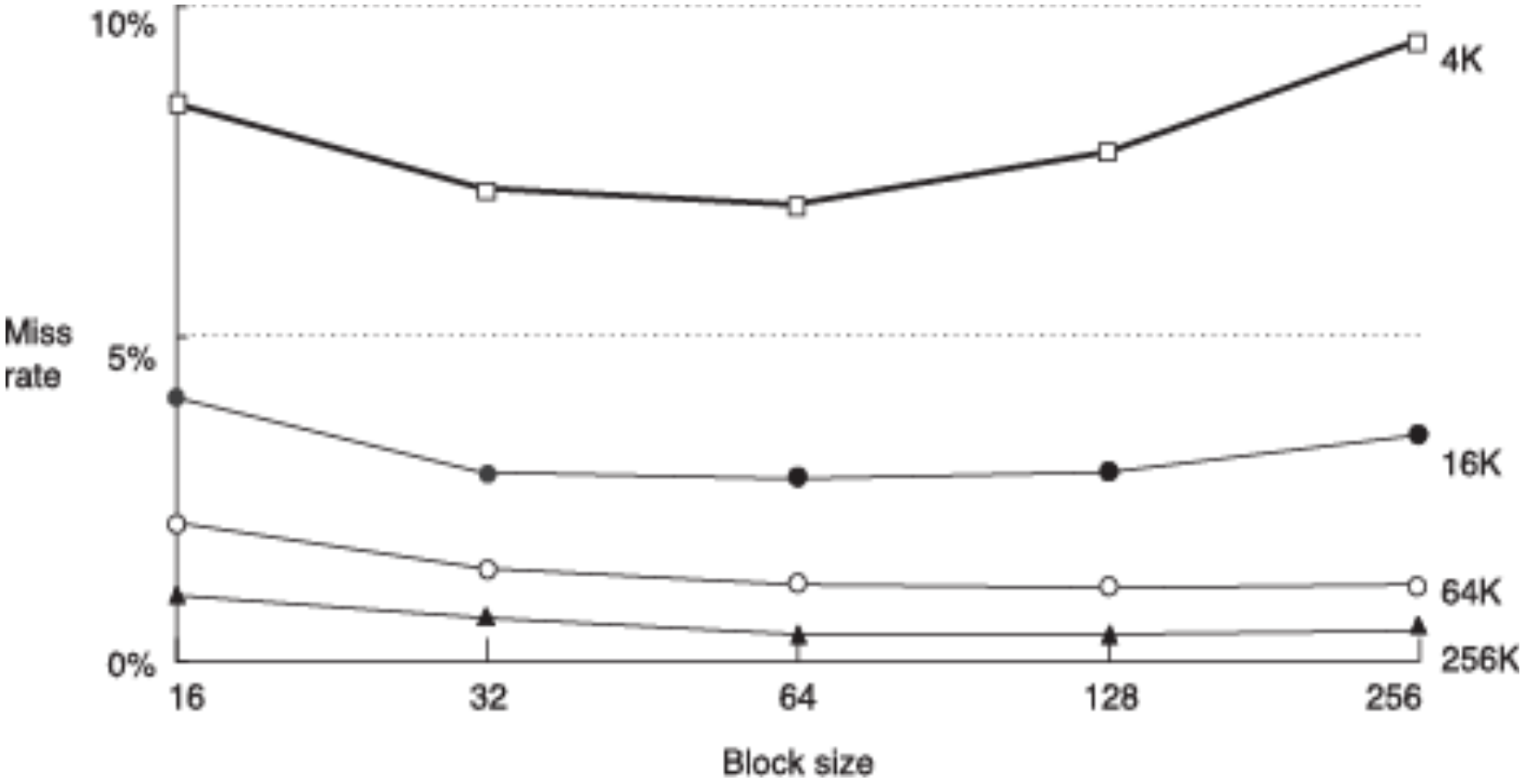
Größerer Caches reduzieren die Anzahl der Capacity Misses

Größerer Assoziativität reduziert Conflict Misses

LEHRE WIKI TEST IN DREI FOLIEN

Adapted from Patterson & Hennessy, *Computer Architecture: A Quantitative Approach*

Miss Rate



Höhere Blockgröße reduziert die Anzahl der Compulsory Misses

Höhere Blockgröße erhöht die Anzahl der Conflict Misses

Caching Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Was für Daten werden im Cache gespeichert?
  - Vor kurzer Zeit genutzte Datenpunkte (zeitliche Lokalität)
  - Daten drumherum (örtliche Lokalität, mit höherer Blockgröße)
- Wie werden Daten gefunden?
  - Assoziativitätsmenge “Set” wird aus der Adresse bestimmt
  - Das Wort im Block wird auch aus der Adresse bestimmt
  - In Mengenassoziativen Caches gibt es hierzu viele Ansätze
- Welche Daten werden ersetzt?
  - Ersatz des am längsten ungenutzen Eintrags (Least-recently used way in the set)

LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

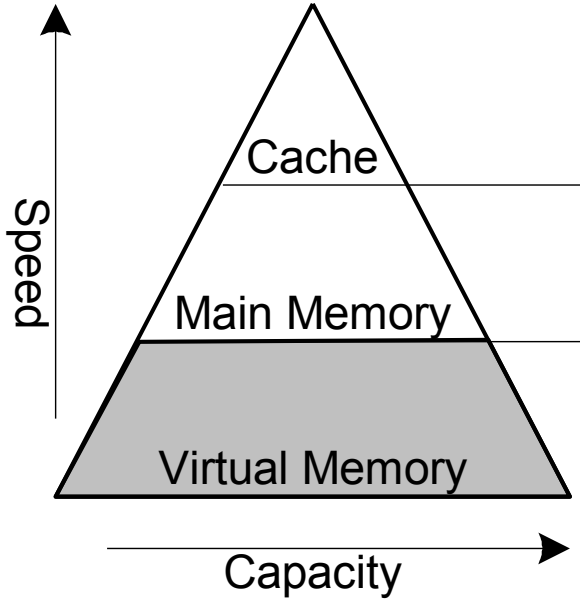
Multilevel Caches

- Wir haben gelernt:
  - Größerer Caches reduzieren die Anzahl der Capacity Misses
  - Größerer Caches sind teuerer und haben bei gleichem Preis meist langsamere Zugriffszeiten.
- Daher: Ausbau einer Hierarchie von Caches!
  - Level 1: klein und schnell (e.g. 16 KB, 1 cycle)
  - Level 2: größer und langsamer (e.g. 256 KB, 2-6 cycles)
- Heute oft noch mehr Levels.

Virtual Memory

- Eine “Illusion” eines größeren Hauptspeichers!
  - Langsamer als Hauptspeichers.
  - Billiger als Hauptspeichers
- Hauptspeichers (implementiert mit DRAM) ist der Cache der Festplatte.

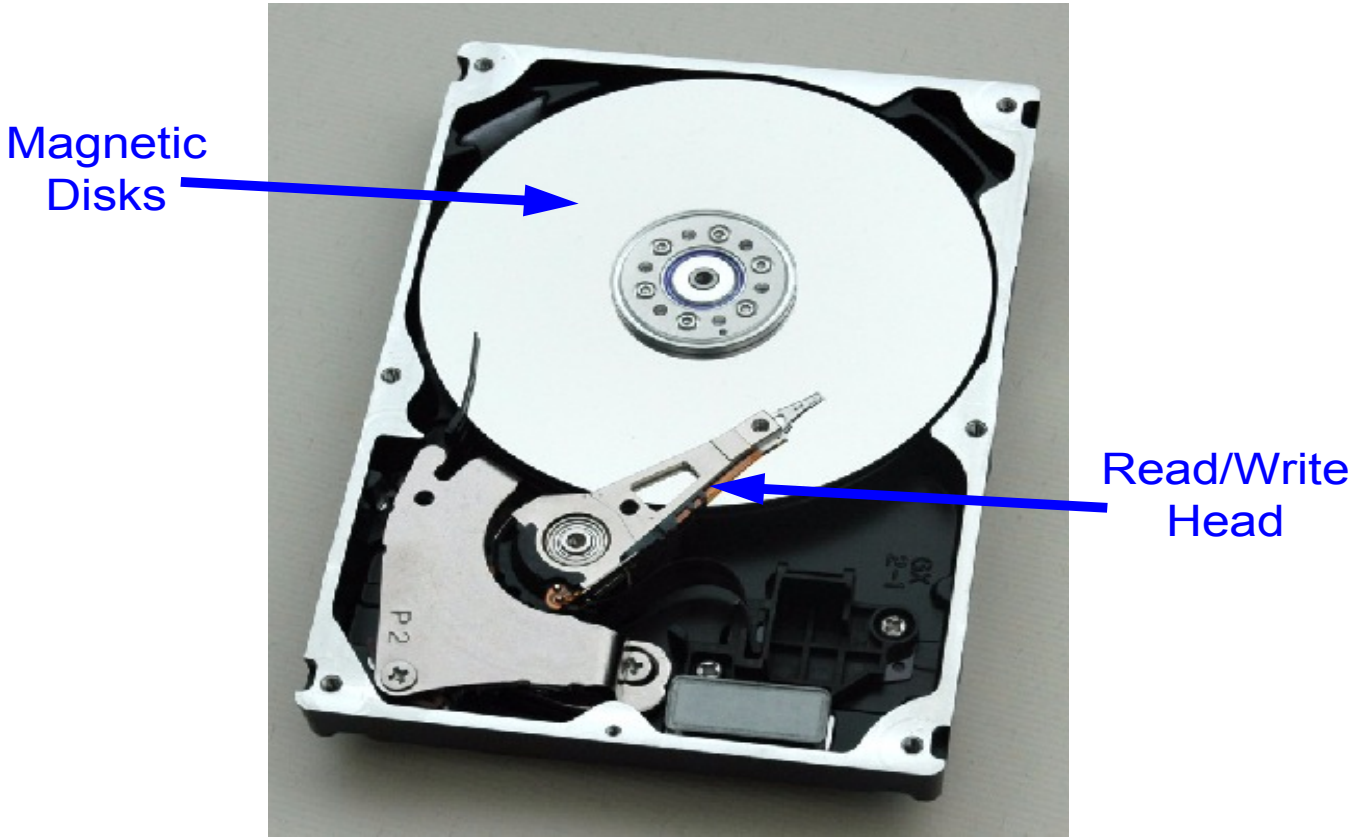
Speicherhierarchie: Beispiele für Ebenen



Technology	cost / GB	Access time
SRAM	~ $10,000	~ 1 ns
DRAM	~ $100	~ 100 ns
Hard Disk	~ $1	~ 10,000,000 ns

- **Physikalischer Hauptspeicher:** DRAM
  - Langsamer, grösser und günstiger als der Cache
- **Virtueller Hauptspeicher:** Festplatte
  - Langsamer, grösser und günstiger als der Hauptspeicher

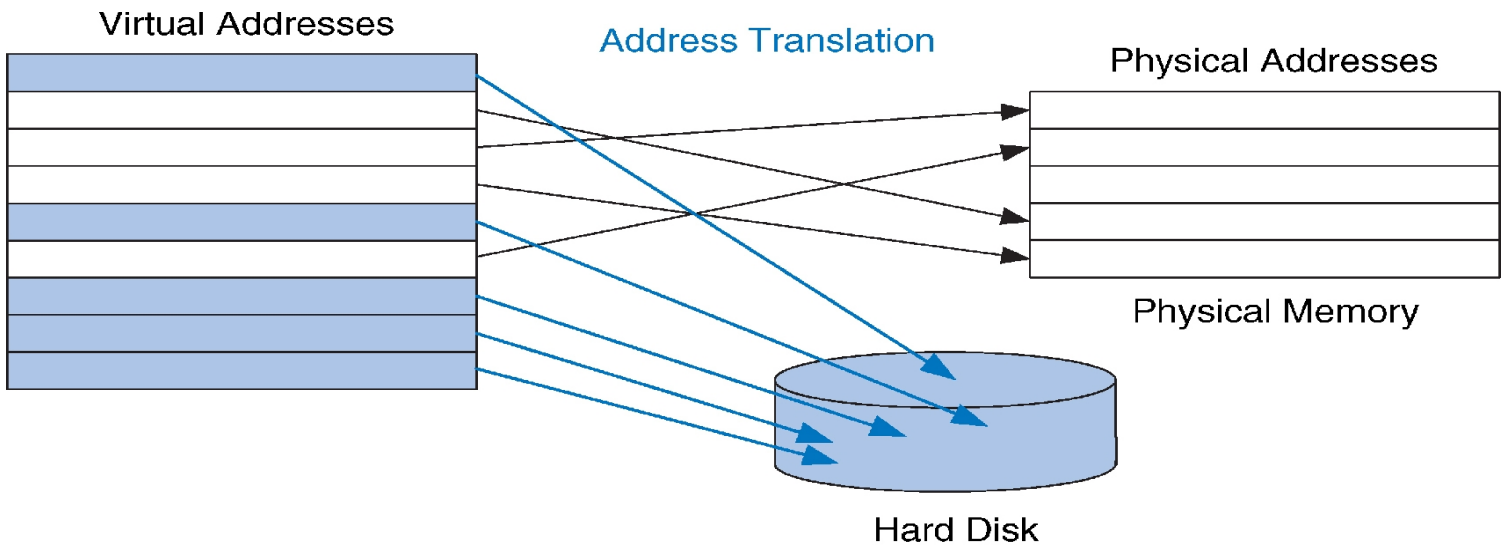
Festplatten



Auffinden eines Ortes benötigt Millisekunden und keine Nanosekunden!

Virtueller Hauptspeicher

- Jedes Programm verwendet ***Virtuelle Adressen***
  - Der gesammte virtuelle Adressraum ist auch auf der Festplatte gespeichert.
  - Ein Teil des virtuellen Adressraums ist im Hauptspeicher.
  - Die CPU übersetzt virtuelle Addressen in ***Physikalische Adressen***
  - Fehlende Daten werden von der Festplatte in den Speicher geholt.
  - **Fast wie beim Cache!**



© 2007 Elsevier, Inc. All rights reserved

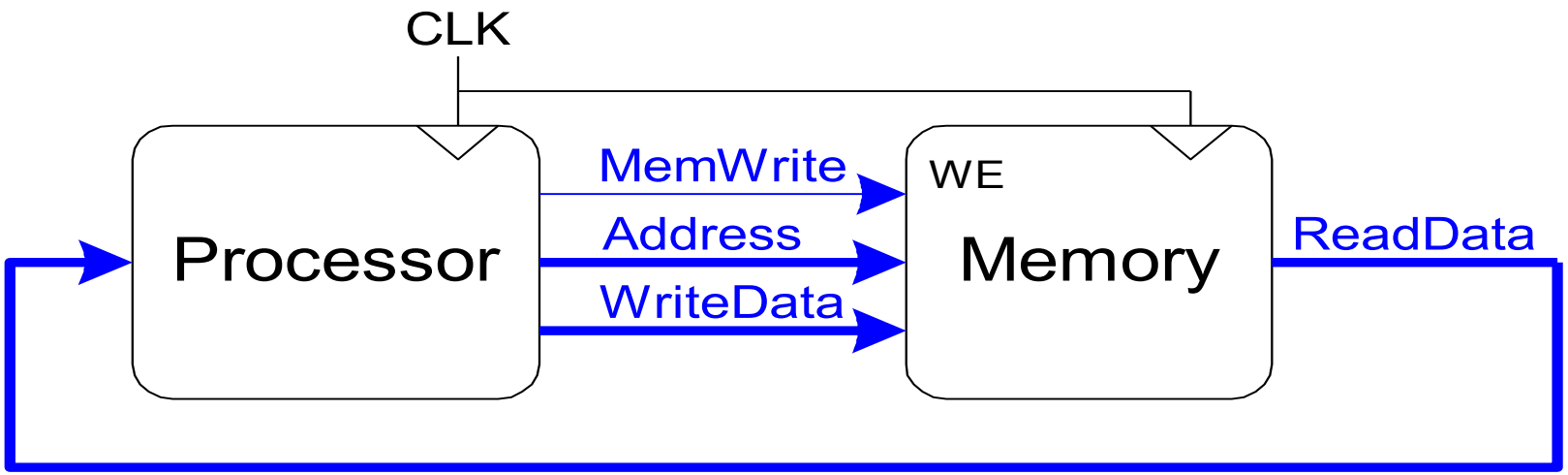
Memory Protection

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jedes Programm hat seine eigene Zuordnung von virtuellen auf physikalische Adressen
  - Zwei Programme können diegleiche virtuelle Adresse für unterschiedliche Daten nutzen.
  - Beide Programme wissen nichts über einander.
  - ***Memory protection*** schützt die virtuellen Speicher beider Programme voreinander.

- ***Vielmehr Details in GDI-3! Hier nur ein kurzer Einblick!***

The Memory Interface



Memory-Mapped Eingabe/Ausgabe Geräte

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Prozessor greifft auf Eingabe/Ausgabe Geräte (I/O Devices), z.B. Tastaturen, Monitore, Drucker, wie auf Speicher zu!
- Jedes I/O Device hat eine oder mehrere Adressen.
- Wenn dieses Gerät angesprochen wird, werden Daten aus dem I/O Device in den Hauptspeicher gelesen oder aus diesem auf das I/O Device geschrieben.
- Ein Teil des Adressraums ist für I/O Device reserviert (z.B., Adressen 0xFFFF0000 to 0xFFFFFFFF)

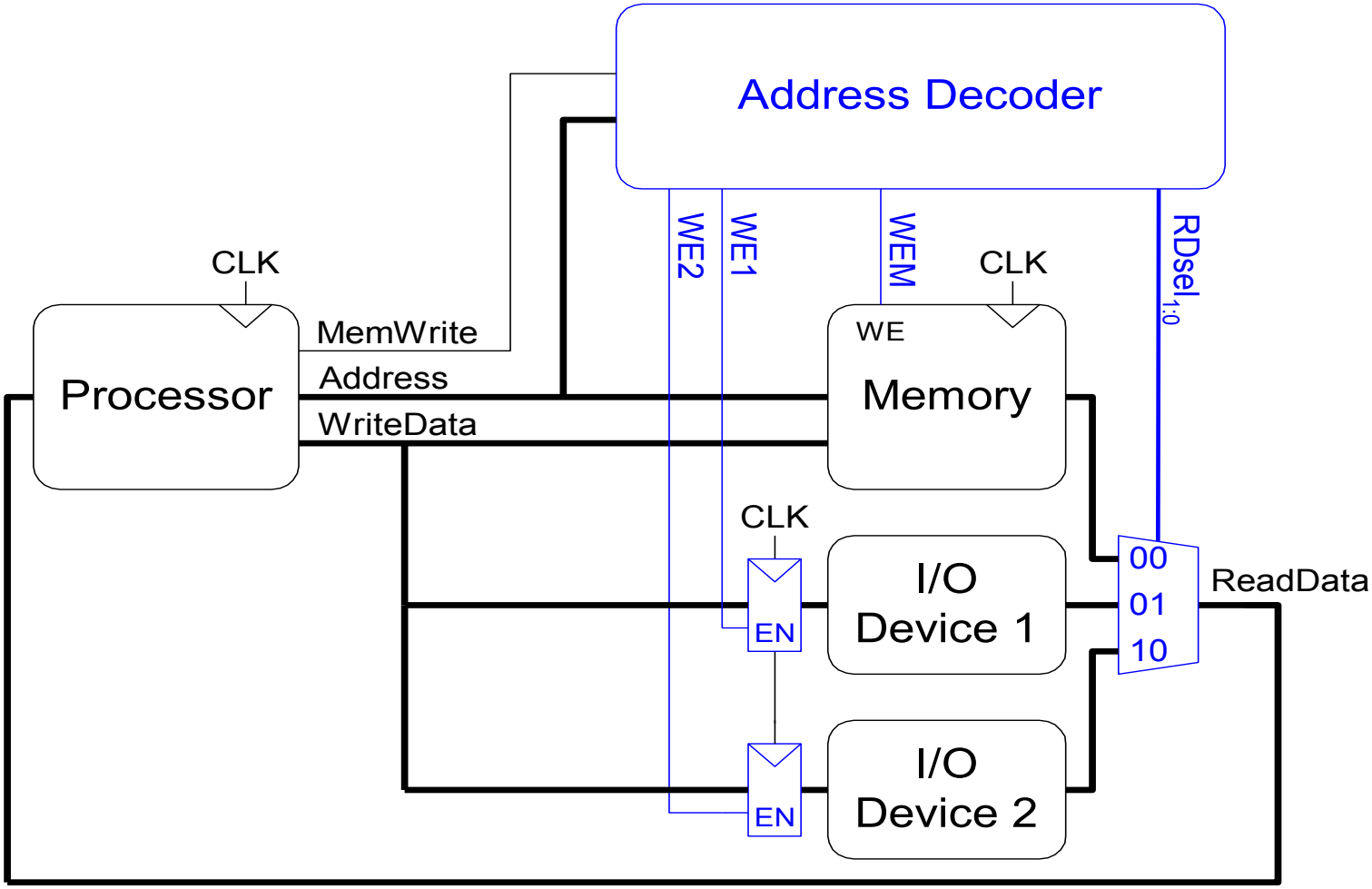
Memory-Mapped I/O Hardware

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Address Decoder:**
  - Stellt von der Adresse aus fest, welcher Speicher oder welches I/O Device mit dem Prozessor kommuniziert.
- **I/O Registers:**
  - Speichern Daten aus den I/O Devices
- **ReadData/WriteData Multiplexer:**
  - Wählt zwischen Hauptspeicher und I/O Devices als Datenquelle für den Prozessor

LEHRE WIKI TEST IN DREI FOLIEN

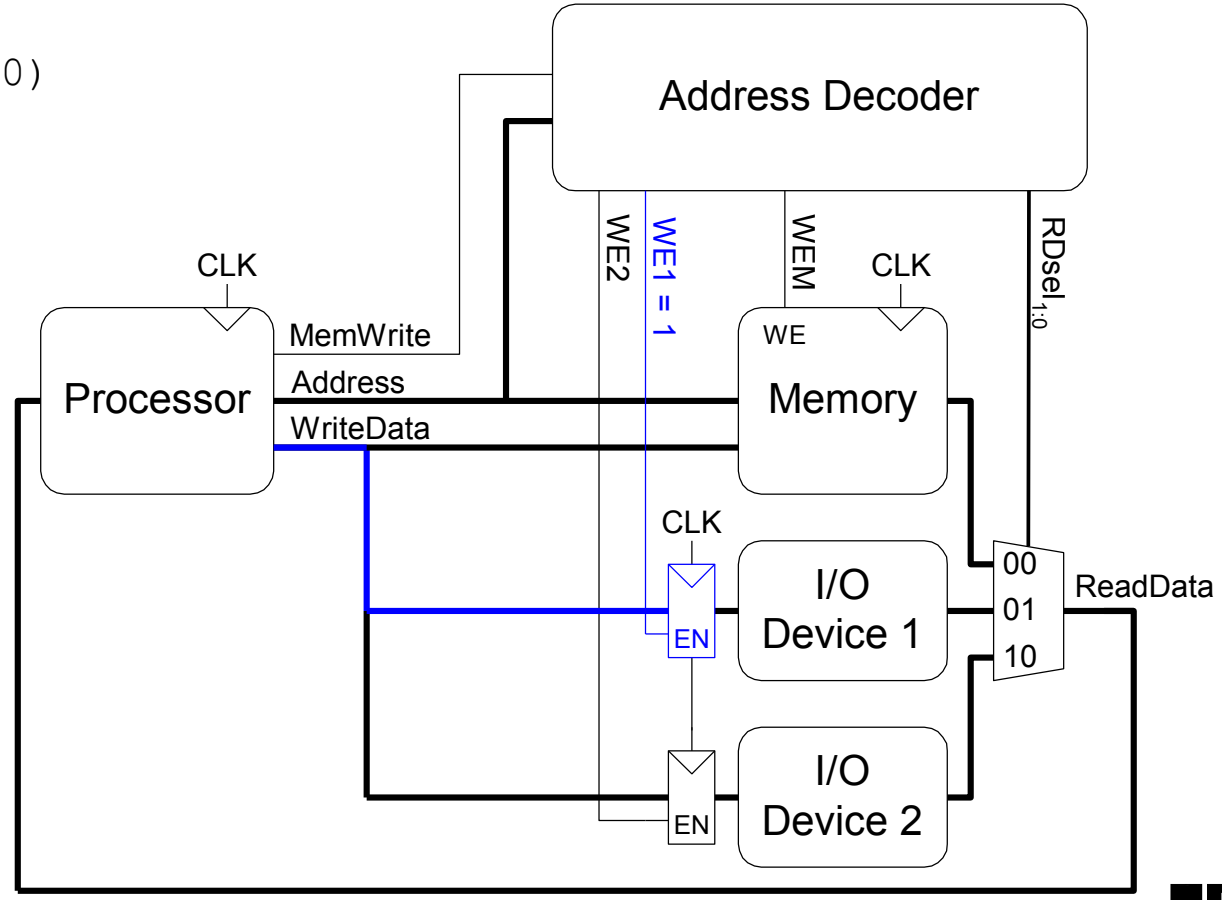
Memory-Mapped I/O Hardware



Memory-Mapped I/O Code

- **Schreiben von 42 zum I/O Device 1 (0xFFFFFFF4)**

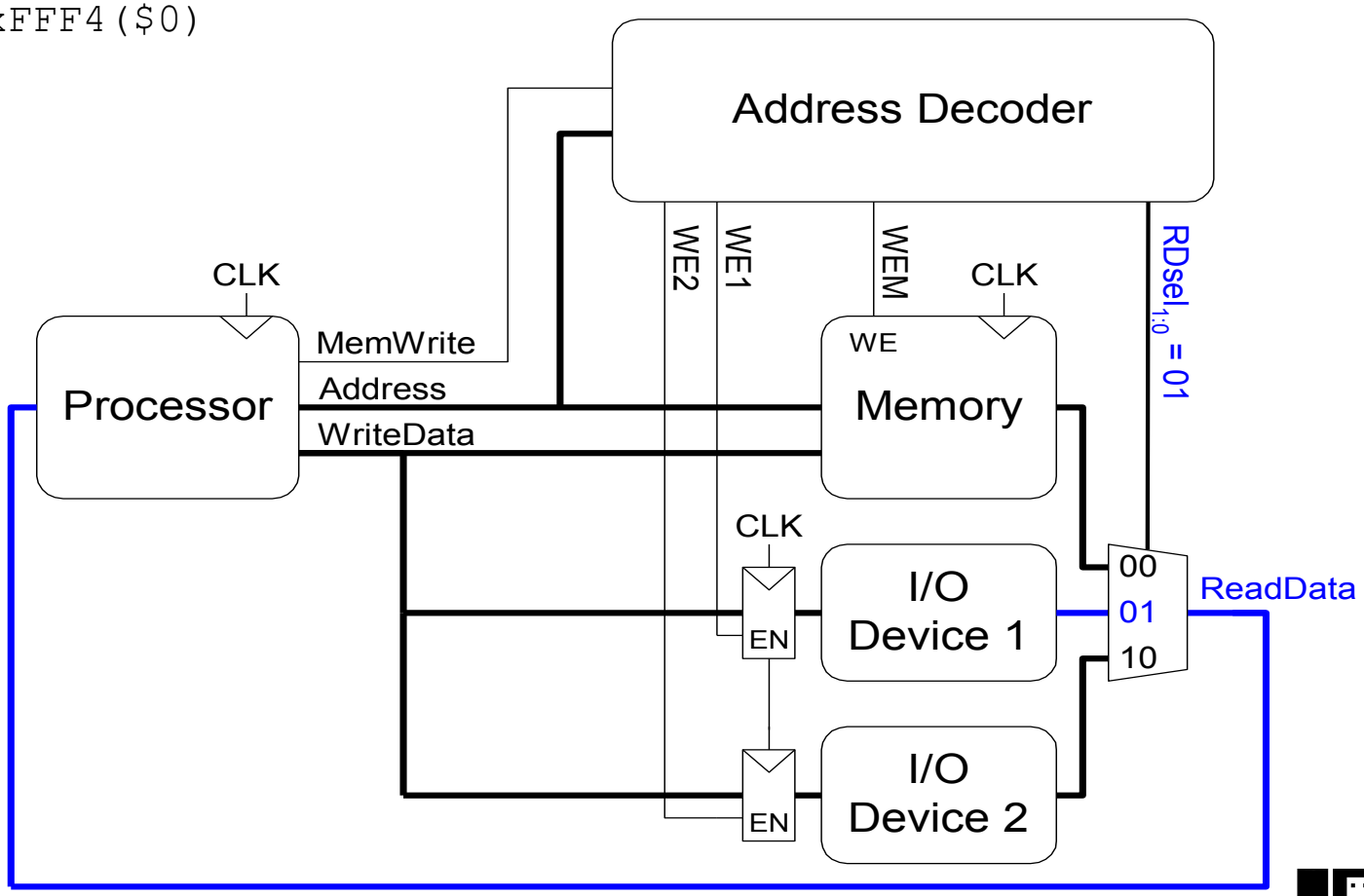
```
addi $t0, $0, 42
sw $t0, 0xFFF4($0)
```



Memory-Mapped I/O Code

- **Lesen von I/O Device 1 und Abspeicherung in Register `$t3`**

```
lw $t3, 0xFFF4($0)
```



LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

TGDI Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ihr habe jetzt die wichtigsten technischen Grundlagen der Informatik verstanden:
  - Kombinatorischer und sequentieller Logik
  - Digitale Grundbausteinen: adders, ALUs, multiplexers, decoders, memories, etc.
  - HDLs: Verilog, VHDL und SystemVerilog!
  - Assembler und Computer Architektur
  - Prozessorentwurf und Microarchitektur
  - Speichersysteme
- Der Stoff ist wichtiger als das Verständnis von Autos, Kenntnis von Goethe, …
- Ihr habt den Schlüssel zu allen technischen Themen!
  - Es liegt an Euch, das “Tor zur digitalen Welt zu öffnen” und diese zu verändern!
  - Ich erwarte Grosses von Euch!

Resttermine

- **Jan steht Frage und Antwort:** Mi, 4. Feb. 2015 (09:50-11:30)
- **Fällt aus** – Jan ist am in Pittsburgh, Pensylvania! Do, 5. Feb. 2015
- **Probeklausur:** Mi, 11. Feb. 2015 (09:50-11:30)
- **Besprechung der Probeklausur:** Do, 12. Feb. 2015 (16:15-17:55)