

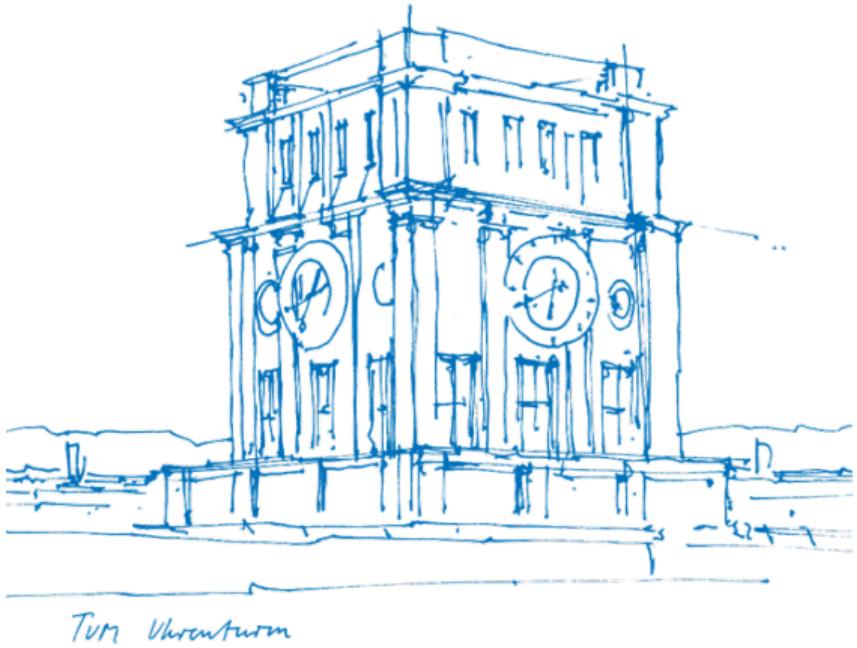
Übung 05: Caches

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

14. November 2025



Keine Garantie für die Richtigkeit der Tutorfolien.
Bei Unklarheiten/Unstimmigkeiten haben VL/ZÜ-Folien recht!

Motivation

- Zugriffe auf Hauptspeicher (\equiv RAM) sind **extrem** langsam. Lösung: Caches
- „Zwischenstation“ zwischen Registern (sehr schnell, sehr klein) und Hauptspeicher (sehr langsam, sehr groß)
- Idee: Häufig genutzte Daten im Cache zwischenspeichern, der Rest wird bei Bedarf aus dem Hauptspeicher geholt
- Die Position eines Datums im Cache wird durch die Adresse des Zugriffs bestimmt
- Heutzutage meist L1/L2/L3-Caches: Caches aufsteigender Größe, aber absteigender Zugriffszeit

Terminologie

- **Hit:** Datum liegt im Cache
- **Miss:** Datum nicht im Cache, muss erst aus Hauptspeicher geholt werden
→ Ziel: möglichst hohe **Hitrate** (Hits/Anfragen), d.h. häufig genutzte Daten im Cache

Terminologie

- **Hit:** Datum liegt im Cache
- **Miss:** Datum nicht im Cache, muss erst aus Hauptspeicher geholt werden
→ Ziel: möglichst hohe **Hitrate** (Hits/Anfragen), d.h. häufig genutzte Daten im Cache
- **zeitliche** Lokalität¹: Zugriff auf $x \rightarrow$ Zugriff auf x in Zukunft wahrscheinlicher
- **räumliche** Lokalität: Zugriff auf $x \rightarrow$ Zugriff auf Daten in der Nähe wahrscheinlicher
→ Konsequenz: Cachen von ganzer Cachezeile

¹ Definition der Lokalitätsprinzipien ist eine beliebte Klausuraufgabe!

Direktabbildender Cache²

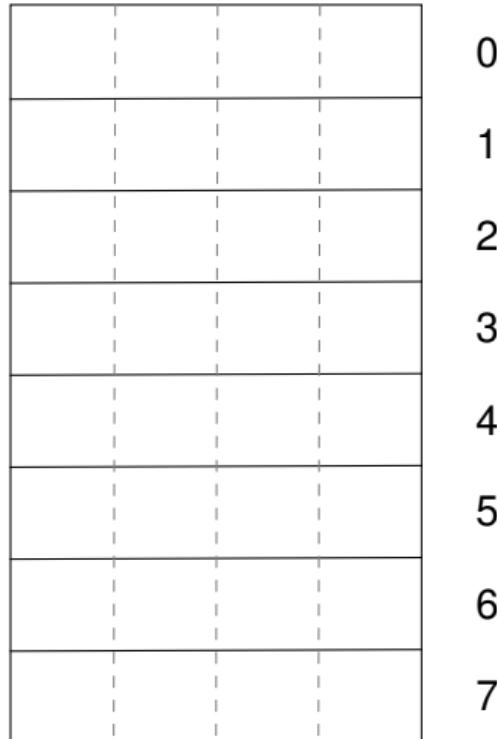
- direkte (surjektive) Abbildung von
Hauptspeicheradresse \mapsto Cacheadresse
- jedes Datum aus dem Hauptspeicher
kann nur in *eine* bestimmte Cachezeile
gecached werden

²Englisch: Direct Mapped Cache

Direktabbildender Cache²

Index

- direkte (surjektive) Abbildung von
Hauptspeicheradresse \mapsto Cacheadresse
- jedes Datum aus dem Hauptspeicher
kann nur in *eine* bestimmte Cachezeile
gecached werden
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen
zu je 4 Byte
 - 1. Anzahl Offset-Bits:
 - 2. Anzahl Index-Bits:
 - 3. Anzahl Tag-Bits:

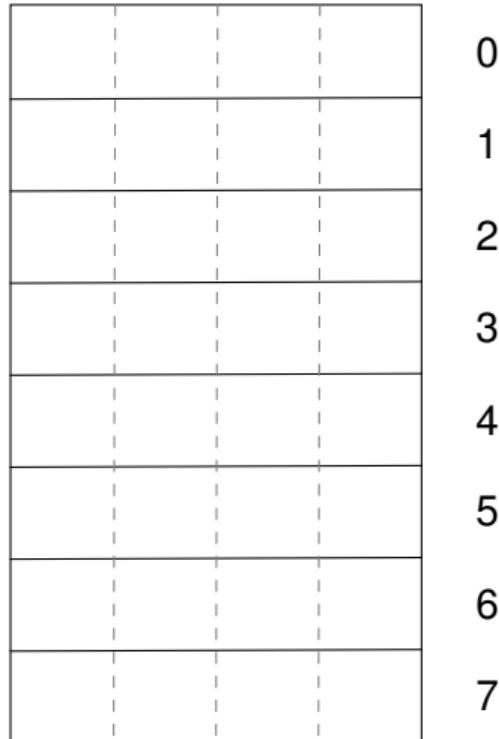


²Englisch: Direct Mapped Cache

Direktabbildender Cache²

Index

- direkte (surjektive) Abbildung von
Hauptspeicheradresse \mapsto Cacheadresse
- jedes Datum aus dem Hauptspeicher
kann nur in *eine* bestimmte Cachezeile
gecached werden
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen
zu je 4 Byte
 1. Anzahl Offset-Bits: $\log_2(4) = 2$
 2. Anzahl Index-Bits: $\log_2(8) = 3$
 3. Anzahl Tag-Bits: $16 - 3 - 2 = 11$



²Englisch: Direct Mapped Cache

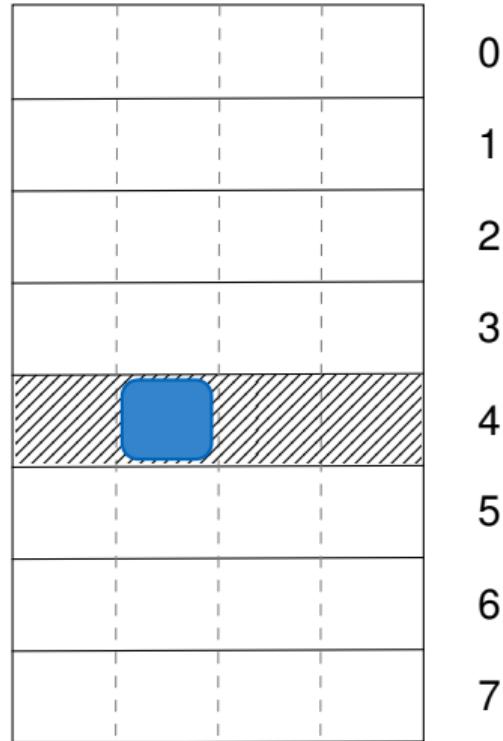
Direktabbildender Cache²

- 0x6291 =

Tag	Index	Offset
01100010100	100	01

11 bit 3 bit 2 bit

Es wird immer die gesamte Cachezeile ausgetauscht!



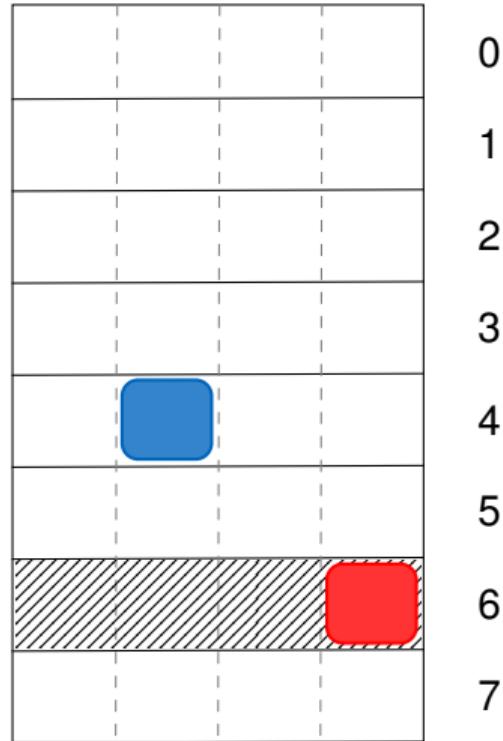
²Englisch: Direct Mapped Cache

Direktabbildender Cache²

● $0x6291 = \begin{array}{c|c|c} \text{Tag} & \text{Index} & \text{Offset} \\ \hline 01100010100 & 100 & 01 \\ \hline 11\text{ bit} & 3\text{ bit} & 2\text{ bit} \end{array}$

● $0xccdb = \begin{array}{c|c|c} \text{Tag} & \text{Index} & \text{Offset} \\ \hline 11001100110 & 110 & 11 \\ \hline 11\text{ bit} & 3\text{ bit} & 2\text{ bit} \end{array}$

Es wird immer die gesamte Cachezeile ausgetauscht!



²Englisch: Direct Mapped Cache

Vollassoziativer Cache³

- Daten werden „wo gerade Platz ist“ gecached
- jedes Datum aus dem Hauptspeicher kann in eine *beliebige* Cachezeile gecached werden

³Englisch: Fully Associative Cache

Vollassoziativer Cache³

- Daten werden „wo gerade Platz ist“ gecached
- jedes Datum aus dem Hauptspeicher kann in eine *beliebige* Cachezeile gecached werden
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen zu je 4 Byte
 1. Anzahl Offset-Bits:
 2. Anzahl Index-Bits:
 3. Anzahl Tag-Bits:

³Englisch: Fully Associative Cache

Vollassoziativer Cache³

- Daten werden „wo gerade Platz ist“ gecached
- jedes Datum aus dem Hauptspeicher kann in eine *beliebige* Cachezeile gecached werden
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen zu je 4 Byte
 1. Anzahl Offset-Bits: $\log_2(4) = 2$
 2. Anzahl Index-Bits: 0
 3. Anzahl Tag-Bits: $16 - 2 - 0 = 14$

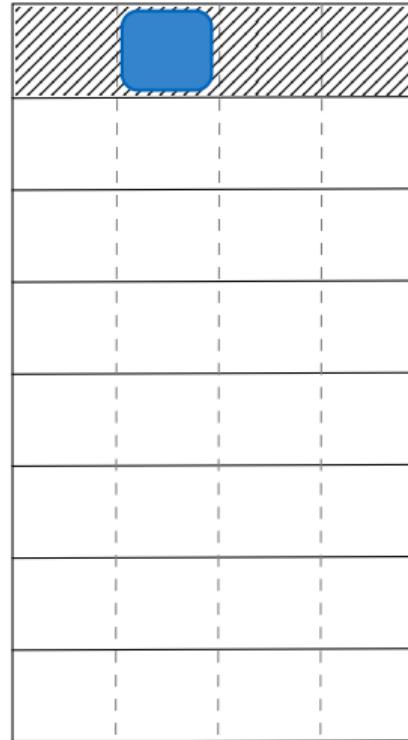
³Englisch: Fully Associative Cache

Vollassoziativer Cache³

- 0x6291 =

01100010100100	01
14 bit	2 bit

Es wird immer die gesamte Cachezeile ausgetauscht!



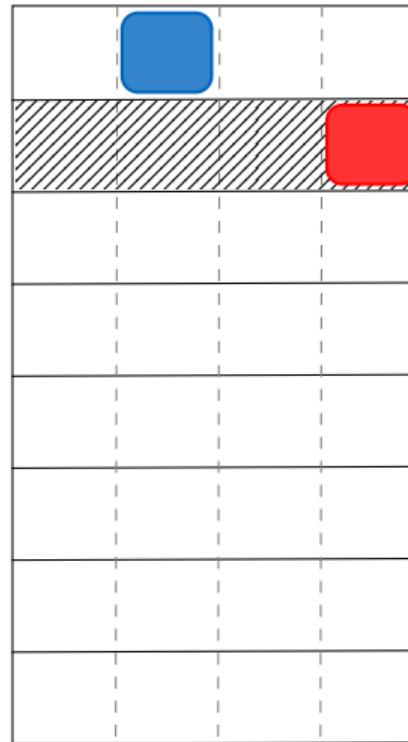
³Englisch: Fully Associative Cache

Vollassoziativer Cache³

Tag Offset
● $0x6291 = \boxed{01100010100100}$ $\boxed{01}$
 14 bit 2 bit

Tag Offset
● $0xccdb = \boxed{11001100110110}$ $\boxed{11}$
 14 bit 2 bit

Es wird immer die gesamte Cachezeile ausgetauscht!



³Englisch: Fully Associative Cache

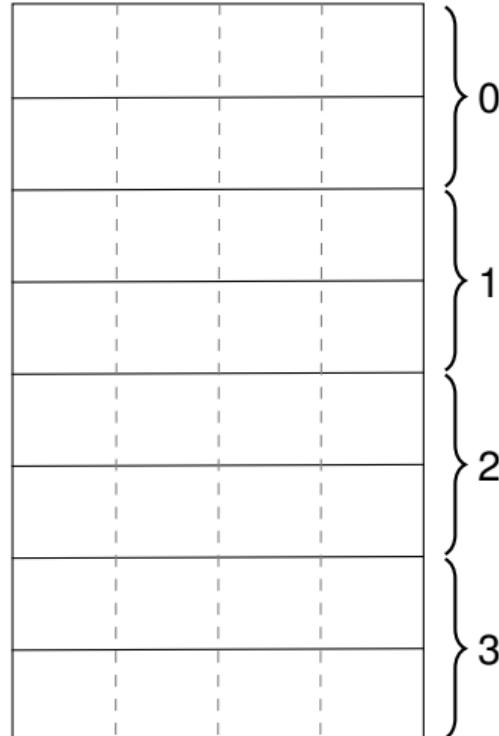
Mengenassoziativer Cache⁴

- Aufteilung des Cache in Sets zu je n Cachezeilen
- jedes Datum aus dem Hauptspeicher kann nur in *einem* bestimmtem Set gecached werden, im Set aber in einer *beliebigen* Cachezeile stehen

⁴Englisch: Set Associative Cache

Mengenassoziativer Cache⁴

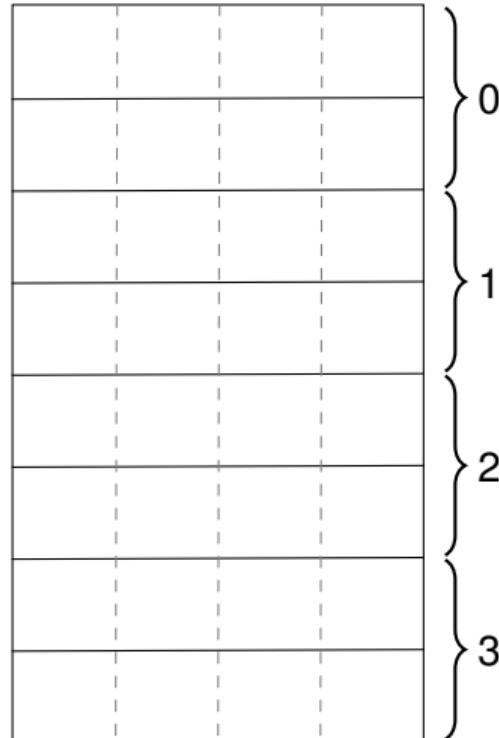
- Aufteilung des Cache in Sets zu je n Cachezeilen
- jedes Datum aus dem Hauptspeicher kann nur in *einem* bestimmtem Set gecached werden, im Set aber in einer *beliebigen* Cachezeile stehen
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen zu je 4 Byte, 2-fach assoziativ
 1. Anzahl Cache-Sets:
 2. Anzahl Offset-Bits:
 3. Anzahl Index-Bits:
 4. Anzahl Tag-Bits:



⁴Englisch: Set Associative Cache

Mengenassoziativer Cache⁴

- Aufteilung des Cache in Sets zu je n Cachezeilen
- jedes Datum aus dem Hauptspeicher kann nur in *einem* bestimmtem Set gecached werden, im Set aber in einer *beliebigen* Cachezeile stehen
- Beispiel: 16-Bit-Architektur, 8 Cachezeilen zu je 4 Byte, 2-fach assoziativ
 1. Anzahl Cache-Sets: $\frac{8}{2} = 4$
 2. Anzahl Offset-Bits: $\log_2(4) = 2$
 3. Anzahl Index-Bits: $\log_2(4) = 2$
 4. Anzahl Tag-Bits: $16 - 2 - 2 = 12$



⁴Englisch: Set Associative Cache

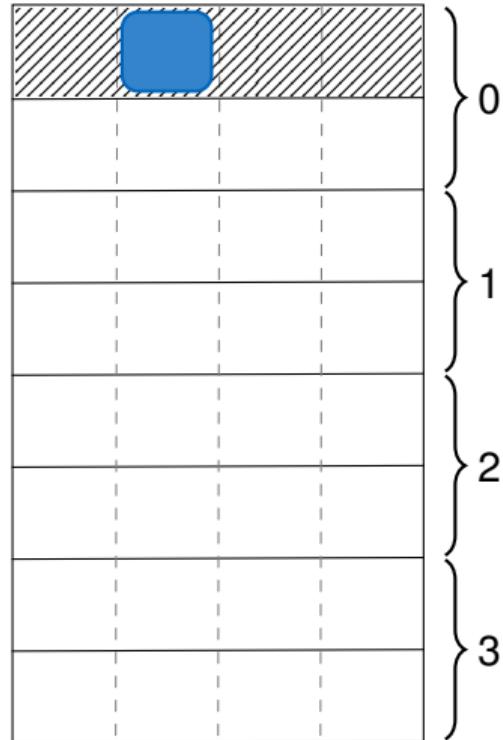
Mengenassoziativer Cache⁴

- 0x6291 =

Tag	011000101001
12 bit	

Index	00	Offset	01
2 bit		2 bit	

Es wird immer die gesamte Cachezeile ausgetauscht!



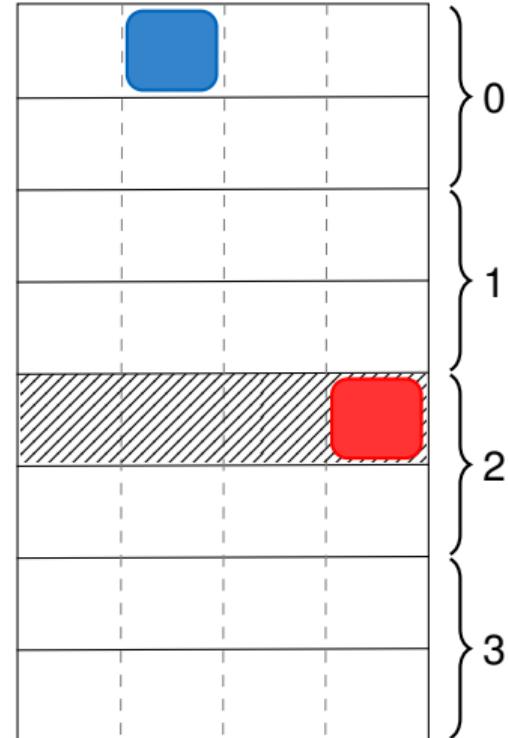
⁴Englisch: Set Associative Cache

Mengenassoziativer Cache⁴

● $0x6291 = \begin{array}{c|cc} \text{Tag} & 011000101001 \\ \hline 12 \text{ bit} & \end{array} \quad \begin{array}{c|c} \text{Index} & \text{Offset} \\ \hline 00 & 01 \\ 2 \text{ bit} & 2 \text{ bit} \end{array}$

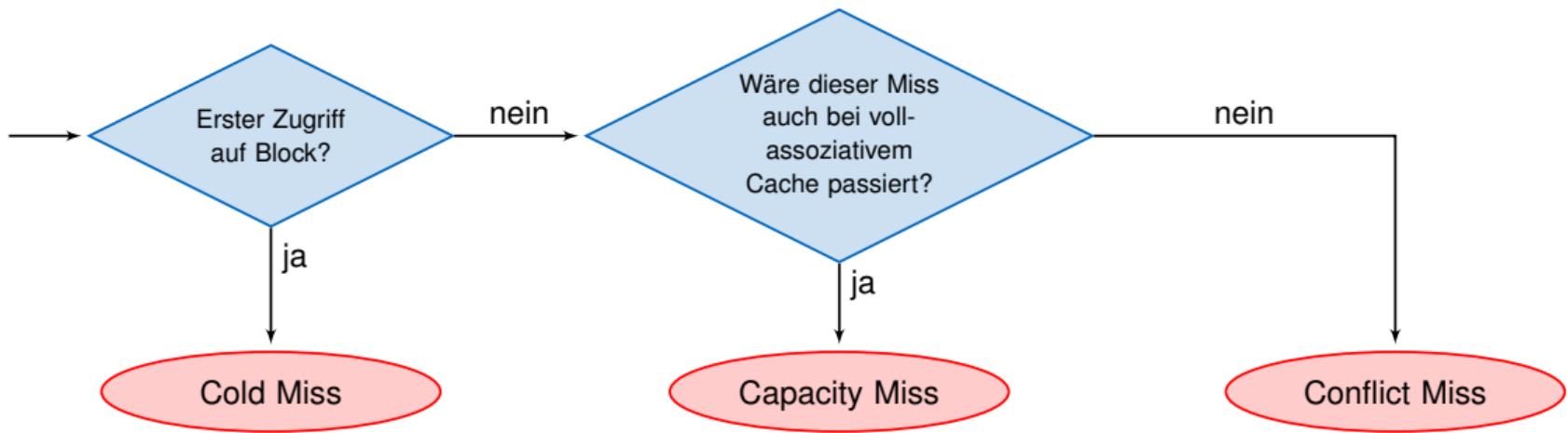
● $0xccdb = \begin{array}{c|cc} \text{Tag} & 110011001101 \\ \hline 12 \text{ bit} & \end{array} \quad \begin{array}{c|c} \text{Index} & \text{Offset} \\ \hline 10 & 11 \\ 2 \text{ bit} & 2 \text{ bit} \end{array}$

Es wird immer die gesamte Cachezeile ausgetauscht!



⁴Englisch: Set Associative Cache

Klassifikation von Misses



Nach Gschoßmann et al., 2023

Fragen?

Links

- Zulip: „ERA Tutorium – Mi-1600-3“ bzw. „ERA Tutorium – Fr-1500-1“
- ERA-Moodle-Kurs
- ERA-Artemis-Kurs
- Wikipedia zu Caches
- Elektronik-Kompendium zu Caches

Übung 05: Caches

Einführung in die Rechnerarchitektur

Niklas Ladurner

School of Computation, Information and Technology
Technische Universität München

14. November 2025

