

Table Lookup Buffer Simulation

Gruppe 420

Technische Universität München

Grundlagenpraktikum Rechnerarchitektur

19. August 2024



Gültigkeit der Masterfolien

Dieser Folienmaster gilt bei offiziellen Präsentationen im Rahmen der TUM.
Es ist darauf zu achten, dass wir uns in einem durchgängigen Layout präsentieren.

Abweichungen vom vorgegebenen Layout bitte auf ein Minimum reduzieren.

Grundlage der Masterfolien

Als Grundlage dient der Corporate Design Style Guide der TUM.

Die Präsentationsvorlage ist auf gute Lesbarkeit und klare Darstellung von Informationen optimiert.

Hier steht eine Überschrift max. 2-zeilig

Als Grundlage dient der Corporate Design Style Guide der TUM.

Die Präsentationsvorlage ist auf gute Lesbarkeit und klare Darstellung von Informationen optimiert.

Schrift

Das Grundprinzip ist, Informationen bestmöglich zu transportieren. Dazu muss vor allem die Schrift einheitlich und für alle im Raum lesbar sein.

Schriftart: Arial

Schriftgrößen: 25 | 18 | 14 | 11

Zeilenabstand: 1,15mm

Die Einstellungen sind in den Textfeldern und Textfeldvorlagen dieses ppt-Masters als Standard eingestellt. Bei Diagrammen und Tabellen muss die Schriftgröße ggf. angepasst werden. Für Auszeichnungen im Fließtext kann auch fett markiert werden. Bei großer Distanz bzw. kleinem Präsentationsmedium kann der Schriftgrad notfalls proportional erhöht werden.

Farben

Als erstes soll mit schwarz und weiß gearbeitet werden.

Für Aufwändigere Darstellungen sind Farben mit Bedacht und in möglichst geringem Umfang einzusetzen.
In diesem Folienmaster ist die Farbpalette festgelegt.

Zuerst mit den Primärfarben arbeiten.



Für z.B. komplexe Diagramme stehen noch Sekundärfarben zur Verfügung.



Gering im Einsatz sind die Akzentfarben.



Texte

Kurze und knappe Texte, Fließtexte linksbündig, kein Blocksatz

Beispiel:

Tem soluptam, nisi as verum ereprehendam at acculpa quidisq uissit volupta tusdant utem as etur, odi odis
es doluptiae dem nimaion con nossinctenis pora quam voloria consenimus blabore everfer epeliquo maio
etur.

Bilder - Allgemein

schlichte Darstellung von Informationen

reduzierte Farben

Rahmen und Überlagerungen nach Möglichkeit vermeiden

Aufzählung

Bei kleinen Aufzählungen auf Aufzählungszeichen verzichten und ggf. zusätzliche Leerzeile

Nur die wesentlichen Punkte nennen und Themen auf verschiedene Seiten splitten.

Punkt 1

Punkt 2

Wenn Unterpunkte in einer Aufzählung nötig sind ist ein Einrücken mit – möglich

- Unterpunkt 1
 - Unterpunkt 1
 - Unterpunkt 2

Bei größeren Listen die Standardeinstellung • verwenden

- Unterpunkt 1
- Unterpunkt 2

Bilder

Bildbeschreibung

oberer Bildrand: Begrenzung durch Text

Bilder

Überschrift 2

Hier steht ein einleitender oder beschreibender
Fließtext und nach Wunsch eine Aufzählung

Punkt 1

Punkt 2

Punkt 3

Punkt 4

Bilder

Bildbeschreibung

oberer Bildrand: Begrenzung durch Text

Bilder

Bildbeschreibung

oberer Bildrand: Begrenzung durch Text

Nicht formatfüllende Bilder

Weißer bzw. transparenter Hintergrund
mit genug Freiraum anordnen

Bilder Format füllend - maximale Bildgröße

Nicht Format füllende Bilder

Alternativ mit formatfüllendem Hintergrund: 5 % schwarz

Beschriftungen können zusätzlich neben den Bildern angebracht werden

Bilderklärung

Tabelle – Beispiel 1

Tabelle ohne Farbe und kein Rand

innerer Seitenrand links 0 cm, oben z.B. 0,5 cm (für genug Zeilenabstand innerhalb)

Ø - Strecke	39 km/Tag (14.360 km/Jahr)
Ø - Geschwindigkeit	25 km/h
Ø - Verfügbare Ladezeit	22 h/Tag
Kosten	Kleinwagen mit Verbrennungsmotor
Einsatzgebiet	Stadt und Umland

Tabelle – Beispiel 2

Tabelle mit schwarzem Rand

innerer Seitenrand links 0,15 cm, oben z.B. 0,5 cm (für genug Zeilenabstand innerhalb)

Ø - Strecke	39 km/Tag (14.360 km/Jahr)
Ø - Geschwindigkeit	25 km/h
Ø - Verfügbare Ladezeit	22 h/Tag
Kosten	Kleinwagen mit Verbrennungsmotor
Einsatzgebiet	Stadt und Umland

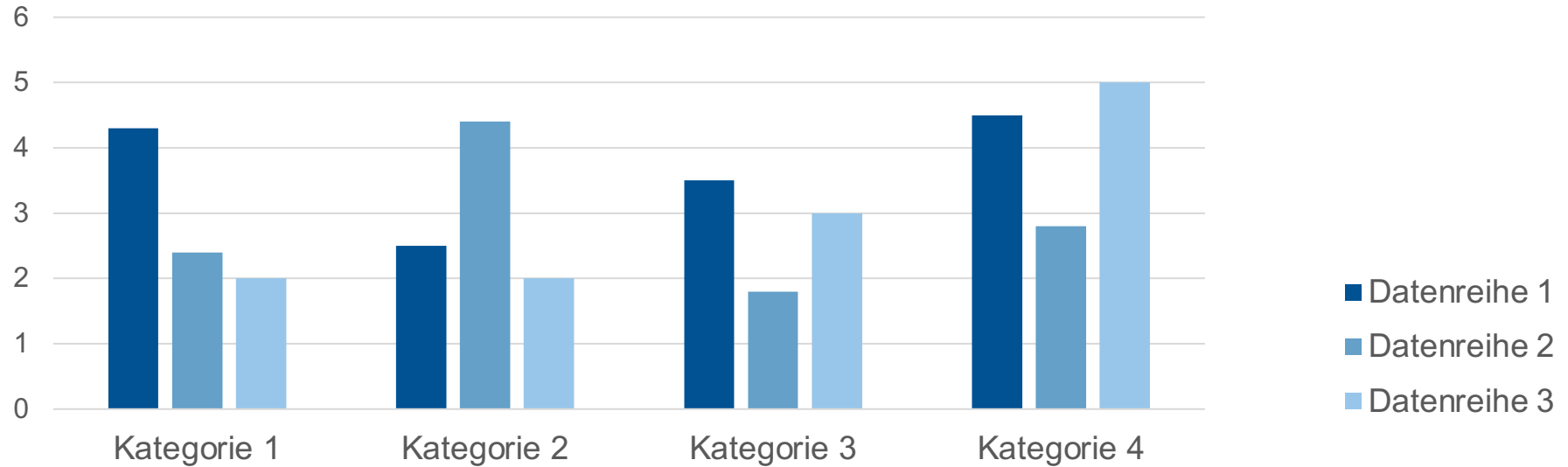
Diagramme – Beispiel 1

Nach Möglichkeit linksbündig bleiben

Unnötige Striche und Balken vermeiden

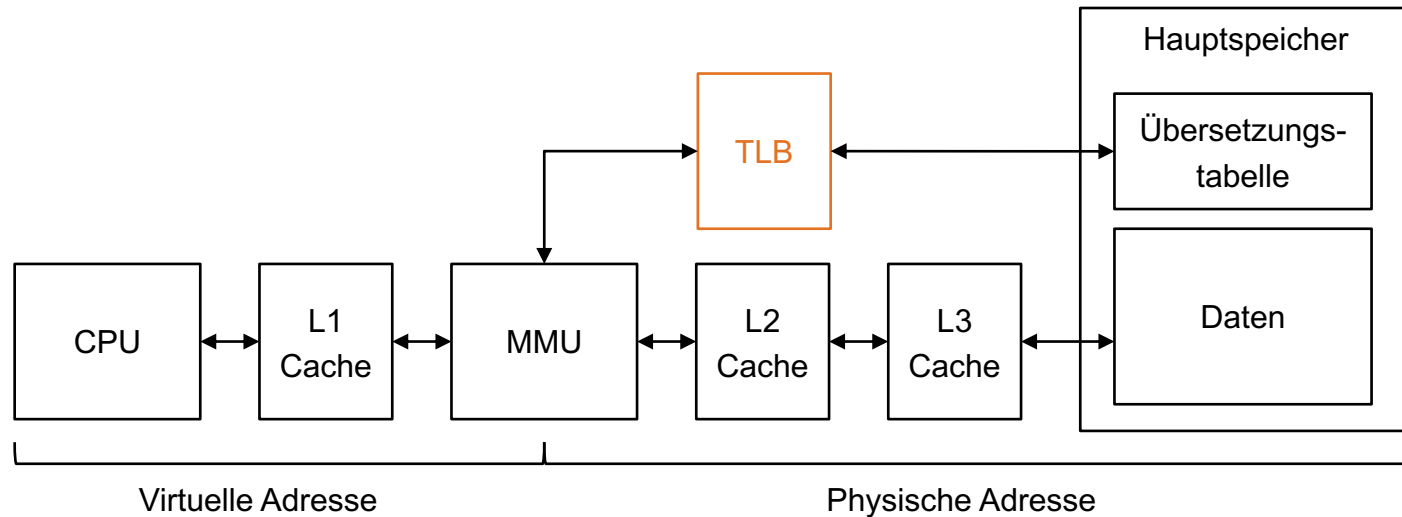


Diagramme



Was ist ein TLB

Levelled Cache Architektur in einem Prozessor



Translation Lookaside Buffer

Technische Universität München

Grundlagenpraktikum Rechnerarchitektur

19. August 2024

Gruppe 151

Lukas Wolf

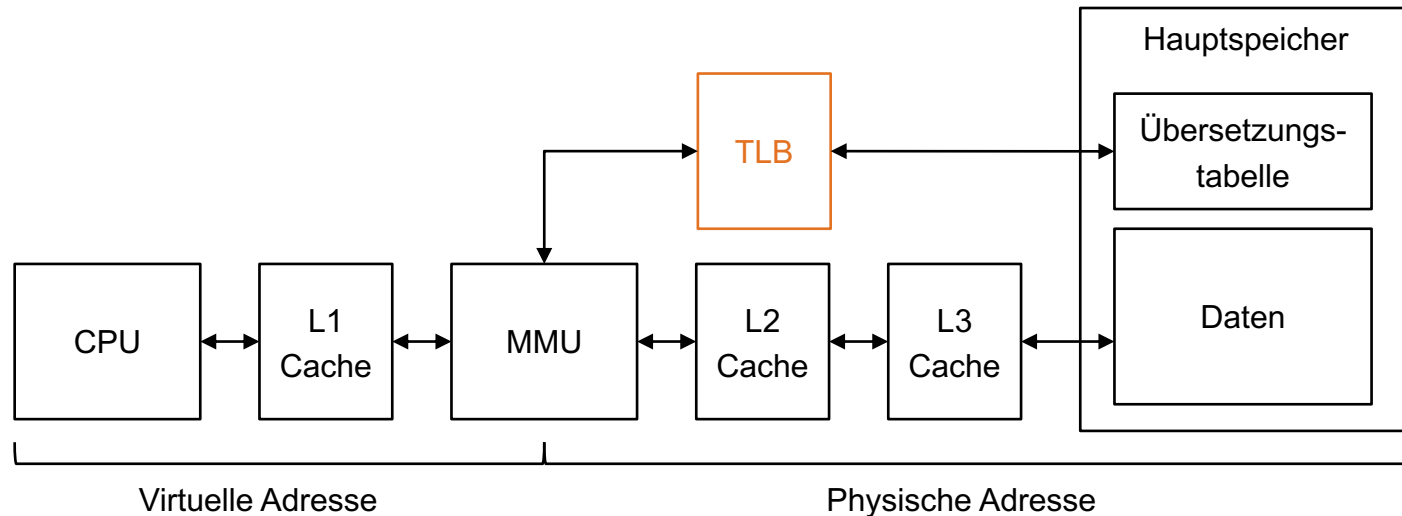
Elena Reinbold Fraire

Jonah Zabel



Was ist ein TLB

Leveled Cache Architektur in einem Prozessor



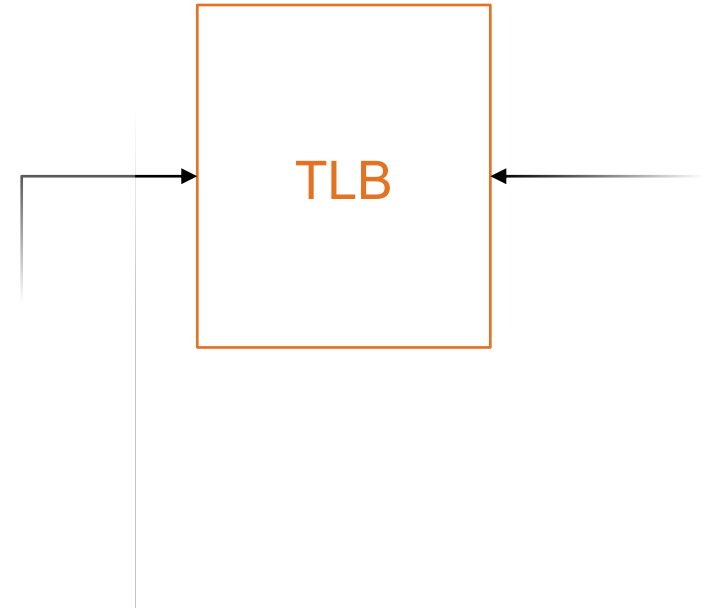
Was ist ein TLB

Simulation mittels SystemC

Rahmenprogramm in C-Programmiersprache

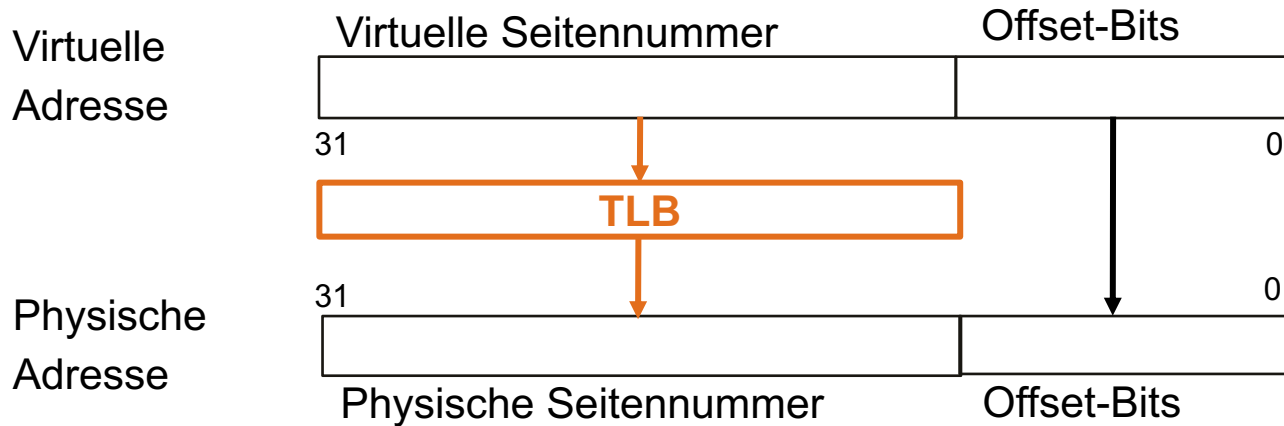
Ziele der Simulation:

- Auswirkungen auf Hauptspeicherzugriffe?
- Welche Bauart liefert optimale Ergebnisse?
- Performanzverbesserung?



Unser Lösungsansatz

Modul 1: Berechnung der Adresse



Unser Lösungsansatz

Modul 1: Berechnung der Adresse

Virtuelle Seitennummer	Physische Seitennummer
0x0	0x0
0x56	0x78
0x783	0x956
0x3	0x7

Unser Lösungsansatz

Modul 1: Berechnung der Adresse

Wichtige Abstraktionen:

- Speicherung der im TLB enthaltenen virtuellen Seitennummern in einem `std::vector`
- Abfangen von Cold Misses durch ein `std::set`
- Page Table durch eine einfache Berechnung ersetzen
- Ergebnis direkt berechnen und danach die benötigten Clock-Zyklen durchlaufen lassen

Unser Lösungsansatz

Modul 2: Speicherung der Daten

Wichtige Abstraktionen:

- Simulation des Speichers durch eine `std::map`
- Auch hier: Ergebnis direkt berechnen und danach die benötigten Clock-Zyklen durchlaufen lassen

Unser Lösungsansatz

Modul 3: Requests abarbeiten

- Synchronisation der beiden Untermodule
- Die Requests in dem übergebenen Array nacheinander abarbeiten → durch for-Schleife abstrahiert
- Anzahl der benötigten Gatter berechnen
- Abstraktion: fast alles → gehört nicht zur Logik, sondern zur Messung der Effizienz

Berechnung der Gatteranzahl

$$(2 * (32 - \log(\text{blocksize})) + 1) * 4 * \text{tlb_size} + 182$$

Anzahl Bits pro Zeile

Anzahl Gatter für die Speicherung von einem Bit

Anzahl Zeilen insgesamt

Anzahl Gatter für einen Subtrahierer

Übliche Größen eines TLB und des Hauptspeichers

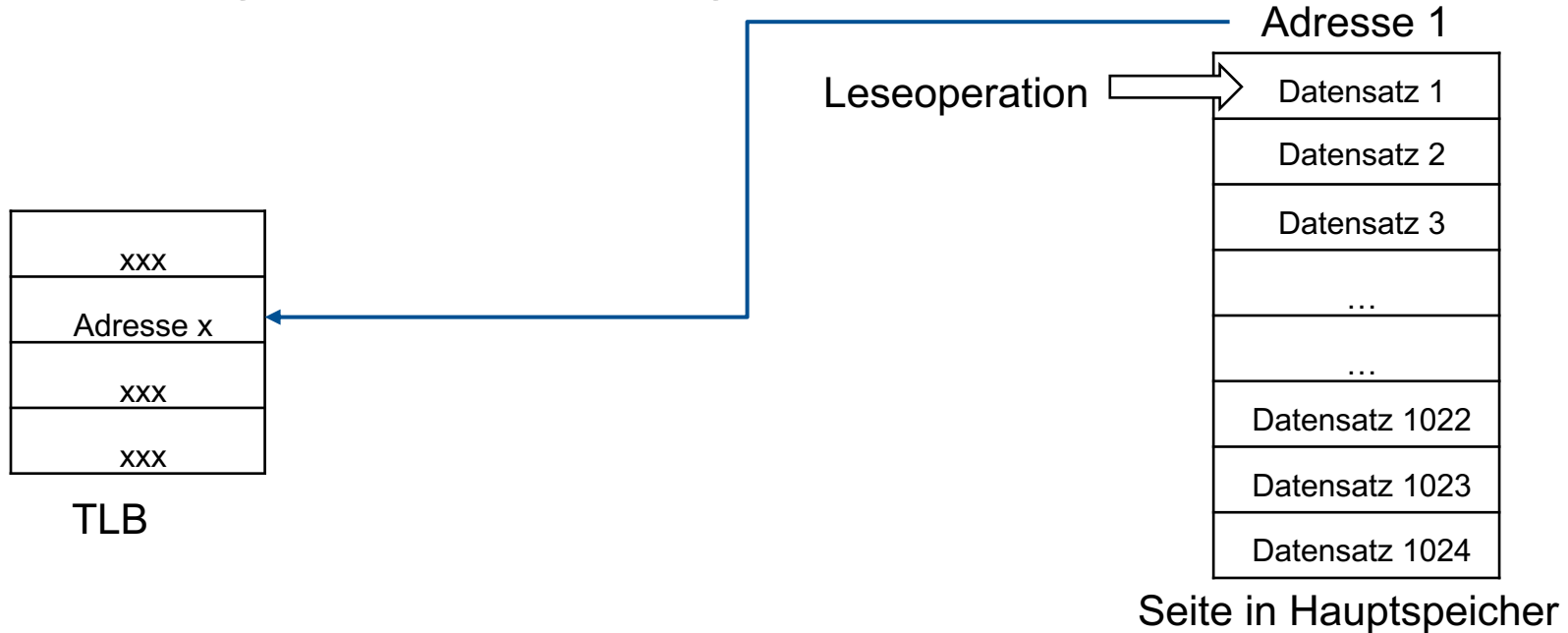
- Größe: 16 – 512 Einträge
- Größe eines geladenen Blockes: 1 Seite (4096 Bytes)
- Hit-Rate: 99 % - 99,9 %
- Assoziativität: Voll- / Mengenassoziativ
- Zugriffszeit: ~ 1 Zyklus

- Zugriffszeit Hauptspeicher: 100 Zyklen

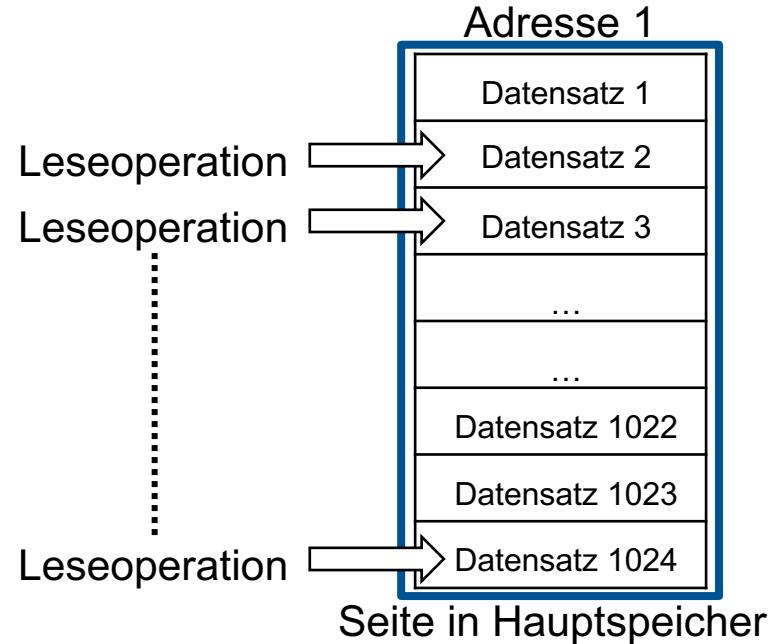
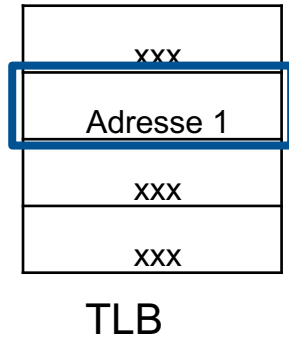
Speicherzugriffsverhalten über eine verkettete Liste

- Auswirkung der Größe des TLB auf die Auswirkungszeit
- Verkettete Liste: Verweis auf Daten

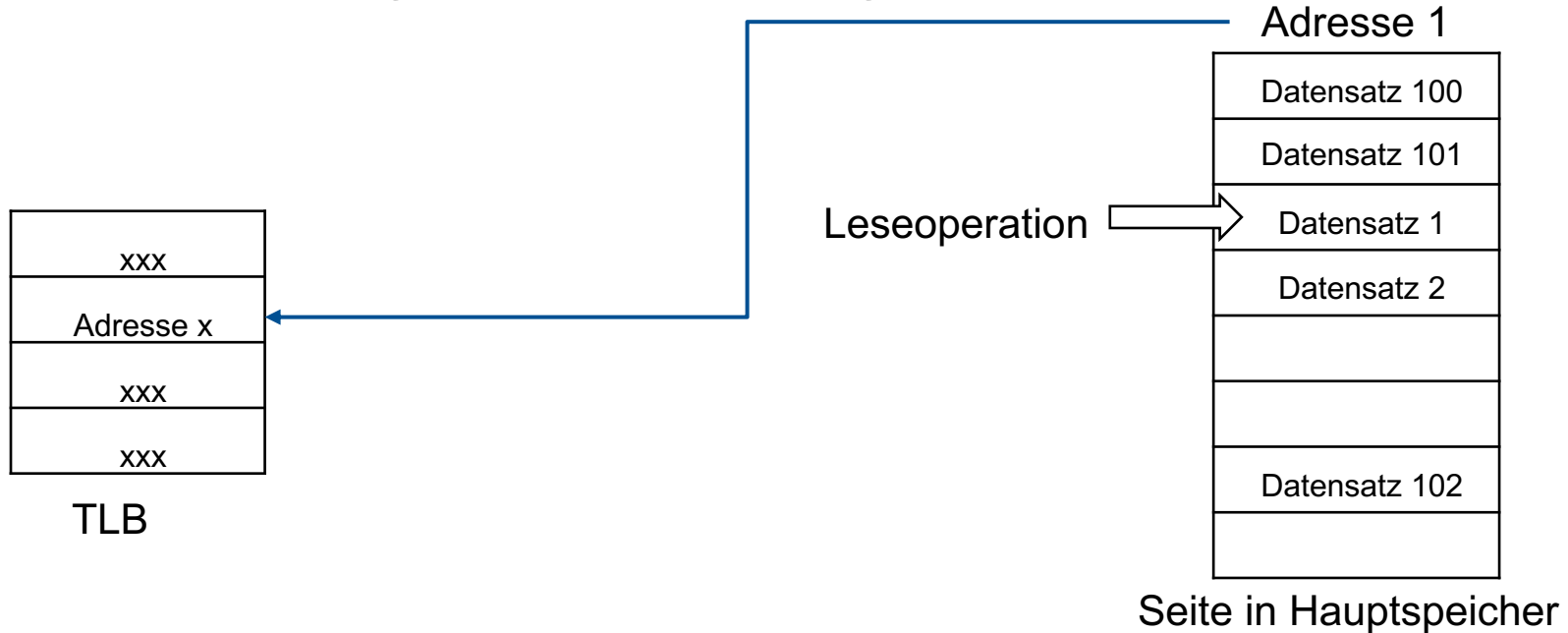
Chronologischer Datenzugriff



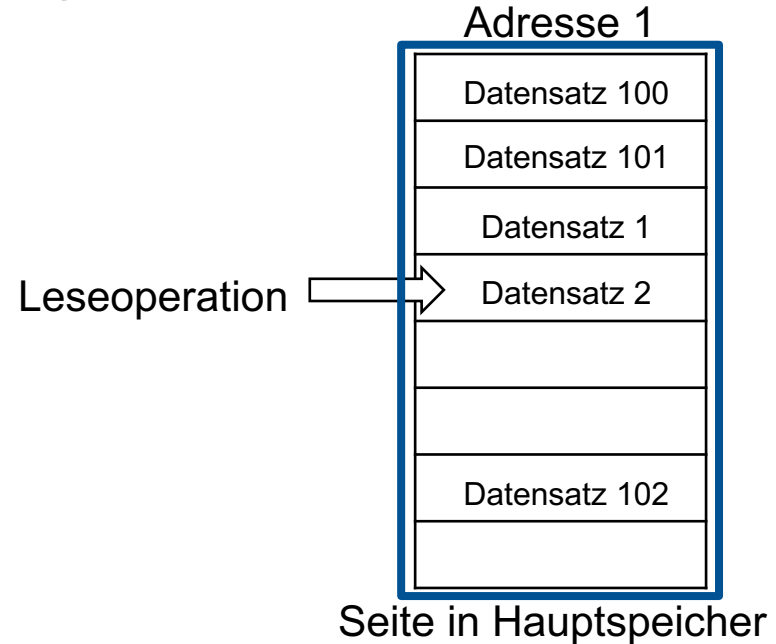
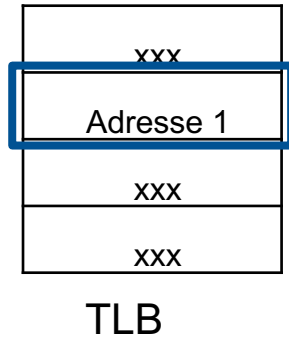
Chronologischer Datenzugriff



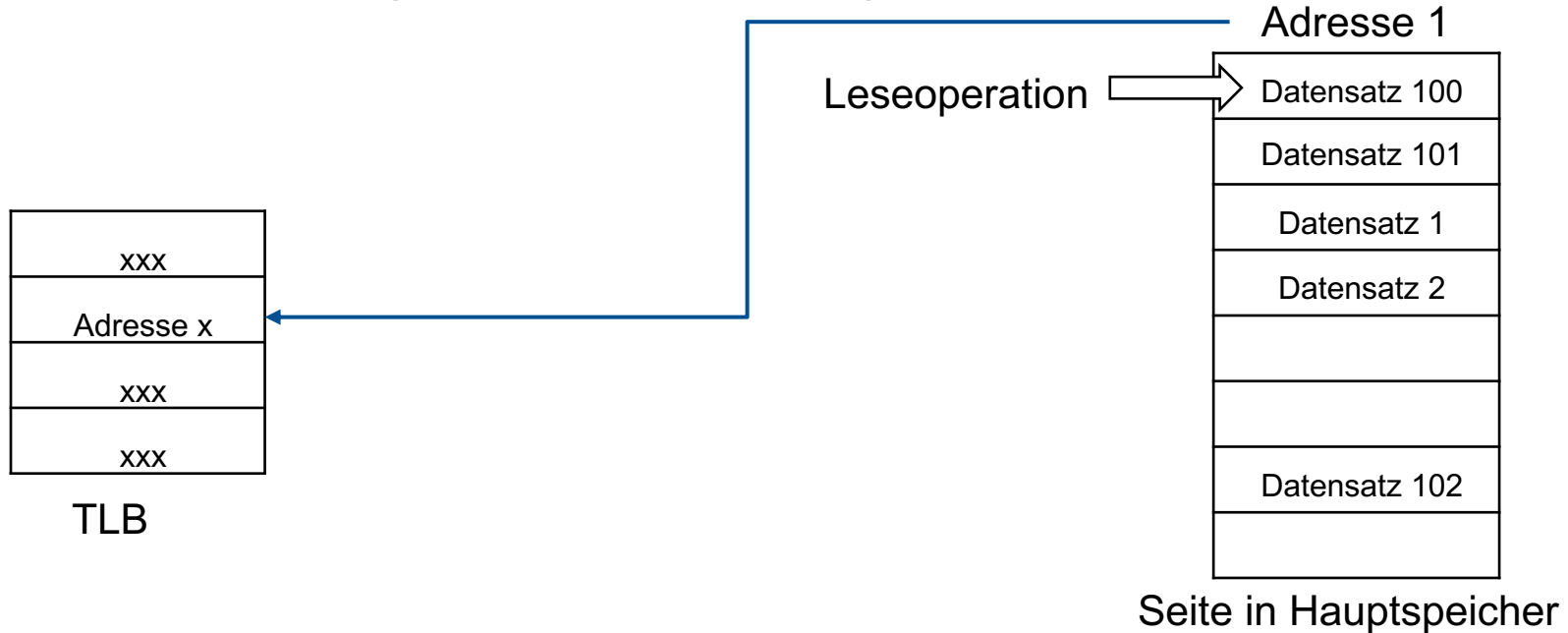
Nicht chronologischer Datenzugriff



Nicht chronologischer Datenzugriff








Nicht chronologischer Datenzugriff

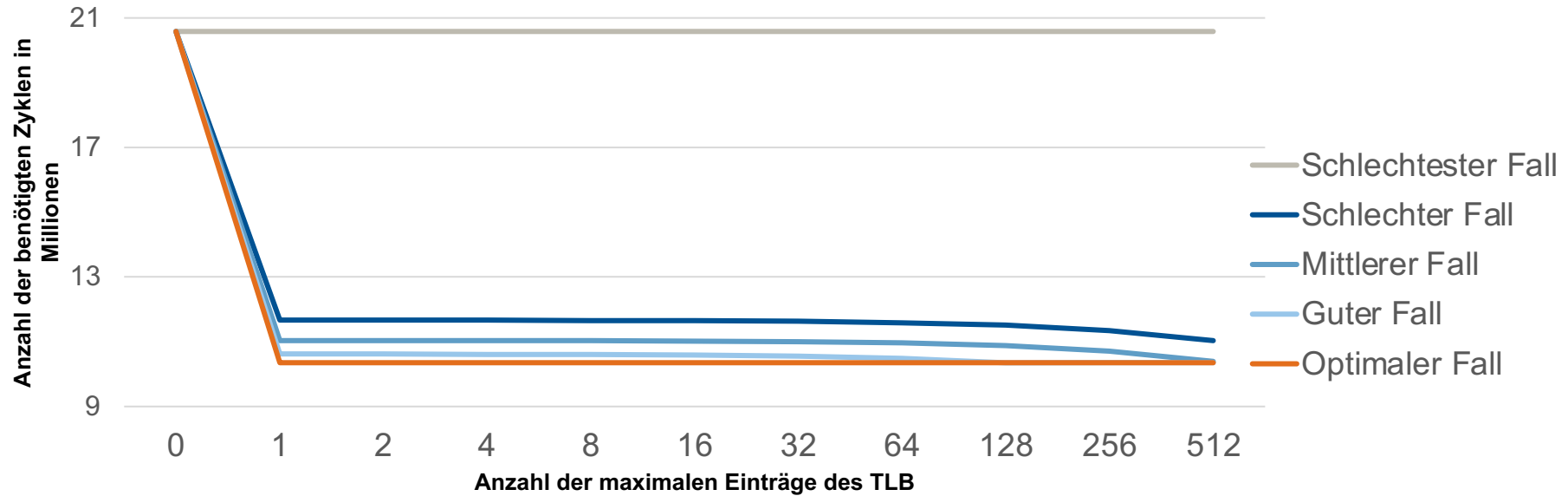


Messung des TLB's

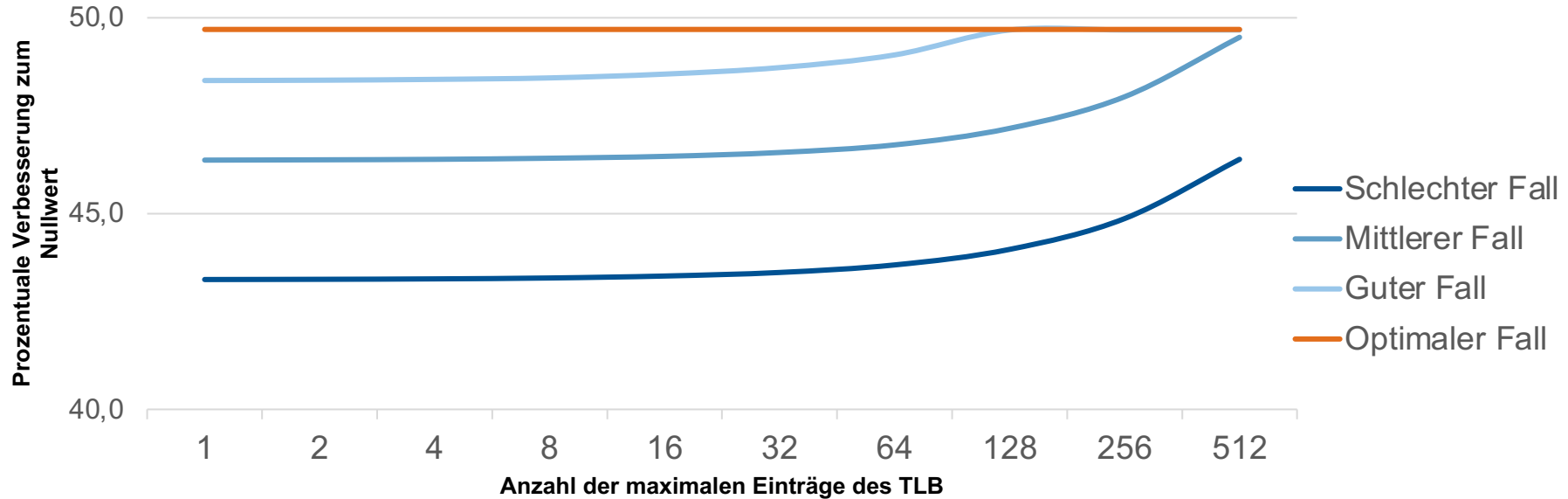
Jeweils 102400, 32-Bit Integer Werte

	Anzahl der Seiten	Ø-Anzahl von Einträgen pro Seite
Optimale Verteilung 	100	1024, chronologisch
Gute Verteilung 	128	800, nicht chronologisch
Mittlere Verteilung 	512	200, nicht chronologisch
Schlechte Verteilung 	1024	100, nicht chronologisch
Schlechteste Verteilung 	102400	1

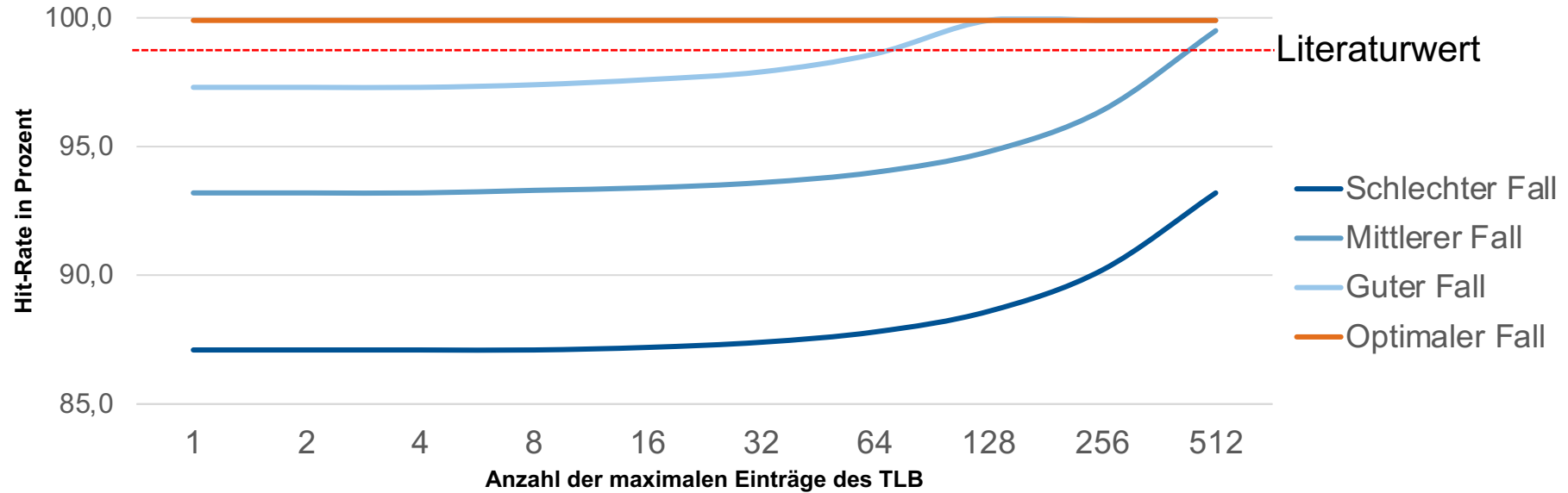
Anzahl der benötigten Zyklen



Anzahl der eingesparten Zyklen



Hit-Rate

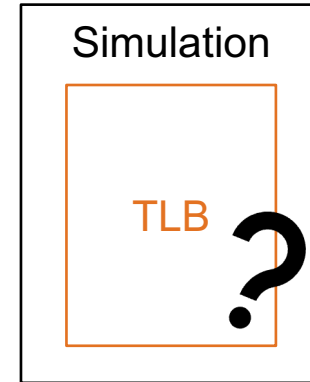


Korrektheit

TLB Simulation

Diskrepanz in Hit-Rate?

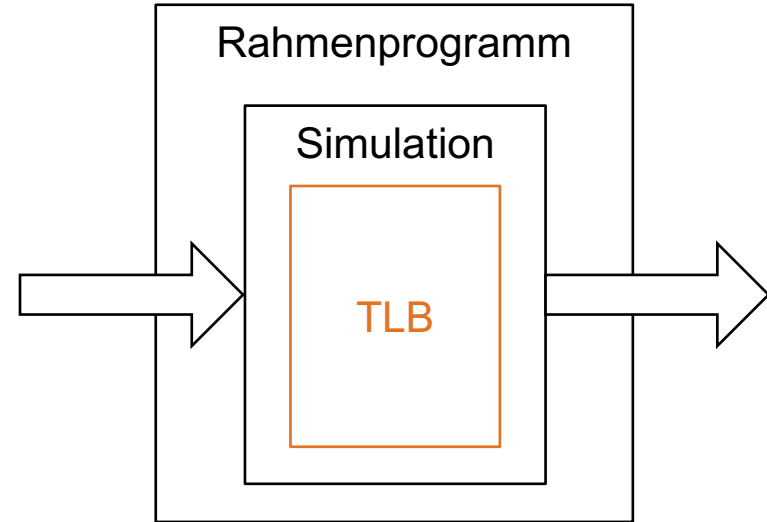
- Abhängig vom Memory Management
- Direct Mapped-Cache nicht optimal
- Keine Heuristik für Ersetzungsstrategie



Korrektheit

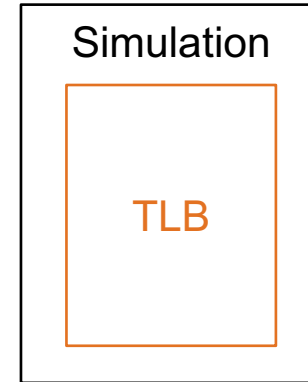
Rahmenprogramm

- Input-Fuzzing für Startargumente
- Request-Fuzzing für Lese/Schreibzugriffe
- Verifizierung der Ausgegeben Werte
- Benutzung des Request-Generation-Tools



Ergebnisse

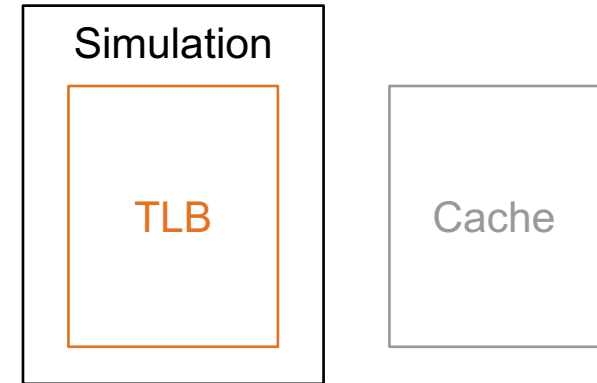
- Bauweise und vom TLB nicht sehr bedeutend
- Insbesondere Größe des TLB (bei Directly Mapped)
- Wichtig ist Verteilung der Übersetzungszugriffe
- Fokus auf gutes Memory Management
- Ebenfalls Cache Optimierte Programmierung



Ergebnisse

- Ergänzendes Cache zum aktuellen System
- Wichtig ist Kombination mit Daten Cache
- Schon bei kleinen Größen sehr effizient
- Dementsprechend geringe Gatterkosten beim Verbau
- Geeignete Cache Assoziativität verwenden
- Verdrängungs-Heuristiken verwenden

Simulation dementsprechend erweitern,
um realitätsnähere Ergebnisse zu erhalten



Literatur

- Patterson, D. A. (2009). Computer Organization and Design: The Hardware/Software Interface. (4th ed.). Morgan Kaufman.
- Arpaci-Dusseau, R. H. (2023). Operating Systems: Three Easy Pieces. Arpaci-Dusseau Books

Translation Lookaside Buffer

Technische Universität München

Grundlagenpraktikum Rechnerarchitektur

19. August 2024

Gruppe 151

Lukas Wolf

Elena Reinbold Fraire

Jonah Zabel

