

Vorlesung: Rechnerstrukturen, Teil 2 (Modul IP7)

J. Zhang

zhang@informatik.uni-hamburg.de

Universität Hamburg
Fachbereich Informatik
AB Technische Aspekte Multimodaler Systeme

Inhaltsverzeichnis

8. Die Speicherhierarchie 262

 Festplatten 262

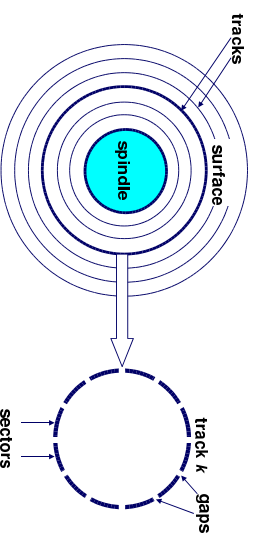
 Cache Speicher 277

 Virtuelle Speicher 292

Festplattengeometrie

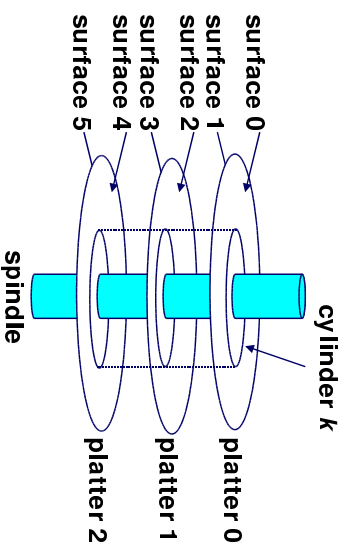
Festplatten bestehen aus Platten mit jeweils zwei Oberflächen ("surfaces").
Jede Oberfläche besteht aus konzentrischen Ringen, die Spuren ("tracks") genannt werden.

Jede Spur besteht aus Sektoren ("sectors"), die durch Lücken ("gaps") getrennt sind.



Festplattengeometrie (Ansicht mehrerer Platten)

Untereinander liegende Spuren bilden einen Zylinder.



Festplattenkapazität

Kapazität: Höchstzahl speicherbarer Bits

- Verkäufer geben Kapazität in Gigabyte (GB) an, wobei $1GB = 10^9 \text{ Byte}$.

Kapazität wird von diesen technologischen Faktoren bestimmt:

- Aufnahmedichte (bits/in): Anzahl der Bits, die in ein 1-Inch Segment einer Spur passen.
- Spurdichte (tracks/in): Anzahl der Spuren, die in ein radiales 1-Inch Segment passen.
- Flächendichte (bits/in²): Resultat aus Aufnahme- und Spurdichte.

Moderne Festplatten unterteilen Spuren in getrennte Unterbereiche, die Aufnahmezonen ("Recording Zones")

- Jede Spur einer Zone hat gleichviele Sektoren, bestimmt durch die Ausdehnung der innersten Spur.
- Jede Zone hat unterschiedlich viele Sektoren/Spuren.

Seite 264

Berechnen der Festplattenkapazität

$$\text{Kapazität} = (\# \text{Bytes/Sektor}) \times (\text{Durchschn. \# Sektoren/Spur}) \times (\text{Spuren/Oberfläche}) \times (\# \text{Oberflächen/Platten}) \times (\# \text{Platten/Festplatte})$$

Beispiel:

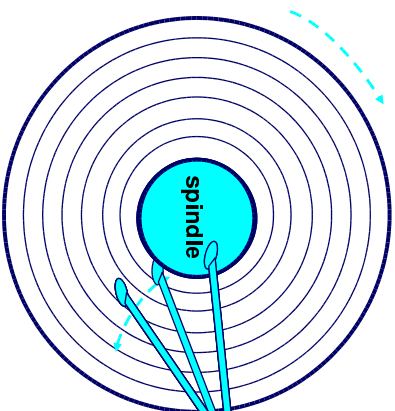
- 512 Bytes/Sektor
- 300 Sektoren/Spuren (im Durchschnitt)
- 20.000 Spuren/Oberfläche
- 2 Oberflächen/Platten
- 5 Platten/Festplatte

$$\begin{aligned} \text{Kapazität} &= 512 \times 300 \times 20000 \times 2 \times 5 \\ &= 30.720.000.000 \\ &= 30.72 \text{ GB} \end{aligned}$$

Seite 265

Festplatten-Operation (Ansicht einer Platte)

The disk
surface
spins at a
fixed
rotational
rate



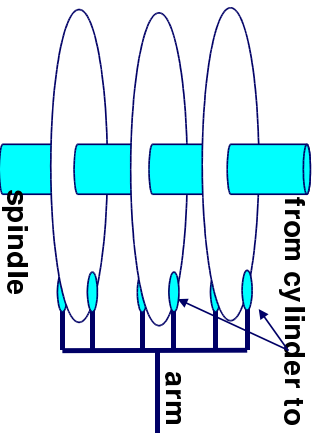
The read/write *head*
is attached to the end
of the *arm* and flies over
the disk surface on
a thin cushion of air.

By moving radially, the
arm can position the
read/write head over
any track.

Festplatten-Operation (Ansicht mehrerer Platten)

read/write heads
move in unison

from cylinder to cylinder



Festplatten-Zugriffszeit

Durchschnittliche (avg) Zugriffszeit auf einen Zielsektor wird angenähert durch:

- $T_{\text{Zugriff}} = T_{\text{avg Suche}} + T_{\text{Durchschn Rotationslatenz}} + T_{\text{avg Transfer}}$

Suchzeit ($T_{\text{avg Suche}}$)

- Zeit in der Schreib-Lese-Köpfe ("heads") über den Zylinder mit dem Targetsektor positioniert werden
- Üblicherweise $T_{\text{avg Suche}} = 9 \text{ ms}$

Festplatten-Zugriffszeit II

Rotationslatenzzeit ($T_{\text{avg Rotationslatenz}}$)

- Wartezeit, bis das erste Bit des Targetsektors unter dem r/w Schreib-Lese-Kopf durchrotiert.
- $T_{\text{avg Rotationslatenz}} = 1/2 \times 1/\text{RPMs} \times 60 \text{ Sek}/1 \text{ Min}$

Transferzeit ($T_{\text{avg Transfer}}$)

- Zeit, in der die Bits des Targetsektors gelesen werden
- $T_{\text{avg Transfer}} = 1/\text{RPM} \times 1/(\text{Durchschn \# Sektoren/Spur}) \times 60 \text{ Sek}/1 \text{ Min}$

Beispiel für Festplatten-Zugriffszeit

Gegeben:

- Umdrehungszahl = 7.200 RPM ("Rotations per Minute")
- Durchschnittliche Suchzeit = 9 ms.
- Avg # Sektoren/Spur = 400

Abgeleitet:

- Tavg Rotationslatenz = $1/2 \times (60 \text{ Sek}/7200 \text{ RPM}) \times 1000 \text{ ms/Sek} = 4 \text{ ms}$.
- Tavg Transfer = $60/7200 \text{ RPM} \times 1/400 \text{ Sek/Spur} \times 1000 \text{ ms/Sek} = 0.02 \text{ ms}$
- Tavg Zugriff = 9 ms + 4 ms + 0.02 ms

Seite 270

Beispiel für Festplatten-Zugriffszeit

Wichtige Punkte:

- Zugriffszeit wird von Suchzeit und Rotationslatenzzeit dominiert
- Das erste Bit eines Sektors ist das "teuerste", der Rest ist quasi umsonst
- SRAM Zugriffszeit ist ca. 4ns/Doppelwort, DRAM ca. 60 ns
 - ◆ Festplatte ist ca. 40.000 mal langsamer als SRAM,
 - ◆ 2.500 mal langsamer als DRAM.

Seite 271

Logische Festplattenblöcke

Moderne Festplatten haben eine simple abstrakte Ansicht der komplexen Sektorengeometrie:

- Der Bereich verfügbarer Sektoren wird als eine Sequenz logischer Blöcke der Größe b modelliert $(0, 1, 2, \dots)$

Abbildung zwischen logischen Blöcken und tatsächlichen (physikalischen) Sektoren

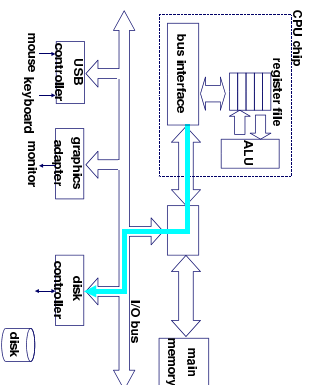
- Wird ausgeführt durch eine Hardware/Firmware Einheit: names Festplattencontroller
- Konvertiert Anfragen nach logischen Blöcken zu Tripeln (Oberfläche, Spur, Sektor).

Erlauben dem Controller, für jede Zone Ersatzzylinder bereit zu stellen.

- Bedingt den Unterschied zwischen "formatierter Kapazität" und "Maximaler Kapazität"

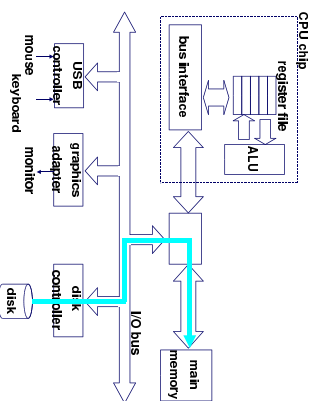
Lesen eines Festplattensektors (1)

Die CPU initiiert einen Lesevorgang von einer Festplatte, indem sie Befehl, logische Blocknummer und Zielspeicheradresse auf einen Port (Adresse) schreibt, der dem Festplattencontroller zugeordnet ist.

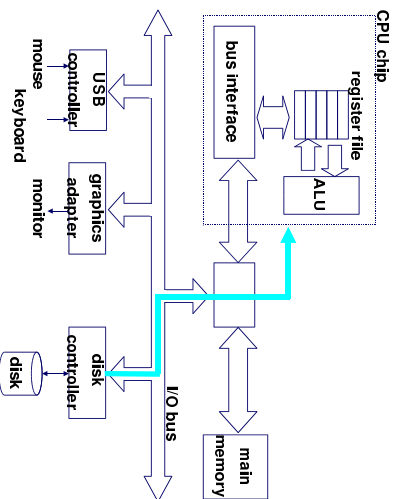


Lesen eines Festplattensektors (2)

Der Festplattencontroller liest den Sektor aus und führt einen direkten Speicherzugriff (DMA) auf den Hauptspeicher aus.



Lesen eines Festplattensektors (3)



Speicherhierarchien

Einige fundamentale und bleibende Eigenschaften von Hard- und Software:

- Schnelle Speichertechnologien kosten mehr pro Byte und haben geringere Kapazität.
- Der Abstand zwischen CPU und Hauptspeichergeschwindigkeit vergrößert sich.
- Gut geschriebene Programme weisen meist eine gute Lokalität auf

Diese fundamentalen Eigenschaften ergänzen sich sehr gut.

Sie legen einen Ansatz zur Organisation von Speichern und Speichersystemen nahe, der als Speicherhierarchie bekannt ist.

Seite 276

Cache Speicher

Themen

- Allgemeine Cache Speicher Organisation
- Direkt abgebildete Caches ("direct mapped")
- Bereichsassoziative Caches ("set associative")
- Einfluss der Caches auf die Leistung

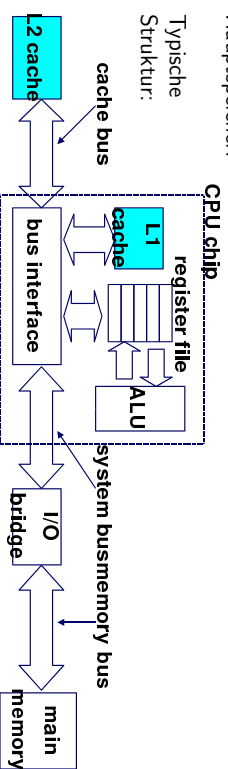
Seite 277

Cache Speicher

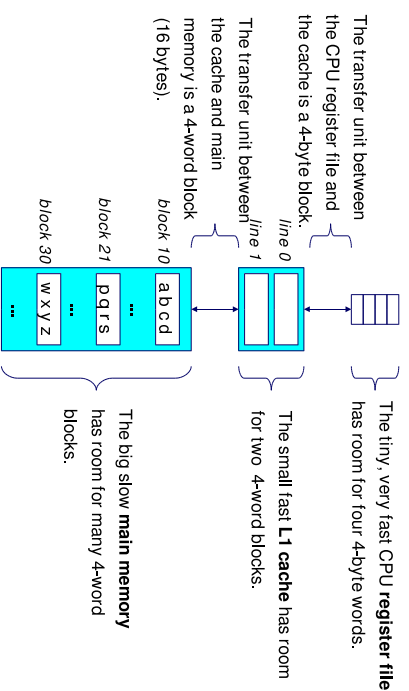
Cache Speicher sind kleine, auf schnellen SRAM basierende Speicher, die in der Hardware automatisch verwaltet werden

- Enthalten Blöcke des Hauptspeichers, auf die oft zugegriffen wird

CPU sucht erst nach Daten in L1 (level 1), dann in L2 (level 2), dann im Hauptspeicher.

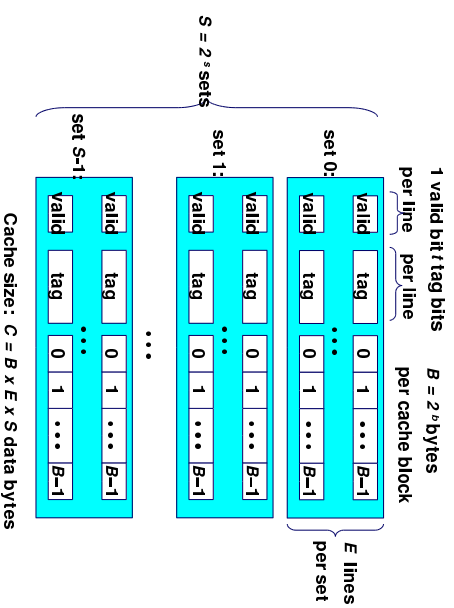
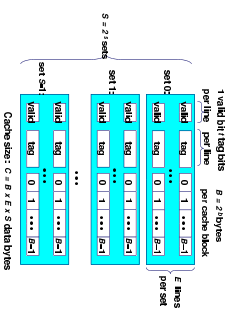


Einschieben eines L1 Cache zwischen CPU und Hauptspeicher



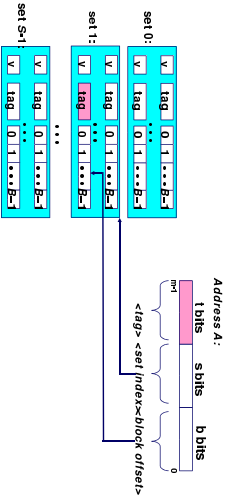
Allgemeine Organisation eines Cache Speichers

- Cache ist ein Array von Speicher-Bereichen ("sets")
- Jeder Bereich enthält eine oder mehrere Zeilen
- Jede Zeile enthält einen Datenblock



Adressierung von Caches

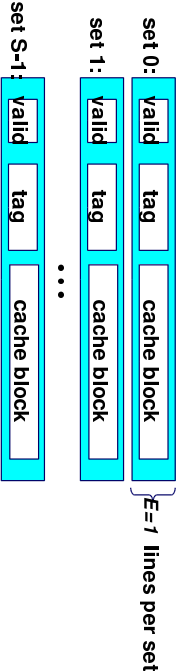
- Das Wort bei Adresse A liegt im Cache, wenn die "tag" Bits in einer der als gültig markierten Zeilen $\leq$ "valid" $>$ im aktuellen Bereich $\leq$ "set index" $>$ mit dem $\leq$ "tag" $>$ der Adresse A übereinstimmen.
- Das gesuchte Wort beginnt im Cache unter dem angegebenen Offset and der Stelle $\leq$ "block offset bytes" $>$ hinter dem Anfang des Blocks.



Direkt abgebildete Cache (direct mapped)

Einfachste Cache-Art

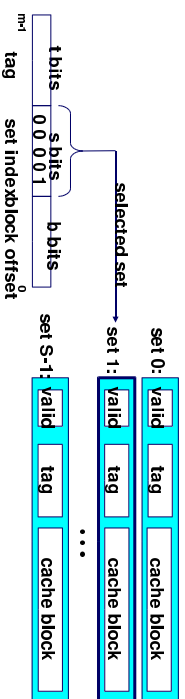
Verfügt über genau 1 Zeile pro Bereich.



Zugriff auf direkt abgebildete Caches

Bereichsauswahl

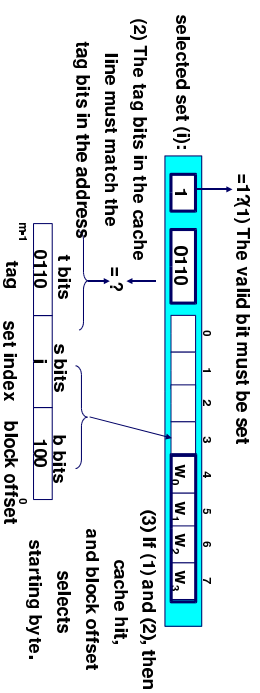
- Benutze die Bereichsindex-Bits ("set index") um den fraglichen Bereich zu bestimmen



Zugriff auf direkt abgebildete Caches

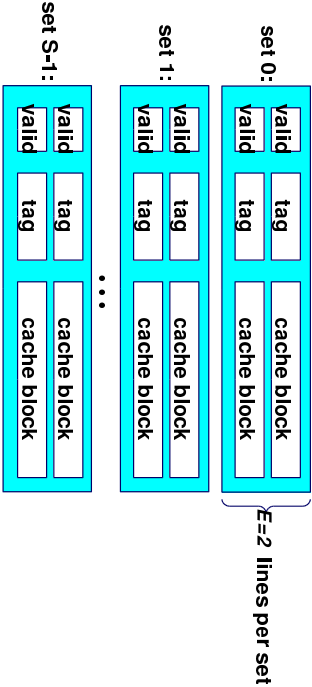
"Line matching" und Wortselektion

- "Line matching": Finde eine gültige Zeile im ausgewählten Bereich mit übereinstimmendem "tag".
- Wortselektion: Dann extrahiere das Wort



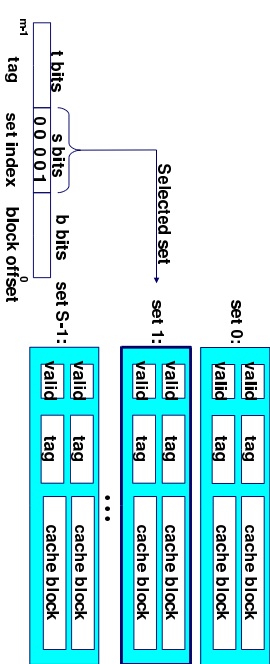
Bereichsassoziative Caches ("set associative")

Verfügen über mehr als eine Zeile pro Bereich.



Zugriff auf Bereichsassoziative Caches

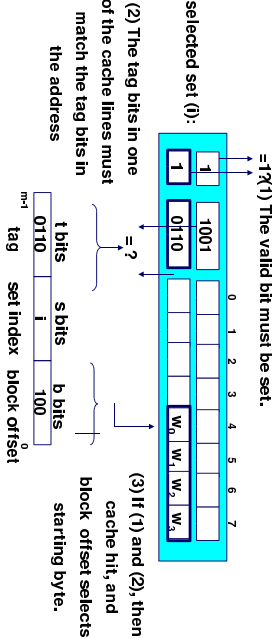
Bereichsauswahl: identisch mit direkt abgebildetem Cache



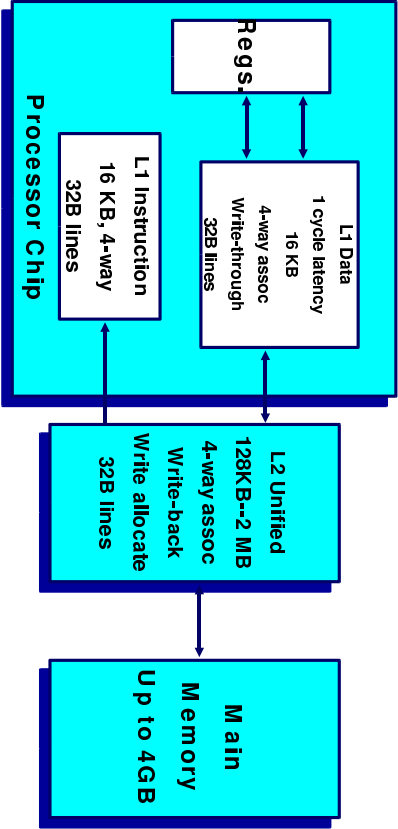
Zugriff auf Bereichsassoziative Caches

"Line matching" und Wortselektion

- muss das "tag" in jeder gültigen Zeile im ausgewählten Bereich ("set") vergleichen



Intel Pentium Cache Hierarchie



Abschließende Bemerkungen

Der Programmierer kann sein Programm für maximale Cacheleistung optimieren

- durch entsprechende Organisation der Datenstrukturen
- durch Steuerung des Zugriffs auf die Daten
 - ◆ Geschachtelte Schleifenstruktur
 - ◆ Blockbildung ist eine übliche Technik

Seite 290

Abschließende Bemerkungen (Forts.)

Alle Systeme bevorzugen einen "Cache-freundlichen Code"

- Erreichen der optimalen Leistung ist sehr plattformspezifisch
 - ◆ Cachegrößen, Zeilengrößen, Assoziativität etc.
- einen großen Teil der optimalen Leistung erhält man mit allgemeinem Code, wenn man
 - ◆ "working set" angemessen klein hält (zeitliche Lokalität)
 - ◆ kleine Adressfortschaltungen ("strides") benutzt (räumliche Lokalität)

Seite 291

Motivation für einen virtuellen Speicher

Benutzung von Hauptspeicher als "Cache" für die Festplatte

- Adressraum eines Prozesses kann physikalische Speichergröße übersteigen
- Die Summe der Adressräume mehrerer kleiner Prozesse kann physikalischen Speicher übersteigen

Vereinfachung der Speicherverwaltung

- Viele Prozesse liegen im Hauptspeicher
 - ◆ Jeder Prozess mit seinem eigenen Adressraum
- Nur "aktiver" Code und Daten sind tatsächlich im Speicher
 - ◆ Teile dem Prozess bei Bedarf mehr Speicher zu

Seite 292

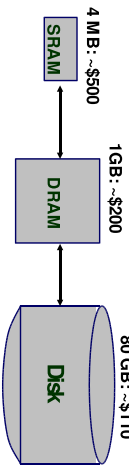
Motivation für einen virtuellen Speicher (Forts.)

Bereitstellung von Schutzmechanismen

- Ein Prozess kann einem anderen nicht ins Gehege kommen
 - ◆ da sie in verschiedenen Adressräumen operieren
- Benutzerprozess kann nicht auf privilegierte Informationen zugreifen
 - ◆ die verschiedenen Abschnitte der Adressräume haben verschiedene Zugriffsrechte

Seite 293

Motivation Nr.1: DRAM ist ein "Cache" für die Festplatte



Vollständiger Adressraum ist ziemlich groß:

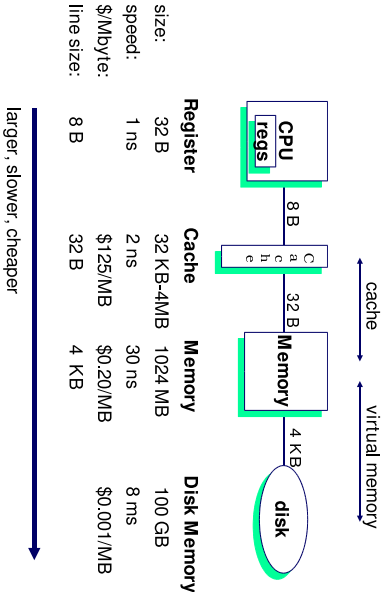
- 32-Bit Adressen: $\approx 4.000.000.000$ (4 Milliarden) Bytes
- 64-Bit Adressen: $\approx 16.000.000.000.000.000.000$ (16 Quintillionen) Bytes

Speichern auf der Festplatte ist $\approx 300\times$ billiger als im DRAM

- 80 GB DRAM: $\approx \$33.000$
- 80 GB Festplatte: $\approx \$110$

Um kostengünstig auf große Datenmengen zuzugreifen, muss der Hauptanteil an Daten auf der Festplatte gespeichert werden.

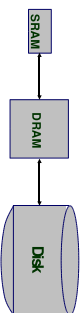
Ebenen in der Speicherhierarchie



DRAM versus SRAM als "Cache"

DRAM versus Festplatte ist extremer als SRAM versus DRAM

- Zugriffswartenzeiten
 - ◆ DRAM $\approx 10\times$ langsamer als SRAM
 - ◆ Festplatte $\approx 100.000\times$ langsamer als DRAM
- Wichtigkeit der Nutzung der räumlichen Lokalität:
 - ◆ Erstes Byte ist $\approx 100.000\times$ langsamer als nachfolgende Bytes
→ deutlich dramatischer als z. B. 4malige Verbesserung durch Seitenmodus ("Page-Mode") gegenüber regulärem Zugriff auf RAM
- Schlussfolgerung
 - ◆ Designentscheidung für DRAM als Cache hängt von den enormen Kosten bei Fehlzugriff ("Cache miss") ab

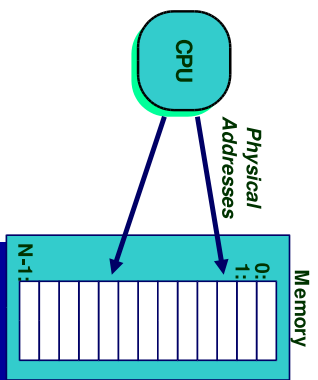


Seite 296

Ein System mit ausschließlich physikalischem Speicher

Beispiele:

- die meisten Cray Maschinen, frühe PCs, fast alle eingebetteten Systeme etc.

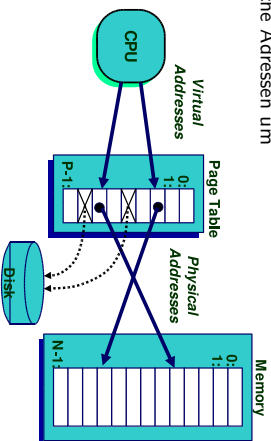


Seite 297

Ein System mit virtuellem Speicher

Beispiele:

- Workstations, Server, moderne PCs etc.
- Adressübersetzung: Hardware wandelt via vom Betriebssystem (OS) verwalteter "lookup table" (Seitentabelle) virtuelle Adressen in physikalische Adressen um

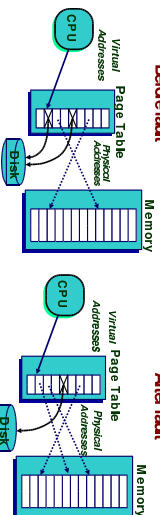


Seite 298

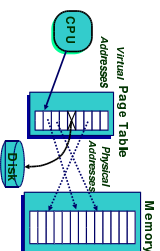
Seitenfehler (ähnlich wie "Cache Misses")

- Seiten-Tabelleneintrag gibt virtuelle Adresse für eine Seite an, die nicht im Speicher liegt
- OS "exception handler" wird aufgerufen, um Daten von der Festplatte in den Speicher zu laden
- ♦ laufender Prozess wird unterbrochen, andere können weiterlaufen
- ♦ OS hat die volle Kontrolle über die Platzierung der neuen Seite im Hauptspeicher (Ersetzungsstrategien) etc.

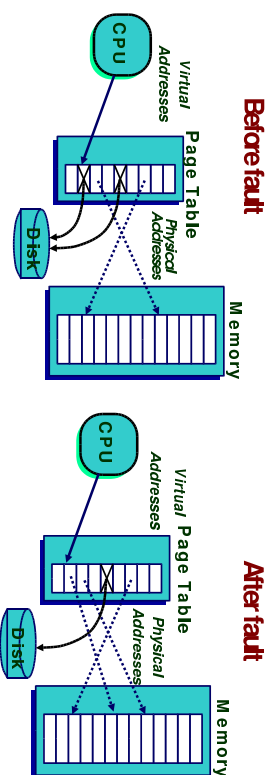
Before fault



After fault



Seite 299



Einen Seitenfehler behandeln

Prozessor meldet dem Controller

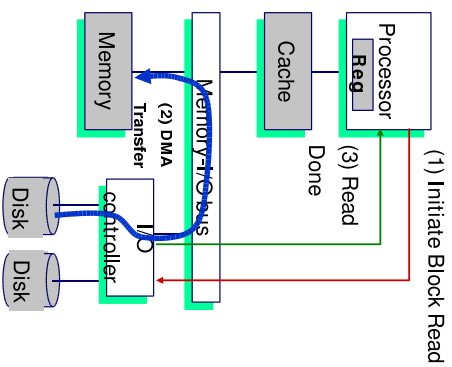
- Lies Block der Länge P ab der Festplattenadresse X und speichere sie ab Speicheradresse Y ab.

Lesezugriff erfolgt in Form von:

- Direct Memory Access (DMA)
- Kontrolle durch I/O Controller

I/O Controller meldet Abschluss

- Gibt Interrupt an den Prozessor
- OS lässt unterbrochenen Prozess weiterlaufen



Ergänzende Literatur

Zur Rechnerarchitektur (Teil 2, 7. Termin):

Literatur

- [1] Randal E. Bryant and David O'Hallaron. Computer systems. pages 454–531, 690–700. Pearson Education, Inc., New Jersey, 2003.