Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (19.12.2012)

Date: Wed Dec 19 13:14:21 CET 2012

Duration: 45:13 min

Pages: 16

Speicherverwaltung

Fragestellungen

Dieser Abschnitt beschäftigt sich mit den Adressräumen für Programme und deren Abbildung auf den physischen Arbeitsspeicher einer Rechenanlage:

Programmadressraum vs. Maschinenadressraum.

Direkte Adressierung, Basisadressierung.

Virtualisierung des Speichers; virtuelle Adressierung, insbesondere Seitenadressierung.

Einige fundamentale Eigenschaften von Hardware und Software

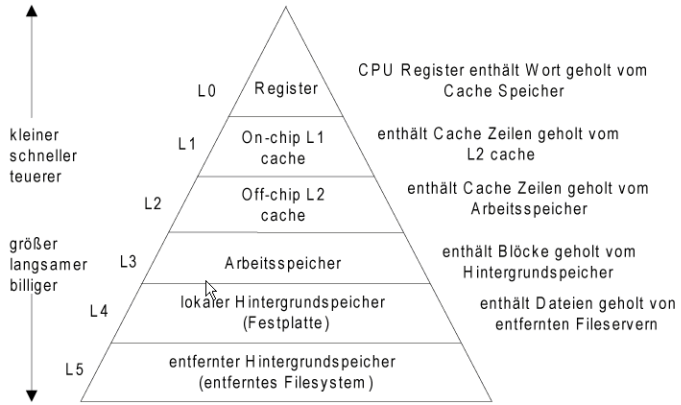
schnelle Speichertechnologien kosten mehr Geld pro Byte und haben kleinere Kapazitäten.

Lücke zwischen CPU und Arbeitsspeichergeschwindigkeit wächst.

gut geschriebene Programme tendieren dazu, gute Lokalität zu haben

zeitliche Lokalität: kürzlich referenzierte Objekte werden in naher Zukunft wieder referenziert.

räumliche Lokalität: Objekte mit beieinander liegenden Adressen, tendieren dazu, ungefähr zur gleichen Zeit referenziert zu werden.



Cache: Bereitstellungsraum für eine Teilmenge der Daten einer größeren, langsameren Speichereinheit.

Fundamentale Idee hinter der Speicherhierarchie:

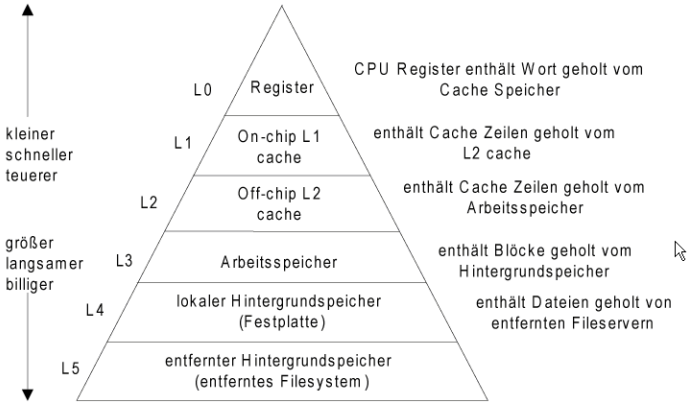
für jedes k agiert die schnellere, kleinere Einheit auf Schicht k als Cache für die größere, langsamere Einheit auf Schicht k+1

Warum funktionieren Speicherhierarchien

Programme tendieren dazu auf Daten der Schicht k häufiger zuzugreifen, als auf Daten der Schicht k+1.

deshalb okay, wenn Speicher auf Schicht k+1 langsamer und preiswerter ist.

Speicherhierarchie - Beispiel



Cache Speicher

Cache: Bereitstellungsraum für eine Teilmenge der Daten einer größeren, langsameren Speichereinheit.

Fundamentale Idee hinter der Speicherhierarchie:

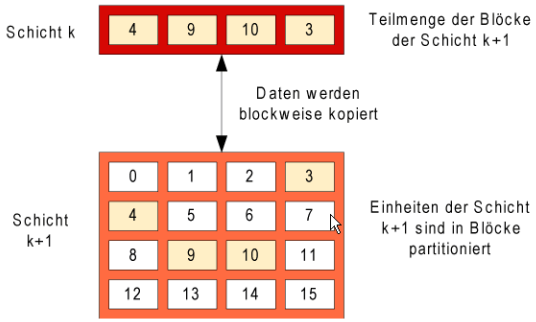
für jedes k agiert die schnellere, kleinere Einheit auf Schicht k als Cache für die größere, langsamere Einheit auf Schicht k+1

Warum funktionieren Speicherhierarchien

Programme tendieren dazu auf Daten der Schicht k häufiger zuzugreifen, als auf Daten der Schicht k+1.

deshalb okay, wenn Speicher auf Schicht k+1 langsamer und preiswerter ist.

kleinere, schnellere, teuere Einheiten auf Schicht k cachen Daten von größeren, langsameren, preiswerteren Einheiten der Schicht k + 1.



Programm braucht Objekt d in Block b

Cache-Treffer (hit): Programm findet b im Cache der Schicht k, z.B. in Block 10

Cache-Fehler (miss): b ist nicht auf Schicht k; Cache muss b von Schicht k+1 holen (z.B. Block 8).

wenn Schicht k Cache voll, muss ein Block entfernt werden, z.B. nach LRU.

Typen von Cache Problemen

Beispiele aus der Cache Hierarchie

Es können unterschiedliche Arten von Problemen unterschieden werden:

kalter Cache: der Cache ist leer

Kapazitätsfehler: resultieren daraus, dass die Summe der aktiven Cache Blöcke größer ist als der Cache

Konfliktfehler

Caches limitieren meist Platzierungsmöglichkeiten der Blöcke der Schicht k+1 auf eine 1-elementige Teilmenge der Blöcke der Schicht k,

z.B. Block i muss auf Block i mod 4 platziert werden

Konfliktfehler entstehen, wenn der Cache groß genug ist, aber mehrere Datenobjekte auf denselben Block der Schicht k abgebildet werden.

z.B. Zugriff auf Blöcke 0, 8, 0,8, 0, 8, ... jedesmal ein Fehler

Caches werden auf unterschiedlichen Ebenen in modernen Rechensystemen genutzt.

Cache Typ	was wird gecached	wo wird gecached	kontrolliert
Register	4 Byte Wort	CPU Register	Compiler
TLB	Address Translation	On-Chip TLB	Hardware
L1 Cache	32 Byte Block	On-Chip L1	Hardware
L2 Cache	32 Byte Block	Off-Chip L2	Hardware
virtual memory	4 KB page	Arbeitsspeicher	Hardware + BS
buffer cache	Teile von Dateien	Arbeitsspeicher	BS
network buffer cache	Teile von Dateien	lokale Platte	NFS client
Browser cache	Web Seiten	lokale Platte	Web Browser
Web cache	Web Seiten	entfernte Server Platte	Web Proxy Server

Realisierung von Caches

Obwohl je nach der Ebene der Cache Hierarchie unterschiedliche Varianten der Umsetzung existieren, folgen diese doch gemeinsamen allgemeinen Prinzipien.

Allgemeine Cache Organisation

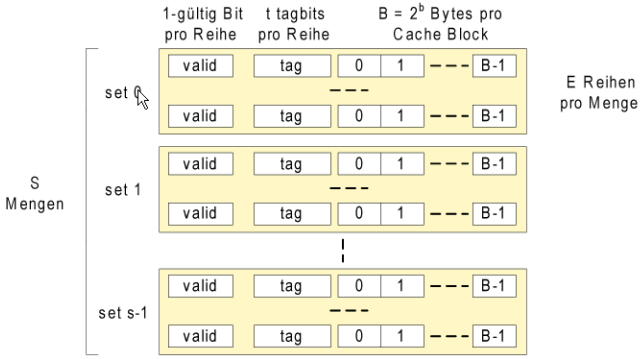
Angenommen, das Rechensystem hat Arbeitsspeicheradressen mit m bits ⇒ $M = 2^m$ eindeutige Adressen.

Cache ist ein Array von Mengen (S = Anzahl von Mengen).

jede Menge enthält eine oder mehrere Reihen (E = Anzahl von Reihen pro Menge).

jede Reihe enthält einen Datenblock (B ist Blockgröße pro Reihe).

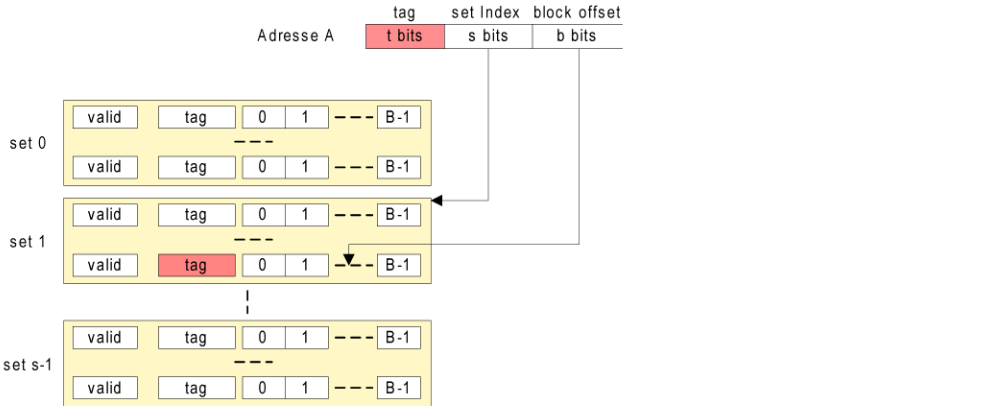
Cachegröße = B * E * S Bytes.



Die Cache Organisation bedingt eine Strukturierung der Speicheradresse

das Wort an Adresse A ist im Cache, falls die Tagbits einer der gültigen Reihen der Menge "set Index" den Tag "tag" besitzen

der Wortinhalt beginnt am Offset "block offset" Bytes vom Blockanfang



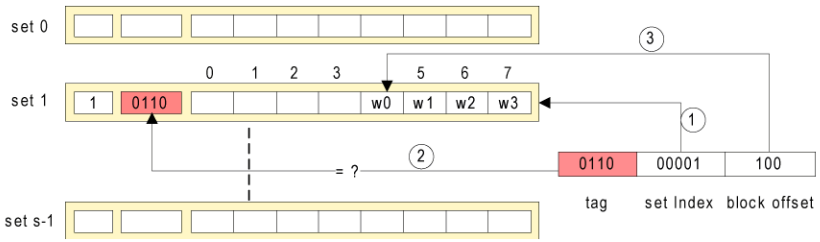
Direct-Mapped Cache

Einfachste Art des Caches mit genau einer Reihe pro Menge; Zugriff erfolgt in der Reihenfolge

Mengenselektion: Benutzung der "set Index" bits zum Bestimmen der relevanten Menge.

Reihenabgleich: Finden der gültigen Reihe in der selektierten Menge mit dem richtigen Tag.

Wortselektion: Benutzung des block offsets zum Bestimmen des relevanten Wortes.



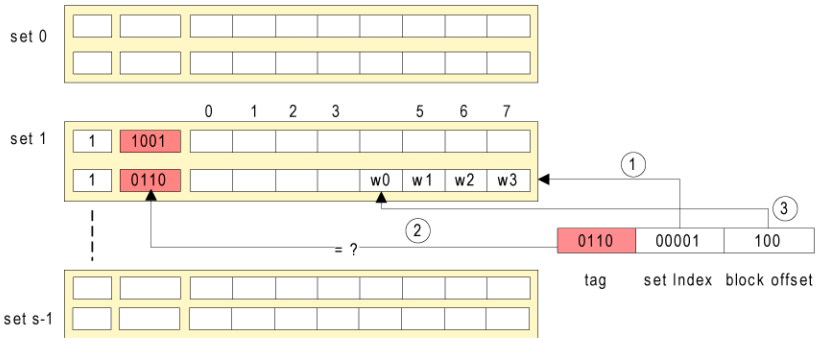
Mengen Assoziativer Cache

Mehr als eine Reihe pro Menge, d.h. E > 1;

Mengenselektion: identisch zu direct-mapped Cache.

Reihenabgleich: Vergleich der Tags für jede gültige Reihe in der selektierten Menge.

Wortselektion: identisch zu direct-mapped Cache.



Die Performanz eines Caches wird anhand der auftretenden Cache Misses und Cache Hits gemessen.

Fehlerrate = #Cache Misses / # Speicherzugriffe

Beeinflussung der Fehlerrate durch günstige / ungünstige Programmierung (Berücksichtigung der Lokalität).

Beispiel: Summe einer Matrix

Beispiel: Summe einer Matrix

Betrachten wir 4-Byte Worte. Weiterhin umfasse der Cache nur eine Menge mit genau einer Reihe.

Ein Cacheblock der Reihe umfasse vier 4-Byte Worte.

pro Matrixelement (vom Typ int) ist ein Wort (d.h. 4 Bytes) notwendig.

die Matrix a[M,N] sei zeilenweise im Speicher abgelegt und passt nicht vollständig in den Cache.

Zeilenweises Aufsummieren

```
int i, j, sum = 0;
for (i = 0; i < M; i++)
    for (j = 0; j < N; j++) sum += a[i][j]
return sum
```

Fehlerrate der inneren Schleife = 1/4 = 25%.

Spaltenweises Aufsummieren

```
int i, j, sum = 0;
for (j = 0; j < N; j++)
    for (i = 0; i < M; i++) sum += a[i][j]
return sum
```

Fehlerrate der inneren Schleife = 100%.

Disjunkte Prozesse, d.h. Prozesse, die völlig isoliert voneinander ablaufen, stellen eher die Ausnahme dar. Häufig finden Wechselwirkungen zwischen den Prozessen statt ⇒ Prozesse interagieren. Die Unterstützung der Prozessinteraktion stellt einen unverzichtbaren Dienst dar.

Fragestellungen

Dieser Abschnitt beschäftigt sich mit den Mechanismen von Rechensystemen zum Austausch von Informationen zwischen Prozessen.

Kommunikationsarten.

nachrichtenbasierte Kommunikation, insbesondere Client-Server-Modell.

Netzwerkprogrammierung auf der Basis von Ports und Sockets.