Ostfalia
Hochschule für angewandte Wissenschaften

Cache-Speicher

Design Digitaler Systeme

Prof. Dr.-Ing. Rainer Bermbach

Übersicht Cache-Speicher

- Warum Cache-Speicher?
- Cache-Strukturen
- Aufbau und Organisation von Caches
- Cache-Architekturen
- Cache-Strategien
- Zusammenfassung

1 Warum braucht man Cache-Speicher?

- idealer Speicher:
  - unendlich groß
  - beliebig schnell
  - beliebig billig
- Kompromiss nötig zwischen
  - Größe – Geschwindigkeit – Kosten
- Die Anforderungen müssen nur im Mittel erfüllt werden!

1.1 Speicherhierarchie

- Aufteilung in verschieden schnelle Speicherebenen = Speicherhierarchie:
  - CPU-Register/On-Chip-Speicher ca. 1- 10 ns
  - Arbeitsspeicher (DRAM) ca. 100 ns
  - Plattenspeicher ca. 10 ms
  - Bandspeicher ca. 1 s - 1 h
- => große zeitliche Diskrepanz zwischen CPU- und Hauptspeicherzugriff

1.1 Speicherhierarchie (Forts.)

- zunehmende Steigerung der CPU-Geschwindigkeit
- Arbeitsspeicher kaum schneller
- zusätzliche Hierarchiestufe zwischen CPU und Arbeitsspeicher nötig:
  der Cache-Speicher
- Cache arbeitet für die CPU transparent
  - häufigst benutzt
  - => kein Verwaltungsaufwand für die CPU (HW)
  - zeigt sich im Namen: Cache = Versteck

1.1 Speicherhierarchie (Forts.)

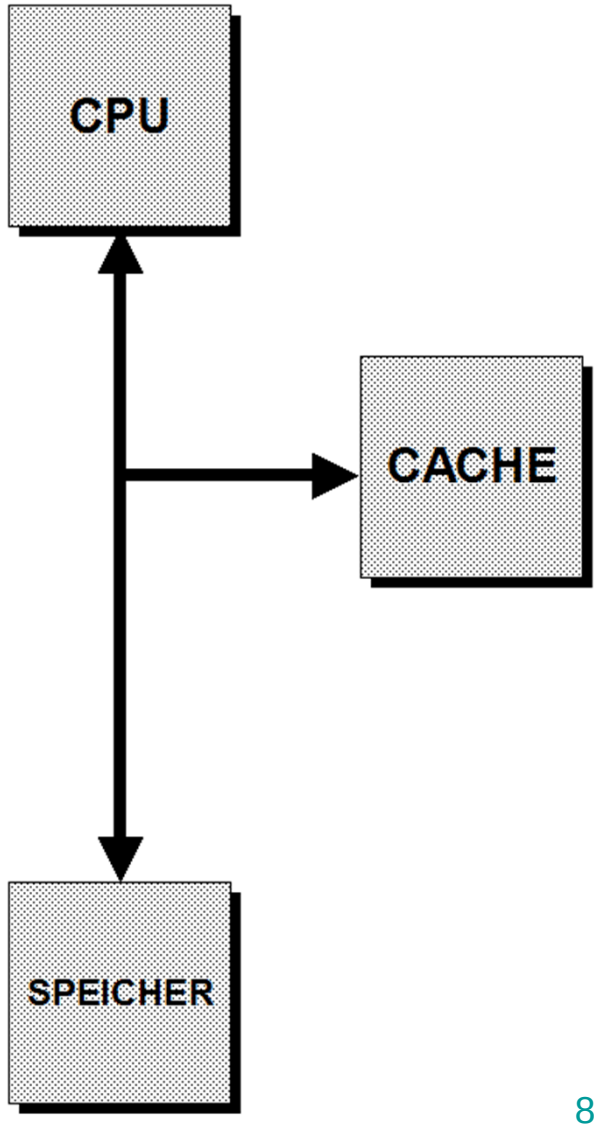
- Aufgabe des Caches:
  - Bereithalten der häufiger benötigten Daten und Befehle in einem schnellen Zwischenspeicher
  - Prozessor wird nicht durch langsamen Hauptspeicher „ausgebremst“
- Warum funktioniert das ?
  - Daten und Programme besitzen i.d.R. eine hohe Lokalität
  - ca. 95% der „nächsten“ Zugriffe liegen innerhalb eines Bereiches von wenigen Bytes

1.2 Anwendungen von Caches

- zwischen Prozessor und Hauptspeicher
- zwischen Multiprozessoren und dem gemeinsamen Arbeitsspeicher
- zwischen CPU/Arbeitsspeicher und der Peripherie z.B.
  - als Plattencache, beim CD-ROM-Laufwerk
  - beim Netzwerkanschluss, bei Schnittstellen
  - bei der Speicherverwaltung (Translation Look-aside Buffer)
- allgemein vor Einheiten mit langsamem Zugriff und hoher Lokalität der Daten

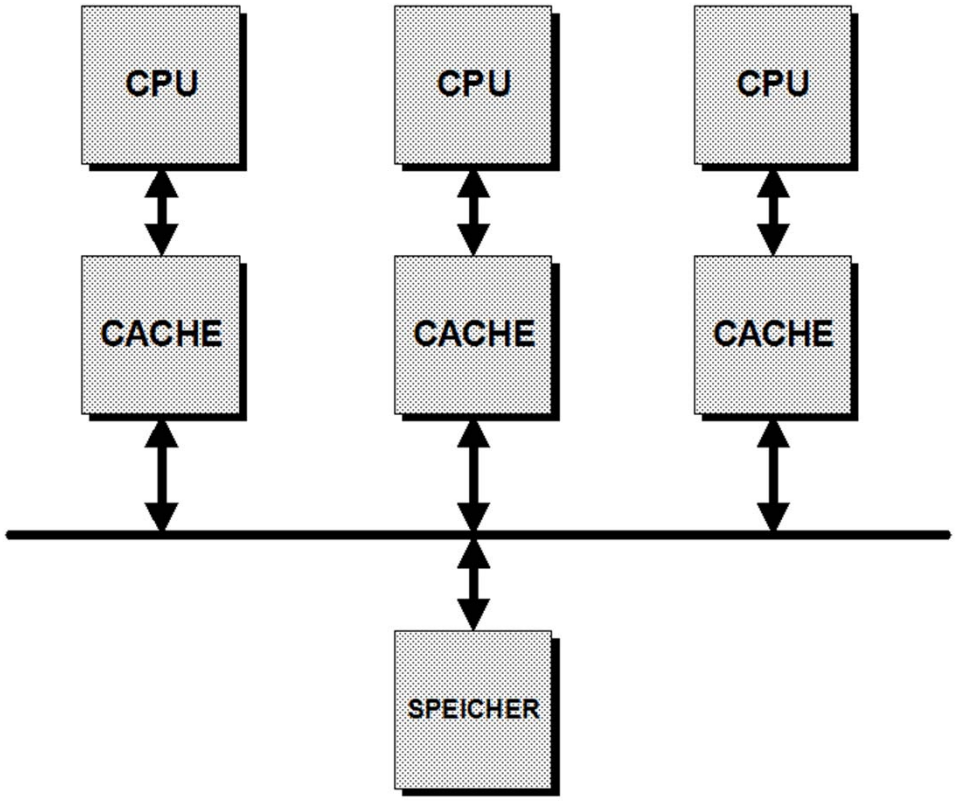
2 Cache-Strukturen

- Look-side Cache
  - Cache liegt parallel zum Daten-/Adressbus
  - Zugriff der CPU auf Cache- und System-RAM
  - nur bei Single-Prozessorsystemen



2 Cache-Strukturen (Forts.)

- Look-through Cache
  - CPU sieht nur den Cache
  - kein direkter Zugriff auf den Arbeits-speicher
  - vorrangig bei Multi-prozessorsystemen (Busentlastung) und Peripherie-Caches



2 Cache-Strukturen (Forts.)

- kombinierte Befehls/Datencaches
  - von-Neumann-Architektur
- getrennte Befehls/Datencaches
  - Harvard-Architektur
  - Daten und Befehle belegen verschiedene Bereiche und zeigen unterschiedliches Verhalten
- Mehrfach-Caches
    - 1st Level Cache (L1)
    - 2nd Level Cache (L2)
    - 3rd Level Cache (L3), Victim Cache

3 Aufbau und Organisation

- Data RAM
  - schnelles, statisches RAM (angepasst an CPU-Zykluszeit)
  - speichert die Daten in Form von Lines
- Tag RAM
  - noch schnelleres RAM
  - enthält Infos, welche Daten im Cache sind
- Cache Controller
  - gewährleistet transparente Nutzung
  - stellt die Kohärenz sicher

3 Aufbau und Organisation (Forts.)

- Speicherung in sog. Lines
  - Eintraglänge (Line Size) typ. 32 - 256 Bit
  - angepasst an die Prozessorwortbreite
- ein Tag/Line
  - Tag (Merker) für die Adresse und den Status der Daten im Cache
  - enthält die Info, ob gesuchtes Datum bereits im Cache vorliegt
- Komparator(en)
  - vergleicht den Tag (Adresse) mit der aktuell benötigten Adresse

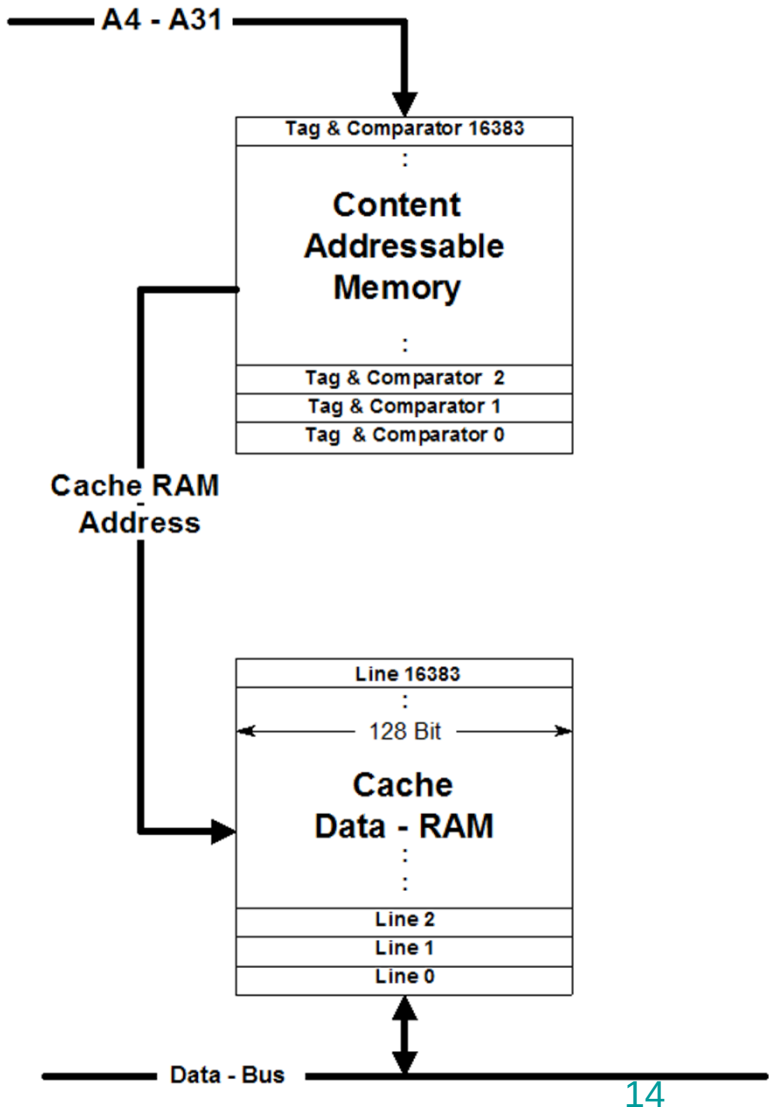
3 Aufbau und Organisation (Forts.)

- spezielle Begriffe:
  - Lokalität
    - Daten/Befehle liegen i.A. in direkter Nachbarschaft
  - Kohärenz
    - Übereinstimmung der Daten in Cache und Speicher
  - Cache Hit
    - gesuchte Daten liegen im Cache vor
  - Cache Miss
    - gesuchte Daten liegen noch nicht im Cache vor
  - Cacheable Area /Memory
    - max. Speicherbereich, den der Cache abdeckt

4 Cache-Architekturen

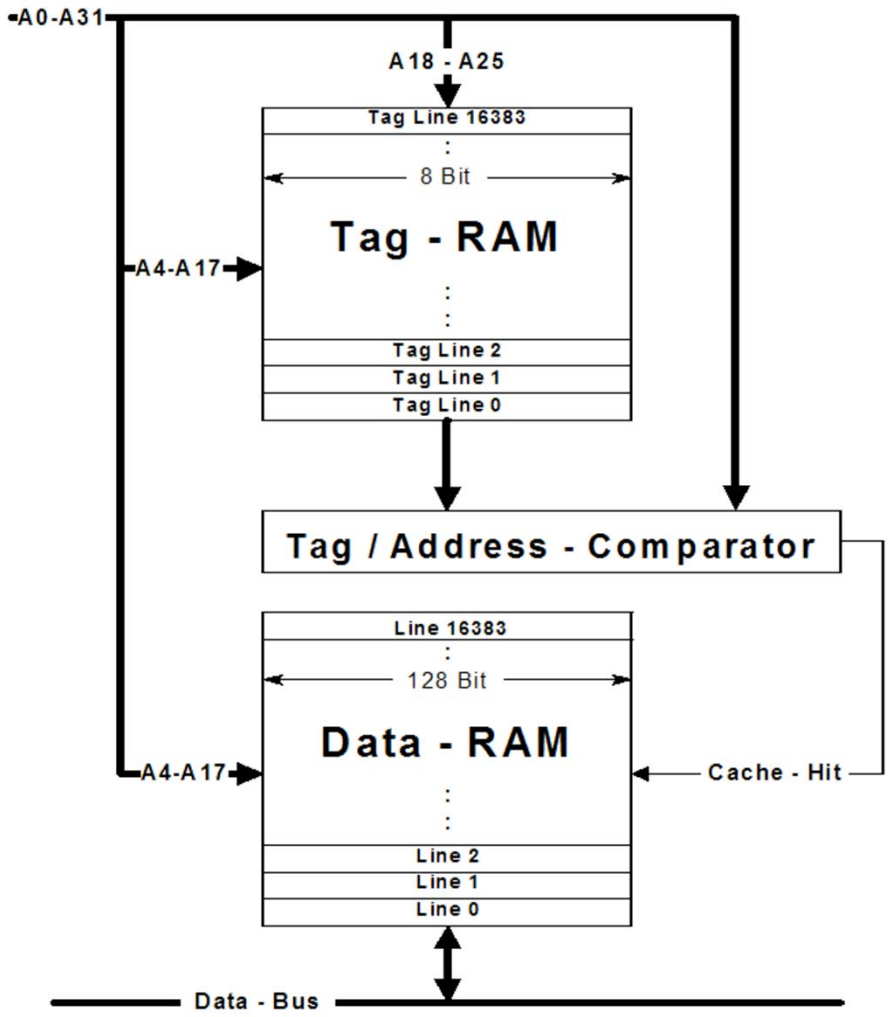
4.1 Voll-assoziativer Cache

- jede Line kann Daten von jeder Stelle des Haupt-speichers enthalten
- hoher Vergleicher-aufwand
- nur für sehr kleine Caches geeignet



4.2 Direct-Mapped Cache

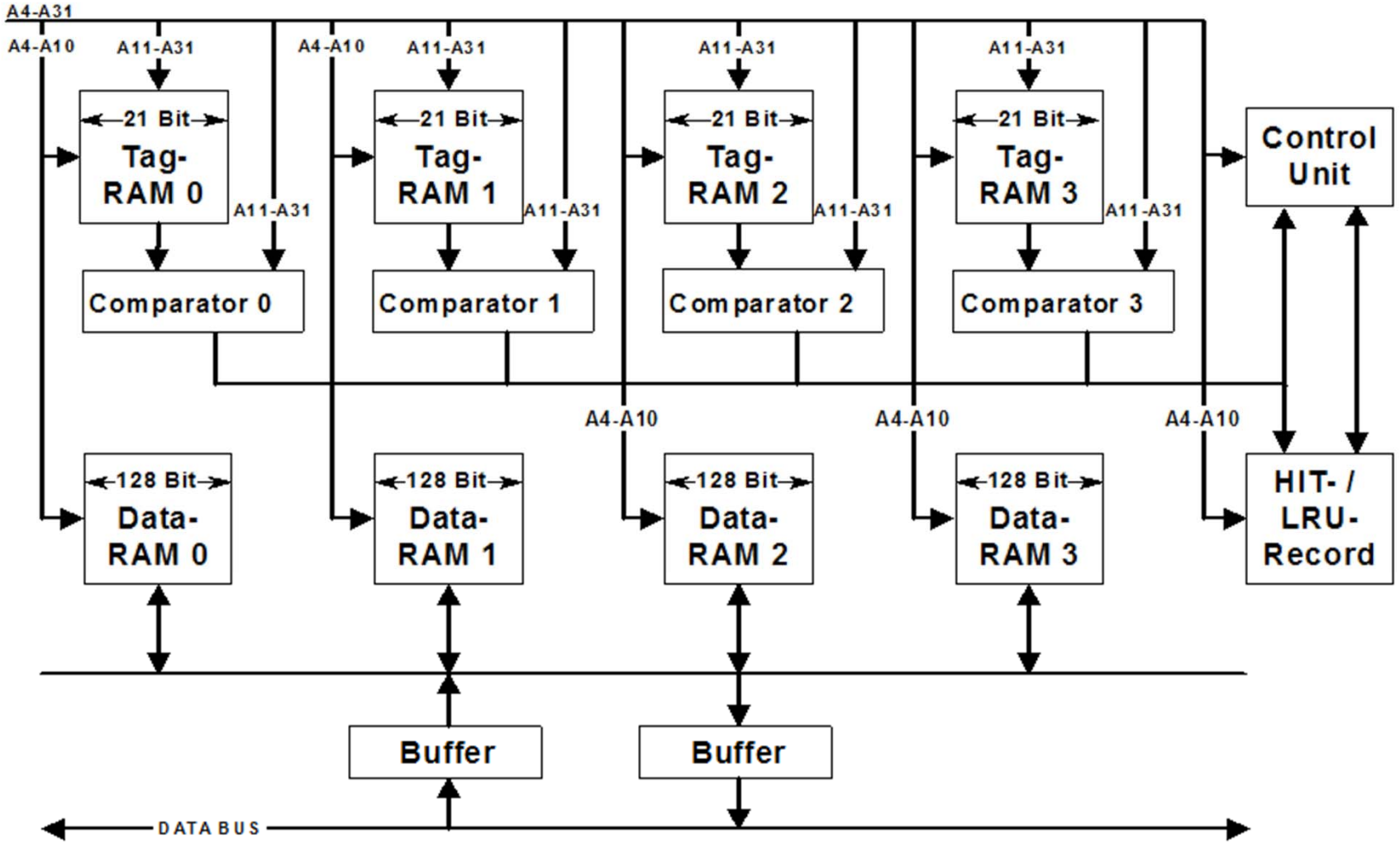
- untere Adressbits dienen direkt der Adressierung der Cache Line
- einfacher Aufbau
- Wettbewerb der potentiellen Lines um jeden Cache-Eintrag = Gefahr von Trashing



4.3 Mehrfach-assoziativer Cache

- z.B. vierfach-assoziativ
  = 4-way set-associative
- d.h. vier Sets pro Line, vergleichbar vier Direct-mapped Caches parallel
- erträglicher Vergleicheraufwand
- hohe Effektivität auch bei kleineren Cache-Größen
- benötigt Update-Algorithmus (z.B. LRU)

4.3 Mehrfach-assoziativer Cache (Forts.)



4.3 Mehrfach-assoziativer Cache (Forts.)

- Cache-Größe 8 KB (4 x 2 KB)
- cacheable Area 4 GB
  - Tag RAM arbeitet bis A31
- Line-Größe 16 Byte
- 512 K Memory-Lines, die im Wettbewerb liegen ($2^{21}/4$, A10-A4 gleich, A21-A31 verschieden)
- Beispiel aus L1-Cache eines i486

5 Cache-Strategien

- Write Through (auch Write Thru)
  - Caching nur für das Lesen (Write direkt in den Speicher)
  - bei Cache Read Hit lesen aus dem Cache (gesuchtes Datum, meist die ganze Line per *Burst*)
  - bei Cache Read Miss nachladen (in Cache und CPU) und aktualisieren des Tag RAM
  - bei Cache Write Hit schreiben in den Speicher und Update des Cache (*Kohärenz*)
  - bei Cache Write Miss kein Cache Update
  - Abhilfe gegen Geschwindigkeitsverluste beim Schreiben und Reduzierung der Busbelastung:
    - Buffered Write Through oder Posted Write Cache

5 Cache-Strategien (Forts.)

- Write Back (auch Write Copy)
  - Caching teilweise auch für Schreiben
  - bei Cache Read Hit lesen aus dem Cache (wie Write Through)
  - bei Cache Read Miss evtl. retten der alten Line (*Cast-Off*), dann erst nachladen
  - bei Cache Write Hit nur Update des Cache, Setzen eines *Dirty-* oder *Modify*-Bits (für Cast-Off)
  - bei Cache Write Miss kein Cache Update, nur schreiben in den Hauptspeicher

Zusammenfassung

- Warum Cache-Speicher?
  - Speicherhierarchie, kurze Zykluszeiten der CPUs, transparente Zwischenspeicherstufe, hält häufig benötigte Daten/Befehle bereit
- Cache-Strukturen
  - Look-aside Cache, Look-through Cache, kombinierte oder getrennte Befehls/Datencaches, Mehrfach-Caches
- Architektur von Caches
  - Tag RAM, Data RAM, Controller, Comparator
  - voll-assoziativ, direct-mapped, mehrfach-assoziativ
- Strategien
  - Write Through & Write Back

Zusammenfassung

- Cache-Speicher sind aus modernen, schnellen Prozessorsystemen nicht mehr wegzudenken
- Struktur, Architektur und Strategie eines Caches müssen immer wieder auf den Einzelfall optimiert werden

Vielen Dank für Ihre Aufmerksamkeit !

ENDE