Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Hardwarenahe Softwareentwicklung

Cache, MMU, DMA

V5.1, ©2023 roger.weber@bfh.ch

Lernziele

Sie sind in der Lage:

- Die Funktion von Cache, MMU und DMA zu erklären.
- Den Einfluss eines Caches auf die Programmausführungszeit zu messen.

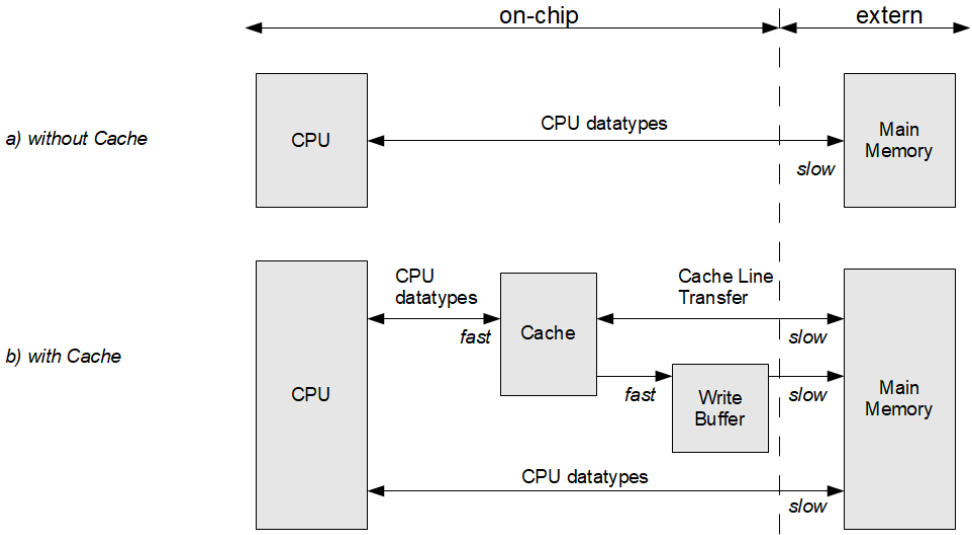
Inhaltsverzeichnis

Cache

Aufgaben Cache

- Aus dem PC-Bereich kennen Sie den Begriff Cache.
- Was sind die Aufgaben des Caches?

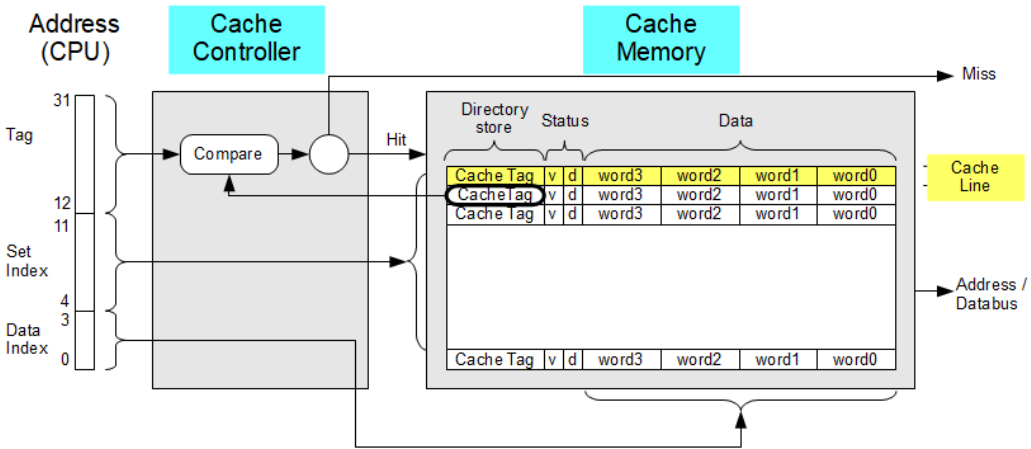
Cache


on-chip
extern
a) without Cache
CPU
CPU datatypes
Main Memory
slow
b) with Cache
CPU
CPU datatypes
fast
Cache
Cache Line Transfer
slow
fast
Write Buffer
slow
Main Memory
CPU datatypes
slow

Cache Ablauf

- **Cache Miss**
  - Daten oder Programmcode sind noch nicht im Cache
  - Speicher → Cache → CPU, langsam
- **Cache Hit**
  - Daten oder Programmcode sind im Cache
  - Beispielsweise Code-Schleifen, Array von Daten
  - Cache → CPU, schnell
- Modifizierte Daten werden über den Write-Buffer ins Main Memory zurückgeschrieben.
- Von Neumann Architektur: 1 Cache für Code und Daten
- Harvard Architektur: 1 Cache für Daten, 1 Cache für Code

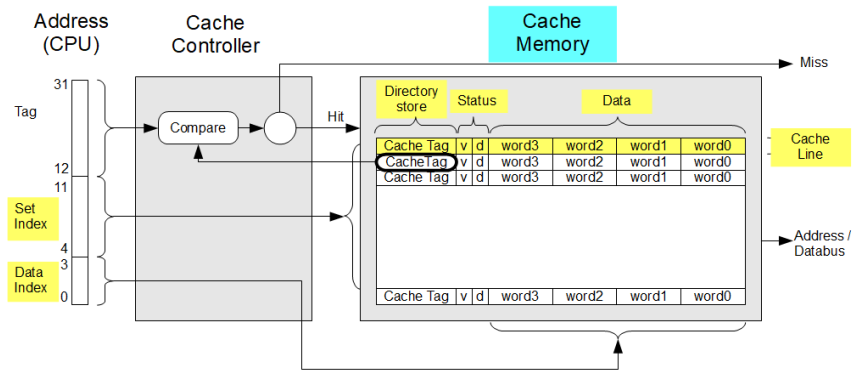
Cache Architektur



Cache Memory

Das Cache Memory besteht aus folgenden Elementen:

- **Directory Store** → Cache Tag gibt an, aus welchem Bereich im Main Memory die Information im Cache Memory stammt.
- **Status** Information → Zustand der Cache Line (z.B. valid, dirty)
- **Data** Section → Daten oder Programmcode



Cache Memory

- Wie gross ist im im Beispiel auf der vorangehenden Folie das Cache-Memory in Byte?
- Wie gross ist das Cache-Memory, wenn der Data-Index 5 Bit und der Set-Index 10 Bit ist?

Cache Controller

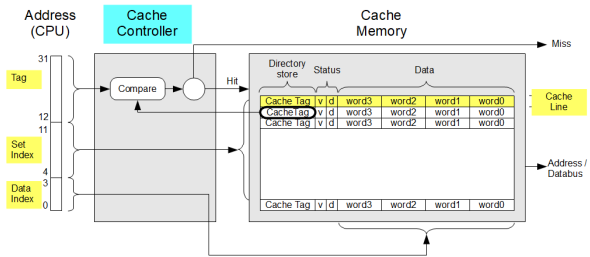
- In Hardware implementiert, für den Anwender “transparent“.
- Prüft, ob Daten oder Programmcode im Cache vorhanden sind.
- CPU-Adresse wird in drei Felder aufgeteilt:
  - Tag Field → wird mit Cache Tag im Cache Memory verglichen
  - Set Index Field → lokalisiert die Cache-Line
  - Data Index Field → selektiert Byte / Halfword / Word in der Cache-Line

Bei Cache Miss:

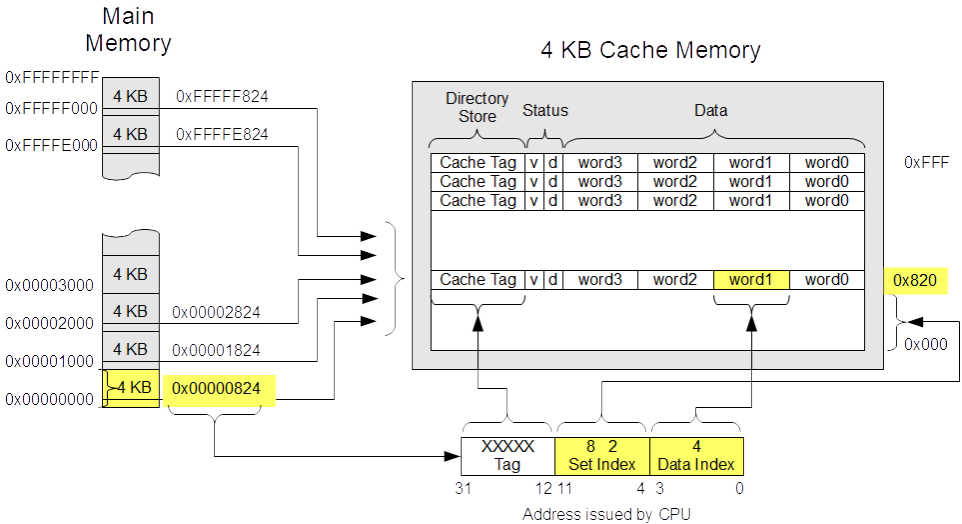
- Verwerfen einer aktuellen Cache Line, ev. Daten zurückschreiben (Writeback)
- Kopieren der neuen Cache Line vom Programm- / Datenspeicher

Bei Cache Hit:

- Cache Memory → CPU



Direct Mapped Cache



Cache Policy und Write Buffer

Write Buffer:

- Zusätzlicher FIFO-Buffer für das Schreiben der Daten ins Main Memory
- Befreit CPU vor langsamem Schreibzugriff auf den Speicher

Cache Policy:

- Writeback:
  - Cache Controller schreibt Daten nur ins Cache Memory
  - Daten im Cache Memory und im Main Memory sind nicht konsistent
  - Daten müssen vom Cache Memory ins Main Memory kopiert werden

Writeback

→ schnell, aber inkonsistente Daten

- Writethrough
  - Cache Controller schreibt Daten ins Cache Memory und ins Main Memory

Writethrough

→ schlechtere Performance (langsamer), aber konsistente Daten

MMU

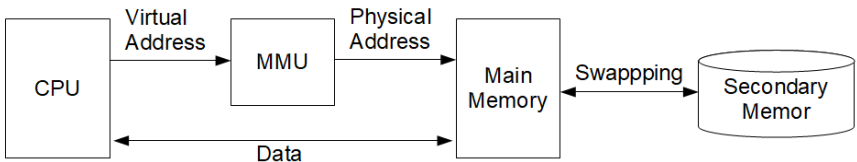
Denksportaufgabe

- Was Sie schon kennen: Auf einer CPU läuft nur eine Applikation, die Applikation startet auf Adresse 0.
- Wie funktioniert das auf einem PC, auf welchen Adressen starten die einzelnen Applikationen?

Memory Management Unit (MMU)

Aufgaben der MMU:

- Speicherverwaltung
- Address Translation: Zuordnen der virtuellen Adressen zu physikalischen Adressen.
- Zugriffschutz: Schutz von Speicherbereichen einzelner Programme.
- Swapping: Nicht benutzter Code oder Daten wird auf den Massenspeicher ausgelagert.

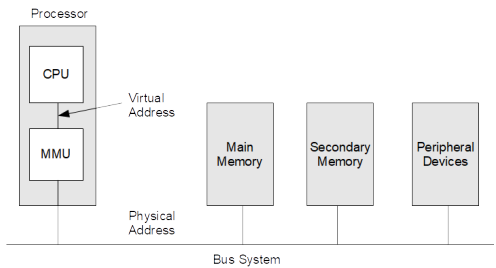


Virtuelle und physikalische Adressen

- **Virtuelle oder logische Adressen**
  Adressen, die der Compiler / Linker einem Programm zuweist und das Programm auch während der Ausführung auf der CPU erhält.
- **Physikalische Adressen**
  Adressen, die bei der Ausführung des Programms von der CPU (oder von der MMU) auf den Adressbus ausgegeben werden.

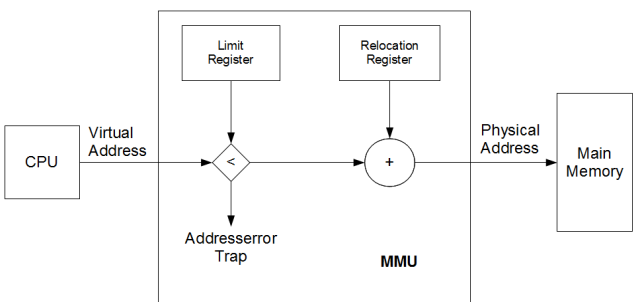
Merke

→ Programme und CPU arbeiten mit virtuellen Adressen, der Zugriff auf die Speicherbausteine erfolgt über die physikalischen Adressen.



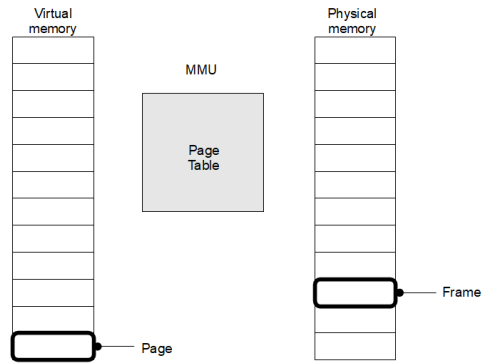
Vereinfachter Aufbau einer MMU

- Zuordnung virtuell → physikalisch: **Relocation**
- MMU addiert zur virtuellen Adresse den zugehörigen Wert aus dem **Relocation-Register** → physikalische Adresse.
- MMU prüft, ob der Zugriff in einem erlaubten Bereich stattfindet. → **Limit-Register**



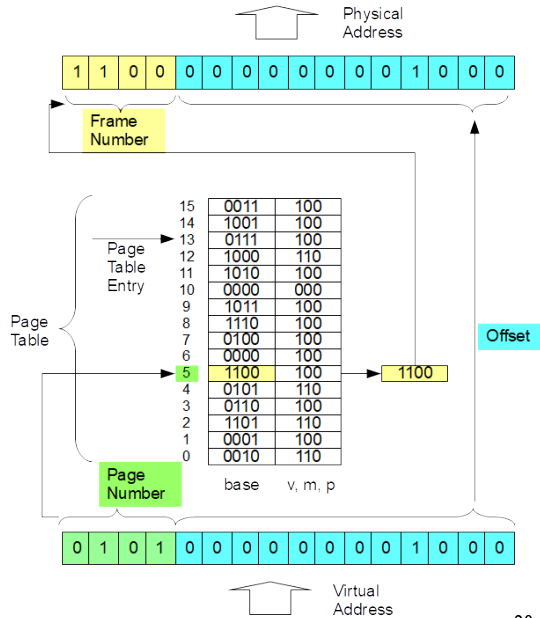
Pages und Frames, Page Table Übersicht

- Der virtuelle Adressbereich wird in **Pages** unterteilt (Blöcke gleicher Grösse).
- Der physikalische Adressbereich wird in **Frames** unterteilt.
- Grössen der Blöcke: 512 Bytes bis 16 MBytes.
- Die Umrechnung der virtuellen Adresse in die physikalische Adresse erfolgt über die **Page-Table**.



Page Table

- Die virtuelle Adresse wird wie folgt aufgeteilt:
  - **Page-Number** Sie gibt den Index in der Page-Table an. Die Page-Table enthält den Eintrag der (physikalischen) Basis-Adresse jedes Frames.
  - **Page-Offset** Er wird zum Inhalt der Page-Table (Basis-Adresse) addiert. Die Summe ist die physikalische Adresse.
- Page Tables werden vom Betriebssystem im RAM angelegt.
- Pro Applikation eine Page-Table.
- v bit (valid): gibt an, ob die physikalische Adresse, auf welche der Inhalt der Page-Table zeigt, gültig ist.

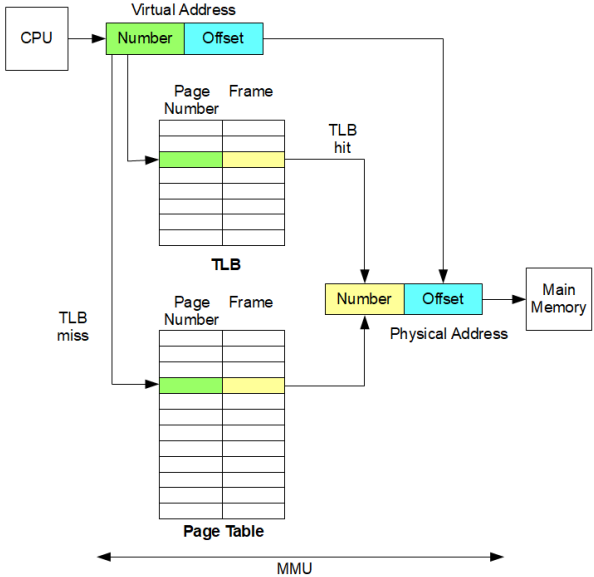


TLBs – Translation Lookaside Buffers

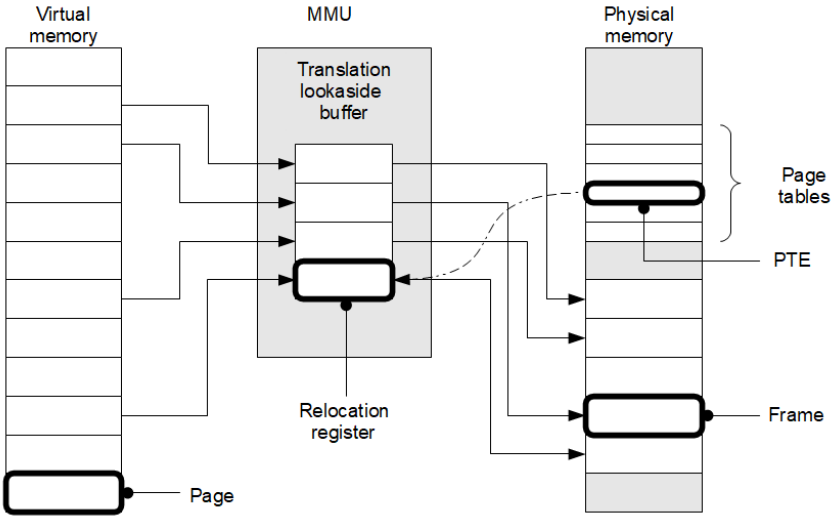
Page-Table:

- Zugriff auf Page-Table ist langsam (RAM).
- Programmlauf: nur wenige Einträge aus der gesamten Page-Table werden verwendet (Gründe: Programmschleifen, Daten in Arrays usw.).
  $\rightarrow$ Verwendung von TLBs
- Die Berechnung der physikalischen Adresse wird durch den TLB optimiert.
- TLB: eine Art Cache-Einheiten für die Speicherzuordnung
  Im TLB stehen immer die zuletzt verwendeten Page-Number.
- In der MMU integriert $\rightarrow$ Hardware $\rightarrow$ schnell
- Tabelle mit typischerweise 8 bis 64 Einträgen, welche den Einträgen in der Page Table entsprechen.
- 1 Eintrag in der TLB = 1 Relocation Register

MMU mit Page-Table und TLB



Übersicht MMU



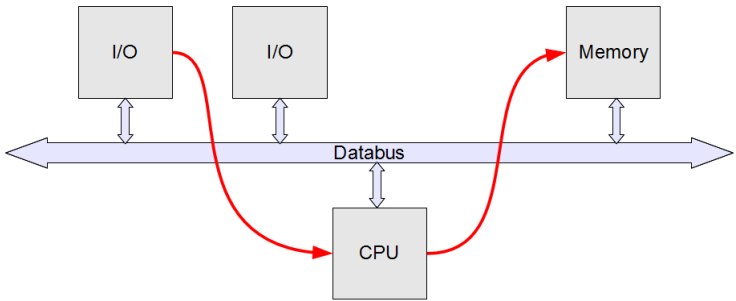
Denksportaufgabe

- Wie gross ist die Page-Table für eine 32-Bit CPU in Kilobyte, wenn eine Page 0.5 Megabyte gross ist? Nehmen Sie als Vereinfachung an, dass ein Page-Table Entry 32 Bit ist. Hilfestellung: Berechnen Sie zuerst die Anzahl Einträge (Entry) der Page-Table.

DMA

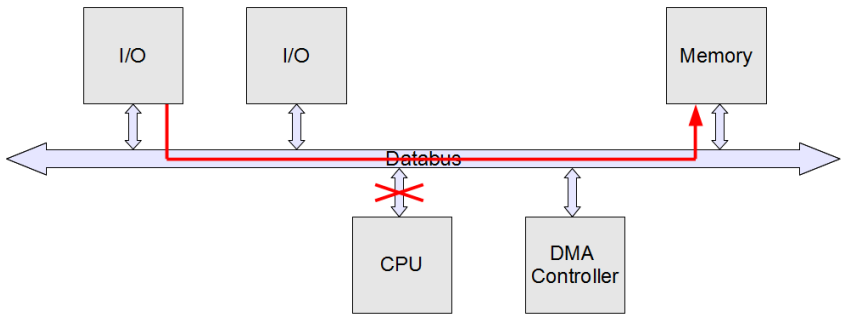
Datentransfer ohne DMA (Direct Memory Access)

- Daten werden von der Peripherie via CPU ins RAM kopiert.
- CPU verwaltet Quelladresse und Zieladresse.
- Ein oder mehrere Register der CPU werden als Zwischenspeicher benötigt.
- Der Transfer muss mit einzelnen Instruktionen programmiert werden.
- Um ein Datenwort zu kopieren, müssen mehrere Buszyklen ausgeführt werde. Dadurch wird der Transfer langsam.



Datentransfer mit DMA

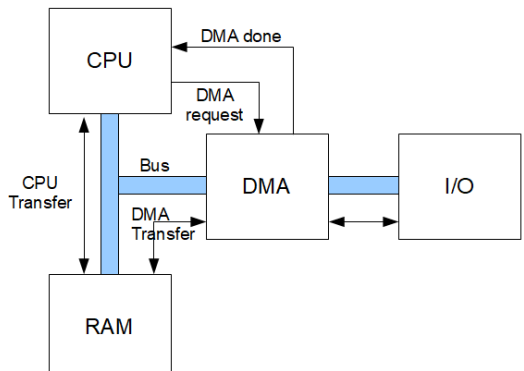
- Daten können von der Quelle zum Ziel ohne Verwendung der CPU kopiert werden.
- Der DMA-Controller übernimmt den Kopiervorgang.



Ablauf Datentransfer mit DMA

Beispiel Datentransfer I/O → RAM

- CPU sendet DMA request inklusive Angaben wie Quelladresse, Zieladresse und Datenmenge an die DMA.
- DMA übernimmt Datentransfer inklusive Adress-Generierung.
- Nach dem Transfer generiert die DMA einen Interrupt für die CPU.
- CPU ist währende dem Datentransfer frei und kann sonstige Berechnungen machen.



DMA

- Der DMA-Controller ist ein komplexer Hardwarebaustein, oft im Prozessor integriert.
- Arbitrierungsverfahren, um Buskonflikte zwischen CPU und DMA-Controller zu vermeiden.
- 2 Adressierungsverfahren für DMA-Controller:
  - Explicite Adressing: 2 Buszyklen, DMA-Controller kopiert Daten in internes Register $\rightarrow$ Datentransfer zwischen Speicherbereichen.
  - Implicite Adressing: Daten werden direkt kopiert, nur 1 Buszyklus. $\rightarrow$ Datentransfer zwischen Speicher und Peripherie