Lernzettel

Speicherhierarchie und Adressierung: Register, Cache, Hauptspeicher, Adressräume

Universität: Technische Universität Berlin
Kurs/Modul: Technische Grundlagen der Informatik
Erstellungsdatum: September 19, 2025

Lernzettel: Speicherhierarchie und Adressierung: Register, Cache, Hauptspeicher, Adressräume

(1) Grundprinzip der Speicherhierarchie. Computer arbeiten mit einer hierarchischen Speicherorganisation:

- Register: ultraschneller, aber sehr limitierter Speicher direkt in der CPU.
- Cache (L1/L2/L3): schneller, aber begrenzter Speicher nahe an der CPU; dient als Zwischenspeicher zwischen Register- und Hauptspeicher.
- Hauptspeicher (RAM): größer, langsamer als Cache; hält aktive Programme und Daten.
- Sekundärer Speicher: deutlich langsamer, aber sehr groß (SSD/HDD); langfristige Speicherung.

(2) Register. Register sind die unmittelbarsten Speicherorte der CPU und speichern Operanden, Adressbestandteile oder Statusinformationen.

- Typen: Allgemeinregister (GPR), spezielle Register (PC, Statusregister, Instruktions-Zähler, Stackpointer).
- Eigenschaften: extrem niedrige Latenz, sehr geringe Kapazität, direkter Zugriff durch den Prozessor.

(3) Cache. Cache-Speicher dient dazu, die Lücke zwischen Registerzugriffen und Hauptspeicher zu schließen.

- Aufbau: Cache-Block (Cache-Line) als kleinste transferierbare Einheit.
- Organisation: direct-mapped, fully associative oder set-associative (z. B. 4-way set-associative).
- Adressaufteilung (typisch):

$$A = [\text{Tag} \mid \text{Index} \mid \text{Offset}]$$

$$\text{Blockgröße} = 2^b \text{ Byte} \quad (\text{Offset} = b)$$

$$\text{Indexbreite} = i, \quad \text{Tag} = \text{AdresseBreite} - (i + b)$$

$$\text{Beispiel (32-Bit-Adresse): Tag} = 32 - (i + b) \text{ Bits}$$

(4) Hauptspeicher (RAM). Hauptspeicher hält die laufenden Programme und Daten in einem deutlich größeren Speicherbereich als der Cache.

- Typen: DRAM (häufigster Typ); schnelle Cache-Ankopplung über Memory-Controller.
- Eigenschaften: Byteadressierbar, Latenzen im Bereich von einigen Dutzend Nanosekunden; Bandbreite wird durch Busbreite und Taktfrequenz bestimmt.
- Adressraum: abhängig von der Adressbreite des Systems (z. B. 32-Bit oder 64-Bit).

(5) Adressräume und Adressierung. Unterscheidung zwischen logischen Adressen (virtuelle Adressen) und physischen Adressen.

- Virtueller Adressraum vs. physischer Adressraum.
- Paging/Seitentabellen: virtuelle Adressen werden durch einen MMU in physische Adressen übersetzt.
- Seitentabellen und TLB (Translation Lookaside Buffer) beschleunigen die Übersetzung.
- Adressaufteilung (Paging):

$$\text{Virtuelle Adresse} = [\text{VPN} \mid \text{Offset}]$$

$$\text{Physische Adresse} = [\text{PPN} \mid \text{Offset}]$$

- Seitengröße typischerweise 4 KB (Offset=12 Bits); größere Seiten (4 MB, 2 MB) möglich.

(6) Adressierung und Cache-Kohärenz. Beim Zugriff auf Daten existieren analog zur Cache-Zuordnung Adressfelder:

$$\text{Adresse} = [\text{Tag} \mid \text{Index} \mid \text{Offset}]$$

Der Compiler/Bus-Controller sorgt dafür, dass aktuell verwendete Adressräume konsistent bleiben (Cache-Kohärenz zwischen mehreren Caches/Bänken).

(7) Leistungskennzahlen.

$$\text{Latenz} = \text{Zeit für einen Zugriff (z. B. } t_{\text{L1}})$$

$$\text{Throughput} = \frac{\text{Anzahl gelesener Wörter}}{\text{Zeit}}$$

$$\text{Missrate} = 1 - \text{Hit-Rate}$$

(8) Beispielhafte Adressaufteilung und Übersetzungsüberlegung. Angenommen, ein System besitzt:

- Adressbreite des virtuellen Adressraums: 32 Bit.
- Page-Size: 4 KB ($2^1 2 Byte$).$VirtuelleAdresseVA\%2^{32}$. Dann gilt:

$$\text{Offset} = \text{VA mod } 2^{12} = \text{untere 12 Bits von VA}$$

$$\text{VPN} = \left\lfloor \frac{\text{VA}}{2^{12}} \right\rfloor$$

Der MMU übersetzt VPN mittels Seitentabell und ggf. TLB in eine physische Adresse.

$$\text{Physische Adresse} = [\text{PPN} \mid \text{Offset}]$$

Der Cache greift zusätzlich über die Adresse in Form von

$$A = [\text{Tag} \mid \text{Index} \mid \text{Offset}]$$

auf die entsprechende Cache-Line zu.

(9) Zusammenhang zwischen Hierarchie und Programmiermodell.

- Programme arbeiten mit logischen Adressen; der MMU übersetzt sie in physische Adressen.
- Caches beschleunigen Zugriff auf häufig verwendete Datenstrukturen, z. B. Arrays und lokale Variablen.
- Das Betriebssystem verwaltet den Adressraum, Speicherschutz und Auslagerung über Paging/Segmentierung.

(10) Kurze Übung (Gedankenanstoß). - Skizziere kurz, wie ein Zugriff auf eine Speicherstelle durch mehrere Ebenen der Speicherhierarchie geht (Register -> Cache -> RAM). - Beschreibe, wie Standort- oder Adresskonventionen die Effizienz beeinflussen (z. B. Cache-Alignment, Blockgröße).