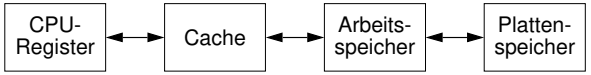
4.2 Pufferspeicher (Caches)

4.2.1 **Speicherhierarchie**



- Zeitliche Lokalität

  Wenn ein Speicherort angesprochen wird, dann wird er innerhalb eines kleinen Zeitintervalls mehrfach angesprochen

- Örtliche Lokalität

  Wenn ein Speicherort angesprochen wird, dann werden innerhalb eines kleinen Zeitintervalls überwiegend Speicherorte mit geringem Adreßabstand angesprochen

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Beispiel

```
int M = 10000;
int main()
{ int i, j, k;
  int a[M][M], b[M][M], c[M][M];
  for (i = 0; i < M; i++)
    for (j = 0; j < M; j++) {
      a[i][j] = i; b[i][j] = j; c[i][j] = 0;
    }

  for (i = 0; i < M; i++)
    for (j = 0; j < M; j++)
      for (k = 0; k < M; k++)
        c[i][j] =
             c[i][j] + a[i][k] * b[k][j];

}
```

`c[i][j]` weist zeitliche Lokalität auf

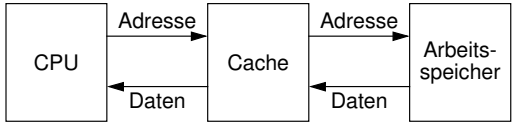
`a[i][j]` weist örtliche Lokalität auf

`b[i][j]` weist keines von beidem auf

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

4.2.2 **Grundlagen**

- Zugriff



  - Fehlzugriff (cache miss)
  - Treffer (cache hit)
  - Trefferrate (hit ratio)
  - Cache-Zeile (cache line)
- Virtuelle oder physikalische Adressen?
- Suchen im Pufferspeicher

  Verwendung von Hash-Techniken
- Ersetzungsstrategien

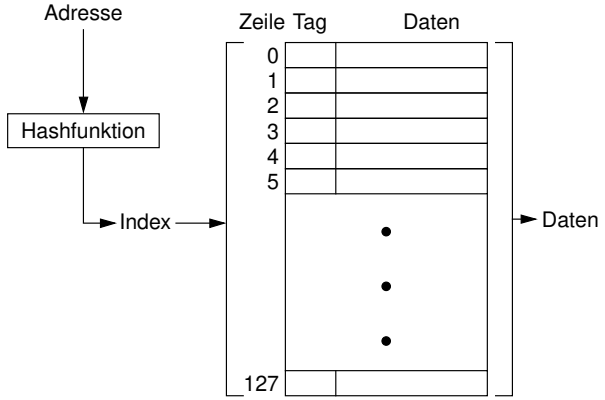
  Typischerweise LRU oder Approximationen von LRU, aber auch zufällige Auswahl

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Schreibstrategien
  - Treffer
    - write-through: Arbeitsspeicher immer auf dem neuesten Stand, zuweilen unnötige Verlangsamung
    - write-back: schreibt nur in den Cache; als Folge entsteht ein zeitweise inkonsistenter Arbeitsspeicherinhalt; erfordert besondere Maßnahmen beim Laden von Cachezeilen.
  - Fehlzugriff
    - write-allocate: erst (soweit erforderlich) aus Arbeitsspeicher lesen, dann schreiben mit einer der beiden vorangehenden Strategien.
    - write-to-memory: Modifikation wird nur im Arbeitsspeicher vorgenommen.

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

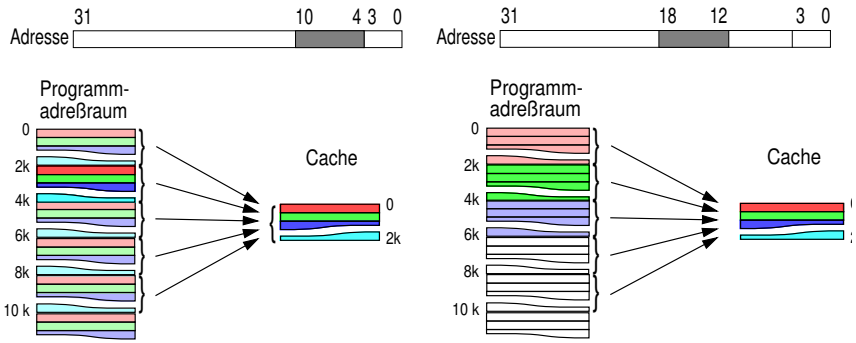
4.2.3 Direkt abgebildete Pufferspeicher



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ Hash-Algorithmen

- Modulo-Hashing, z. B. Bits 0 - 3 zur Auswahl des Bytes innerhalb einer Zeile, Bits 4 - 10 zur Auswahl der Zeile, Bits 11 - 31 sind Tag
- Hashing durch Ausschnitt aus der Adresse, z. B. Bits 12 - 18 als Zeilenadresse

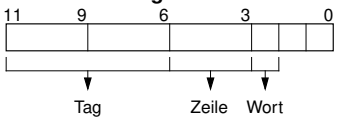


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ Beispiel

- 12 Bit breite Adressen
- Worte zu 4 Byte, ausgerichtet an 4-Byte-Grenze
- Cache mit 8 Zeilen; 8 Bytes pro Cache-Zeile

Aufteilung der Adresse



Adressbeispiele

Adresse	Tag	Zeilen-index	Wort	Ergebnis-daten
01234	012	3	1	0777
01230	012	3	0	052
00130	001	3	0	miss
03574	035	7	1	0
06540	065	4	0	miss

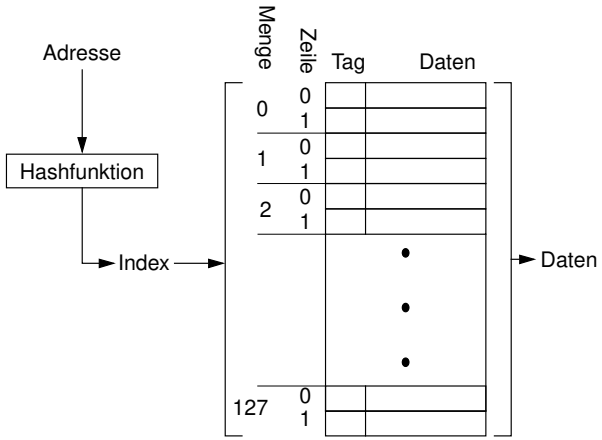
Cache

Zeile	Tag	Daten	
0	---		
1	---		
2	---		
3	012	052	0777
4	---		
5	---		
6	---		
7	035	067	0

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

4.2.4 Zweifach assoziative Pufferspeicher



Beispiel: Daten-Cache des Pentium

4.2.5 Vollständig assoziative Pufferspeicher

Gesamter Cache wird behandelt wie **eine** Menge

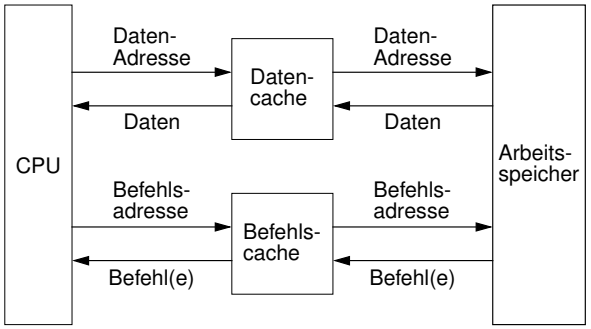
4.2.6 Leeren des Pufferspeichers (cache flushing)

- Alle Realisierungen sehen Möglichkeit zur zwangsweisen Leerung des gesamten Cache oder einzelner Zeilen vor
  - Aktualisierung des Arbeitsspeichers
  - Invalidierung des Cache
- Wird benötigt zur Konsistenzsicherung

4.2.7 Ungepufferte Operationen

- Meist wählbar auf Seitenbasis durch geeigneten Indikator in der Seiten-Kachel-Tabelle
- Wichtig für Speicherbereiche, die sich unabhängig von Schreibaufrufen ändern können (*volatile* in C und C++), z. B. in den Speicher abgebildete Statusregister von E/A-Geräten

4.2.8 Getrennte Pufferspeicher für Daten und Befehle



4.2.9 Leistung von Pufferspeichern

- Zeitliche Lokalität spricht für write-back
- Für kleine Caches spricht für mehrfach-assoziative Vorgehensweisen mit großen Mengen
- Adreßlokalität spricht für große Zeilenlänge
- Erreichbare Leistungssteigerung durch Cache wird wesentlich vom Betriebssystem beeinflußt

4.2.9.1 Unterschied in der Realisierung von Pufferspeichern

- Größe
- Zeilengröße
- Mengengröße
- Nutzung von ‘write-allocate’
- Ersetzungsstrategie
- Aufsuchen mit virtueller oder physikalischer Adresse
- Kennzeichnung der Zeilen
- ‘write-through’ oder ‘write-back’