Lösung Aufgabenblatt 9: Cache 2

1 LRU-Verfahren

Gegeben: Cache 4-fach assoziativ, 8 Sets á 4 Cache-Lines, 1 Byte pro Cache-Line. Verdrängungsstrategie: LRU (*Least Recently Used*). Jedes Set besitzt seine eigene LRU-Matrix. Zugriffsreihenfolge auf Adresse: 0, 8, 16, 24, 16, 0, 32, 48. *Annahme:* am Anfang ist der Cache leer.

Verfahren: Bei einem Treffer in der oder bei einer Einlagerung in die i-te ($i \in \{0, \ldots, 3\}$) Cache-Line eines Set wird die i-te Spalte mit '1' und die i-te Zeile mit '0' beschrieben. Muss Platz im Set geschaffen werden, wird die Cache-Line verdrängt, deren Zeile (binär interpretiert) die grösste Zahl (oder deren Spalte die kleinste Zahl) enthält.

Die Daten der Adressen 0, 8, 16, 24, 32 und 48 werden bei 4-facher Assoziativität alle in Set 0 zwischengespeichert. Das jeweilige Tag ergibt sich folgendermaßen:

$$Tag = \frac{Adr.}{\#Set}$$

Damit ergeben sich die folgenden Inhalte der Matrix für Set 0 nach dem jeweiligen Zugriff:

Adr. 00 → Set 0
Tag 0
Cache Line 0

Cache Line	0	1	2	3	Valid	TAG
0	0	0	0	0	1	0
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	1	0	0	0	0	0

nach Zugriff auf Adresse 0

Adr. 08 → Set 0
Tag 1
Cache Line 1

Cache Line	0	1	2	3	Valid	TAG
0	0	1	0	0	1	0
1	0	0	0	0	1	1
2	1	1	0	0	0	0
3	1	1	0	0	0	0

nach Zugriff auf Adresse 8

Adr. 16 → Set 0
Tag 2
Cache Line 2

Cache Line	0	1	2	3	Valid	TAG
0	0	1	1	0	1	0
1	0	0	1	0	1	1
2	0	0	0	0	1	2
3	1	1	1	0	0	0

nach Zugriff auf Adresse 16

Adr. 24 → Set 0
Tag 3
Cache Line 3

Cache Line	0	1	2	3	Valid	TAG
0	0	1	1	1	1	0
1	0	0	1	1	1	1
2	0	0	0	1	1	2
3	0	0	0	0	1	3

nach Zugriff auf Adresse 24

Adr. 16 → Set 0
Tag 2
Hit
Cache Line 2

Cache Line	0	1	2	3	Valid	TAG
0	0	1	1	1	1	0
1	0	0	1	1	1	1
2	0	0	0	0	1	2
3	0	0	1	0	1	3

nach Zugriff auf Adresse 16

Adr. 00 → Set 0
Tag 0
Hit
Cache Line 0

Cache Line	0	1	2	3	Valid	TAG
0	0	0	0	0	1	0
1	1	0	1	1	1	1
2	1	0	0	0	1	2
3	1	0	1	0	1	3

nach Zugriff auf Adresse 0

Adr. 32 → Set 0
Tag 4
Cache Line 1

Cache Line	0	1	2	3	Valid	TAG
0	0	1	0	0	1	0
1	0	0	0	0	1	4
2	1	1	0	0	1	2
3	1	1	1	0	1	3

nach Zugriff auf Adresse 32

Adr. 48 → Set 0
Tag 6
Cache Line 3

Cache Line	0	1	2	3	Valid	TAG
0	0	1	0	1	1	0
1	0	0	0	1	1	4
2	1	1	0	1	1	2
3	0	0	0	0	1	6

nach Zugriff auf Adresse 48

2. Programmierung

Gegeben: Cache mit 16 KByte für Code und Daten, direct mapped (→ Assoziativität = 1), Cache-Line ist 1 Byte gross, m Dateneinträge á 512 Byte, eine Codeschleife von 1 KByte Grösse betrachtet jedes Element, zweite Codeschleife wird 100mal durchlaufen.

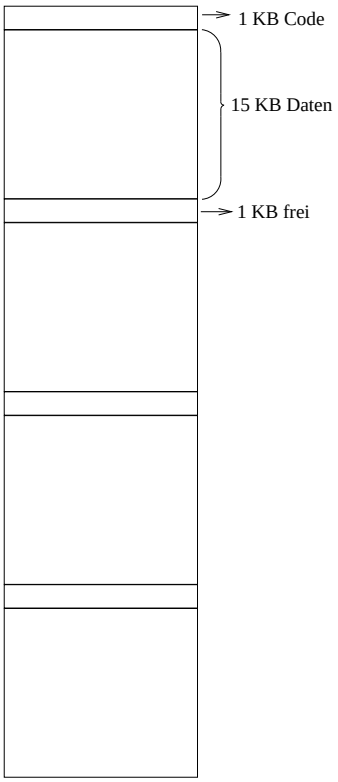

Abbildung 1: Hauptspeicher

1. Wie müssen die Schleifen geschachtelt sein?
   Der Code für die innere Schleife sollte immer im Cache bleiben. Die innere Schleife betrachtet jedes Element 100mal nacheinander $\Longrightarrow$ ab dem zweiten Zugriff auf dasselbe Element (in der inneren Schleife) gibt es Treffer.

2. Wie sollten Daten und Programm im Speicher angeordnet sein?
   Bei *direct mapped* wird der Hauptspeicher in 16 KByte Blöcke unterteilt (siehe auch *Abb. 1*). Im ersten (oder zweiten oder dritten oder ... 16.) Kilobyte des Cache sollte die innere Codeschleife zwischengespeichert werden $\Longrightarrow$ jeweils das erste (oder zweite oder dritte oder ... 16.) Kilobyte eines Haupspeicherblocks darf nicht von Daten genutzt werden. $\Longrightarrow$ 30 Elemente á 512 Byte können in 15 KByte zwischengespeichert werden. Danach muss eine 1 KByte ungenutzt (Lücke) bleiben, und dann wieder Daten.

3. Wie sollten die Daten und das Programm angeordnet werden, wenn man einen 4-fach assoziativen Cache verwendet?
   Ab der Assoziativität 2 können Programm und Daten beliebig angeordnet werden.