Grundlagen der Informatik III

Wintersemester 2010/2011

Wolfgang Heenes, Patrik Schmittat

TECHNISCHE UNIVERSITÄT DARMSTADT

7. Aufgabenblatt

13.12.2010

Hinweis: Der Schnelltest und die Aufgaben sollen in den Übungsgruppen bearbeitet werden. Die Hausaufgaben sind in der Kalenderwoche 2 (10.01. bis 14.01.) bei den Tutoren in **physikalischer Form** (handschriftlich oder gedruckt) abzugeben. Bei allen Abgaben ist der Name des Tutors und die Übungsgruppe deutlich anzugeben. Bei Teamabgaben wird nur eine Lösung eingereicht, die alle Namen der Teammitglieder enthält.

Aufgabe 1: Schnelltest

Fragen	Antworten
1. Gegeben sei ein vollassoziativer 32 Byte-Cache mit 4 Wort Blöcken und 32 Bit Wortgröße. Auf die Daten im Hauptspeicher wird in folgender Sequenz zugegriffen: `0x001, 0x004, 0x008, 0x004, 0x01E, 0x004, 0x0FF`. Die Ersetzungsstrategie soll LRU sein. Welcher Block wird beim letzten Zugriff der Sequenz ersetzt?	☐ Keiner ☐ 0x000 - 0x00F ☐ 0x010 - 0x01F ☐ 0x0F0 - 0x0FF ☐ 0x100 - 0x10F
2. Ein vierfach satzassoziativer Cache habe 32 Blöcke. Die Wortgröße betrage 32 Bit. Wie groß muss dann der Cache sein (ohne Valid-Bit oder Tag-Bits)?	☐ 256 Bytes ☐ 512 Bytes ☐ 512 KBytes ☐ 32 Bytes ☐ 128 Bytes
3. Welche Aussagen über Caches sind korrekt?	☐ Ein vierfach satzassoziativer Cache ist doppelt so groß wie ein zweifach satzassoziativer Cache. ☐ Je höher die Assoziativität, desto höher ist in der Regel die Miss-Rate. ☐ Je höher die Assoziativität, desto aufwändiger ist das Suchen. ☐ Je höher die Assoziativität, desto mehr Index-Bits sind nötig.
4. Welche Aussagen zur Daisy Chain sind korrekt?	☐ Andere Geräte können „aushungern“. ☐ Andere Geräte können nie „aushungern“. ☐ Subsystem mit höchster Priorität befindet sich am Ende der grant-Kette. ☐ Schnelligkeit, da grant-Signal direkt an E/A-Gerät gesendet wird.

Aufgabe 2: Direct-Mapped Cache

Betrachtet wird ein Direct-Mapped Cache. Folgende Eigenschaften sind gegeben:

- 16-Bit Adressgröße
- Blockgröße 256 Bytes, ansprechbar zu 8 Bit
- 16 Cache-Blöcke (Sets)

a) Ermitteln Sie die Bitbreite von Tag, Index und Blockoffset.

b) Ermitteln Sie für die folgenden Zugriffe, ob es sich um Hits oder Misses handelt. Tragen Sie in die untere Tabelle jeweils ein, wie sich der Cacheinhalt verändert. Der erste Eintrag ist vorgegeben.

Zugriff	0x0D3A	0x017D	0x0A9C	0x2752	0x510B	0x78B3	0x0D8D
Hit/Miss	Miss						

0x0AD0	0x32C7	0x1888	0x4F0E	0x6D6A	0x14B3	0x014B	0x0AA3

Index	Valid	Tag	Daten
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
A			
B			
C			
D	x	0	Mem[0x0D00 - 0x0DFF]
E			
F			

c) Im Folgenden wird ein zweifach satzassoziativer Cache mit denselben Werten wie oben (und derselben Gesamtzahl von Cacheblöcken) betrachtet. Welche Bitbreiten ändern sich hierdurch? Ändert sich die Gesamtgröße des Caches (einschließlich Verwaltungsinformationen)?

d) Inwieweit ist eine solche Variante des Caches „besser“ als die aus Aufgabenteil a)? Würden sich weitere Verbesserungen ergeben, wenn man einen 16-fach satzassoziativen Cache verwenden würde?

Aufgabe 3: Direct-Mapped Cache

Betrachten Sie den folgenden Pseudo-Code:

```
int[][] dst = new int[2][2];
int[][] src = new int[2][2];

... // Hier wird src mit Werten gefuellt

for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        dst[j][i] = src[i][j];
    }
}
```

Gehen Sie davon aus, dass die Elemente von `dst` und `src` der Reihe nach hintereinander im Speicher abgelegt sind und jeder `int`-Wert 4 Bytes benötigt, wobei `src` an Speicheradresse 0 startet und `dst` direkt dahinter an Speicheradresse 16 ($= 10000_2$).

Die Reihenfolge eines Arrays `a[2][2]` im Speicher sei: `a[0][0], a[0][1], a[1][0], a[1][1]`

Der Zugriff auf die beiden Arrays erfolgt über einen direkt abbildenden, byte-adressierten Cache mit Write-Back-Strategie. Die Blockgröße beträgt 8 Byte, die Gesamtgröße 16 Byte.

a) Skizzieren Sie die Aufteilung einer 32-Bit-Adresse in Tag, Index und Blockoffset.

b) Tragen Sie in den folgenden Tabellen ein, welche Arrayzugriffe bei Ausführung des o.a. Codestücks Hits (h) und welche Misses (m) sind.

(a) `src`-Array

	Col 0	Col 1
Row 0		
Row 1		

(b) `dst`-Array

	Col 0	Col 1
Row 0		
Row 1		

Tabelle 3.2: 16-Bit-Cache

c) Betrachten Sie jetzt dieselbe Aufgabenstellung, jedoch mit einem 32-Byte-Cache. Wie ist die Aufteilung von Speicheradressen, wie sehen die Tabellen aus?

(a) `src`-Array

	Col 0	Col 1
Row 0		
Row 1		

(b) `dst`-Array

	Col 0	Col 1
Row 0		
Row 1		

Tabelle 3.3: 32-Bit-Cache

Aufgabe 4: Vollassoziativer Cache

a) Gegeben sei ein Cache mit folgenden Eigenschaften:
   - Vollassoziativ
   - Blockgröße 64 Kilobyte
   - Gesamtgröße (ohne Verwaltungsinformationen) 256 Kilobyte
   - FIFO[1] Ersetzungsstrategie

[1] First In First Out

Weiterhin sei eine Zugriffsfolge in nachfolgender Tabelle gegeben, bei der byteweise lesend auf die Daten im Hauptspeicher zugegriffen wird.

Notieren Sie zu jedem Zugriff, ob es sich um einen Hit (h) oder einen Miss (m) handelt sowie den Zustand der Tag-Felder des Datencaches. Tragen Sie die Werte für die Tag-Felder in Hexadezimalschreibweise ein. Leere Felder versehen Sie bitte mit einem Strich. Gehen Sie davon aus, dass der Datencache zu Beginn leer ist.

Benutzen Sie die folgende Tabelle:

Zugriff (Byteadresse)	Hit/Miss	Tag-Felder des Caches			
		Eintrag 1	Eintrag 2	Eintrag 3	Eintrag 4
0x9A44 D124					
0xA4B3 7486					
0x21CF EA04					
0xA4B3 7976					
0xFE67 D7E3					
0x9A44 8B23					
0x315A 16AD					
0x9A44 D0A0					
0xA4B3 2497					
0xFE67 F10C					
0x21CF 1234					

b) Gegeben sei derselbe Cache wie in der vorherigen Teilaufgabe, jedoch soll in dieser Aufgabe eine LRU-Ersetzungsstrategie verwendet werden. Das heisst, es wird bei einem Cache-Miss derjenige Eintrag ersetzt, auf den am längsten nicht mehr zugegriffen wurde.

Tragen Sie in die nachfolgende Tabelle wieder für jeden Zugriff ein, ob er ein Hit (h) oder ein Miss (m) ist und notieren Sie die Cachebelegung, indem Sie die Tagfelder eintragen:

Zugriff (Byteadresse)	Hit/Miss	Tag-Felder des Caches			
		Eintrag 1	Eintrag 2	Eintrag 3	Eintrag 4
0x9A44 D124					
0xA4B3 7486					
0x21CF EA04					
0xA4B3 7976					
0xFE67 D7E3					
0x9A44 8B23					
0x315A 16AD					
0x9A44 D0A0					
0xA4B3 2497					
0xFE67 F10C					
0x21CF 1234					

Hausaufgabe 1: Vollassoziativer Cache (6 Punkte)

Gegeben sei ein Datencache mit folgenden Eigenschaften:

- Vollassoziativ
- Blockgröße 4 KByte, byte-adressierbar
- 4 Einträge
- LRU Ersetzungsstrategie

a) Wie viele Nutzdaten (d. h. keine Tag- und Verwaltungsbits) kann der Cache aufnehmen? Wie viele Bits besitzt ein Tag-Feld des Caches?

b) Ergänzen Sie die folgende Tabelle. Gehen Sie davon aus, dass der Datencache zu Beginn leer ist.

Zugriff		Tag-Felder des Caches			
—- (Byteadresse)	Hit/Miss	Eintrag 1	Eintrag 2	Eintrag 3	Eintrag 4
0x1234 0001					
0x1A13 0002					
0x1A13 1006					
0x1A13 1876					
0x17D8 0A02					
0x1234 0001					
0x1A13 0002					
0x2478 2174					
0x1998 4446					
0xFA12 060E					
0xD728 8802					
0x213F 2A0B					

Hausaufgabe 2: Bus-Arbitration (4 Punkte)

Mehrere Subsysteme teilen sich einen gemeinsamen Datenbus. Es soll anhand eines einfachen Beispiels die Bus-Arbitration, d. h. die Zuteilung des Busses an die Subsysteme, untersucht werden. Die Bus-Arbitration soll gemäß festen Prioritäten (beispielsweise in Form einer „Daisy-Chain“) erfolgen. Subsystem 1 hat dabei die höchste, Subsystem 3 die niedrigste Priorität. Gehen Sie dabei zunächst von drei Subsystemen mit folgenden Buszugriffsanforderungen aus:

- Subsystem 1 fordert, beginnend in Takt 0, alle vier Takte Zugriff auf den Bus an. Wenn die Zuteilung erfolgt wird der Bus jeweils einen Takt lang belegt.
- Subsystem 2 fordert, beginnend in Takt 1, alle sechs Takte Zugriff auf den Bus an. Wenn die Zuteilung erfolgt wird der Bus jeweils drei Takte lang belegt.
- Subsystem 3 fordert, beginnend in Takt 2, alle acht Takte Zugriff auf den Bus an. Wenn die Zuteilung erfolgt wird der Bus jeweils zwei Takte lang belegt.

a) Wie würde in diesem Fall die Bus-Zuteilung aussehen? Geben Sie in einem Diagramm die Anforderungszeitpunkte so wie die jeweilige Buszuteilung für die ersten 30 Takte an. Den Zeitaufwand für die Bus-Arbitration können Sie dabei vernachlässigen.

b) Angenommen es wird ein Subsystem 4 hinzugefügt welches eine niedrigere Priorität als Subsystem 3 haben soll. Welche Probleme treten auf wenn zusätzlich zur obigen Anforderungsreihenfolge noch Anforderungen eines vierten Subsystems hinzukommen würden?

c) Welches der auftretenden Probleme kann durch eine andersartige Bus-Arbitration verhindert werden? Was müsste hierzu an der Bus-Arbitration geändert werden?