

TI II

Sommersemester 2010
PD Dr. Katinka Wolter

6. Aufgabenblatt

Ausgabe Abgabe
25.06.2010 09.07.2010

Kontakt bei Fragen

Matthias Dräger, Viet Do, Marco Jeschke, Uwe Kuehn, Markus Rudolph
{mdraeger/do/kuehn/mjeschke/rudolph}@mi.fu-berlin.de

Problem 1: Speicherhierarchie (1+4+4=9)

- Warum wird Speicher überhaupt hierarchisiert?
- Welche zwei Strategien werden dabei verwendet? Erläutern Sie diese.
- Welche Speicherebenen existieren in der Regel zwischen der Register- und der Festplattenebene? Welcher Sprung im Zeitverbrauch zwischen den Ebenen ist größenordnungsmäßig der bedeutendste?

Problem 2: Cache-Zugriff (6+1=7)

Gegeben sei ein vollassoziativer Datencache, Größe 1024 KByte, am Anfang leer, mit 64 Byte pro Cache-Zeile. Es werde folgende Programmschleife zur Addition von 128 16-Bit-Zahlen, die in einem Array `g[]` liegen, durchgeführt:

```
int i=0;
for(j=0; j<128; j++)
i = i+g[j];
```

Weitere Annahmen: Bei einem Cache-Miss wird immer die ganze Cache-Zeile in den Cache geladen. Die Daten des Arrays `g[]` liegen im Speicher an direkt aufeinander folgenden Adressen und in der Reihenfolge, in der sie auch von der Schleife eingelesen werden.

- Tragen Sie die Gesamtzahl der Cache-Misses und Cache-Hits je Zeile in die Abbildung ein.

		Miss	Hit
	<code>write i = 0</code>		
	<code>write j = 0</code>		
<code>loop:</code>	<code>read j</code>		
	<code>if (j >= 128) exit</code> <code>else</code>		
	<code>read g[j]</code>		
	<code>read i</code>		
	<code>compute i + g[j]</code>		
	<code>write i</code>		
	<code>read j</code>		
	<code>compute j + 1</code>		
	<code>write j</code>		
	<code>jump to loop</code>		

- Berechnen Sie die Cache-Miss-Quote des gesamten Programms.

TI II

Sommersemester 2010
PD Dr. Katinka Wolter

Problem 3: Virtueller Speicher (2+2+2+3=9)

Ein Computer habe einen virtuellen Adressraum mit 128 Seiten, aber lediglich 4 Seitenrahmen. Anfangs ist der Speicher bereits mit den Seiten 9, 4, 2, 5 (in dieser Reihenfolge) gefüllt worden. Ein Programm referenziere die virtuellen Seiten in folgender Reihenfolge:

7, 1, 2, 4, 2, 4, 3, 6, 3, 4, 1, 9

- Welche Referenzen verursachen einen Seitenfehler bei LRU als Ersetzungsstrategie?
- Welche Referenzen verursachen einen Seitenfehler bei LIFO als Ersetzungsstrategie?
- Welche Referenzen verursachen einen Seitenfehler bei FIFO als Ersetzungsstrategie?
- Nennen/Beschreiben Sie jeweils eine Zugriffsfolge aus acht Seitenanforderungen der Seiten eins bis fünf, die in den obigen Verfahren zu einem Maximum an Seitenfehlern führt.

Problem 4: Tag-Größe (4)

Eine erhöhte Assoziativität benötigt mehr Vergleiche und mehr Tag-Bits pro Cache-Block. Ermitteln Sie für einen Cache mit 4k-Blöcken, einer Blockgröße von 4 Wörtern und einer 32-Bit-Adresse die Gesamtzahl der Sätze sowie die Gesamtzahl der Tag-Bits für direkt abgebildete, zweifach und vierfach satzassoziative und vollassoziative Cache-Organisationen.

Problem 5: Fragmentierung (6)

Weisen Sie den folgenden Verschwendungserscheinungen die Label interne, bzw. externe Fragmentierung zu:

- eine Person, die ein Taxi für sich bestellt
- Zelte auf dem Zeltplatz, die so aufgebaut sind, dass keiner mehr dazwischen paßt, aber auch nicht total dicht nebeneinander stehen
- Werbepausen im Fernsehen, die zu kurz sind, um was anderes zu machen
- Rockstars, die eine ganze Hotelebene mieten, aber nur eine Suite bewohnen
- 100 Fußballfans, die drei Busse a 40 Personen mieten, um zum Spiel zu kommen
- Geschenkpapierreste, die so klein sind, dass man sie nicht zum Umwickeln eines neuen Geschenkes verwenden kann

Problem 6: MMIX – Seitenverdrängungsstrategie LRU (8+2+3=13)

In dieser Aufgabe soll die Seitenverdrängungsstrategie LRU (Least Recently Used) für einen Cache mit n Seiten implementiert werden. Auf der Veranstaltungsseite (bzw. im Blackboard) finden Sie das Rahmenprogramm `rahmen.mms`, das die Eingabe-Routine und die Ausgabe des Caches enthält.

- Implementieren Sie an der markierten Stelle im Rahmenprogramm das Unterprogramm LRU, das als Parameter die zu suchende Zahl im Cache enthält und als Rückgabewert 1 (Cache-Hit) und 0 (Cache-Miss) zurückgibt.

Hinweis: Benutzen Sie das Unterprogramm `cacheLookup`, um eine Zahl im Cache zu suchen.

TI II

Sommersemester 2010
PD Dr. Katinka Wolter

2. Führen Sie das Programm aus und geben Sie die folgenden Cache-Anfragen ein:
7, 1, 2, 4, 2, 4, 3, 6, 3, 4, 1, 9

Bei welchen Cache-Anfragen treten Cache-Misses auf und wie viele sind es insgesamt?

3. Der Cache soll nun auf die Größe 6 erweitert werden. Suchen Sie im Quellcode nach folgenden Abschnitt:

```
CacheSize  IS  4  
Cache      OCTA 5,2,4,9
```

und setzen die Konstante *CacheSize* auf 6 und erweitern die Startsequenz des Caches:

```
CacheSize  IS  6  
Cache      OCTA 5,2,4,9,0,0
```

Führen Sie nun die Cache-Anfragen aus Aufgabenteil 2 erneut aus. Welche Cache-Anfragen sorgen nun für Cache-Misses und wie viele sind es insgesamt?

Hinweis: Wenn Sie es nicht schaffen Aufgabenteil 1 zu implementieren, dann können Sie Aufgabenteil 2 und 3 durch logisches Denken lösen.