Freie Universität Berlin

Übung 11

Rechnerarchitektur
Wintersemester 20/21
Prof. Dr.-Ing. Jochen Schiller

Abgabe 19.02.2021, 10:15

Aufgabe 1: Speicher

Beantworten Sie folgende Fragen, geben Sie ggf. Ihre Quellen an.

1. Warum wird Speicher hierarchisch aufgebaut?
2. Erklären Sie *kurz* jeweils wie S-RAM und D-RAM funktionieren und was die jeweiligen Vor- und Nachteile sind.
3. Erklären Sie Speichervirtualisierung und warum das sinnvoll ist.
4. Würden Sie ein Cache mit physikalischen oder virtuellen Adressen organisieren? Begründen Sie Ihre Antwort.
5. Erklären Sie warum i. d. R. Cache-Koheränz und keine Cache-Konsistens angestrebt wird.
6. Erklären Sie *kurz* die Verdrängungsstrategien Least Frequently Used (LFU), Clock, Second Chance (SC), Random (RAND) und Optimal (OPT)
7. Warum ist LIFO als Verdrängungsstrategie für Caches i. d. R. ungeeignet?
8. Diskutieren Sie *kurz* die Vor- und Nachteile von **write-through** und **write-back**.

Caches Gegeben sei folgender Cache:

Set	Tag	V	D	C	0	1	2	3
3	0xA4E	1	0	0	0x9D	0xA6	0xC4	0x79
	0xD8E	0	1	3	0x83	0x2C	0xCD	0x07
	0xAB6	1	1	1	0x08	0x5B	0xB6	0xF5
	0xDA9	0	0	2	0x3B	0xA1	0x43	0x0C
2	0x396	1	0	3	0xA9	0x49	0xD4	0x9A
	0x008	0	0	1	0xD3	0x11	0xB3	0x5C
	0xC6C	1	0	2	0x12	0xA8	0xA4	0x01
	0x530	0	0	0	0x13	0xE3	0xF0	0x78
1	0xFED	1	1	1	0xC4	0xD2	0x18	0x7D
	0xA0B	1	0	0	0x6D	0x7A	0x75	0xA2
	0x238	1	1	3	0xF0	0xD1	0x92	0x74
	0xB39	0	1	2	0x45	0xCD	0x6E	0x8A
0	0xFBF	1	0	1	0x61	0x99	0x12	0xED
	0x922	1	0	2	0xE4	0x78	0xB2	0x94
	0x396	1	1	3	0x44	0x2E	0x2C	0x95
	0x8BB	1	0	0	0xA1	0x2C	0x03	0x7D

Ein gesetztes Valid Bit zeigt an, dass der Cacheeintrag gültig ist gegenüber allen anderen non-shared Caches. Ein gesetztes Dirty Bit zeigt an, dass der Cacheeintrag verändert wurde seit dem letzten Zurückschreiben in den Hauptspeicher. Der Counter (C) beschreibt die Position der Cacheline in der Queue des Sets bzw. welche Cacheline zuletzt geladen, gelesen oder aktualisiert wurde. Ein höherer Wert bedeutet eine höhere Aktualität.

Die CPU möchte lesend auf die folgenden binären Adressen zugreifen und das jeweilige Byte auslesen:

- 1111 1110 1101 0110
- 1010 1000 0110 1101
- 0000 0000 1000 1000
- 1001 0010 0010 0001
- 1000 0011 1001 0111

Geben Sie jeweils an, ob ein Cache-Hit oder ein Cache-Miss vorliegt. Wenn ein Cache-Hit vorliegt, geben Sie zusätzlich noch die Daten mit an, die ausgelesen werden.

Geben Sie nach jedem Zugriff an, was sich im Cache ändern würde, wenn die Verdrängungsstrategie FIFO bzw. LRU eingesetzt wird.

Aufgabe 2: Matrix Summe

Im KVV ist Ihnen ein C-Framework gegeben, welches zwei Funktionen, die eine Matrix aus M Zeilen und N Spalten aufsummieren, vergleicht. Die Matrix wird dabei durch ein Array dargestellt. Wenn auf das Element an der Position (x, y) zugegriffen werden soll, geschieht dies mit `A[N*y+x]`.

Implementieren Sie in NASM die folgenden Funktionen:

```
int64_t asmRowAdd(int64_t matrix[], uint64_t m, uint64_t n);
int64_t asmColAdd(int64_t matrix[], uint64_t m, uint64_t n);
```

Die Funktion `asmRowAdd` soll die Matrix dabei zeilenweise aufsummieren, während die Funktion `asmColAdd` die Matrix spaltenweise aufsummiert. Versuchen Sie dabei, die Anzahl der Befehle in beiden Funktionen möglichst gleich zu halten, damit der Benchmark möglichst aussagekräftig wird.

Das C-Framework gibt bei Ausführung eine Tabelle aus, in welcher angegeben wird, wie viel Zeit die Funktionen jeweils für ihre Berechnung benötigen. Erklären Sie die Zeitunterschiede. Das C-Framework implementiert zusätzlich die Funktionen in C, damit Sie Ihre Ergebnisse überprüfen können.