Grundlagen der Informatik III

Wintersemester 2010/2011

Wolfgang Heenes, Patrik Schmittat

TECHNISCHE UNIVERSITÄT DARMSTADT

6. Aufgabenblatt mit Lösungsvorschlag

06.12.2010

Hinweis: Der Schnelltest und die Aufgaben sollen in den Übungsgruppen bearbeitet werden. Die Hausaufgaben sind in der Kalenderwoche 50 (13.12. bis 17.12.) 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. *Welche der folgenden Aussagen zu RAM-Speichern sind richtig?*	☐ *DRAMs erfordern keinen Refresh.* ■ *Die Stromaufnahme von SRAMs ist höher als die von DRAMs.* ☐ *Die Speicherung eines Wertes erfolgt bei SRAMs durch Ladung eines Kondensators.* ■ *SRAMs sind i. A. schneller als DRAMs, benötigen aber auch mehr Platz auf dem Chip.*
2. *Wo werden (zeitliche und räumliche) Lokalitäten ausgenutzt?*	■ *Hardware* ■ *Betriebssystem* ■ *Anwendungsebene* ■ *Pipelining*
3. *Bei welchem der Beispiele wird zeitliche (temporale) Lokalität ausgenutzt?*	☐ *Addition zweier Matrizen* ■ *While-Schleife* ☐ *Sprung zu einem Label* ☐ *sequentielle Instruktionsfolge*
4. *Was sind Vorteile von Write-Through gegenüber Write-Back?*	■ *Einfacher zu implementieren.* ■ *Misses sind einfacher zu behandeln.* ☐ *Einzelne Worte werden mit Cache-Geschwindigkeit geschrieben.* ☐ *Beim Rückschreiben kann effizient von großen Speicherbandbreiten Gebrauch gemacht werden.*
5. *Welche Ersetzungsstrategie sind bei einem direkt abbildenen Cache sinnvoll?*	■ *keine* ☐ *LRU* ☐ *FIFO* ☐ *LIFO*

Aufgabe 2: Speichergrößen

Auf Aufbau eines Rechnersystems sind die folgenden Speicherbausteine gegeben:

- *ROM (8 Daten-, 16 Adressleitungen, Steuerleitungen)*
- *RAM (8 Daten-, 19 Adressleitungen, Steuerleitungen)*

a) *Welche Speicherkapazität hat der obige ROM Baustein?*

Lösungsvorschlag:

2^{16} Byte = 64 KByte

b) *Welche Kapazität hat der obige RAM Baustein?*

Lösungsvorschlag:

2^{19} Byte = 512 KByte

c) *Wie viele ROM Bausteine benötigen Sie zur Realisierung von 512 KByte?*

Lösungsvorschlag:

512 KB / 64 KB = 8

d) *Wie viele RAM Bausteine benötigen Sie zur Realisierung von 8 MByte?*

Lösungsvorschlag:

8 MB / 512 KB = 16

Aufgabe 3: Direct-Mapped Cache I

Betrachten Sie einen direkt abbildenden Cache mit 8 Einträgen. Es wird nacheinander auf die folgenden Hauptspeicheradressen (Darstellung Hexadezimal) zugegriffen:

`0x01, 0x02, 0x03, 0x13, 0xF8, 0xF9, 0xAA, 0xCC, 0x57`

a) *Wie viele Adressbits hat der Hauptspeicher?*

Lösungsvorschlag:

Es handelt sich um Byte-Adressen, also werden 8 Adress-Bits benötigt.

b) *Wie viele Index- bzw. Tag-Bits werden für einen Cacheeintrag benötigt?*

Lösungsvorschlag:

$\log_2(8) = 3$, also werden 3 Bits für die Index- und 5 für die Tagbits benötigt.

c) *Geben sie den Inhalt des Caches am Ende der oben angegebenen Zugriffsfolge an. Neben Index- und Tag-Bits sollen auch die Valid-Bits angegeben werden.*

Lösungsvorschlag:

Index	V	Tag	Data
000	Y	1111 1	Speicher(`0xF8`)
001	Y	1111 1	Speicher(`0xF9`)
010	Y	1010 1	Speicher(`0xAA`)
011	Y	0001 0	Speicher(`0x13`)
100	Y	1100 1	Speicher(`0xCC`)
101	N		
110	N		
111	Y	0101 0	Speicher(`0x57`)

Aufgabe 4: Direct-Mapped Cache II

Betrachten Sie ein 32-Bit System mit 1024 MB Hauptspeicher. Es wird ein direkt abbildender Cache mit 2048 Einträgen verwendet. Die Blockgröße beträgt 16 Byte.

a) *Wie viele Bits sind für die Adressierung nötig, um den kompletten Hauptspeicher anzusprechen?*

Lösungsvorschlag:

1024 MByte entspricht $1024 \cdot 2^{20} = 2^{10} \cdot 2^{20} = 2^{30}$ Byte. Für die Hauptspeicheradressierung sind somit 30 Bit nötig.

b) *Wie viele Bits werden für das Index-Feld verwendet?*

Lösungsvorschlag:

$\log_2(2048) = 11$ Bits werden für den Index benötigt.

c) *Wie viele Bits werden für das Tag-Feld verwendet?*

Lösungsvorschlag:

Mit 11 Bits für den Index sowie $2^4 = 16$ Byte Blockgröße bleiben $30 - (11 + 4) = 15$ Bits für das Tag-Feld.

d) *Wie groß ist der Cache ohne Valid- und Tag-Bits insgesamt?*

Lösungsvorschlag:

Der Cache enthält $2048 = 2^{11}$ Einträge, von denen jeder einen Block der Länge 16 Byte enthält. Ohne Valid- und Tag-Bits ergeben sich damit $2048 \cdot 128$ Bit. Das sind 32 KByte.

e) *Wie viele Blöcke können sich gleichzeitig im Cache befinden?*

Lösungsvorschlag:

2048 Blöcke

f) *Wenn das System erweiterbar wäre, wieviel Hauptspeicher könnte es dann adressieren?*

Lösungsvorschlag:

Insgesamt könnten $2^{32} = 4$ GB adressiert werden.

Die Adresse für einen Cachezugriff bei einem 32 Bit System teilt sich wie folgt auf:

T T T T T T T T T T T T T T T T T	I I I I I I I I I I I I I	0 0 0 0 0

Ein `T` *steht für ein Tag-Bit, ein* `I` *für Indexbit und ein* `0` *für ein Offsetbit.*

g) *Wie viele Blöcke werden pro Cacheeintrag gespeichert? Wie viele Bytes enthält jeder Block (durch die Offsetbits kann ein Byte aus dem Block ausgewählt werden)?*

Lösungsvorschlag:

Es wird pro Cacheeintrag ein Block gespeichert, da es sich um einen Direct-Mapped Cache handelt. Da 5 Bits für den Offset vorgesehen sind, ergibt sich daraus $2^5 = 32$ Bytes (8 Worte).

h) *Wie viele KByte an Daten kann der Cache maximal aufnehmen?*

Lösungsvorschlag:

Mit 13 Indexbits und 8 Worten Blockgröße erhalten wir $2^{13} \cdot 8 \cdot 4$ Byte $= 2^3 \cdot 32$ KByte $= 256$ KByte.

i) *Wie groß ist der Cache insgesamt?*

Lösungsvorschlag:

Mit 14 Bit Tag, einem Valid Bit und 8 Worten Blockgröße ($32 \cdot 8 = 2^8$ Bits) erhalten wir $2^{13} \cdot (2^8 + 1 + 14) = 2220032$ Bit bzw. 271 KByte

Hausaufgabe 1: Vergleich zweier Architekturen (5 Punkte)

In dieser Aufgabe sollen zwei Architekturen verglichen werden, wobei die eine Architektur einen Coprozessor für Fließkommaberechnungen besitzt (MFP: Machine with Floating Point), und die zweite nicht (MNFP: Machine with no Floating Point).

Das zu analysierende Programm P besitzt folgende Befehlshäufigkeiten:

Befehlsklasse	*Häufigkeit*	*Befehlsklasse*	*Häufigkeit*
floating-point multiply	*12%*	*floating-point divide*	*7%*
floating-point add	*13%*	*integer instructions*	*68%*

Die Architektur MFP kann die benötigten Fließkommaberechnungen direkt durchführen, wobei Sie die jeweiligen CPI der folgenden Tabelle entnehmen können:

Befehlsklasse	*CPI*	*Befehlsklasse*	*CPI*
floating-point multiply	*6*	*floating-point divide*	*22*
floating-point add	*3*	*integer instructions*	*2*

Die Architektur MNFP muss dagegen die Fließkommaberechnungen mit Integer-Instruktionen simulieren, wobei Sie die jeweils benötigten Operationen der folgenden Tabelle entnehmen können:

Befehlsklasse	*Anzahl Instruktionen*
floating-point multiply	*30*
floating-point add	*20*
floating-point divide	*50*

Sie können davon ausgehen, dass auf der Architektur MNFP jede Integer-Instruktion ebenfalls 2 Zyklen benötigt. Beide Maschinen besitzen eine Taktfrequenz von 1.5 GHz.

a) *Geben Sie die MIPS-Raten beider Architekturen für das Programm P an.*

Lösungsvorschlag:

Für die Maschine MFP erhält man eine CPI von $6 \cdot 0.12 + 3 \cdot 0.13 + 22 \cdot 0.07 + 2 \cdot 0.68 = 4.01$. Pro Sekunde sind das $1.5 \cdot 10^9$ Zyklen. Mit $1.5 \cdot 10^9 Hz/4.01 \approx 374064837/s$ ergibt das ca. 374 MIPS.

Für MNFP erhält man eine CPI von $60 \cdot 0.12 + 40 \cdot 0.13 + 100 \cdot 0.07 + 2 \cdot 0.68 \approx 20.76$. Mit $1.5 \cdot 10^9 Hz/20.76 \approx 72254335/s$ ergeben sich etwa 72 MIPS.

b) *Angenommen, die Maschine MFP benötigt 350 Millionen Instruktionen für P. Wie viele Integer-Operationen benötigt dann MFNP für das Programm?*

Lösungsvorschlag:

Es ergeben sich $350 \cdot 10^6$ Instruktionen für P. Davon fallen auf `mult` $0.12 \cdot 350 \cdot 10^6 = 42 \cdot 10^6$ Instruktionen, auf `add` $0.13 \cdot 350 \cdot 10^6 = 45.5 \cdot 10^6$ Instruktionen, auf `div` $0.07 \cdot 350 \cdot 10^6 = 24.5 \cdot 10^6$ Instruktionen sowie auf `int` $0.68 \cdot 350 \cdot 10^6 = 238 \cdot 10^6$ Instruktionen. Die MFNP-Architektur benötigt somit $(42 \cdot 30 + 45.5 \cdot 20 + 24.5 \cdot 50 + 238) \cdot 10^6 = 3633 \cdot 10^6$ Instruktionen.

c) *Wie lange ist die Ausführungszeit (in Sekunden) für Programm P auf beiden Maschinen?*

Lösungsvorschlag:

$T = B \cdot \text{CPI}/f$ und erhalten $T_{\text{MFP}} = 350 \cdot 10^6 \cdot 4.01/(1.5 \cdot 10^9 Hz) \approx 0.936$s. Für MNFP muss beachten werden, dass jede Integer-Instruktion 2 Zyklen benötigt, d. h. CPI=2: $T_{\text{MFNP}} = 3633 \cdot 10^6 \cdot 2/(1.5 \cdot 10^9 Hz) \approx 4.84$s. Alternativ: 350 Mio. Instruktionen und CPI=20.76 rechnen: $T_{\text{MNFP}} = 350 \cdot 10^6 \cdot 20.76/(1.5 \cdot 10^9 Hz) \approx 4.84$s.

Hausaufgabe 2: Caches (5 Punkte)

Die folgende Tabelle enthält die Parameter (vgl. Vorlesung 14) zweier Caches.

Cache	*m*	*C*	*B*	*E*	*S*	*t*	*s*	*b*
1.	*32*	*2048*	*8*	*1*				
2.	*32*	*2048*	*32*	*1*				

a) *Berechnen Sie die fehlende Einträge S, t, s und b.*

Lösungsvorschlag:

Cache	m	C	B	E	S	t	s	b
1.	32	2048	8	1	256	21	8	3
2.	32	2048	32	1	64	21	6	5

Gegeben seien zwei Arrays a und b mit der Größe $n = 2^k$, sowie ein Programm zur Berechnung des Skalarproduktes

$$s = a_0 \cdot b_0 + a_1 \cdot b_1 + \ldots + a_{n-1} \cdot b_{n-1}.$$

Die Arrays sind im Datensegment wie folgt abgelegt:

```
.data
a: .long ...   # n Words
b: .long ...   # n Words
```

b) *Geben Sie die Speicherzugriffsfolge des Programms für $k = 3$ an, unter der Annahme, dass das Datensegment bei Adresse* `0x1001 0000` *beginnt.*

Lösungsvorschlag:

Zugriffsfolge für $k = 3 (n = 8)$:

```
0x1001 0000, 0x1001 0020, 0x1001 0004, 0x1001 0024,
0x1001 0008, 0x1001 0028, 0x1001 000C, 0x1001 002C,
0x1001 0010, 0x1001 0030, 0x1001 0014, 0x1001 0034,
0x1001 0018, 0x1001 0038, 0x1001 001C, 0x1001 003C
```

c) *Erwarten Sie eine Verbesserung der Performanz des Programms auf einer Architektur, die einen 64 Byte großen Direct-Mapped Cache mit 4 Einträgen besitzt (Wortgröße 32 Bit)? Begründen Sie Ihre Antwort. Überlegen Sie auch, was passiert, wenn k andere Werte als 3 annimmt.*

Lösungsvorschlag:

Das kommt darauf an! Für $k = 3$ verdrängen sich die Zugriffe auf a_i und b_i nicht gegenseitig, für $k = 4$ schon. Für den gegebenen Cache mit 4 Einträgen ergibt sich aus der Gesamtgröße (64 Byte) eine Blockgröße von 16 Byte bzw. 4 Worten. Es ergibt sich somit ein Blockoffset von 4 Bit sowie 2 Bit Index. Die Adresse teilt sich für den Cache daher wie folgt auf:

T T T T T T T T T T T T T T T T T T T T T T T T T T T T	I I	0 0 0 0

Bei jedem Cache-Miss wird der gesamte Block in den Cache geladen. Beispielsweise werden beim Zugriff auf Adresse `0x1001 0000` die Worte bis einschließlich Adresse `0x1001 000C` in den Cache geladen. Um zu sehen, ob der Cache eine Effizienzsteigerung bringt, muss überprüft werden, ob aufeinanderfolgende Speicherzugriffe auf denselben Cache-Block abgebildet werden. Für $k = 3$ sieht man, dass der Block beginnend ab Adresse `0x1001 0000` in den Cache-Block mit Index 0 geladen wird und der Block beginnend ab Adresse `0x1001 0020` in Cache-Block 1.

Für $k = 4$ dagegen beginnt `b` an Adresse `0x1001 0040` und verdrängt sofort den zuvor geladenen Cache-Block aus Adresse `0x1001 0000`. Allgemein verdrängen die aufeinanderfolgenden Zugriffe auf `a,b` sich für $k \geq 4$ gegenseitig und der Cache bringt keine Effizienzsteigerung.