Aufgabenblatt 8: Cache 1 - Musterlösung

1 Cache-Berechnungen

Bezeichner	**Formel**	**Eingesetze Werte (ggf. als Bruch)**	**Wert (gekürzt, ggf. als $y * 2^x$)**
Assoziativität	-	4	2^2
Hauptspeicher (in Byte)	-	$512 * 2^{20} Byte$	$2^{29} Byte$
Cachegröße (in Byte) Nutzdaten	-	$64 * 2^{10} Byte$	$2^{16} Byte$
# Adressbits	-	$32 Bit$	$2^5 Bit$
Cachelinegröße	-	$32 Byte$	$2^5 Byte$
# Offsetbits	-	$5 Bit$	$5 Bit$
# Cachelines im Hauptspeicher	$\frac{Hauptspeicher}{Cachelinegroesse}$	$\frac{2^{29}}{2^5}$	$2^{24} = 16M$
# Cachelines im Cache	$\frac{Cachegroesse}{Cachelinegroesse}$	$\frac{2^{16}}{2^5}$	$2^{11} = 16M$
# Sets	$Cachegroesse = Assoziativitaet * \#Sets * Cachelinegroesse$	$2^{16} = 2^2 * x * 2^5$	2^9
# Setbits	-	9 Bit	9 Bit
# Tagbits	$\#Adressbits = \#Tagbits + \#Setbits + \#Offsetbits$	$32 - 9 - 5 = 18$	18 Bit
$\frac{\#CachelinesimCache}{Set}$	$= Assoziativitaet = \#Blockrahmen$	4	4

2 Direct mapped Cache

In dieser Aufgabe wird ein sogenannter "Direct mapped Cache" betrachtet in dem jedes Set genau eine Cacheline hat. Da jedes Set hier nur eine Cacheline enthält wird der Set-Teil der Adresse hier manchmal auch einfach als Line (statt als Set) bezeichnet. Wir wollen hier aber aus Konsistenzgründen weiter den Begriff Set verwenden.
Der Hauptspeicher soll 256 Byte betragen und der Cache soll 16 Byte davon zwischenspeichern können. Die Cachlinegröße beträgt 1 Byte.

a) Da unsere Cachelinegröße 1 Byte beträgt benötigen wir für die Adressierung innerhalb der Cacheline (Offset l) kein Bit. Dadurch müssen wir die 8 Bit Adresse (um $256 = 2^8$ Speicherzellen zu adressieren) nur noch in Tag- und Set-Teil aufteilen. Da wir einen direct mapped Cache betrachten haben wir eine Assoziativität von 1 und können daher 16 Sets in unseren 16 Bytes Cache unterbringen. Daher benötigen wir $log_2 16 = 4$ Bits für den Set-Teil der Adresse. An dieser Stelle kann man mit Hilfe der Gesamtlänge der Adresse (8 Bit) und der Länge der Offset (0 Bit) und Set-Teile (4 Bit) bereits die Länge des Tag-Teils bestimmen (8-0-4=4 Bit). Man kann jedoch auch direkt überlegen wie lang der Tag-Teil sein muß. Der Tag-Teil wird mit der Cacheline im Cache gespeichert um zu erkennen welche der Speicherstellen für

die eine gegebene Cacheline zuständig ist gerade in dieser gecached ist. Daher müssen wir nur überlegen wie viele Cacheline-lange Speicherstücke jedem Set hier zugeordnet sind. Da wir 16 Sets haben und 256 Cacheline-lange Speicherstücke kommen wir auf einen Wert von $\frac{256}{16} = 16$ und damit auf eine benötigte Bit-Zahl von $log_2 16 = 4$ Bits für den Tag-Teil.

b) Wie eben erläutert wird mit jeder Cacheline der Tagteil der entsprechenden Adresse gespeichert um festzustellen zu können welche der Adressen die zu diesem Set gehören gerade gecached wurde. Ausserdem benötigen wir das sog. Valid-Bit das kennzeichnet ob überhaupt ein gültiger Wert in dieser Cacheline gespeichert ist. Insgesamt werden also mit jeder Cacheline 5 Bits an Verwaltungsdaten gespeichert. Der Cache soll 16 Byte in 1 Byte (8 Bit) Cachelines speichern können womit wir auf 16 Cachelines kommen. Jede dieser Cachelines hat $8+5 = 13$ Bits zu speichern. Damit kommen wir auf eine Gesamtgröße von $208Bit = 26Byte$.

c) Der Aufwand des Cache ist durch das Verhältnis

$$\frac{\#\text{ Verwaltungsbits}}{\#\text{ Nutzungbits}}$$

gegeben. Da wir hier 5 Verwaltungs und 8 Nutzbits haben erhalten wir also

$$\frac{5}{8} = 0.625$$

als Aufwand.

d) Eindeutiger Vorteil dieses Cache ist daß hier keine Strategie (und keine Hardware um diese zu implementieren) benötigt wird die festlegt welche Cacheline eines Sets bei einem Cache-Miss ausgetauscht wird. Dies ist aber gleichzeitig auch der größte Nachteil denn jede Cacheline wird sofort ausgetauscht wenn ein Cache-Miss auftritt der eine Speicherzelle im Speicherbereich betrifft der dieser Cacheline zugeordnet ist. Dadurch wird u.U. die Miss-Rate stark erhöht wenn abwechselnd mehrere dieser Speicherzellen verwendet werden.

e) Spielen Sie einen Zugriff auf die Speicherzellen 0, 8, 16, 24, 8, 16 durch!

Da jedem Set 16 1 Byte große Speicherbereiche zugeordnet sind ist jeder 16. Speicherzelle die selbe Cacheline zugeordnet. In der Tabelle sind alle Zeilen ausgelassen die nicht verwendet werden. Zellen die nicht verändert werden sind mit einem "-" gekennzeichnet. Diese Zellen behalten ihren alten Wert. Bei "Cache-Hit" behalten die Cache-Zellen ebenfalls den alten Wert und dieser wird statt eines Hauptspeicher-Zugriffs verwendet. Mit Cacheline ist die Cacheline innerhalb des jeweiligen Sets gemeint. Set- und Cacheline-Nummerierung beginnen jeweils bei 0.

	Set	0	8
Zugriff auf	Cacheline	0	0
0		MM[0],Tag=0	-
8		-	MM[8],Tag=8
16		MM[16],Tag=1	-
24		-	MM[24],Tag=1
8		-	MM[8],Tag=0
16		Cache-Hit	-

3 N-assoziativer Cache

Im sogenannten "n-assoziativen Cache" stehen für jede gegebene Speicherstelle n Cachelines zur Verfügung in denen diese Speicherstelle gecached werden kann. Alle diese Cachelines zusammen werden als Set bezeichnet. Alle Cachelines eines Set können die gleichen Speicherstellen cachen. Die Anzahl der Cachelines eines Sets wird als Assoziativität bezeichnet. Dieser Wert soll hier 2 betragen. Die restlichen Werte entsprechen denen von Aufgabe 2.

a) Bei einem n-assozitiven Cache wird die Speicheradresse in ein r-bit großen Tag- einen k-Bit großen Set- und einen l-Bit großen Offsetabschnitt eingeteilt. Wie groß sind r, k und l in unserem Beispiel wenn wir einen n-assozitiven Cache verwenden?

   Wie auch in der vorherigen Aufgabe beträgt der Offsetanteil natürlich 0 Bit da eine Cacheline nur 1 Byte lang ist. Da wir nur die 16 Cachelines des Cache in Sets zu je 2 aufteilen müssen haben wir nur noch 8 Sets und damit auch nur noch $log_2 8 = 3$ Bit Set-Teil der Adresse. Jedem der 8 Sets sind jetzt $\frac{256}{8} = 32$ Cacheline-große Speicherstücke zugeteilt wodurch sich die Tag-Größe auf $log_2 32 = 5$ Bit erhöht. Die Gesamtadresslänge beträgt natürlich weiterhin 8 Bit da sich ja an der Größe des Speichers durch eine Änderung des Caching-Verfahrens nichts ändert.

b) Der Cache benötigt nun für jede seiner 16 Cachelines zusätzlich zu den 8 Bit Nutzdaten noch 1 Valid-Bit und 5 Bit für das Tag. Daher werden insgesamt $16 * (8 + 1 + 5) = 16 * 14 = 224Bit = 28Byte$ benötigt.

c) Der Aufwand des Cache ist durch das Verhältnis

$$\frac{\#\text{ Verwaltungsbits}}{\#\text{ Nutzungbits}}$$

   gegeben. Da wir hier 6 Verwaltungs und 8 Nutzbits haben erhalten wir also

$$\frac{6}{8} = \frac{3}{4} = 0.75$$

   als Aufwand.

d) Der Vorteil des n-assoziativen Cache besteht darin daß hier Hardware-Aufwand gegen Cache-Hit-Ratio abgewogen werden kann indem man die Assoziativität variiert (beim Entwurf, nicht während des Einsatzes). Im Grunde sind die beiden anderen betrachteten Cache-Arten nur Sonderfälle dieses Cache.

4 Vollassoziativer Cache

Jetzt wird ein vollassoziativer Cache verwendet. Dieser erlaubt das beliebige abspeichern einer zu cachenden Speicherstelle. Eine andere Betrachtungsweise sieht diese Art des Cache als Sonderform des n-assoziativen Cache mit nur einem Set. Die Größe des Hauptspeichers soll wieder 256 Byte betragen wovon der Cache wieder 16 Byte cachen soll.

a) Wie in den vorherigen Aufgaben ist der Offset-Teil 0 Bit lang. Da der Set-Teil hier ebenfalls nicht vorhanden ist muß der Tag-Teil also die gesamte Adresse umfassen. Dies macht auch Sinn, da jede Adresse in jeder Cacheline abgelegt werden kann muß der Tag auch die gesamte Adresse umfassen um eindeutig feststellen zu können welche Adresse in einer gegebenen Cacheline aktuell gespeichert ist.

b) Wie bereits in der Aufgabenstellung erwähnt handelt es sich beim vollassoziativen Cache um eine Sonderform des n-assoziativen Cache mit 1 Set. Daher muß man kein Set auswählen und benötigt also auch keine Bits für die Adressierung eines Set.

c) Da jede Cacheline die volle Adresse ihrer aktuellen Speicherstelle im Tag speichert benötigen wir 8 Bit Tag und 1 Valid-Bit an Verwaltungsdaten. Daher haben wir $8+1+8=17$ Bit pro Cacheline und damit $16*17=272Bit=34Byte$ Gesamtgröße.

d) Der Aufwand des Cache ist durch das Verhältnis

$$\frac{\#\text{ Verwaltungsbits}}{\#\text{ Nutzungbits}}$$

gegeben. Da wir hier 9 Verwaltungs und 8 Nutzbits haben erhalten wir also

$$\frac{9}{8}=1.125$$

als Aufwand.

e) Spielen Sie einen Zugriff auf die Speicherzellen 0, 8, 16, 24, 8, 16 durch!

In der Tabelle sind alle Zeilen ausgelassen die nicht verwendet werden. Zellen die nicht verändert werden sind mit einem "-" gekennzeichnet. Diese Zellen behalten ihren alten Wert. Bei "Cache-Hit" behalten die Cache-Zellen ebenfalls den alten Wert und dieser wird statt eines Hauptspeicher-Zugriffs verwendet. Mit Cacheline ist die Cacheline innerhalb des jeweiligen Sets gemeint. Set- und Cacheline-Nummerierung beginnen jeweils bei 0.

	Set	0	0	0	0
Zugriff auf	Cacheline	0	1	2	3
0		MM[0],Tag=0	-	-	-
8		-	MM[8],Tag=8	-	-
16		-	-	MM[16],Tag=16	-
24		-	-	-	MM[24],Tag=24
8		-	Cache-Hit	-	-
16		-	-	Cache-Hit	-