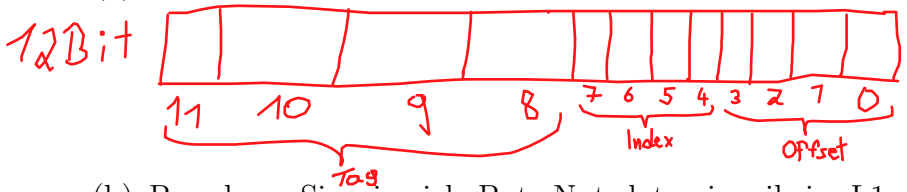
9. *(_____ / 8 Punkte)* Caching

Ein Prozessor besitzt eine Hierarchie aus zwei Caches. L1 ist als *Direct Mapped Cache* organisiert, L2 als *8-Way Set Associative Cache*. Die Adresslänge beträgt 12 Bit, die kleinste adressierbare Einheit ist 1 Byte. Beide Caches verwenden 4 Bit Offset und 4 Bit Index.

Die CPU führt 100000 Adressanfragen aus. Dabei gibt es 90000 Hits beim L1-Cache und 8000 Hits beim L2-Cache.

(a) Wieviele Bits werden für den Tag beim L2-Cache benötigt?



4 Bits für den Tag

(b) Berechnen Sie wie viele Byte Nutzdaten jeweils im L1- und im L2-Cache Platz haben.

L1: Blockgr. 2^4 Bytes = 16 Bytes
Blockgr. 2^4 Bytes = 16 Bytes
Nutzdaten = 16 · 16 = 256 Bytes

L2: # Nutzdaten = 8 · 256 Bytes = 2048 Bytes
$2 \cdot 2^{10}$ Bytes = 2 KiB

(c) Geben Sie die Hit-Rate des L2-Cache in Prozent an.

$h_{L_2} = \frac{8000}{100000 - 90000} = \frac{8000}{10000} = \frac{8}{10} \Rightarrow 80\%$

Um die Hitrate zu berechnen, muss man die Hits durch die Misses dividieren. Da insgesamt 100000 Adressanfragen ausgeführt werden und der L1 Cache davon 90000 davon beantwortet, bleiben für den L2 Cache 10000 Misses.

(d) Es gelten folgende reine Zugriffszeiten (vorangehende Misses *nicht* inkludiert):

- L1-Cache: $t_{L1} = 10$ ns
- L2-Cache: $t_{L2} = 90$ ns
- Hauptspeicher: $t_{main} = 900$ ns

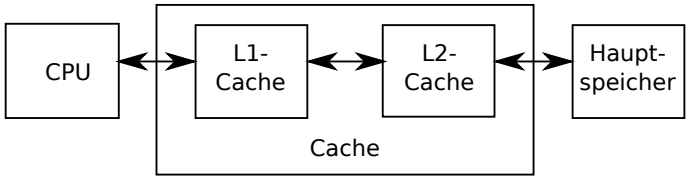
Berechnen Sie die durchschnittliche Speicherzugriffszeit t_{mem} mithilfe der sich aus der Angabe ergebenden Hit-Raten und Miss-Raten. Bedenken Sie, dass sich die Einzelzugriffszeiten bei einem Miss entsprechend summieren, da **alle** Anfragen zunächst an den L1-Cache gestellt werden. Notieren Sie auch den **Rechenweg.**

$t_{mem} = 0.9 \cdot 10\text{ns} + \quad$ ((1)
$0.1 \cdot [0.8 \cdot (10\text{ns} + 90\text{ns}) + \quad$ ((2)
$0.2\,(10\text{ns} + 90\text{ns} + 900\text{ns})] = \quad$ (mem)

$9\text{ns} \quad$ ((1)
$0.1\,[80\text{ns} \quad + \quad$ (2)
$200\text{ns}] \quad$ (mem)

$9\text{ns} + 0{,}1 \cdot 280\text{ns} = 37\text{ns}$

6. *(____ / 9 Punkte)* Ein Prozessor besitzt eine Speicherhierarchie bestehend aus L1-Cache, L2-Cache und Hauptspeicher. Der Prozessor wird mit 5 GHz getaktet. Die Zugriffszeit t_{L1} auf den L1-Cache beträgt 2 Taktzyklen. Die Zugriffszeit t_{L2} auf den L2-Cache beträgt 40 Taktzyklen, wobei der vorhergehende Miss beim L1-Cache bereits inkludiert ist. Die Zugriffszeit t_{main} auf den Hauptspeicher beträgt 200 Taktzyklen, wobei ebenfalls die beiden vorhergehenden Misses bereits inkludiert sind.



Während der Ausführung eines Prozesses wurden die folgenden Werte beobachtet:

- 10000 Adressanfragen an den L1-Cache.
- 1000 Adressanfragen an den L2-Cache.
- 80 Adressanfragen an den Hauptspeicher.

(a) Geben Sie die Hit-Rate des L1-Caches in Prozent an.

$\frac{9000}{10000} = 90\%$

(b) Geben Sie die Miss-Rate des L2-Caches in Prozent an.

$\frac{80}{1000} = 8\%$

(c) Geben Sie folgende Zugriffszeiten in ns an.

$t_{L1} = \frac{2}{5}\,ns$

$t_{L2} = \frac{40}{5}\,ns = 8\,ns$

$t_{main} = 40\,ns$

5 GHz = $5 \cdot 10^9$ / s

1 ns ist 1 Milliardstel

$: 10^9$

= 5 / ns

Zyklen

(d) Sei H_X die absolute Anzahl der Hits auf den jeweiligen Speicher ($X \in \{L1, L2, main\}$). Berechnen Sie die akkumulierte Zugriffszeit t_a nach folgender Formel, und geben Sie das Ergebnis in μs an.

1 Taktzyklus passiert in $\frac{1}{5}$ ns

$10^9 = 1\,s$

$10^6 = 1\,s$

$t_a = H_{L1} \cdot t_{L1} + H_{L2} \cdot t_{L2} + H_{main} \cdot t_{main}$

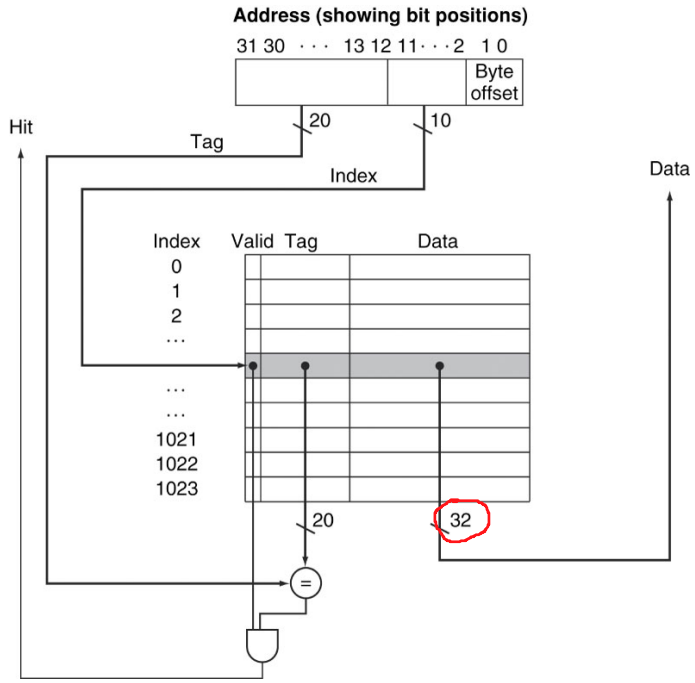
$9000 \cdot \frac{2}{5} + 920 \cdot 8 + 80 \cdot 40 = 14.160\,ns$

$: 10^3$ ↓ µs

14,16 µs

5. *(_____ / 6 Punkte)* Caching

(a) Um welche 32-Bit Cache-Architektur handelt es sich in der folgenden Abbildung? Wie groß ist ein adressierbares Datenwort in Byte?



Direct-Mapped-Cache

Offset Größe

$32Bit/2^2$

$\frac{2^5}{2^2} = 2^3 = 8Bit$

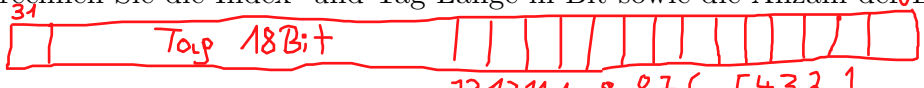
$8Bit/8 = 1Byte$

ist ein addressierbares Datenwort

(b) Der ARM Cortex A8 (Raspberry v1) verwendet für den L1-Cache eine Blockgröße von 64 Byte und kann Datenwörter der Länge 1 Byte adressieren. Dabei verwendet er für Instruktionen und Daten zwei getrennte L1-Caches mit 2 Ways und je 32 KiB. Zur Adressierung werden 32 Bit verwendet.

Berechnen Sie die Index- und Tag-Länge in Bit sowie die Anzahl der Bits für den Offset.

32Bit



Offset:

#Länge Offset = 6Bit

Index-Länge:

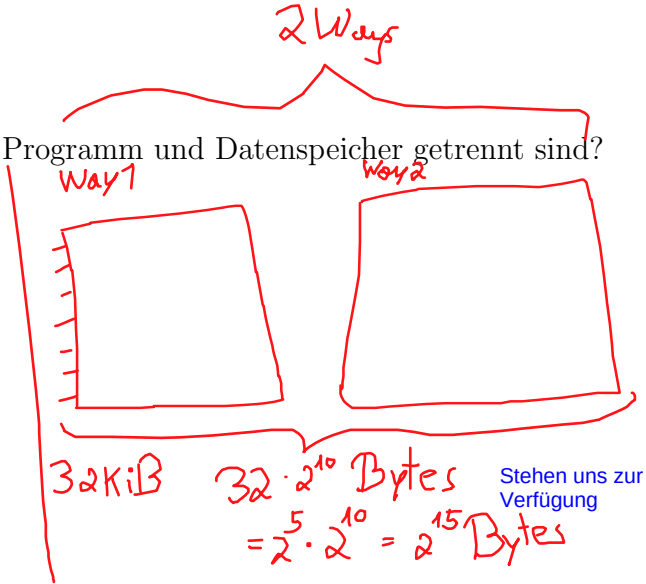
$\frac{2^{15}}{2 \text{ Ways}} = 2^{14} \Rightarrow \frac{2^{14}}{2^6} = 2^8$ unterschiedliche Blöcke $\Rightarrow$ 8Bit Index Länge

Blockgrösse (64Bytes = 2^6)

Tag-Länge:

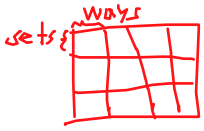
Adresslänge = Tag + Index + Offset
Tag = Adresslänge - Index - Offset
Tag = 32 - 8 - 6 = **18 Bit**

(c) Um welche Architektur handelt es sich, wenn Programm und Datenspeicher getrennt sind?



Havard-Architektur

2018S



Cache-Grösse =
Ways * Sets * Blockgrösse

Nur die Nutzdaten gemeint.

9. *(____ / 10 Punkte)* Ein Prozessor besitzt eine Hierarchie aus zwei Caches. L1 ist 32KiB groß und als *8-Way Set Associative Cache* organisiert, L2 als *16-Way Set Associative Cache*. Die Adresslänge beträgt 64 Bit, die kleinste adressierbare Einheit ist 1 Byte. Beide Caches verwenden eine Blockgröße von 64 Byte. L1 hat eine Hit-Rate von 80 Prozent und L2 von 90 Prozent.

Offset 6bit $2^6 = 64$

$32\,KiB = 2^{15}$, $8\,ways = 2^3\,ways$, $64\,Byte = 2^6\,Byte$

$\frac{2^{15}}{2^3} = \frac{2^{15}}{2^{12}} = 2^3$

(a) Berechnen Sie für den L1-Cache die Länge des Tags in Bits.

#Blöcke => Cache-Grösse/ Blockgrösse

AL: Adresslänge

Wenn man von der Grösse des Caches spricht, dann meint man nur die Nutzdaten.

AL = TAG + Index + Offset =
64bit = 52bit + 6bit + 6bit

$\#Blöcke = \frac{32 \cdot 2^{10}\,Byte}{2^6} = 2^9$

$\#Sets = \frac{2^9}{2^3} = 2^6\,Bit\,Index$

$TAG = 64 - 6 - 6 = 52\,bit$

(b) Der Tag des L2 Cache ist um 2 Bit kürzer als der des L1 Cache. Wie groß ist der L2 Cache?

Der Tag ist 2Bit kürzer, daher müssen wo anders addiert werden. Blockgrösse ist die gleiche, daher muss der Index grösser sein. Laut Angabe is die Adresslänge gleich

Anzahl Sets · Ways des L2 · Blockgröße

$2^8\,Sets \cdot 16\,Ways \cdot 2^6 = 2^{18} = 256\,KiB$

50	8	6
52-2 Tag	Index	Offset

(c) Betrachten Sie L1 und L2 als ein System und geben Sie die Hit-Rate des gesamten Cache in Prozent an.

$0,2 \cdot 0,1 = 0,02$ Miss-Rate des L1 * Miss-Rate des L2 = Miss-Rate des gesamten Caches

$1 - 0,02 = 0,98 = 98\%$

1 - Miss-Rate des Caches = Hit Rate des Caches

Hit-Rate: $0.8 + 0.2 \cdot 0,9 = 98\%$

Wsl-Baum

Hit-Rate
Miss-Rate

(0.8 Hit-Rate des L1) + (Miss-Rate des L1 * Hit-Rate des L2) -> siehe Baum

(d) Der Prozessor arbeitet mit einer Taktrate von 2 GHz und der Cache hat die folgenden Zugriffszeiten (vorangehende Misses inkludiert):

- L1-Cache: 2 Takte
- L2-Cache: 4 Takte
- Hauptspeicher: 20 Takte

Es wird ein Programm mit 100.000 Adressanfragen ausgeführt. Wie lange dauern diese 100.000 Adressanfragen durchschnittlich. Geben Sie das Ergebnis in Mikrosekunden (μs) an.

$\frac{1}{2\,GHz} = \frac{1}{2 \cdot 10^9\,Hz} = \frac{1}{2} \cdot 10^{-9} = 0,5\,ns$ $(10^{-9} = nano)$

100K·0.8 (Hit-Rate L1) · 2 Takte + (100K·0.8·0.9 = Hit L2) · 4 Takte + (100K·0.8·0.1 = Hit RAM) · 20 Takte

$80000 \cdot 2 + 18000 \cdot 4 + 2000 \cdot 20 = 272\,000\,Takte$

L1 L2 HS

$272000 \cdot 0,5$
$= 136000\,ns$
$= 136\,\mu s$

$(10^{-6} = mikro)$

Der L1-Cache hat eine Hit-Rate von 0.8, daher bearbeitet er 80000 Anfragen von den 100000. Der L2-Cache hat eine Hit-Rate von 0.9, daher bearbeitet er von den übrigen 20000 18000. Die restlichen 2000 werden vom Hauptspeicher beantwortet. Zusätzlich werden die Takte multipliziert. Darauf ergibt sich die Gesamtanzahl der Takte. Dies wird mit der Taktzeit multipliziert und man bekommt heraus, wie lange es dauert bis alle Takte abgearbeitet sind.

2017W

Cache = Blockgröße • Sets • Ways

2^{10} Index

8. *(____ / 15 Punkte)* Ein 1 MiB großer n-Way Set-Associative L3-Cache besitzt 1024 Cache-Sets. Die Blockgröße beträgt 512 Bit und beinhaltet 16 adressierbare Datenwörter. In diesem System können Datenwörter im vollen Umfang zur Adressierung verwendet werden.

Off: $2^4 = 4$ Bit

(a) Berechnen Sie die Adresslänge und Datenwortlänge in Bit sowie die Anzahl der Ways des Caches.

Adresslänge: 32 Bit

Datenwortlänge: 512 / 16 = 32 bit

Ways: $2^{20} \cdot 2^3 / 2^{10} / 2^9 = 2^1 = 2^4$ Ways = 16 Ways

(b) Berechnen Sie für den gegebenen Cache die Längen von Tag, Offset und Index in Bit.

Tag-Länge: 18

Index-Länge: 10

Offset-Länge: 4

} Bit

(c) Es wird die zufällige Erstetzungsstrategie (Random) verwendet, wobei nie der zuletzt verwendete Block ersetzt wird. Wie viele Bits sind für die Verwaltung der Ersetzungsstrategie pro Cache-Set minimal notwendig?

Verw: Tag, Dirty, Valid, Ersetzungsstrategie

16 Ways = 2^4 = 4 Bit um genau darzustellen

(d) Der L1-Cache hat eine *Hit-Rate* von 80% und der L2-Cache eine Hit-Rate von 95%. Wie viele Anfragen in Prozent werden durchschnittlich an den L3-Cache gestellt?

$0{,}2 \cdot 0{,}05 = 0{,}01$ (1%)

oder Gegenwschl.:

$0{,}8 + 0{,}2 \cdot 0{,}95 = 0{,}99$ und dann $(1 - 0{,}99) = 0{,}01$ (1%)

Hit-Rate von beiden zusammen

(e) Sie bauen den L3-Cache zu einem Fully Associative Cache um. Steigt oder sinkt die Hit-Rate statistisch? Welche technischen Gründe sprechen gegen einen solchen Umbau?

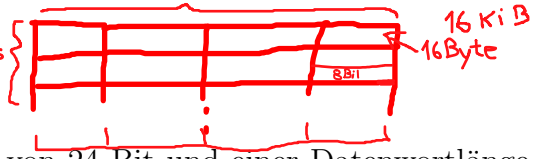
Steigt

Suche dauert länger (Zugriffszeiten brauchen länger)

Ways

2017S

Es macht Sinn den Cache zu visualisieren. Einen Cache kann man sich als eine Tabelle vorstellen.



5. *(15 Punkte)* Ein Prozessor mit einer Adresslänge von 24 Bit und einer Datenwortlänge von 8 Bit hat einen 16 KiB großen *4-way Set-Associative Cache* mit einer Blockgröße von 16 Byte.

Die Ways sind die Spalten und die Sets sind die

(a) Aus wievielen Cache-Sets besteht dieser Cache?

Der gesamte Cache ist 16 KiB gross. Ein Block besteht aus Datenwörtern. Wieviele Datenwörter passen in einen Block.

(b) Berechnen Sie für den gegebenen Cache die Längen von Tag, Offset und Index in Bit.

Tag-Länge: 12 Bit

Index-Länge: 8 Bit

Offset-Länge: 4 Bit

(c) Angenommen, das Cache-Set 18 ist vor jedem der unten angegebenen Lesezugriffe folgendermaßen belegt:

Tag	Valid	Dirty
FF0	1	0
003	1	1
7E8	0	0
3FF	1	0

Welche Zugriffe führen zu einem *miss*, welche zu einem *hit*? Kreuzen Sie entsprechend an.
(richtig: +1 Punkte, falsch: -1 Punkte, keine Antwort: 0 Punkte)

Die Adresszugriffe sind hexadezimal angegeben, daher ist eine Ziffer in Hexadezimal 4 Bit in Binär.
Die letzten 4 Bit ist der Offset und dieser dient zur Navigation im Block.

FF0181	☒ hit	○ miss
3EE18C	○ hit	☒ miss
003180	☒ hit	○ miss
7E8183	○ hit	☒ miss

Da sich die Adresse nicht im Cache befindet

Da das Valid-Bit 0 ist

Wichtig ist nur der Tag beim Zugriff. 18 ist das Cache-Set, das oben angegeben wurde.

(d) Auf welche Strategie bei Schreiboperationen kann man aufgrund der im vorigen Teilbeispiel angegebenen Verwaltungsinformationen schließen?
(richtig: +2 Punkte, falsch: -2 Punkte, keine Antwort: 0 Punkte)

○ Write Through ☒ Copy Back ○ Random Write

Wegen dem Dirty-Bit

(e) Wieviele Adressen des Hauptspeichers sind dem selben Cache-Set zugeordnet?

Adressen sind einem Cache-Set zugeordnet.

4. *(12 Punkte)* Ein Prozessor mit Adresslänge 20 Bit besitzt einen *Fully Associative Cache* mit 4 Blöcken, ein Block enthält 16 Datenwörter. Die Adressierung erfolgt auf Datenwort-Ebene, die Datenwortlänge beträgt 32 Bit.
Die verwendete Ersetzungsstrategie ist *Least Recently Used (LRU)*. Initial ist der Cache leer, die Blöcke werden in aufsteigender Reihenfolge beschrieben. Für Schreibzugriffe gilt: Bei einem *Hit* wird *Copy-Back* verwendet, bei einem *Miss* wird *Fetch-On-Write* verwendet.

   (a) Geben Sie die Längen von Tag, Index und Offset in Bit an!

   - Tag: 16 (20-4)
   - Index: 0 wegen Fully Associative Cache
   - Offset: 4 (16DW = 2^4)

   (b) Vervollständigen Sie die nachfolgende Tabelle! Es müssen nur Änderungen eintragen werden. *Hinweis:* Zahlen sind hexadezimal gegeben, wobei führende Nullen nicht dargestellt werden.

					Hit/	Block 0			Block 1			Block 2			Block 3		
	Adr.	r/w	Tag	Offset	Miss	Tag	V	D	Tag	V	D	Tag	V	D	Tag	V	D
0	–	–	–	–	–	0	0	0	0	0	0	0	0	0	0	0	0
1	A48	r	A4	8	Miss	A4	1	0	0	0	0	0	0	0	0	0	0
2	A49	w	A4	9	Hit	A4	1	1	0	0	0	0	0	0	0	0	0
3	0	r	0	0	Miss	A4	1	1	0	1	0	0	0	0	0	0	0
4	B58	w	B5	8	Miss							B5	1	1			
5	B50	r	B5	0	Hit												
6	C00	r	C0	0	Miss										C0	1	0

Nur änderungen in die Tabelle

   (c) Wie viele Bytes Nutzdaten können in diesem Cache gespeichert werden?

   4 Ways · 16 DW · 4 Byte (DWL, 32 Bit) = 256 Byte

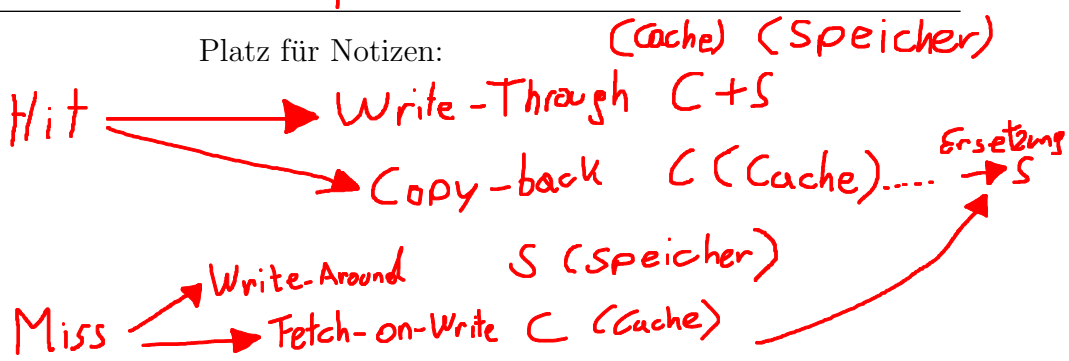
   (d) Wie viele Bytes Verwaltungsdaten benötigt dieser Cache, wenn für die Realisierung der Ersetzungsstrategie zusätzliche 6 Bit pro Block (in obiger Tabelle nicht dargestellt) benötigt werden?

   16 (Tag) + 1 (Valid-Bit) + 1 (Dirty-Bit) + 6 (Ersetzung) = 24 Bit

   Das gilt für einen Way. Danach muss man noch mal den Ways rechnen.

   24 · 4 (Ways) = 96 Bit → 12 Byte

Platz für Notizen:



9. *(12 Punkte)* Schreiben Sie ein Micro16-Programm, das prüft, ob zwei Bitfolgen, die in den Registern `R1` und `R2` abgelegt sind, ident sind. Sind sie ident, soll am Ende im Register `R0` die Zahl 0 hinterlegt sein, anderenfalls die Zahl 1.

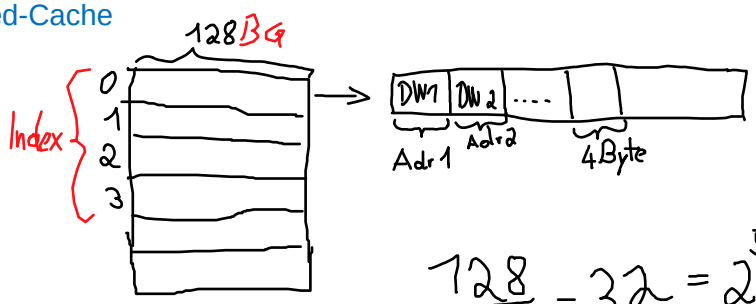
```
R2 <- ~R2
R2 <- R2 +1
R0 <- R1 + R2; if Z goto .end
R0 <- 1
:end
```

Idee:
Um zu ueberpruefen, ob zwei Zahlen ident sind, muss man die beiden Zahlen subtrahieren. Da der Micro16 nicht subtrahieren kann, muss man das Zweierkomplement einer der beiden Zahlen bilden. Dadurch ist die Zahl negativ und kann addiert werden. Wenn das Ergebnis der Subtraktion 0 ist, dann sind die Zahlen gleich und das Programm kann beendet werden.

Platz für Notizen:

Cache

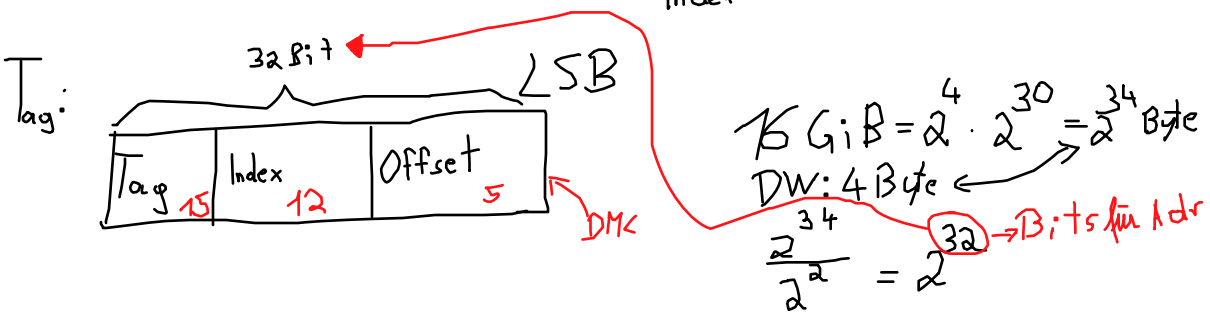
Direct-Mapped-Cache



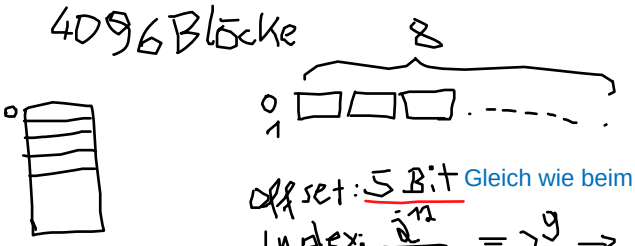
16 GiB Adressierung
Datenwortlänge: 4 Byte
Cache: 512 KiB
Blockgrösse: 128 Byte

$\frac{128}{4} = 32 = 2^5 \rightarrow$ Offset

Index: $\frac{512\,KiB}{128\,Byte} = \frac{2^9 \cdot 2^{10}}{2^7} = 2^{12} = 4096$ Blöcke



8-Way Set Cache



Offset: 5 Bit — Gleich wie beim direct-mapped

Index: $\frac{2^{12}}{2^3} = 2^9 \rightarrow$ 9 Bit — Index von oben durch die Ways (8) dividieren.

Tag: 18 Bit — 32 - 9 - 5 = 18

Fully-Associative Cache

Offset: 5
Index: 0, weil wir nur 1 Set haben
Tag: 27 (32-5)

2014 S

6. *(12 Punkte)* Gegeben ist der aus der Übung bekannte Prozessor mit einer Adresslänge von 12 Bit (7 Bit Tag, 2 Bit Index, 3 Bit Offset) sowie einer Datenwortlänge von 1 Byte. Der integrierte *Direct Mapped Cache* besitzt 4 Blöcke und speichert 8 Byte pro Block. Für Schreibzugriffe gilt: Bei einem *Hit* wird *Copy-Back* verwendet, bei einem *Miss* wird *Fetch-on-Write* verwendet.

   Ausgehend von einem anfangs leeren Cache haben sich die Verwaltungsdaten für die gegebenen Speicherzugriffe wie folgt verändert:

   rd(0x2EA) – rd(0x2E6) – wr(0x2EF) – rd(0x7A2) – rd(0x1D) – wr(0x7A3) – rd(0xA82) – wr(0x7A3)

Speicherzugriff				Block $(00)_2$			Block $(01)_2$			Block $(10)_2$			Block $(11)_2$		
Adr.	r/w	H/M	Adr. binär	Tag	V	D	Tag	V	D	Tag	V	D	Tag	V	D
				0x00	0	0	0x00	0	0	0x00	0	0	0x00	0	0
0x2EA	r	Miss	0010111 01 010	0x00	0	0	0x17	1	0	0x00	0	0	0x00	0	0
0x2E6	r	Miss	0010111 00 110	0x17	1	0	0x17	1	0	0x00	0	0	0x00	0	0
0x2EF	w	Hit	0010111 01 111	0x17	1	0	0x17	1	1	0x00	0	0	0x00	0	0
0x7A2	r	Miss	0111101 00 010	0x3D	1	0	0x17	1	1	0x00	0	0	0x00	0	0
0x01D	r	Miss	0000000 11 101	0x3D	1	0	0x17	1	1	0x00	0	0	0x00	1	0
0x7A3	w	Hit	0111101 00 011	0x3D	1	1	0x17	1	1	0x00	0	0	0x00	1	0
→ 0xA82	r	Miss	1010100 00 010	0x54	1	0	0x17	1	1	0x00	0	0	0x00	1	0
0x7A3	w	Miss	0111101 00 011	0x3D	1	1	0x17	1	1	0x00	0	0	0x00	1	0
0x015	w	Miss	0000000 10 101							0x00	1	1			
0x78D	r	Miss	0111100 01 101				0x3C	1	0						

Beim Direct-Mapped Cache bestimmt der Index den Block

Fetch on Write

   (a) Bei welchem Speicherzugriff wurde zum ersten Mal ein gültiger Block im Cache überschrieben? Geben Sie die zugehörige Adresse in hexadezimaler Notation an!

   0x 7A2

   Der Tag wird mit etwas komplett neuem überschrieben

   (b) Bei welchem Speicherzugriff wurde zum ersten Mal ein Block in den Hauptspeicher geschrieben? Geben Sie die zugehörige Adresse in hexadezimaler Notation an!

   0x A82

   (c) Wie hoch ist die *Miss-Rate* der oben ausgeführten Speicherzugriffssequenz (8 Zugriffe)?

   $\frac{6}{8} \triangleq 75\%$

   6 Misses bei 8 Zugriffen

   (d) Tragen Sie in obige Tabelle ein, wie sich die Verwaltungsdaten durch folgende Zugriffe ändern:

   wr(0x015) – rd(0x78D)

   Hinweis: Bei den Blöcken müssen Sie nur Felder befüllen, bei denen sich eine Änderung ergibt!

Platz für Notizen:

2013 W

5. *(10 Punkte)* Ein Prozessor besitzt eine Hierarchie aus zwei Caches. L1 ist als *Direct Mapped Cache* organisiert, L2 als *8-Way Set Associative Cache*. Die Adresslänge beträgt 12 Bit, die kleinste adressierbare Einheit ist 1 Byte. Beide Caches verwenden 4 Bit Offset und 4 Bit Index.

   Die CPU führt 10.000 Adressanfragen aus. Dabei gibt es 1.000 Misses beim L1-Cache und 200 Misses beim L2-Cache.

   (a) Berechnen Sie die Größe des L1-Caches!

   kl. adr. Einheit

   2^4 = 16 Blöcke à 2^4 x 1 Byte = 16 · 16

   $= 2^4 \cdot 2^4 = 2^8 = 256$ Byte

   (Offset, Index)

   (b) Um wie viel Prozent ist der L2-Cache größer als der L1-Cache?

   256 Byte · 8 Ways

   $= 2^8 \cdot 2^3 = 2^{11}$

   = 2048 Byte

   → 700 % Größer

   Der Cache ist nur 700% grösser, da der L1 256 Byte hat und der L2 2048, daher sind nur 7 Einheiten Unterschied.

   (c) Geben Sie die *Miss-Rate* des L2-Cache in Prozent an!

   Die Misses des L1 Cache sind die Zugriffe auf den L2 Cache.

$$\text{Miss-Rate} = \frac{\#\text{Misses}}{\#\text{Zugriffe}} = \frac{200}{1000} = 20\,\%$$

   Der L1 Cache hat 1000 Misses, daher sind das 1000 Zugriffe auf den L2 Cache. Von diesen 1000 Zugriffen sind 200 Misses. Für die Miss-Rate muss man die Misses/Zugriffe divideren.

   (d) Angenommen, die Zugriffszeit auf den L1-Cache (t_{L1}) beträgt bei einem Hit 10ns, auf den L2-Cache (t_{L2}) 100ns und auf den Hauptspeicher (t_{main}) 1000ns. H bezeichnet die Anzahl der Hits auf den jeweiligen Cache. Berechnen Sie die kumulierte Zugriffszeit (t_{kum}) nach folgender Formel und geben Sie das Ergebnis in Millisekunden (ms) an!

$$t_{kum} = H_{L1} \times t_{L1} + H_{L2} \times t_{L2} + H_{main} \times t_{main}$$

$$9000 \cdot 10\,ns + 800 \cdot 100\,ns + 200 \cdot 1000\,ns$$

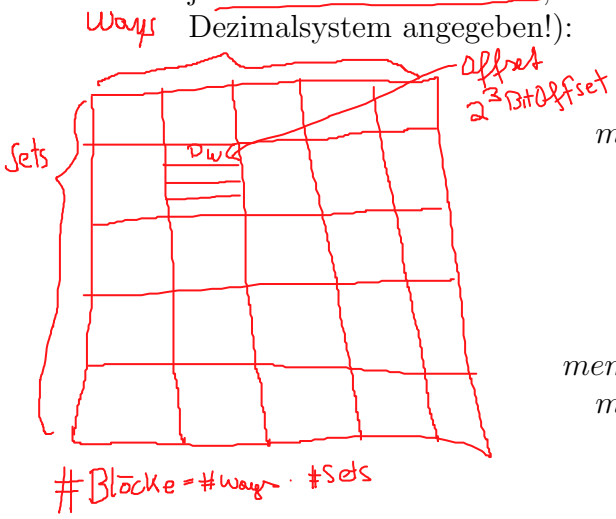
$$= 90\,\mu s + 80\,\mu s + 200\,\mu s = 370\,\mu s$$

$$= 0{,}37\,ms$$

   Hier muss man die insgesamte Zugriffszeit berechnen. Wie oben hervorgeht werden 10000 Anfragen gemacht. Von diesen Zugriffen gibt es 1000 Misses, daher wird auf den L1 9000 gehittet. Die Zugriffszeit für den L1 Cache sind 10ns, daher 9000*10. Die 1000 Misses ergeben 1000 Zugriffe auf den L2, von denen 800 gehittet werden. Die Zugriffszeit auf den L2 beträgt 100ns, daher 800 * 100. Die restlichen 200 Zugriffe erfolgen auf den Hauptspeicher, welcher eine Zugriffszeit von 1000ns hat. Daher 200 * 1000. Das ganze am Ende noch in Milisekunden umwandeln.

2012 W

6. *(23 Punkte)* Auf einem Prozessor, der einen *Direct Mapped Cache* bestehend aus 2 Blöcken zu je 8 Datenwörtern besitzt, wird folgende Befehlssequenz ausgeführt (alle Konstanten sind im Dezimalsystem angegeben!):



$R1$	$\leftarrow$	13
$memory[26]$	$\leftarrow$	$R1$
$R0$	$\leftarrow$	6
$R1$	$\leftarrow$	$memory[(R0)+]$
$R2$	$\leftarrow$	$memory[(R0)+]$
$R3$	$\leftarrow$	$memory[(R0)+]$
$R4$	$\leftarrow$	10
$memory[(R4)]$	$\leftarrow$	$R0$
$memory[27]$	$\leftarrow$	$R2$
$R4$	$\leftarrow$	$R4 + 1$
$R2$	$\leftarrow$	$memory[R4 + 20]$
$R5$	$\leftarrow$	$memory[R4 - 5]$
$R6$	$\leftarrow$	$memory[-(R4)]$
$memory[(R6)]$	$\leftarrow$	$R2$

Post-Dek.

Ein Direct Mapped Cached hat nur einen Way.

Offset: 3Bit

Mit dem Index werden die Sets angesprochen. Daher ein 1 Bit (0, 1). Der Offset dient dazu im Block zu navigieren. Mit dem Tag kann der Block identifiziert werden.

(a) *(11 Punkte)* Bestimmen Sie die Datenadressen, die sich aus dem Ablauf der Befehlssequenz ergeben, und tragen Sie diese in die Tabelle ein.

(b) *(10 Punkte)* Geben Sie für jede Adresse an, von welchem Cache-Block die Daten gelesen werden und ob es sich um einen *hit* oder *miss* handelt. Tragen Sie bei einem *miss* zusätzlich ein, von welchen Adressen die Daten geladen werden.

Block wird durch den Index bestimmt, daher kann es in diesem Beispiel nur 0 oder 1 sein, da der Index nur 1 Bit lang ist. Aus der Adresse ist es das 4te Bit von rechts gelesen.

Jeder Block besitzt 8 Datenwörter,

1-Bit Index, 3-Bit Offset

$(26)_{10} = 11010$

Adresse	hit/miss	Cache-Block	Inhalt
26	m	1	24 31
6	m	0	0 ... 7
7	h	0	0 7
8	m	1	8 15
10	h	1	8 15
27	m	1	24 31
31	h	1	24 ... 31
6	h	0	0 ... 7
10	m	1	8 15
9	h	1	8 15

0 7
8 15
16 23
24 31

(c) *(2 Punkte)* Berechnen Sie die *miss-rate* des Caches:

$$\textit{miss-rate} = \frac{\text{Anz. Misses}: 5}{\text{Anz. Zugriffe}: 10} = \frac{1}{2} = 50\,\%$$