

---

# 6. Übung

## Cache-Speicher



---

### Arbeitsweise eines Cache-Speichers

- ❑ Cache-Steuerung prüft, ob
  - Der zur Speicheradresse gehörende Hauptspeichereintrag als Kopie im Cache steht (**Bedingung 1**) und
  - Dieser Cache-Eintrag durch das Gültigkeits-Bit (Valid-Bit) als gültig gekennzeichnet ist (**Bedingung 2**)
- ❑ Prüfung führt zu einem Cache-Treffer oder zu einem Fehlzugriff.
- ❑ Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.



# Arbeitsweise eines Cache-Speichers

---

- ❑ Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.

## Lesezugriffe (read miss)

- Lesen des Datums aus dem Hauptspeicher und Laden des Cache-Speichers
- Kennzeichnen der Cache-Eintrag als gültig (V-Bit setzen)
- Speichern der Adressinformation im Adress-Speicher des Cache-Speichers



# Arbeitsweise eines Cache-Speichers

---

- ❑ Cache-Fehlzugriff (Cache-miss): eine der beiden Bedingungen ist nicht erfüllt.

## Schreibzugriffe (write miss)

Aktualisierungsstrategie bestimmt, ob

- der entsprechende Block in den Cache geladen und dann mit dem zu schreibenden Datum aktualisiert wird oder, ob
- nur der Cache aktualisiert wird und der Hauptspeicher unverändert bleibt



# Arbeitsweise eines Cache-Speichers

---

- Cache-Treffer (Cache-hit, read hit, write hit):
  - Beide Bedingungen 1 und 2 sind erfüllt
  - Zugriff erfolgt auf den Cache-Speicher



## Aufgabe 1

---

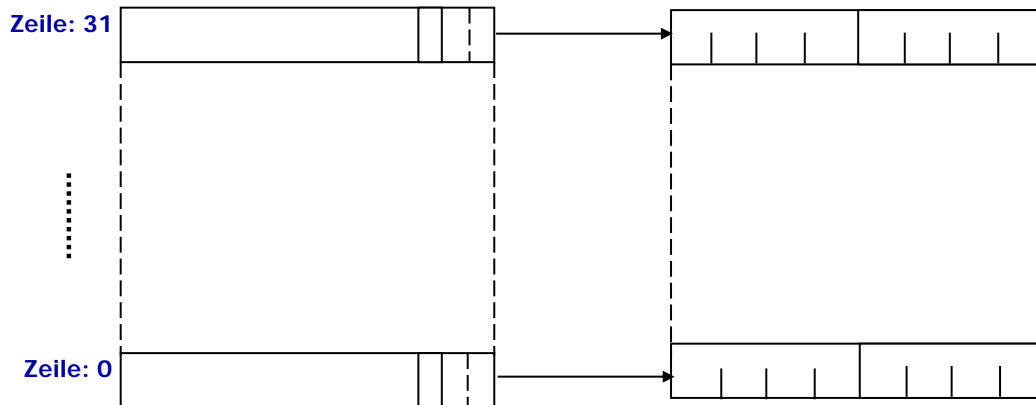
In einem Mikroprozessorsystem mit 32-bit-Datenzugriff auf den Hauptspeicher ist ein Daten-Cache vorhanden. Das Laden des Caches erfolgt in Blöcken von je **acht** Bytes, d. h. von zwei Wörtern. Die Hauptspeicheradresse umfasst 24 Bits; die Cache-Kapazität beträgt **256** Bytes.

1. Geben Sie die Anzahl der Cache-Zeilen an und skizzieren Sie die Unterteilung der Hauptspeicheradresse für einen *direct-mapped*-Cache (DM) , vollassoziativen Cache (AV) und 4-way-set-assoziativ-Cache (A4).



# Lösung 1.1

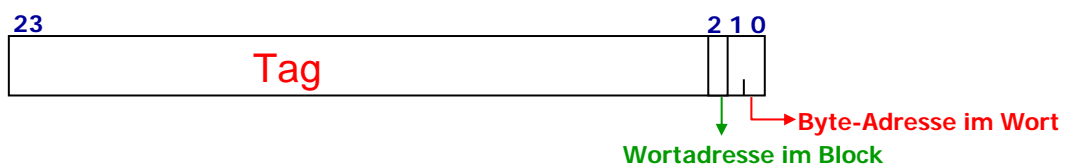
$$\text{Cache-Zeilen} = \frac{\text{Cache-Kapazität}}{\text{Blockgröße}} = \frac{256}{8} = 32 \text{ Zeilen}$$



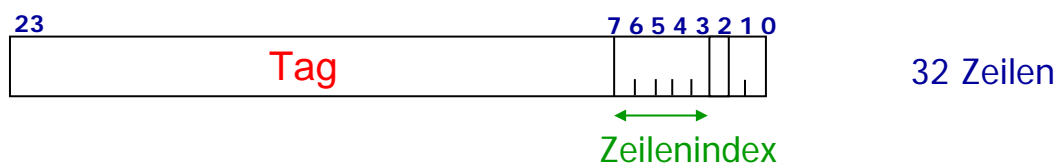
# Lösung 1.1

Unterteilung der HPS-Adresse:

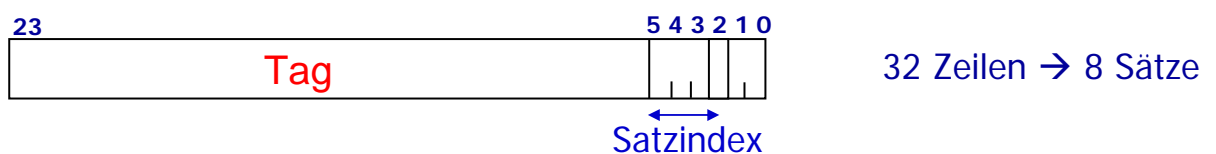
AV



DM



A4



# Lösung 1.2

2. Geben Sie für die drei Cachespeicher die Anzahl der benötigten Vergleiche und die zu vergleichende Bitzahl an.

Anzahl der zu vergleichenden Bits = Tag-Länge

32	Vergleicher	Tag-Länge
VA	32	$24 - 3 = 21$
DM	1	$24 - 8 = 16$
A4	4	$24 - 6 = 18$



# Lösung 1.3

3. Welche Zeilen sind bei diesen Cachespeichern hinsichtlich der Blockersetzung als zusammengefasst zu betrachten, z. B. für einen Alterungsmechanismus nach dem LRU-Prinzip.

- **VA:** Zeile 0 bis 31
- **DM:** nur die durch den Zeilenindex ausgewählte Zeile
- **A4:** jeweils die 4 Zeilen des durch den Satzindex angewählten Satzes



# Aufgabe 2

In die drei Cachespeicher der Aufgabe 1 sind drei Hauptspeicherblöcke mit den folgenden in hexadezimaler Schreibweise angegebenen Adressen in der angeführten Reihenfolge zu laden:

\$000008

\$0000F8

\$000108

Die Caches seien zu Beginn leer, und das Laden soll jeweils in die Zeile mit der niedrigsten verfügbaren Zeilennummer erfolgen. Geben Sie bei der Zuordnung von Blöcken zu Zeilen nicht die Adressen sondern die Blocknummern an.



## Lösung 2

Block: 2 Wörter = 8 Byte

Adressen:

	8	7	6	5	4	3	2	1	0
\$000008:	0000	0000	0000	0000	0000	1	0	0	0
\$0000F8:	0000	0000	0000	0000	1111	1	0	0	0
\$000108:	0000	0000	0000	0001	0000	1	0	0	0

Blocknummer (Index/ Tag)      Byte auswählen      Wort auswählen

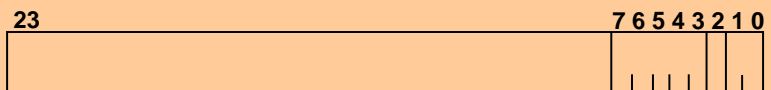
Blocknummer: 21 Bit lang

\$000008 → Block \$000001 =  $1_{10}$

\$0000F8 → Block \$00001F =  $31_{10}$

\$000108 → Block \$000021 =  $33_{10}$





Block: 2 Wörter = 8 Byte

Adressen:

	8	7	6	5	4	3	2	1	0
\$000008 :	0000	0000	0000	0000	0000	1	0	0	0
\$0000F8 :	0000	0000	0000	0000	1111	1	0	0	0
\$000108 :	0000	0000	0000	0001	0000	1	0	0	0

← Blocknummer (Index/ Tag)      ↓ Byte auswählen  
    Wort auswählen

Zeilennummer: 5 Bit lang (Bits 3-7)

\$000008 → Zeile 00001 =  $1_{10}$   
 \$0000F8 → Zeile 11111 =  $31_{10}$   
 \$000108 → Zeile 00001 =  $1_{10}$



Block: 2 Wörter = 8 Byte

Adressen:

	8	7	6	5	4	3	2	1	0
\$000008 :	0000	0000	0000	0000	0000	1	0	0	0
\$0000F8 :	0000	0000	0000	0000	1111	1	0	0	0
\$000108 :	0000	0000	0000	0001	0000	1	0	0	0

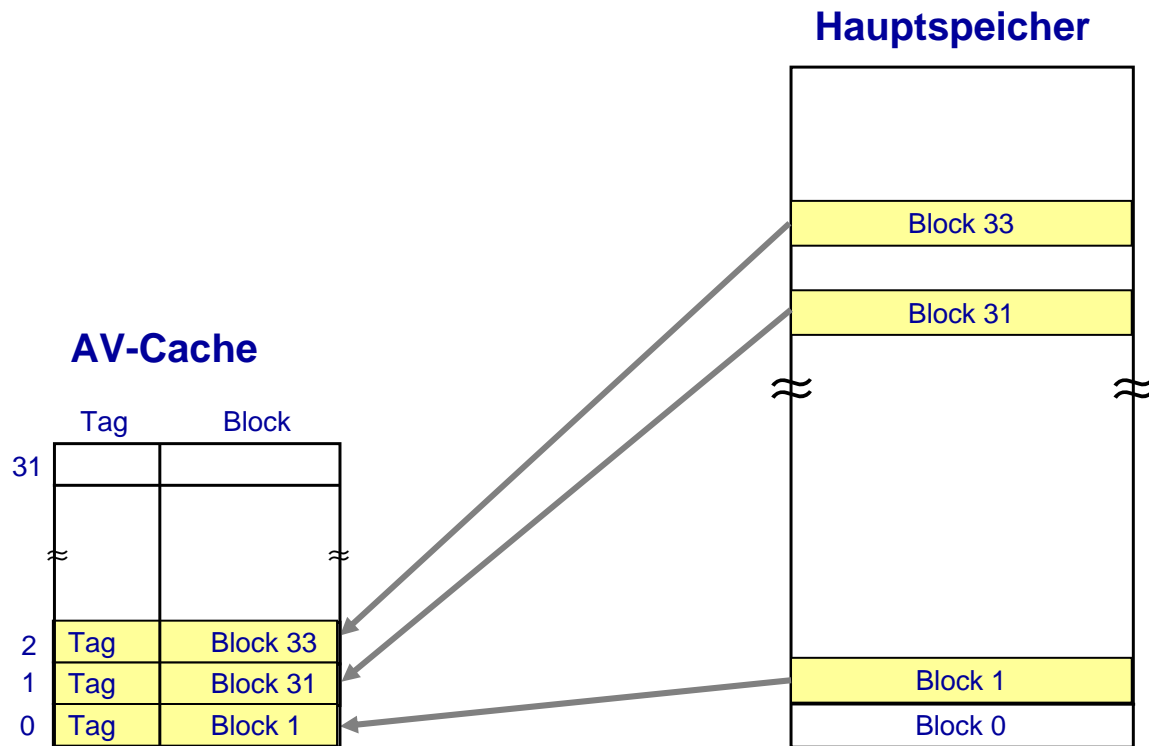
← Blocknummer (Index/ Tag)      ↓ Byte auswählen  
    Wort auswählen

Satznummer: 3 Bit lang (Bits: 3 -5)

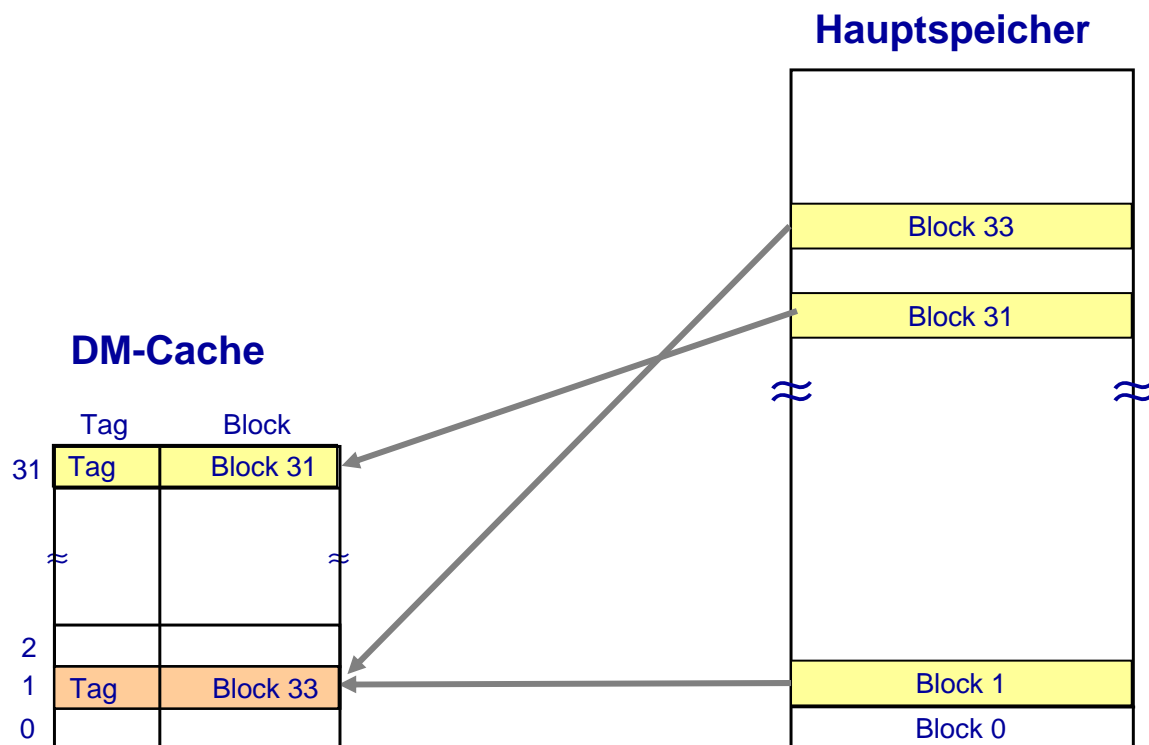
\$000008 → Satz 001 =  $1_{10}$   
 \$0000F8 → Satz 111 =  $7_{10}$   
 \$000108 → Satz 001 =  $1_{10}$



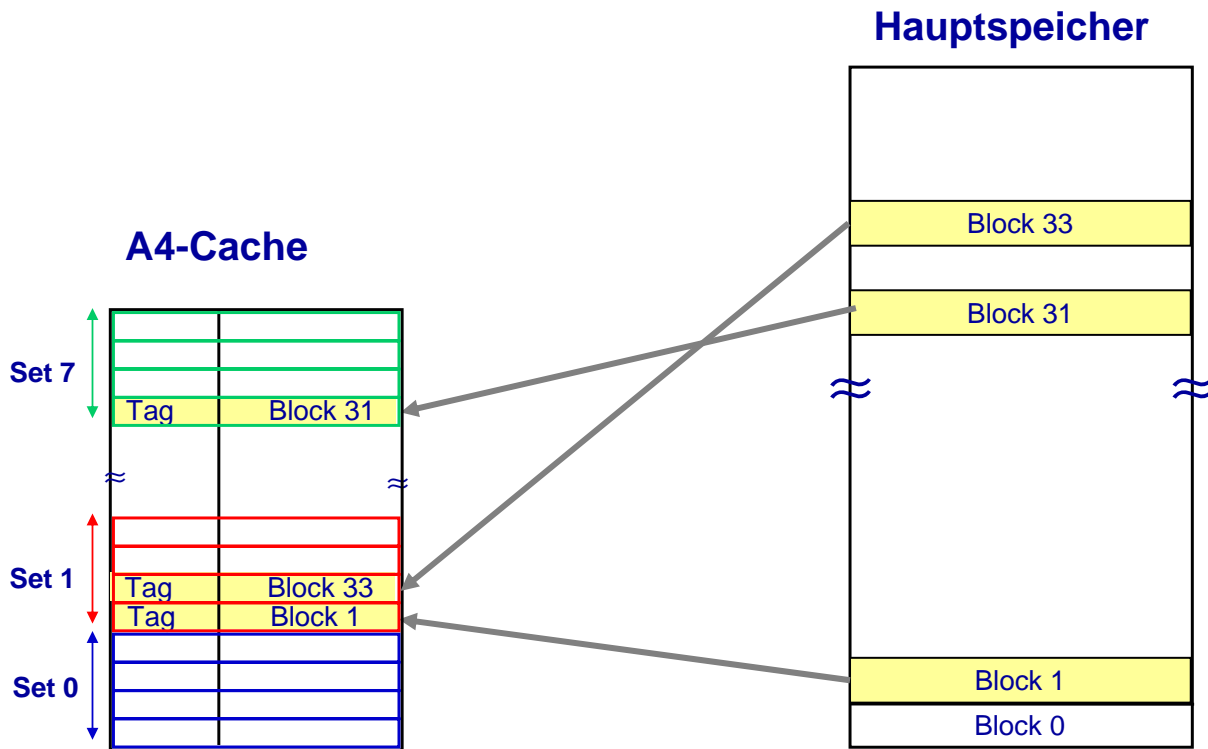
# AV-Cache



# DM-Cache



# A4-Cache



## Wohin wird ein Block abgebildet?

Blocknummer = (Hauptspeicheradresse) **div** (Blockgröße)

Zeilennummer = (Blocknummer) **mod** (Zeilenanzahl)

Satznummer = (Blocknummer) **mod** (Satzanzahl)

- $8_{16}$       Blocknummer:  $8_{16} \text{ div } 8 = 1$   
                  Zeilennummer:  $1 \text{ mod } 32 = 1$   
                  Satznummer:  $1 \text{ mod } 8 = 1$
- $F8_{16}$       Blocknummer:  $F8_{16} \text{ div } 8 = 31$   
                  Zeilennummer:  $31 \text{ mod } 32 = 31$   
                  Satznummer:  $31 \text{ mod } 8 = 7$
- $108_{16}$      Blocknummer:  $108_{16} \text{ div } 8 = 33$   
                  Zeilennummer:  $33 \text{ mod } 32 = 1$   
                  Satznummer:  $33 \text{ mod } 8 = 1$



# Aufgabe 3

Gegeben seien drei Cache-Speicher DM, A2 und AV, die jeweils **vier** Cache-Blöcke besitzen mit je **einem Byte**. Der Cache DM ist als direkt-abgebildeter Cache (direct-mapped) organisiert; Cache A2 als 2-fach assoziativer Cache (2-way-set-associativ); Cache AV ist vollassoziativ (fully-associativ). Bei den Cachespeichern A2 und AV soll die „least recently used“-Ersetzungsstrategie LRU angewendet werden.

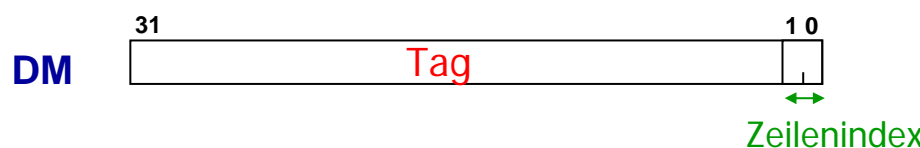
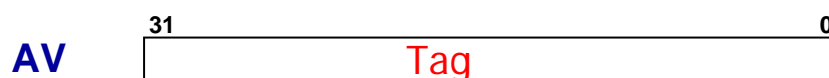
Nehmen Sie an, die Cachespeicher seien zu Beginn leer, und es soll eine Serie von einzelnen Bytes mit den folgenden 32-Bit-Adressen gelesen werden:

**9, 42, 13, 9, 23, 12, 13, 42**

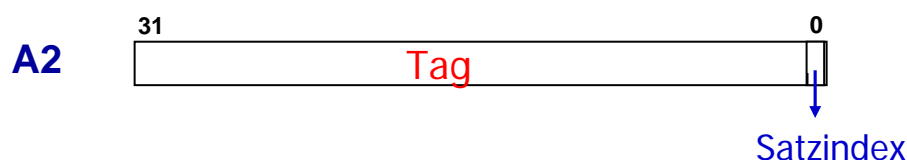


# Aufgabe 3

1. Geben Sie für die drei Cachespeicher an, wie viele Bits zur Verwaltung eines Cacheblocks benötigt werden. Dabei sollen für den Zustand des Cache-Blocks zwei Statusbits verwendet werden (Valid-Bit und Dirty-Bit).



4 Zeilen



4 Zeilen → 2 Sätze



# Lösung 3.1

	Tag-Länge	Tag+Statusbits
VA	32	$32 + 2 = 34$
DM	30	$30 + 2 = 32$
A2	31	$31 + 2 = 33$



Anzahl der notwendigen Bits zur Verwaltung eines Cacheblocks



# Lösung 3.2

2. Geben Sie für die drei Cachespeicher die Anzahl der erforderlichen Vergleiche und die jeweils zu vergleichende Bitanzahl an.

	Vergleicher	Tag-Länge
VA	4	32
DM	1	30
A2	2	31



# Lösung 3.3

3. Geben Sie nun tabellarisch für jeden Cache an, ob es sich beim Lesezugriff auf die jeweilige Adresse um einen Cache-Hit oder um einen Cache-Miss handelt.

Lesezugriffe auf die Adressen (Hier: Adresse = Blocknummer)

**DM: Zeilennummer = Blocknummer mod Zeilenanzahl**

$9 \bmod 4 = 1$  Miss  
 $42 \bmod 4 = 2$  Miss  
 $13 \bmod 4 = 1$  Miss  
 $9 \bmod 4 = 1$  Miss  
 $23 \bmod 4 = 3$  Miss  
 $12 \bmod 4 = 0$  Miss  
 $13 \bmod 4 = 1$  Miss  
 $42 \bmod 4 = 2$  Hit

3	23
2	42 42 hit
1	<del>9</del> <del>13</del> <del>9</del> 13
0	12



# Lösung 3.3

**AV:**

Cache ist voll: Ersetzungsstrategie LRU

$9 \rightarrow$  Zeile 0 Miss  
 $42 \rightarrow$  Zeile 1 Miss  
 $13 \rightarrow$  Zeile 2 Miss  
 $9$  in Zeile 0 Hit  
 $23 \rightarrow$  Zeile 3 Miss  
 $12 \rightarrow$  Zeile 1 Miss  
 $13$  in Zeile 2 Hit  
 $42$  in Zeile 0 Miss

3	23
2	13 13 Hit
1	<del>42</del> 12
0	9 <del>9</del> Hit 42

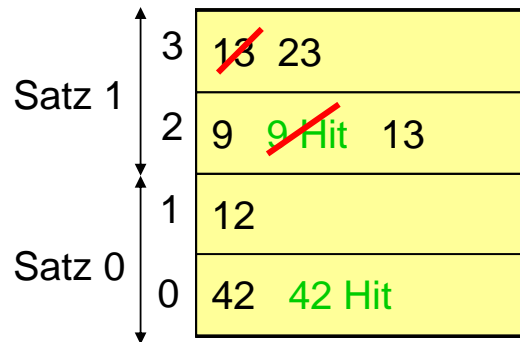


# Lösung 3.3

## A2: Satznummer = Blocknummer mod Satzanzahl

$9 \bmod 2 = 1$	Miss
$42 \bmod 2 = 0$	Miss
$13 \bmod 2 = 1$	Miss
$9 \bmod 2 = 1$	Hit
$23 \bmod 2 = 1$	Miss
$12 \bmod 2 = 0$	Miss
$13 \bmod 2 = 1$	Miss
$42 \bmod 2 = 0$	Hit

Bei vollem Satz: LRU



## Aufgabe 4

### Cache Hit/Miss-Rate

Programmschleife zur Multiplikation von 512 16-Bit-Zahlen

```
for (mul=1, j=0; j<512; j++) mul *=g[j];
```

Vollassoziativer Daten-Cache (am Anfang leer) mit 64 Bytes pro Cache-Zeile.

→ 32 Zahlen pro Cache-Zeile



# Lösung 4

	Miss	Hit
mul = 1	1 1	
j = 0	1 1	
loop: read j		1 1 512
if (j >= 512) exit		
else		
read g[j]	1 16	1 496
read mul		1 1 512
compute mul *g[j]		
write mul		1 1 512
read j		1 1 512
compute j+1		
write j		1 1 512
jump to loop		
1. Durchlauf	18	3056
2. Durchlauf	0,06 %	99,994 %



## Aufgabe 5

Bei einem Cache-Speicher mit einer Speicherkapazität von **128 Kbyte** ist die Hauptspeicheradresse in ein **16 Bit** Tag-Feld, ein **12 Bit** Index-Feld und ein **4 Bit** Byte-Offset unterteilt.

1. Bestimmen Sie die Blockgröße in Bytes.
2. Wieviele Einträge besitzt der Cache-Speicher?
3. Wie ist der Cache-Speicher organisiert?



# Aufgabe 5.1

---

Cache-Kapazität = **128 Kbyte**

**16 Bit** Tag-Feld, **12 Bit** Index-Feld, **4 Bit** Byte-Offset

Blockgröße:



# Aufgabe 5.2

---

Cache-Kapazität = **128 Kbyte**

**16 Bit** Tag-Feld, **12 Bit** Index-Feld, **4 Bit** Byte-Offset

Anzahl der Einträge im Cache:



# Aufgabe 5.3

---

Cache-Kapazität = **128 Kbyte**

**16 Bit** Tag-Feld, **12 Bit** Index-Feld, **4 Bit** Byte-Offset

Organisation:



# Aufgabe 6

---

Es soll ein 3-fach-assoziativer (*3-way set associative cache*) Cache-Speicher mit **128 Sätzen** und einer Blockgröße von **8 Byte** realisiert werden.

Nehmen Sie an, dass die Hauptspeicheradresse 32 Bit breit ist. Zur Verwaltung eines Cacheblocks wird zwei Statusbits (Valid-Bit: V und Dirty Bit: D) verwendet.

Bestimmen Sie den insgesamt erforderlichen Speicherbedarf zur Realisierung dieses Cache-Speichers.



# Lösung 6

---

3-fach-assoziativer (*3-way set associative cache*)

Cache-Speicher mit **128 Sätzen** und einer Blockgröße von **8 Byte**

Speicherbedarf für eine Zeile:

**Tag-Länge + Anzahl der Statusbits + Blockgröße**

