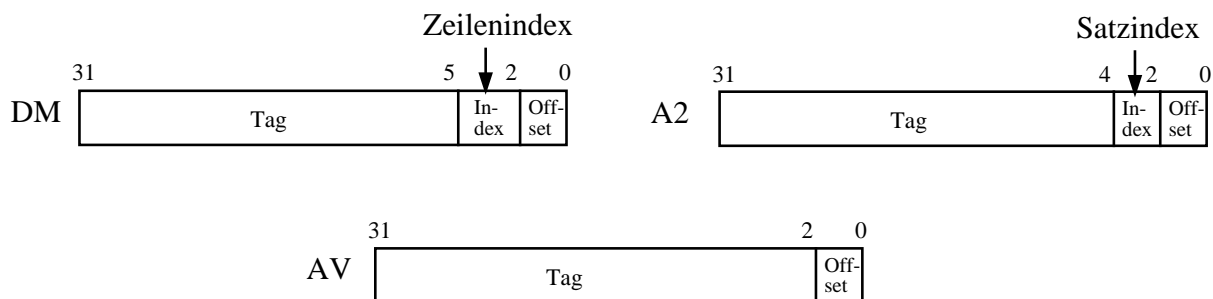




### Lösung 1

1. Aus dem folgenden Bild kann man ablesen, wieviels Bits jeweils auf den „Tag“ entfallen. Zusammen mit den beiden Zustandsbits (*Valid* und *Dirty*) werden daher 29 Bits beim DM-Cache, 30 Bits beim A2-Cache und 32 Bits beim AV-Cache für die Verwaltung eines Cache-Blocks benötigt.



2. Beim AV-Cache sind **acht** 30-Bit-Vergleicher, beim DM-Cache ist ein **einziger** 27-Bit-Vergleicher und beim A2-Cache sind **zwei** 28-Bit-Vergleicher erforderlich.
3. Beim AV-Cache: Zeile 0 bis Zeile 7  
Beim DM-Cache: nur die durch den Zeilenindex angewählte Zeile  
Beim A2-Cache: jeweils die zwei durch den Satzindex angewählten Zeilen
4. Cache-Hit oder Cache-Miss beim Lesezugriff auf die jeweilige Adresse:

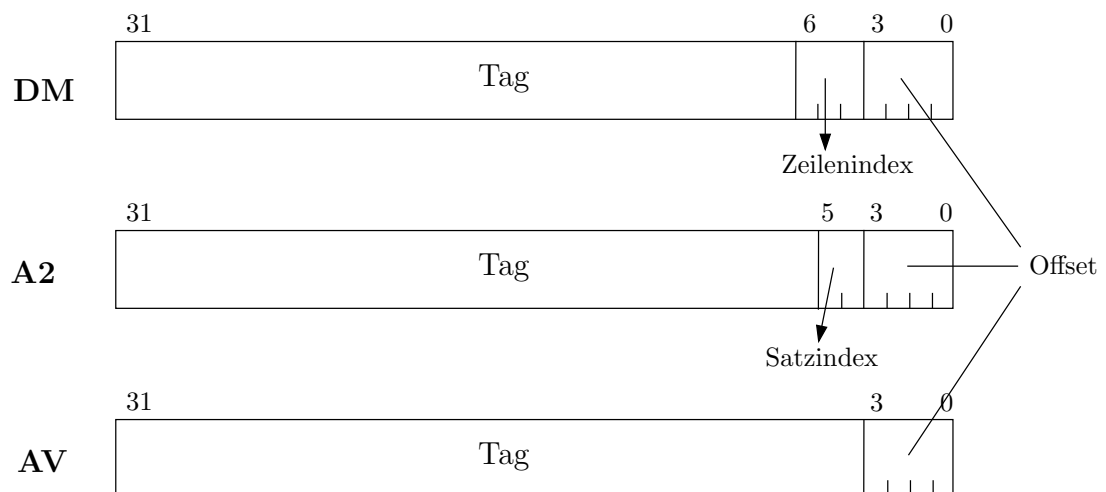
Adresse	70	9	39	83	66	68	35	80	93	67	79	37	84	9	Hits
Block-Nr.	17	2	9	20	16	17	8	20	23	16	19	9	21	2	
DM-Cache	-	-	-	-	-	-	-	×	-	-	-	-	-	×	2 ×
Zeilen-Nr.	1	2	1	4	0	1	0	4	7	0	3	1	5	2	
A2-Cache	-	-	-	-	-	×	-	-	-	-	-	×	-	×	3 ×
Satz-Nr.	1	2	1	0	0	1	0	0	3	0	3	1	1	2	
AV-Cache	-	-	-	-	-	×	-	×	-	×	-	×	-	-	4 ×

5. Zustand der drei Caches nach dem letzten Speicherzugriff:

DM-Cache:			A2-Cache:			AV-Cache:		
Set	Tag	Daten	Set	Tag	Daten	Set	Tag	Daten
0	2	$m[64 - 67]$	0	4	$m[64 - 67]$	0	2	$m[08 - 11]$
1	1	$m[36 - 39]$		5	$m[80 - 83]$		21	$m[84 - 87]$
2	0	$m[08 - 11]$	1	5	$m[84 - 87]$		9	$m[36 - 39]$
3	2	$m[76 - 79]$		2	$m[36 - 39]$		20	$m[80 - 83]$
4	2	$m[80 - 83]$	2	0	$m[08 - 11]$		16	$m[64 - 67]$
5	2	$m[84 - 87]$		-	- - - -		8	$m[32 - 35]$
6	-	- - - -	3	5	$m[92 - 95]$		23	$m[92 - 95]$
7	2	$m[92 - 95]$		4	$m[76 - 79]$		19	$m[76 - 79]$

## Lösung 2

1. Unterteilung der Hauptspeicheradresse:



2.

Cache	Anzahl der Vergleiche
<b>DM</b>	1
<b>A2</b>	2
<b>AV</b>	8

3.

Adresse	0x44	0xA0	0xC3	0x9E	0x66	0x2D	0x6B	0x49
<b>DM</b>	×	—	—	×	×	—	×	—
<b>A2</b>	×	×	—	×	—	—	×	×
<b>AV</b>	—	—	—	—	—	×	×	×

Lösung 3

Adresse	64	32	E4	18	E0	7A	A2	F0	E3
read/write	w	r	w	r	r	r	r	w	r
Index	6	3	6	1	6	7	2	7	6
Tag	0	0	1	0	1	0	1	1	1
Hit/Miss	Miss	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Hit
write back?	nein	nein	ja	nein	nein	nein	nein	nein	nein

Lösung 4

Die Programmschleife sieht in C-Notation etwa wie folgt aus:

```
for ( sum = 0, i = 0 ; i < 128 ; i++ ) sum += a[i] ;
```

Die Anweisung lässt sich in die folgenden Einzeloperationen zerlegen:

	a)		b)	
	Miss	Hit	Miss	Hit
sum = 0	1		1	
i = 0	1		1	
loop: read i		128		
i < 128 ?				
exit, if false				
read a[i]	16	112	16	112
read sum		128		
compute sum+a[i]				
write sum		128		
read i		128		
compute i+1				
write i		128		
jump to loop				
	18	752	18	112
	2,3%	97,7%	13,8%	86,2%
c)	10	760	10	120
	1,3%	98,7%	7,7%	92,3%