

7. 可以方便地实现虚拟内存到物理内存的映射, 可以直接使用地址的组索引和块内偏移来计算物理地址, 提高访问速度。

6. 使用高位作为组索引, 中间位作为标签的话, 每T组的大小会变得非常小, 导致组相对较多, 而每组的块数相对较少, 这会导致缓存的利用率降低, 同时影响缓存的命中率。反之, 则可以更好利用缓存存储空间, 提高缓存命中率。

8. (1) $3\% \times 110 + 97\% = 4.27$

(2) 1GB 被划分为 64KB 的块

$$2^{20} / 2^6 = 2^{14} = 16384 \text{ T块}$$

块命中率 $1 / 16384$ 缺失率: $1 - \frac{1}{16384}$

$$4.27 \times (1 - \frac{1}{16384}) \approx 4.27$$

(3) 局部性原理:

空间局部性: 程序在某一段时间内访问的数据通常在空间上是相邻的

时间局部性: 程序在某一段时间内的数据通常是相同或相似的

数组在空间上相邻, 通过缓存可以大大提高访问性能, 使得访问数组的随机位置的平均访问时间几乎只取决于缓存。

(4) 平均缓存命中率尽可能大 $x + (1-x) \times 110 < 105 \quad x > 46\%$

9. 组数量	组索引位数	标签位数	偏移位数
1 32	5	21	6
2 8	3	23	6
3 1	0	26	6
4 2 ⁸	8	18	6
5 2 ⁶	6	19	7
6 2 ⁸	8	18	6
7 2 ⁶	6	20	6
8 2 ⁵	5	20	7



$$10. \Delta T_1 = 0.22 \times (1 - P_1) + 100 \times P_1$$

$$\Delta T_2 = 0.52 \times (1 - P_2) + 100 \times P_2$$

$$\Delta T_1 < \Delta T_2$$

$$\therefore 99.78P_1 - 99.48P_2 < 0.3$$

$$12) 0.22(1 - P_1) + 10^2 k P_1 < 0.52(1 - P_2) + 10^2 k P_2$$

$$0.22 + 0.22(k-1)P_1 < 0.52 + 0.52(k-1)P_2$$

$$0.22P_1 - 0.52P_2 < \frac{0.3}{k-1}$$

$$\therefore \frac{0.3}{k-1}$$



块地址

11.	索引	块内偏移	直接映射: 块内偏移 6 位	索引 4 位
0x100	0001	0000 0000 000	0x100 与 0x1005, 0x102	2 次替换
0x1005	0001	0000 0000 010	0x1305 与 0xff05	1 次替换
0x102	0001	0000 0010 000	共 3 次替换	
0x1045	0001	0000 0100 010		
0x1305	0001	0011 0000 010	2 路: $16/2 = 8$	3 位索引
0x2ee5	0010	1110 1110 010	000 000 000 00	100 011 100
0xff05	1111	1111 0000 010	1 次替换	

4 路: $16/4 = 4$ 2 位索引 00 00 00 01 00 11 00

1 次替换

8 路: 无替换



12. $256B - 2^8B - 00000000$

16个字节可存放 4个数组元素

直接映射 $a[0]$ $a[1]$ $a[2]$ $a[3]$ $a[0]$ 缺失 $a[1] \sim a[3]$ 命中

缺失率 $\frac{1}{4}$

2路组相联 缺失率 $\frac{1}{8}$



```
13. for (int i=0; i<128; ++i) {  
    for (int j=0; j<64; ++j) {  
        A[i][j] = A[i][j] + 64;  
    }  
}
```

```
for (int i=0; i<64; ++i) {  
    for (int j=0; j<128; ++j) {  
        B[i][j] = A[j][i];  
    }  
}
```

```
for (int i=0; i<128; ++i) {  
    for (int j=0; j<64; ++j) {  
        B[i][j] = B[i][j] + 1;  
    }  
}
```

```
for (int i=0; i<64; ++i) {  
    for (int j=0; j<128; ++j) {  
        A[j][i] = B[i][j];  
    }  
}
```



14. 4KB — 2^{12} 0000 0000 0000

1) 块大小为2字节可存放8个数组元素

优化前: $a[0][0]$ $a[0][1]$ $a[0][2]$ $a[0][3]$... $a[0][7]$

每次均缺失

$64 \times 128 = 2^{13}$ 且每次数据访问2

优化后: $a[0][0]$... $a[0][7]$

访问14次中, 仅写入 $a[0][0]$ 时缺失

$\frac{2^{14}}{14}$ 次缺失

2) 优化前 $64 \times 128 \times 2 = 2^{14}$ 次

优化后: 0次

13) $64 \times 128 = 2^{13}$ 容量



Date. . .

15. ~~2~~ 2块 每块16字节可存4个数

#

input				output			
0	1	2	3	0	1	2	3
0	miss	miss	miss	miss	miss	miss	miss
1	hit	hit	hit	miss	miss	miss	miss
2	hit	hit	hit	miss	miss	miss	miss
3	hit	hit	hit	miss	miss	miss	miss



No.

Date.

16.11) $512/16 = 32$ $32/2 = 16$ 个块 $16/4 = 4$ 个数组元素
命中率 $\frac{1}{4}$

12) 是的。缓存总大小增大可以存储更多的数据,从而提高命中率

13) 不一定。块大小增大可提高命中率,但也可能导致更多的缓存未命中
更多的数据。

