

6. 解: 为了避免冲突, 如果使用高位作为组索引, 由于高位的取值范围较小, 容易导致多个不同的地址被映射到同一个组中, 造成冲突.

7. 解: 可以使缓存系统的管理与虚拟内存管理保持一致, 从而提高系统的整体性能和稳定性.

8. 1) 解: 缺失惩罚 = $10 - 1 = 109$ (周期)

$$\text{平均访问延时} = 1 + 109 \times 3\% = 4.27 \text{ (周期)}$$

2) 解: $1GB = 2^{20}KB$

$$\text{缓存缺失率} = 1 - \frac{64}{2^{20}} = 1 - 2^{-14}$$

$$\begin{aligned} \text{平均访问延时} &= 1 + 109 \times (1 - 2^{-14}) \\ &\approx 110 \text{ (周期)} \end{aligned}$$

3) 解: 局部性原理可以使缓存命中率提高, 减少缓存缺失带来的影响, 从而使程序运行效率更高.

4) 解: 令平均缓存命中率为 η .

则有

$$1 + (1 - \eta) \times 109 < 105$$

$$\eta > \frac{5}{109} \approx 4.59\%$$

9	组数量	组索引数 Bit	标签位数 Bit	偏移位数 Bit
1.	32	5	21	6
2.	8	3	23	6
3.	1	0	26	6
4	256	8	18	6
5.	64	6	19	7
6	256	18	18	6
7.	64	6	20	6
8	32	5	20	7

10. 11 解: $\pi = 0.22(1-p_1) + 100p_1 = 0.22 + 99.78p_1$
 21 解: $0.22 + 0.22Kp_1 < 0.52 + 0.52Kp_2$
 即: $p_1 < \frac{15}{11K-10} + \frac{26}{11}p_2$

11. 解: 直接映射

0x1001 0x1021 替换 1 次

0x1005 0x1045. 0x1305. 0x2005, 0x1105.
 替换 4 次. 共 5 次.

2路组相联 前 24 不替换

185 替换 3 次 共 3 次

4... 共 4 组 替换 3 次.

8... 共 2 组, 不替换.

12. 缓存A:

array[0]~[3], [4]~[7] ... [92]~[95]

每个数组元素存入一个缓存块中.

其中 array[0]~[3] ... [64]~[67] } 映射到1号

array[28]~[31] ... [92]~[95] / 一个块

存入 [64]~[95] 时, [0]~[31] 刷新

[32]~[63] 不变.

缺失率 = 2/3.

B: 分为8组

每次都是刷新后访问

缺失率 = 1

13. 解: for (int j=0; j<128; ++j) {
 for (int i=0; i<64; ++i)
 A[j][i] = A[j][i] + 1
}

14. 1) 解: $4KB = 4 \times 2^{10} B = 2^{12} B$

由 $\frac{2^{12}}{32} = 2^7 B$ 存储需要 815 标签

$128 \times 64 = 2^{13}$ 个数据 64×2^6 个数据

每次取 $2^7/2^6 = 2$. 每2个数据发生缺

失. $2^{13}/2 = 2^{12}$ 次

每次后: 缓存环不缺失. 缓存环开始时缺失共

127次

17. 优化前: 127次

优化后: 63次

37. 优化前: $4 \times 64 = 256KB$

优化后: 4KB

	input				output			
15.	0	1	2	3	0	1	2	3
0	miss	ϕ	ϕ	ϕ	miss	miss	miss	miss
1	miss	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
2	miss	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
3	miss	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

16. (1) 命中率 = $\frac{15}{16} = 93.75\%$

(2) 可以. 缓存变大 + 增大后, 块内存储的数据变多, 每次请求之间的访问次数增大.

(3) 不可以. 每个块存储一个内存数据.