5/18 第四章

6. 为了利用数据的局部性、提高缓存的容量和命中率，并能够更方便地进行标记匹配和管理缓存。如果使用高位作为组索引，中间位作为标签，那么中间位作为标签的位数就会相应增加，导致每个组内能够存放的缓存行数量减少。

7. 可以在硬件设计上采取更简化的缓存访问逻辑，可以共享硬件逻辑来处理地址的转换和缓存索引，从而减少硬件复杂性和成本。同时可以更方便地管理缓存与虚拟内存的一致性。

8. (1) 3% × 110 + 97% × 1 = 4.27 周期

(2) $1GB = 1024MB = 1024^2KB = 2^{20}KB$

$\frac{2^6}{2^{20}} \times 1 + \frac{2^{20}-2^6}{2^{20}} \times 110 =$ ~~110.01 周期~~ 110 周期

(3) 在(1)中，平均缓存缺少率R有3%，导致系统平均访问延迟较低，而(2)中，缓存过小，几乎每次访问都会导致缓存缺失，使得系统平均访问延迟非常高。所以充分利用数据的局部性，尽可能减少缓存缺失，可以降低访存延迟，提高系统性能。

(4) 设平均缓存命中率到少为 x.

则 $x \cdot 1 + (1-x) \cdot 110 \leq 105 \Rightarrow x \geq 4.59\%$

9.

缓存大小 ÷ 块大小 ÷ 相联度 = 组数量.　　块内偏移↑ 块大小

编号	地址位数bit	缓存大小KB	块大小Byte	相联度	组数量	组索引位数bit	标签位数bit	偏移位数bit
1	32	4	64	2	32	5	21	6
2	32	4	64	8	8	3	23	6
3	32	4	64	全相联	1	0	26	6
4	32	16	64	1	256	8	18	6
5	32	16	128	2	64	6	19	7
6	32	64	64	4	256	8	18	6
7	32	64	64	16	64	6	20	6
8	32	64	128	16	32	5	20	7

10. (1) $(1-p1)\cdot 0.22ns + p1\cdot 100ns < (1-p2)\cdot 0.52ns + p2\cdot 100ns$

$99.78p1 + 0.22 < 99.48p2 + 0.52$

$100(p1-p2) < 0.3$

$\therefore p1 < p2 + 0.003$ 时，A优于B.

(2) $(1-p1)\cdot 0.22 + p1\cdot 0.22k < (1-p2)\cdot 0.52 + 0.52k\cdot p2$

$11 - 11p1 + 11kp1 < 26 - 26p2 + 26kp2$

$(11k-11)p1 < 15 + (26k-26)p2$

$(k-1)(11p1 - 26p2) < 15$

$11p1 - 26p2 < \frac{15}{k-1}$

2005: 12005, ff05: 65285. → 5+4080×16, 1005=13+16×62, 1001=9+62×16, 1021=5+63×16, 1045=5+65×16, 1305=9+81×16.

11. 直接映射：~~0x1005→0x1001, 0x1021→0x1005, 0x1045→0x1021 ∈第一块.~~

~~0x2005⇒741+11×64×16=(11×64+37)+11×64×16, 在第12块.~~

~~0xff05⇒(12×64+5)+63×64×16, 在第13块~~

~~=替换3次.~~ ①0x1021→0x1005, ②0x1305→0x1001, ③0x2005→0x1045, ④0xff05→0x2ee5.

∴替换4次.

2路组联：1001=组1+125×8, 1005=5+125×8, 1021=5+127×8, 1045=5+130×8, 1305=1+163×8, 2ee5%8=5

①1045→1005或1021, ②2ee5→5号组块, ③ff05→5号组块 ff05%8=5.

∴替换3次.

4路组联：1001%4=1, 1005%4=1, 1021%4=1, 1045%4=1, ①1305%4=1, ②2ee5%4=1, ③ff05%4=1

∴替换3次.

8路组联：1001%2=1, 1005%2=1, ... 都取余为1.

∴替换0次 (7<8)

12. A: 16块 2路组(一组8块, 共8组)

一次外循环.

96×4÷16=24块, 缺失 24-16=8块.

A和B一致: 缺失率 $= \frac{8}{24} = \frac{1}{3} \approx 33.33\%$

13.
```
for(int j=0; j<64; j+=2)
    for(int i=0; i<64; i++){
        A[j][i] = A[j][i]+1; A[j+1][i] = A[j+1][i]+1;
    }
}
```

14. (1) 优化前: $\frac{64\times128}{2} = 4096$次.

优化后: 4096次

(2) 优化前: ?

优化后: $\frac{64\times128}{4} = 2048$次

(3) 优化前: $(\frac{64\times128}{2} \div \frac{4\times1024}{32}) \times 4KB = 128KB$.

优化后: $(\frac{64\times128}{4} \div \frac{4\times1024}{32}) \times 4KB = 64KB$.

15.

	input 列0	列1	列2	列3	output (FIFO?) 列0	列1	列2	列3
行0		✓	✓	✓	×	×	×	×
行1	✓	✓	✓	✓	✓	✓	✓	×
行2	✓	×	×	×	✓	×	×	×
行3	×	×	×	×	×	×	×	×

16.(1) 每个块可以储存4个int数据，一共32块，16组.

32×4=64个int数据.

$\frac{64}{128\times2}=25\%$.

(2) 可以改善。因为缓存大小决定了能够缓存的块数，缓存越大，命中率越高。

(3) 可以改善。块的大小决定了每次从内存读取多少数据，读取数据越多，命中率越高。