

第5题上题作业已做

5/7: 6. 若采用地址的高位进行索引, 可能会使得地址相差一个块大小的两个变量被映射到同一组之中, 产生错误, 而若使用地址中间位索引, 则可将不同元素分别映射到高速缓存的不同组中, 提高高速缓存利用率, 也增加缓存命中率。

7. 使地址的组索引和块内偏移的总位数与虚拟内存存单元的页偏移位数相同, 可使得虚拟地址的组索引可直接用来查找缓存, 而缓存块内偏移量和组索引也可直接访问虚拟内存, 实现TLB和缓存的并行访问。同时, 还可以减少内存冲突和内存管理开销, 更好地利用局部性原理提高缓存命中率, 来整体提高系统性能。

8. (1)  $T_{delay} = 1 \times 0.97 + 110 \times 0.03 = 4.27$  (个) 周期

(2) 由于是完全随机访问, 故缓存命中率近似认为  $\frac{64}{1 \times 1024 \times 1024} \approx 0.006\%$ , 近似似认为命中率为0

因此,  $T_{delay} = 1 \times 2^{-14} + (1 - 2^{-14}) \times 110 \approx 109.99$  (个) 周期, 故此时  $T_{delay} \approx 110$  (个) 周期

(3) 在第1问中, 程序的平均缓存缺失率仅为3%, 说明了程序的局部性较好, 故此时存储系统平均访问延时仅为4.27个周期, 而第2问中完全随机访问故可以认为几乎不存在局部性, 此时  $T_{delay}$  便需要110个周期。由此可见, 有局部性且其较高的程序可以有效利用缓存, 从而减少了访问主存的需要, 可减少平均访问的延时, 从而提高处理器的性能。

(4) 缓存命中率为  $x$   $x + 110(1-x) < 105$   $109x > 5$   $x > 0.04587$

∴ 平均缓存命中率需高于4.587%, 方能使用缓存系统使用L1时获得性能收益。

9. 编号	地址位数 (bit)	缓存大小 (KB)	块大小 (Byte)	相联度	组数	组索引位数	标签 (bit)	偏移 (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



10.11) 由题意可知,  $T_A = 0.22 + 100P_1$ ,  $T_B = 0.52 + 100P_2$   $\therefore 0.22 + 100P_1 < 0.52 + 100P_2$   
 $\therefore 100(P_1 - P_2) < 0.3$   $\therefore P_1 < P_2 + 0.003$  时, 系统A平均内存访问时间小于系统B!

12) 由题意知,  $T_A = 0.22 + 0.22kP_1$ ,  $T_B = 0.52 + 0.52kP_2$   $\therefore 0.22 + 0.22kP_1 < 0.52 + 0.52kP_2$   
 $\therefore 0.22kP_1 < 0.3 + 0.52kP_2$   $\therefore P_1 < \frac{0.3}{0.22k} + \frac{0.52}{0.22} P_2$  即  $P_1 < \frac{26}{11} P_2 + \frac{15}{11k}$  时, 系统A平均内存访问时间小于系统B.

11.①: 有16个块, 若为直接映射, 则需要4位地址索引  $\therefore 0x1001$  与  $0x1021$ ,  $0x1005$  与  $0x1045$ ,  $0x1045$  与  $0x1305$ ,  
 $0x1305$  与  $0x2005$ ,  $0x2005$  与  $0x2ff5$  均会发生一次替换  $\therefore$  共发生5次替换

② 2路相联:  $16 \div 2 = 8$  (组), 需要3位地址索引. 所以此时  $0x1001$  和  $0x1021$  映射到一组,  $0x1005$  与  $0x1045$  映射到另一组  
 而后在  $0x1305$ ,  $0x2005$ ,  $0x2ff5$  均需发生一次替换  $\therefore$  共发生3次替换.

③ 4路相联:  $16 \div 4 = 4$  (组), 需要2位地址索引, 同理可知共需发生3次块替换

④ 8路相联:  $16 \div 8 = 2$  (组), 需要1位地址索引, 由于只有7个地址, 故可全部放下, 共发生0次替换

12.  $256 \div 16 = 16$  (个) 块, 故A为8组每组2块; B为16组, 而缓存中最多能放  $256 \div 4 = 64$  (个) 元素, 每块中放4个元素  
 直接映射: 一块中有4个元素, 而一次访问不涉及96个元素 故32~64位只会第一次查找时miss, 后续均为hit  
 但其他位由于发生替换, 均为miss  $\therefore$  缺失率 =  $\frac{64 \times 100 + 32}{96 \times 100} \times \frac{1}{4} = 16.75\%$ .

2路组相联:  $\therefore$  共有96个元素多于64个元素, 故会不断发生替换 (映射到缓存的相联组), 故即每一次访问新的块  
 即每个元素发生一次miss, 缺失率 =  $\frac{24 \times 100}{96 \times 100} = 25\%$   $\therefore$  在本题中2路组相联缺失率高于直接映射

13.  $\therefore$  A[0][0]与A[0][1]相邻, 故应将j套为外层循环, 代码层

<pre> 代码: for(int j=0; j&lt;18; ++j) {         for(int i=0; i&lt;4; ++i) {             A[i][j]=A[i][j+1];         }     }         </pre>	<p>14.11) 一个块中有8个int变量, <math>4096 \div 32 = 128</math> 个块  <math>\therefore</math> A[0][0]与A[0][1]间隔64个int, 故8个块 <math>\Rightarrow</math> 一直缺失  <math>\therefore</math> 在优化前, miss数 = <math>64 \times 128 = 8192</math> (次).          优化后 A[0][0] <math>\rightarrow</math> A[0][1] 在同一块内, 故不会发生替换  <math>\therefore</math> 优化后 miss数 = <math>64 \times 128 \div 8 = 1024</math> (次).</p>
--	---

14.12)  $\therefore$  全相联, 故当外层循环不执行完的情况, 后续缓存访问均不会发生miss  $\therefore$  优化前 miss数 =  $8192 \div 8 = 1024$  (次)  
 优化后如情况与11)中相同, miss数 = 1024 (次)

15) 优化前: 即需要每个A[i][j]不重复访问, 故需  $128 \times 64 \times 4 = 32768 \text{ Byte} = 32 \text{ KB}$   
 优化后, 由于每次使用完元素后这个元素对后续访问无影响了, 故仅需1个块即可, 故需32 Byte即可.



15.	input				output				
	列0	列1	列2	列3	列0	列1	列2	列3	
行0	miss	hit	hit	hit	miss	miss	miss	miss	因此,一个块中存放4个int变量
行1	miss	hit	hit	hit	miss	miss	miss	miss	缓存之中一块存有2个块
行2	miss	hit	hit	hit	miss	miss	miss	miss	故在同行中,第1次miss,后续hit
行3	miss	hit	hit	hit	miss	miss	miss	miss	在跨行时,均会出现miss
行3	miss	hit	hit	hit	miss	miss	miss	miss	当时每次跨行,故全miss

16. (1)  $512 \div 16 = 32$  (个) 块, 每个块中存放4个int,  $int[0][0]$  与  $int[0][1]$  是映射到同一块 (差了  $2 \times 4 = 8$  个块)

∴ 每个块第一次访问时会发生miss, 后续不会. 命中率 =  $\frac{128-32}{128} \times 100\% = 75\%$

(2) 增加缓存总大小可增加块数量, 从而增加命中率. 而该程序中一次访问并不恰好用于访问32个块,

不能在这程序中无对于组复用的情况出现, 故增加缓存总大小不能改善程序的命中率.

(3) 增加块大小, 即增加一个块中存放的int变量的数量, 而不会减少组复, 但在该程序中

$int[0][0]$  与  $int[0][1]$  映射到同一块, 故在每次miss后, hit的int数会增加. ∴ 增加块大小能改善命中率.