No. Date

6. 用高位作索引位时，相邻地址的内存块会映射到高速缓存的同一组，此时若顺序访问某个数组，那任何时刻都只保存一个块的大小，使用效率很低；同时用中间位作标签的区而已性可能不好（可能易重复）

7. (1)可简化地址转换，可直接利用虚拟地址某些位作缓存的组索引和块内偏移，而无需额外的位操作。

(2)更好地利用缓存局部性，相邻的页在虚拟地址空间中也很可能相邻。

(3)~~保持~~有利于系统设计一致性，减少硬件开销。

8. (1) $1\times(1-3\%)+110\times3\%=4.27$ 同期

(2) ~~缺失~~命中几率 $\frac{64kB}{1GB}=\frac{2^6}{2^{30}}=\frac{1}{2^{24}}$ $1\times\frac{1}{2^{24}}+110\times(1-\frac{1}{2^{24}})\approx110$ 同期

(3) 编写程序时要注意把握读写数据的空间局部性和时间局部性，所读写数据在空间上连续性好，读取的时间连续性好，就可以较好地发挥这种局部性发挥缓存的高速特性，改善缓存性能。

(4). $1\times2+110\times(1-2)\leq105$ $109Q\geq5$ $Q\geq\frac{5}{109}$

9.

编号	地址位数	缓存大小 kB	块大小 Byte	相联度	组数量	组索引位数	标签位数	偏移位数
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

No. Date

10. (1) $(100+0.22)p_1+0.22(1-p_1) < (100+0.52)p_2+0.52(1-p_2)$

$100p_1+0.22 < 100p_2+0.52 \quad p_1-p_2 < 0.3\times10^{-2}=0.003$

(2) $0.22kp_1+0.22(1-p_1) < 0.52kp_2+0.52(1-p_2)$

$0.22(k-1)p_1+0.22 < 0.52(k-1)p_2+0.52$

$(k-1)(0.22p_1-0.52p_2) < 0.3$

11. 16块 对7个地址模16：①②③④⑤⑥⑦ 1、5、1、5、5、5、5 模8：1、5、1、5、5、5、5

模4：1、1、1、1、1、1、1 模2 1、1、1、1、1、1、1

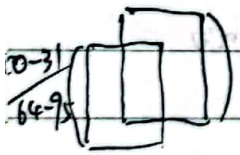
直接映射 ③④⑤⑥⑦ 会替换 5次

2路组相联 ⑤⑥⑦ 会替换 3次

4路组相联 ⑤⑥⑦ 会替换 3次

8路组相联 不会替换 0次

12. A. 2路组相联 LRU替换 按序访问&array[0]~&array[95]



一个块存4个int.

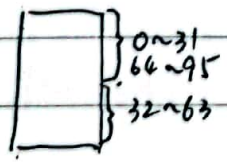
根据LRU，&array[64~95]先会占据&array[0~31]

而后&array[0~31]又会占据&array[32~63]

如此一直相互替换，导致一块总是缺一次击中3次

总缺失率75%

B：直接映射 &array[0~31]和&array[64~95]的映射长域交叠.



同A，一块总是缺1次击中3次

而&array[32~63]可多次复用，一直能击中，无替换

总缺失率 $\frac{2}{3}\times\frac{1}{4}=\frac{1}{6}=16.67\%$

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

内循环Cache利用较好.

No. Date

14. (1) 4KB/32Byte = 2^7 = 128　数组大小 64×128 个 Int，

一个块 32Byte 可容 8 个 Int，　直接映射

&A[j][0] ~ &A[j][63] 会映射到连续的 8 个 line，一行自映射到整个 Cache

~~&A[j]~~　&A[0][0] ~ &A[15][63]，&A[16][0] ~ &A[31][63] …… &A[112][0] ~ &A[127][63]

△ 那么未优化前，~~相等于~~ 内 128 次外 64 次，内循环会反复对特定的 16 个 line 反复覆盖替换（如最初为第 0、8、16 … 120、128、0、8、16 … 128 … 替换 128 次），

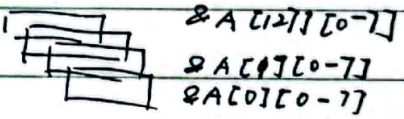
每次都是缺失，故而缺失率 100%。　次数 64×128 = 2^{13}

△ 优化后，循环会访问遍读数据，对 line 的占用变为 0、1、2、3 … 共 16 次

缺失仅会表现为一个块缺 1 次立马后面击中 7 次

缺失率 12.5%　次数 64×128×$\frac{1}{8}$ = 2^{10} = 1024 次。

(2) 全相联　4KB/32Byte = 128　128 路，FIFO 更新。



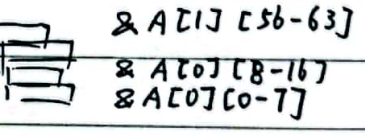
△ 未优化前　外循环一次，会将 &A[j][8k~8k+7]

j = 0、1、2 … 128 恰好填入到 128 个块中，这次外循环内的 128 个内循环全缺失

则后面 7 次外循环都恰好能击中，再一次则进行新的 128 次替换。

表现为缺失率 12.5%　次数 64×128×$\frac{1}{8}$ = 1024 次。

△ 优化后，外循环 2 次，会将 &A[t][0~63]、&A[t+1][0~63] 填入到 128 个块中，这 2 次外循环的共 128 次内循环均为缺失 1 次后立即命中 7 次



缺失率 12.5%　次数 64×128×$\frac{1}{8}$ = 1024 次。

类似 A[j][0-7] 的不连续性

128×8line×32Byte/line。

(3). 直接映射，优化前，需要把容量扩大到 ~~8~~ 8 倍　也即 32kB

优化后，理论上只要有一块就够了，每次缺失都是强制缺失，

也即 32Byte　数据充分利用。

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

16、(1) 512Byte/16Byte = 32　32/2 = 16、　1个块容4个Int型．索引为4位．模16．
可将input切为&input[0][0~63]　&input[0][64~127]　&input[1][0~63]　&input[1][64~127]
~~对每一个块~~ 每次循环 &input[0][x]、input[1][x] 都恰好为同一组的两路
每次乘和、对应块和之前的题基本同理，缺失1次而后立马击中3次　不会出现之间的抢占
因而命中率为75%

(2) 增加缓存总大小，组数会变多，这样 &input[0]… 和 &input[1] 上在缓存中
还是有两路裕度，128次循环中不会对所访问数据的连续性造成影响，
因而增加缓存总大小并不能改善命中率。

(3) 增加块大小，依旧可以"两路并存"，且每个块缺失1次后立马击中的次数会变多，
命中率会提高，~~[illegible]~~。
因而增加块大小可以改善命中率。