T6 ① 高位当索引以相邻数据高位相同，会产生没有冲突，因为这种方法可以直接利用地址，因为索引本身是就是地址的单位，比如：若每个Cache line可存两个字节，地址从0开始，则若Cache有4个line，则内存 0110的单元"11"本身就代表了其映射到了第3个line

T7 这样的话，可以直接用虚拟地址去Cache中访存，能更高效地利用Cache，从而加快数据的读取、写速度

T8

1) $\bar{t} = 110 \times 0.03 + 1 \times 0.97 = 4.27$

2) 缓存中有64KB的数据，而总数据为1GB，访存是完全随机的，则每次访存数据有 $\frac{64}{1048576}$ =0.006%的概率在Cache中，因此

$\bar{t} = 0.06\% \times 1 + 99.4\% \times 110 \approx 110$

3) 局部性原理指的是CPU访存时，都趋于访问聚集在一个较小的连续区域内的数据。

若局部性原理成立，则将访问的数据附近的一些数据也放入缓存中，则后面访存时大概率会访到缓存中的数据，例如小中可大的命中率，则访存周期会大大减少，而若没有局部性原理，则如2)，缓存的利用率会大大降低，从而访存周期大大增加

4) $\bar{p}+(1-\bar{p})\cdot 110 < 105$

32

T9

编号1：32组　5 Bit　21 Bit　6 Bit

编号2：8组　3 Bit　23 Bit　6 Bit

编号3：1组　0 Bit　26 Bit　6 Bit

编号4：256组　8 Bit　18 Bit　6 Bit

编号5：64组　6 Bit　19 Bit　7 Bit

编号6：256组　8 Bit　18 Bit　6 Bit

编号7：64组　6 Bit　20 Bit　6 Bit

编号8：32组　5 Bit　20 Bit　7 Bit

T10

1) $0.22(1-p_1)+100\cdot p_1 < 0.52(1-p_2)+100\cdot p_2$

2) $0.22(1-p_1)+2\cdot 0.22\cdot p_1 < 0.52(1-p_2)+2\cdot 0.52\cdot p_2$

T11

0001	0000	0000	0001
0001	0000	0000	0101
0001	0000	0010	0001
0001	0000	0110	0101
0001	0011	0000	0101
0010	1110	1110	0101
1111	1111	0000	0101

块内偏移为 $\log_2(100)=6$ Bit

索引 8 Bit

直接映射：4 Bit：替换 3 次

2路：3 Bit：替换 1 次

4路：2 Bit：替换 1 次

8路：1 Bit：替换 0 次

T12

1、计算稳态：

直接映射：一路，16组，一组=一块，每块中16字节，4个字，4个数组元素，且稳态时，数组的32~63个元素一直在8~16组中，因此这些元素不会miss，而剩余的，都是4个里面miss一个，命中3个

∴ $miss\_rate = \frac{2}{3} \cdot \frac{1}{4} = \frac{1}{6}$

2路并联：LRU，则每4个miss 1个命中3个，所以

$miss\_rate = \frac{1}{4}$

T13 for(int j=0; j<128; ++j){

for(int i=0; i<64; ++i)

A[i][j]++;}

T14 16~15 A[j][i]

1) 4KB，块大小为32B，也即8个word，能存放8个数组元素，且共有(4×1024)/32=128个块，且A[j][i]与A[j+1] A[i]之间相差64，即8个块

① 未优化之前：每次缺失时存入的8个数据，除了立刻会使用的那一个，剩下的7个，在被利用之前就被冲刷掉了，因此缺失次数是128×64=8192次

② 优化后：每次miss后读入的剩下7个数据会立刻被利用，∴ $miss\_rate = \frac{1}{8}$ ∴ miss=1024次

2) ①未优化前：128个块，每有128行，正好每过一个j循环数组的每行读取8个数据进去，则后7个数据在下一次j循环会利用，所以miss_rate=$\frac{1}{8}$，∴miss=1or0

②：优化后：仍是$\frac{1}{8}$ miss_rate，∴miss=1or0

3) ①优化前：最好是读入块的8个元素，除了强制缺失的一个，其余都能被hit到，则对于优化前代码这等于要求，每次遍历一次j，都不会有缓存冲突，而又因为是直接映射，这等于要求缓存能放下整个数组，即：64×128×4=32KB

②优化后：仅正确次读入之后，强制缺失之外的7个会与hit，所以一个块执行一次32B读取。

T15

output	0	1	2	3
0	miss	miss	miss	miss
1	miss	miss	miss	miss
2	miss	miss	miss	miss
3	miss	miss	miss	miss

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

T16

1) 每次更新缓时，将 input[0][i~i+3] 读入第一路的某一组，再将 input[1][i~i+3] 读到第二路的某一组，这样之后的 i+1~i+3 数据会命中，因此命中率为 $\frac{3}{4}$

2) 不会，因为每个数据只会用一次，而块大小不变，因而都是 input[n][i] | i=0, 3, 7… 的会 miss，而其余的会命中，因此命中率总是 $\frac{3}{4}$

3) 会，若将块大小更新至大于 4 个字，例如 8 个字，则就会成为每 8 个数组元素中 miss 一个，则命中率会变为 $\frac{7}{8}$