6. 中间位作索引，高位作为标签是为了充分利用地址的信息，提高缓存效率和容量

(1) 地址作中间位组索引：提高缓存命中率，因相邻的主存块在时间上往往有更高访问概率

(2) 高位作标签：可以唯一地标识每块，保证缓存一致性和正确性

(3) 如果中间位作标签，高位作索引会导致：

(1) 组内冲突增加：中间位作标签意味组索引的位数减少，导致组的数量减少，这会增加不同主存块之间的映射冲突

(2) 标签冲突增加：将高位作为组索引意味标签位数减少，导致标签空间变小，增加冲突可能性。

7. 有以下好处：

(1) 简化地址转换：可以直接使用虚拟地址的组索引和地内偏移部分进行缓存查找和匹配，而无需进行额外地址操作。这样可减少访问延迟和硬件开销

(2) 提高地址映射一致性：在这种设计中，虚拟地址在这两个系统之间的转换更加直观和一致

(3) 提高空间利用率：将缓存的组索引和地内偏移与页偏移位数相匹配可以更好地对齐内存访问，减少浪费不必要的内存空间

8. (1) $3\% \times 110 + 97\% \times 1 = 4.27$

(2) 2^{30} 数组，2^{16} 缓存

平均周期 $\Rightarrow \frac{2^{16}}{2^{30}} \times 1 + \frac{2^{30}-2^{16}}{2^{30}} \times 110 = 109.99$

(3) 当不满足空间局部性时，缓存效率大大降低

(4) $x \times 1 + (1-x) \times 110 < 105$

$\Rightarrow 110 - 109x < 105 \Rightarrow$ 即 $109x > 5 \Rightarrow x > \frac{5}{109}$ 时，即缓存命中率大于 4.6% 时才有必要收

9. (1) 组数量 $S = 4096 / 64 = 2^{12}/2^6 = 2^6$ 组 2^6 组 = 64 组

组索引位数 $s = \log_2 64 = 6$ 位

标记位数 $T = 32 - 6 - 7 = 19$ 位

偏移位数 $b = \log_2 64 = 6$ 位

(2) $S = 4096/64 = 64$

$s = \log_2 64 = 6$ 位

$T = 32 - 9 - 6 = 17$ 位

$b = \log_2 64 = 6$

⋮

	组数量	组索引位数	标记位数	偏移位数
1	64	7	19	6
2	64	9	17	6
3	64	12	14	6
4	256	8	18	6
5	128	8	17	7
6	1024	12	14	6
7	1024	14	12	6
8	512	13	12	7

10. (1) A: $p_1 \cdot (100\text{ns}) + (1-p_1) \cdot 0.22 = 99.78p_1 + 0.22$

B: $p_2 \cdot (100\text{ns}) + (1-p_2) \cdot 0.52 = 99.48p_2 + 0.52$

A<B 时 A 优于 B $\Rightarrow$ $99.78p_1 + 0.22 < 99.48p_2 + 0.52$

$\Rightarrow 99.78p_1 - 99.48p_2 < 0.3$ 时

(2) A: $p_1 \cdot k \cdot 0.22 + (1-p_1) 0.22$

B: $p_2 \cdot k \cdot 0.52 + (1-p_2) 0.52$

$A<B \Rightarrow (k-1)p_1 \cdot 0.22 + 0.22 < p_2(k-1)0.52 + 0.52$

$\Rightarrow \frac{p_1}{p_2} < \frac{0.3}{0.22} \cdot \frac{0.52}{0.22} \Rightarrow \frac{p_1}{p_2} < 0.707$

11.

转换块地址：$0x1001 = 4097$；$0x1005 = 4101$；$0x1021 = 4129$；$0x1045 = 4165$

$0x1305 = 4869$；$0x2ee5 = 12005$；$0xdd05 = 65285$

直接映射：未命中 → 命中 → 命中 → 未命中(替换) → 未命中(替换) → 未命中(替换)

直接映射：未命中 → 命中 → 命中 → 未命中(替换) → 未命中(替换) → 未命中(替换)

缓存大小：16块，64Byte

直接映射：一次写入1024B

↳ 3-1=2次 $\Rightarrow$ 未命中 命中 → 命中 → 命中 → 命中 → 未命中(替换) → 未命中(替换)

2路：8块+8块，64Byte $\Rightarrow$ 一次写入512B

↳ 4-2=2次 $\Rightarrow$ 未命中写入 → 命中 → 命中 → 命中 → 未命中 → 未命中(替换) → 未命中(替换)

4路：一次写入256B

↳ 5-4次=1次 $\Rightarrow$ 未命中写入 → 命中 → 命中 → 命中 → 未命中 → 未命中 → 未命中 → 未命中

8路：一次写入0B

$\Rightarrow$ 未命中 → 命中 → 命中 → 未命中 → 未命中 → 未命中 → 未命中

$\Rightarrow$ 8-8, 0-0 = 0次

12. 即缓存有16块，直接映射一次写入256Byte，2路组128Byte

1个 int32_t 占4字节

⇒ 直接映射写入数组64个，2路32个

直接：1~64 $\xrightarrow{替}$ 65~96，1~32 $\xrightarrow{替}$ 33~96 $\xrightarrow{替}$ 1~64 ...

2路：1~32 → 33~64 $\xrightarrow{替}$ 65~96 $\xrightarrow{替}$ 1~32 $\xrightarrow{替}$ 33~64 $\xrightarrow{替}$ ...

⇒ 直接：替换次数：$3\times\frac{100}{2}-1=149$ 次

缺失率：$\frac{149}{96\times100}=1.55\%$

⇒ 2路：替换次数：$3\times100-2=298$ 次

$\frac{298}{9600}=3.10\%$

13.

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

14. (1) 缓存有 4096/32 = 128块，假设数组A中一个数据占4字节，则：

1次写入数组1024个元素，即16行，从 A[0][0] 到 A[15][63]

优前 A[0][0] ~ A[15][63] $\xrightarrow{替}$ A[16][0] ~ A[31][63] $\xrightarrow{替}$

A[32][0] ~ A[47][63] $\xrightarrow{替}$ A[48][0] ~ A[63][63] $\xrightarrow{替}$ ...

一次 i 循环中，替换了8次 ⇒ 共替换 8·64次 = 512次，缺失512次

优后：16次 i 循环 替换1次 ⇒ 共替换8次，缺失8次

(2) 优前：A[0][0] → A[1][0] → ... 共替换 48×128次 = 6144次

A[0][1] → A[1][1] → ...

→ A[0][15] → A[1][15] → ... ⇒ 缺失6144次

替 A[0][16] $\xrightarrow{替}$ A[1][16] → ...

替 ... ...

优化后：同优化前，共6144次

(3) 优化前：32kB，优化后：256B

5.

input

m	h	h	h
m	h	h	h
m	h	h	h
m	h	h	h

output

m	m	m	m
m	m	m	m
m	m	m	m
m	m	m	m

16. (1) ~~512/4=128 ⇒ 一次写入~~

512/(2·4) = 64 ⇒ 一次写入64个元素

写 input[0][0] ~ input[0][63] 写 input[1][0] ~ input[1][63]

替 input[0][64] ~ input[0][127] 替 input[1][64] ~ input[1][127]

~~共发生替换~~ 命中率为 $\frac{128\times2-4\times4}{128\times2} = 93.75\%$

(2) ~~可以，input[0][64]~inp~~

没影响。因块大小影响写入和替换

(3) 可以，如块大小增到32字节，命中率会提高至 $\frac{128\times2-4\times2}{128\times2} = 96.88\%$