6 试简要分析为什么缓存一般使用地址的中间位作为组素引、高位作为标签，而不是用高位作为组索引，中间位作为标签?
使用地址的中间位作为组索引、高位作为标签可以在空间利用效率、访问效率和热点数据均衡方面取得更好的平衡，提高缓存性能和访问效率。

7 一些真实的缓存系统在设计时会有意让地址的组索引和块内偏移的总位数与虚拟内存系统的页偏移位数相同，试分析这样做有什么好处。
简化地址转换：简化了地址转换的逻辑和硬件设计，并且可以减少转换过程中的延迟。
硬件实现的便利性：简化硬件电路的结构，减少芯片面积和功耗，并提高设计的可靠性和可维护性。
提高缓存利用率：更好地利用缓存，确保每个缓存块对应一个完整的页，减少了内存碎片和浪费。
简化地址映射：如果缓存的组索引和块内偏移的总位数与页偏移位数相同，那么地址映射过程可以更加简单和直接。

8
1.存储系统平均访问延时计算：
存储系统平均访问延时 = 缓存命中延时 + 缓存缺失延时
= 1 周期 + (缓存缺失率 × 缓存缺失延时)
= 1 周期 + (0.03 × 110 周期)
= 1 周期 + 3.3 周期
= 4.3 周期

2.存储系统平均访问延时计算：
存储系统平均访问延时 = 缓存缺失延时 = 110 周期

3.局部性原理如何影响处理器的访存性能：
局部性原理指出，程序在某一时间段内对存储系统的访问往往会集中在一小部分地址范围内。这包括时间局部性和空间局部性。通过利用局部性原理，缓存可以提供更快的访问速度，减少对主存的访问延迟。这样可以提高处理器的访存性能，并加速程序的执行。

4.当使用 L1 缓存时，平均缓存命中率必须高于直接访问主存时的访问延时。禁用缓存时的访问延时为 105 周期。因此，程序的平均缓存命中率必须高于禁用缓存的访问延时，即高于 105 周期，才能让存储系统在使用 L1 缓存时获得性能收益。

9
编号 1 2 3 4 5 6 7 8
地址位数 32 32 32 32 32 32 32 32
缓存大小 4 4 4 16 16 64 64 64
块大小 64 64 64 64 128 64 64 128
相联度 2 8 全相联 1 2 4 16 16
组数量 2 8 1 16 8 16 16 4
组索引位数 1 3 0 4 3 4 4 2

标签位数 29 29 32 28 29 28 28 30
偏移位数 6 6 6 6 7 6 6 7

10
平均内存访问时间 = 命中延时 + 缺失率 × 缺失代价
A:
平均内存访问时间(A) = 0.22ns + p1 × 100ns
B:
平均内存访问时间(B) = 0.52ns + p2 × 100ns

要使系统 A 的平均内存访问时间优于系统 B，即(A) < (B)，则有:
0.22ns + p1 × 100ns < 0.52ns + p2 × 100ns
p1 < p2 + 0.003ns

因此，当缓存系统 A 的缺失率 p1 小于缓存系统 B 的缺失率 p2 加上 0.003 时，系统 A 的平均内存访问时间优于系统 B。

如果缓存的缺失代价分别是各系统命中延时的 k 倍，则系统 A 的平均内存访问时间优于系统 B 的条件是:
平均内存访问时间 = 命中延时 + 缺失率 × 缺失代价
A:
平均内存访问时间(A) = 0.22ns + p1 × (0.22ns × k)
B:
平均内存访问时间(B) = 0.52ns + p2 × (0.52ns × k)

要使系统 A 的平均内存访问时间优于系统 B，即(A) < (B)，则有:
0.22ns + p1 × (0.22ns × k) < 0.52ns + p2 × (0.52ns × k)
p1 < p2 + 0.3/k

因此，当缓存系统 A 的缺失率 p1 小于缓存系统 B 的缺失率 p2 加上 0.3 除以 k 时，系统 A 的平均内存访问时间优于系统 B。

11
1 直接映射缓存:
给定的块地址序列为:
0x1001、0x1005、0x1021、0x1045、0x1305、0x2ee5、0x1105

0x1001：0
0x1005：1
0x1021：0
0x1045：1
0x1305：1
0x2ee5：0
0x1105：1

因此，直接映射缓存的块替换次数为 4 次。

2 路组相联缓存：
给定的块地址序列为：
0x1001、0x1005、0x1021、0x1045、0x1305、0x2ee5、0x1105

替换次数：

0x1001：0
0x1005：0
0x1021：0
0x1045：1
0x1305：1
0x2ee5：0

12
使用缓存 A
缺失率：
总共访问的元素个数为 N=100。
缓存 A 的总容量为 256 字节，块大小为 16 字节，共有 16 个块。
每个组内有 2 个缓存行，共有 8 个组。
使用缓存 B
缺失率
总共访问的元素个数为 N=100。
缓存 B 的总容量为 256 字节，块大小为 16 字节，共有 16 个块。
每个块地址只能映射到唯一的缓存行。

13

```
for (int j = 0; j < 128; ++j) {
    for (int i = 0; i < 64; ++i) {
        AL[i] = AG[l][i] + 1;
    }
}
```

14
1 于 4KB 大小的直接映射缓存，块大小为 32 字节。
优化前的代码缓存缺失次数：缓存缺失次数为 64 次。
优化后的代码缓存缺失次数：缓存缺失次数为 0 次。

2 对于 4KB 大小的全相联缓存，块大小为 32 字节，使用 FIFO 替换策略。
优化前的代码缓存缺失次数：缓存缺失次数为 64 次。
优化后的代码缓存缺失次数：缓存缺失次数为 0 次。

3 优化前的代码：总共有 64 个元素，所以需要至少 64 个缓存行，即 64 * 32 字节 = 2048

字节的缓存容量。
优化后的代码：循环次数为 64 次，每次访问都命中缓存，所以不需要任何缓存容量。

15

	input 数组列 0	input 数组列 1	input 数组列 2	input 数组列 3	output 数组列 0	output 数组列 1	output 数组列 2	output 数组列 3
行 0	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Miss
行 1	Hit	Miss	Hit	Miss	Hit	Miss	Hit	Miss
行 2	Hit	Miss	Hit	Miss	Hit	Miss	Hit	Miss
行 3	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Miss

16

1 缓存命中率为 0%。

2 增加缓存的总大小可以改善命中率。

3 增加缓存的块大小可能对命中率改善有限。