5. (1) 2^{64} 个可能的虚拟地址.

则 $8\times\frac{2^{64}}{4\times1024}=2^{55}$ Byte

需要 2^{55} 个字节.

(2) 若限制仅使用48位虚拟地址空间.

$8\times\frac{2^{48}}{4\times1024}=2^{39}$ Byte

需要 2^{39} 个字节

(3) 1. 减小页表的大小，在大型内存系统中，页表的大小可能非常庞大，多级页表可以通过层次结构来组织页表，将大的页表拆分成多个小的页表，

2. 惰性加载，采用分级加载，程序运行中，当页面真正被访问时才会加载相应的页表项，意味着只有活动的页面需要加载到内存中，节省存储开销，加速访问速度

3. 局部性原理，程序执行过程中，可能只会访问到一部分连续的虚拟页面，通过多级页表，使得常用的页面在更高级的页表层次中，而不常用的页面则在较低级的页表层次中，可减少整体页表大小，提高内存访问效率.

6. 1. 采用中间位作为组索引可以提高缓存的空间利用率，在缓存中，每个组包含多个缓存行，每个缓存行储存一个数据块，使用中间位做组索引可以使不同的地址映射到不同的组，增加了组的数量.

2. 可以利用空间局部性，相邻的内存地址通常会映射到相同的组，这样可提高缓存的命中率.

3. 地址的高位通常具有较好的随机性和均匀性分布，可以保证缓存之间的均衡负载，若用其作组索引，可能引发一些组经常被访问，造成缓存冲

4. 采用高位作为标记可提供更多的位数来区分不同的数据块，增加缓存容量

7. 1. 简化地址转换过程. 可直接使用相同的位数进行操作. 减少了地址转换之复杂性和开销

2. 提高缓存系统的访问效率. 可用简单之位运算和掩码操作来提取组索引和块内偏移. 加快解码速度. 减少访问延时.

3. 提高缓存之利用率. 数据块的组索引和块内偏移可以直接映射到缓存之组索引和块内偏移, 无需进行额外截断或扩展操作.

4. 简化硬件设计. 硬件可以直接使用相同之位数进行地址解码和计算. 简化设计.

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

2) 若*完全随机访问数据, 缓存之命中率极低. 访问一次延时接近 110 个周期.

访问次数. $\frac{1GB}{64kB} = 2^{14}$ 次.

则访问这个1GB之数组共延时 $(2^{14} \times 110)$ 周期.

3) 影响缓存命中率以及缓存缺失次数, 可将*经常访问的数据放在高速缓中.

将连续之数据块预加载到缓存中.

4). ① 缓存命中率

$$1 \cdot x + (1-x) \times 110 < 105 \Rightarrow x > \frac{5}{109} \approx 4.6\%$$

缓存命中率高于 4.6%, 性能开始提升.

	组数量	组索引位数	标签位数	偏移位数
9. 1	32	5	21	6
2	8	3	23	6
3	1	0	26	6
4	64	6	20	6
5	64	6	19	7
6	16	4	22	6
7	4	2	24	6
8	8	3	22	7

10.

1) 系统A延时: $(1-P_1)\cdot 0.22 + P_1\cdot 100$

系统B延时: $(1-P_2)\cdot 0.52 + P_2\cdot 100$.

当满足 $99.78P_1 < 0.3 + 99.48P_2$ 时. 系统A优于B.

2) A延时: $(1-P_1)\cdot 0.22 + P_1\cdot k\cdot 0.22$ < B延时: $(1-P_2)\cdot 0.52 + P_2\cdot k\cdot 0.52$

满足: $0.22 + 0.22P_1(k-1) < 0.52 + 0.52P_2(k-1)$

亦即: $\frac{1+P_1(k-1)}{1+P_2(k-1)} < \frac{52}{22}$ 时. A 优于B.

4097 4101 4129 4165

12. ~~缓存A 2路组相联~~ 共16个块

11. ① 直接映射：0x1001
0x1005
0x1021 (替换0x1001)
0x1045 (替换0x1005)
0x1305 (替换0x1045)
0x2ee5 (替换0x1305)
0xff05 (替换0x2ee5)

共替换5次

② 2路组相联 0x1001 0x1021
0x1005 0x1045
0x1305 (替换) 0x2ee5 (替)
0xff05 (替)

共替换3次

③ 4路组相联，共替换一次

0x1001 0x1021 0x1005 0x1045 0x1305 0x2ee5
0xff05 (替)

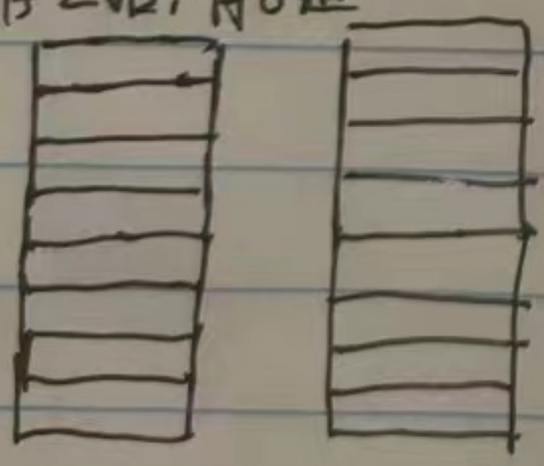
④ 8路组相联，不发生替换

0x1001 0x1021 0x1005 0x1045 0x1305 0x2ee5 0xff05

12. ① 对于2路组相联

内存共 $\frac{96\times4}{16}=24$ 个块.

缓存2路，有8组



数组按顺序访问，每块16个Byte

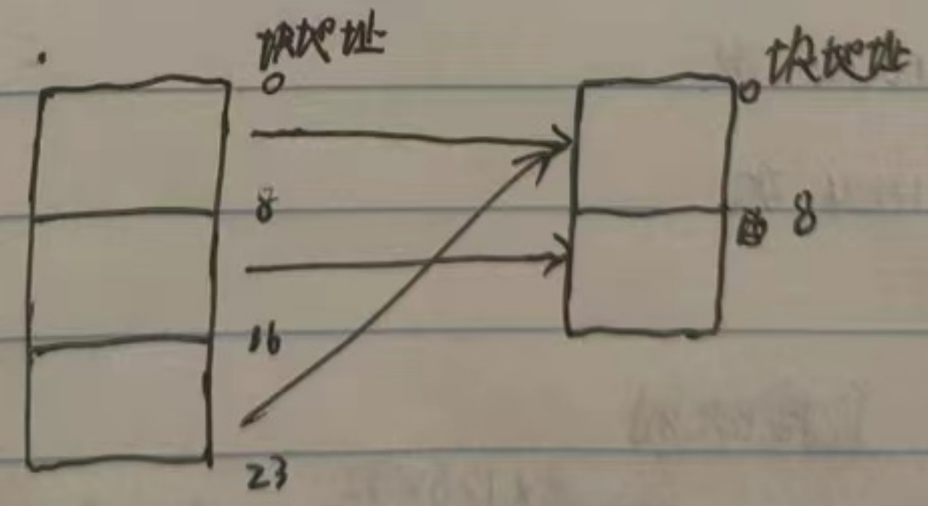
可以放4个数据.

第一个数据miss、后3个hit.

则缺失率 $\frac{1}{4}=25\%$

② 对于直接映射

内存24个块 Cache 16个块



第一次循环缺失24次

后面循环不缺失8次

16

外层循环i>1后，内存中8~15块始终命中.

则 miss rate = $\frac{24+16\times99}{96\times100}$

$=16.75\%$

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

14. 1) 优化前　缺失次数 128×64 = 8192次

$\frac{4\times1024}{32}$ = 128个块

每个块 $\frac{32}{4}$ = 8个数据

j\i	0	1	2	3	4	5	6 …	63
0	a_{01}	a_{02}						a_{063}
1	a_{121}	a_{122}						
2								
3								
4								
…								
127	a_{1271}	a_{1272}						a_{1276}

优化后：i=8k 时，缺失

缺失次数 $\frac{64}{8}\times128$ = 1024次

2) 全相联

优化前：$128\times\frac{64}{8}$ = 1024次

优化后：$128\times\frac{64}{8}$ = 1024次

3). 对于32字节一块，直接映射

优化前，需要 8×128个块，总容量 $\frac{8\times128\times32}{1024}$ = 32KB

优化后：至少需要1个块，总容量：32×1 = 32个字节

15.

input 数组

	列0	列1	列2	列3
行0	miss	hit	hit	hit
行1	hit	hit	hit	hit
行2	miss	hit ~~miss~~	hit	hit
行3	hit	hit	hit	hit

output 数组

	列0	列1	列2	列3
	miss	hit	miss	hit
	miss	hit	miss	hit
	miss	hit	miss	hit
	miss	hit	miss	hit

16. 1) 块的数量: $\frac{512}{16} = 32$ 个块. 每个块放 $\frac{16}{4} = 4$ 个数据

则有 16 组, 2 路. 则每组每块第一个数据缺失, 后3个命中

则命中率 $\frac{3}{4} = 75\%$

2) 增加cache存总大小不能改善命中率. 仍是每个块中第一个缺失, 后3个命中, 命中率仍为 75%

3) 可以. 增加块大小, 块中存放的数据变多 (设为 $N>4$)

则第一个数缺失, 剩下命中.

命中率 $1-\frac{1}{N} > 1-\frac{1}{4} = 75\%$. 提高.