第四章. 计算机存储系统.

1、因为不同的存储层级对于容量、成本、速度的需求各不相同，为了达到各自所适合的最"平衡"而实施分级管理.

2、页面过小可以充分利用内存，但页表会很长（占用内存）；反之页面过大对内存的利用率就很低.

3-1)：DAGUXRWV，其中 D(dirty)指该页是否被写过，A(access)表示是否被访问，即"读/写/执行过"，G表示是否为全局映射(global)，U表示在用户模式下，该页是否可用，V表示整个PTE页表项有效性（V=0则其它位信息全部无效）. RWX分别指页的 read, write, execute 三种权限。

2) 有些进程执行的先后可能会对页表进行覆盖/改错，影响程序的功能.

3) RWX位全部为0表达了"指向下一级页表的物理页号".

4-1)：所指范围不同，页表条目中的XWR针对该页，PMP的XWR更精确指定了小块物理内存区的读写执行权限

2) A字段决定了PMP entry的控制范围，A=0时不匹配地址；L字段表示"lock"，即当L=1时，machine mode也要遵从设定的权限，这也是PMP中还要设定RWX的意义所在.

5-1) 页内偏移有 $\log_2(\frac{4KB}{1B})$= 12位，还有 2^{32} 可用于表示页号，即 一个页表最大需要 $2^{32}\times 8B = 2^{35}B$，那么存储页表需要 $2^{35}/2^{12} = 2^{23}$ 个框，2^{35} B 的空间.

2) $2^{48-12}\times 8B = 2^{39}B$.

3) 因为多级页表"多次分组"，让页表不必全部常驻于内存，相当于多取了几次"$\log_2$".

6. 因为高位作标签使得连续的物理地址一般先变组索引，再改变标签的高位，即中间位在物理地址中改变的频率远高于高位，更加地区分了顺序存储的相邻数据，~~[illegible]~~ 适配了"空间局部性".

7. 块内偏移 + index = cache 地址，cache 地址与页内偏移同位，可以让缓存装配时的"映射算法"相对简单一点，在硬件上简化一些数据传送与运算的电路结构.

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

2) $\frac{64}{(1024)^2} \times 1 + (1 - \frac{2^6}{2^{20}}) \times 110 \approx 110$ (个) 周期.

3) ~~[illegible]~~ 重复访问同一位置(如循环算法)与访问相邻位置都利用预测，因此容易被命中，故此的局部性可以减少对访存的依赖 4) $r \times 1 + (1-r) \times 110 < 105$; $r > \frac{5}{109} \approx 4.59\%$.

9.

	地址位	缓存大小	块大小	相联度	组数	组索引位数	tag 位	偏移位
1	32 bit	4kB	64B	2	32	5	21	6
2		4kB	64B	8	8	3	23	6
3		4KB	64B	all	1	0	25	6
4		16kB	64B	1	256	8	18	6
5		16kB	128B	2	64	6	19	7
6		64kB	64B	4	256	8	18	6
7		64kB	64B	16	64	6	20	6
8		64kB	128B	16	32	5	20	7

10. 1) $0.22 + 100p_1 < 0.52 + 100p_2 \Rightarrow (p_1 - p_2) < 0.003$

2) $0.22 + 0.22kp_1 < 0.52 + 0.52kp_2 \Rightarrow 0.22p_1 - 0.52p_2 < \frac{0.3}{k}$

11. 16个块，① 直接映射，16组，占1个16进制位(4个2进制位)，7个索引位占为 0001, 0101, 0001, 0101, 0101, 0101, 0101 替换5次.

② 2路，3个2进制位，每个有2个位置送替换3次

③ 4路，2位，7个全是一样，每个有4个位置，替换3次.

④ 8个位置，0次替换. 8路.

12. 块大小 16 Byte，一次装入4个整数。A缓存的一个块有两种可能的位置；B的一个缓存一个位置。

A策略：[4][][][][][][][] 8个[]

缺失一次，装配，命中三次（前32个数）；中32个数也一样。

后32个数就上前半段缓存前32个，故也是75%命中；再如此往复，命中率为75%。

B策略：

前32个和后32个交替存入

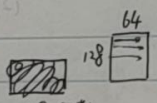
中32个装入后无竞争，之后一直命中

故命中率为 $\frac{2}{3}\times 75\% + \frac{1}{3}\times\left(\frac{99\times 1}{100}+\frac{1\times 75}{100}\right)$

$= 50\% + 33.25\%$

$= 83.25\%$.

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



14. 1) 一个块32个 Byte 有8个整数，4KB共有128块，直接映射，数组大小为 64×128 个数；

优化前在读数时均无法命中，hit rate 50%；优化后8个缺1次，为 $\frac{87.5+100\%}{2\times 100}$ = 93.75%；

而访问共 2×64×128 次；即，优化之前缺失 64×128 = 8192次；优化后缺失 1024 次；

2) 全相联映射不限制块的放置位置，优化后依旧缺失 1024 次；而优化前也能取到数，即缺失64次后命中 3×64 次。

所以都是缺失 1024 次

3). 想要~~全部~~，优化后的代码仅需32B的块装下即可；优化前要让二维装配生效，则要

不出现非必要 8×128×32B = 32KB 的缓存大小。

15.

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

2块，1块可以有4个整数，input 大小为 $\frac{4\times 4}{\cancel{100}}\times 4 = \frac{64}{1}$ Byte；正好是 input 存完到 ~~output~~ output。

由于它是 32 Byte 的缓存 & 直接映射，input [i][j] 和 output [i][j] 映射同一缓存位（是64 Byte）。

16 1) 512 Byte，16 Byte/块 → 4个数一块，索引位 mod 256（数的顺序 mod 64）。

一共32个块，128个数，64个为一组，那么：0~63个与 ~~64~~ 128~191 放在 0~64 (1) 与 cache 0~64(2)。

即前半段命中 75%；64~127个覆盖 0~63个；192~255个数覆盖 128~191个，没有任何利用

命中率为 75%。

2) 可以，至少它能大到不产生地址竞争 3) 可以，如果开到 32KB 一块，命中开到 87.5%。