(3) 若XWR全为0，则其含义为指向页表下一级的指针。

4. (1) 启用分页后，访问虚拟内存的指令可能会导致多次物理内存访问，包括对页表的隐式引用，PMP检查查适用于所有这些访问。最底层的安全保护机制。

(2) L：表项的Lock使能位

A：表项的地址匹配模式

5. (1) 页大小为4KB，则页内索引位数为12位

则虚拟页号占 64-12 = 52位，即 2^{52} 个页表项

而每个页表项使用8个字节空间

共 $2^{52} \times 8 = 2^{55}$ 个字节

即 32 PB.

(2) 虚拟页号减少为 $48 - 12 = 36$ 位 即 2^{36} 个页表项

共 $2^{36} \times 8 = 2^{39}$ 个字节

即 512 GB.

(3) 因为多级页表在进程占用内存空间较小时，可对应地减少页表数目。也就是说多级页表的页表存储是灵活可变的，而单级页表的页表存储不可变，只能取最大值。

6. 因为组索引是块地址对缓存中组数取余，本来就是块地址的较低位。当块地址从最低位开始变化时，组索引会随之变化。标签则为块地址减去最低几位。

7. 组索引与块内偏移的总位数与虚拟内存的页内偏移位数相同时，一个缓存的大小刚好是一页。

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

(2) 命中率 $R = \frac{64kB}{1GB} = \frac{2^6 kB}{2^{20} kB} = \frac{1}{2^{14}}$

平均访问延时 $T = 1 \times R + 110 \times (1-R) \approx 109.993$ 周期

(3) 第(2)问中程序持续地访问一个数组中的随机位置，这正体现了空间局部性，而由于缓存小而数组大，故系统的平均访问延时长。

(4) $T = 1 \times R + 110 \times (1-R) = 110 - 109R$

令 $T < 105$ 得: $R > \frac{5}{109} \approx 0.046$

∴ R需要满足 $R > 0.046$ 时，使用 L1 缓存才能获得性能收益

9.

编号	地址位数	缓存大小 (KB)	块大小 (Byte)	相联度	组数量	组索引位数	标签位数	偏移位数
1	32	4	64	2	32	5	21	6
2	32	4	64	8	8	3	23	6
3	32	4	64	~~64 全相联~~	~~全相联~~ 1	0	26	6
4	32	16	64	1	256	8	18	6
5	32	16	128	2	64	6	~~20~~ 19	~~6~~ 7
6	32	64	64	~~64~~	~~64~~ 256	~~6~~ 8	~~20~~ 18	6
7	32	64	64	16	64	6	20	6
8	32	64	128	16	32	5	20	7

10. (1) $T_A = 0.22 \times (1-p1) + (0.22+100) \cdot p1$ (ns)

$T_B = 0.52 \times (1-p2) + (0.52+100) \cdot p2$ (ns)

令 $T_A < T_B$ 得: $p1 < p2 + 0.003$

(2) $T_A = 0.22 \times (1-p1) + (0.22+0.22k) \cdot p1$ (ns)

$T_B = 0.52 \times (1-p2) + (0.52+0.52k) \cdot p2$ (ns)

令 $T_A < T_B$ 得: $p1 < \frac{26}{11k}(kp2+1) - \frac{1}{k}$

11. 直接映射: 0x1001 与 0x1021 缓存冲突

0x1005 与 0x1045、0x1305、0x2ee5、0xff05 缓存冲突

故缓存发生块替换的次数为 5.

2路组相联：缓存发生块替换的次数为 2次。

4路组相联：块替换 1次.

8路组相联：块替换 0次。

12. 缓存中的块数目为 $\frac{256}{16} = 16$ 个，每个块可以缓存 4个数组项.

对于缓存 A：有 8个组

缺失率为 $\frac{96/4}{96} = 25\%$

对于缓存 B：缺失率为 $\frac{96/4}{96} = 25\%$

13.
```
for (int j=0; j<128; ++j) {
    for (int i=0; i<64; ++i) {
        A[j][i] = A[j][i]+1; }
}
```

14. (1) 每个块存 8个数组项，共 128个块.

~~优化前：缺失 $64\times128\div8 = 1024$ 次~~

优化后：缺失 $64\times128\div8 = 1024$ 次.

~~(2)~~ 优化前：缓存中只有 $128/8 = 16$ 个块被利用.

故缺失 $64\times128 = 8192$ 次.

(2) 优化前后均缺失 $64\times128\div8 = 1024$ 次.

(3) 优化前：需要 $128\times64\div8 = 1024$ 个块.

即 $1024\times32\,B = 32\,KB$ 的缓存容量

优化后：需要 128个块.

即 128× 32B = 4KB 的缓存容量

15.

	input数组				output数组			
	列0	列1	列2	列3	列0	列1	列2	列3
行0	miss	hit	hit	hit	miss	m	m	m
行1	miss	hit	hit	hit	miss	m	m	m
行2	m	h	h	h	m	m	m	m
行3	m	h	h	h	m	m	m	m