6.

使用中间位作为组索引可以提高速度，中间位和块内偏移很接近，当页较大时，页内偏移可包含索引位和块内偏移，从而不必等待页表翻译便可进行索引，然后与页表翻译出的高位的标签匹配。但是如果索引在高位，这种情况不会出现，即处理器必须等页表翻译才能去索引缓存，~~两个访存~~需要两个完整的访存周期，速度慢。

7. 这样做可以提高处理器的运行速度。

~~虚拟地址~~ 内存是按页进行管理的，页的虚拟地址经页表翻译可得到物理地址，而页偏移部分在翻译前后是不变的。设计地址的组索引和块内偏移总数与页偏移位数相同，则不必等待页表翻译出物理地址便可直接利用虚拟地址的页偏移部分进行缓存索引，然后将索引的缓存块的标签与页表翻译之后的标签进行匹配即可，总体来说可以减少总的访存时间，因为进程无需等待页表翻译，减少了这部分时间。

8.

(1) 命中率：97%　缺失率：3%

$1 \times 97\% + 110 \times 3\% = 4.27$ ∴平均访问延时为4.27个周期

(2) 1GB = 1024KB　因为程序完全随机访问数据

又 $\frac{64KB}{1024KB} = 6.25\%$ ∴缓存命中率为6.25%

∴平均访问延时为：$1 \times 6.25\% + 110 \times 93.75\% = 103.1875$个周期

(3) 缓存的基本组织单位时块，一般缓存一次性把内存中一整块数据放入缓存，块中存放相邻地址的数据。如果访存遵循局部性原理，则不会因为块频繁的换入换出导致缺失率的提高。~~存储容量可保证访存时~~ (2)中结果，由于取数据完全随机而缓存的大小又较小，则会产生~~如下问题~~缓存块频繁换入换出的问题，一定程度上使得缺失率提高，影响性能。

(4) 设命中率为 x

则 平均访问延时满足 $x + 110(1-x) < 105$ 时 可获性能收

$\Rightarrow x > \frac{5}{109} \approx 0.04587 \approx 4.59\%$

即命中率要大于 4.59%

9.

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

10.

(1) A：平均访问时间为：$t + 0.22(1-P_1) + 100P_1 = t + 0.22 + 99.78P_1$

B：平均访问时间为：$t + 0.52(1-P_2) + 100P_2 = t + 0.52 + 99.48P_2$

$t + 0.22 + 99.78P_1 < t + 0.52 + 99.48P_2$

∴需满足：$99.78P_1 - 99.48P_2 < 0.3$

(2) A：$t + 0.22(1-P_1) + 0.22kP_1 = t + 0.22 + 0.22(k-1)P_1$

B：$t + 0.52(1-P_2) + 0.52kP_2 = t + 0.52 + 0.52(k-1)P_2$

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

∴需满足：$0.22P_1 - 0.52P_2 < \frac{0.3}{k-1}$

11、先将地址换为二进制地址

0x1001 → 0001 0000 0000 0001

0x1005 → 0001 0000 0000 0101

0x1021 → 0001 0000 0010 0001

0x1045 → 0001 0000 0100 0101

0x1305 → 0001 0011 0000 0101

0x2ee5 → 0010 1110 1110 0101

0xff05 → 1111 1111 0000 0101

(1)直接映射时，因为有16个块，需要用4位进行索引。

0x1001不会发生块替换直接放入0001位置。

0x1005也不会替换，直接放入0101位置

0x1021也要放到0001位置，但与0x1001的标签不同发生替换。

同理0x1045替换、0x1305 0x2ee5 0xff05都替换。

共替换5次

(2) 2路组相联时，有16个块，共有8个组，需要用3位索引，且有2路。

与上述过程类似，共发生替换3次

(3) 4路组相联时有4个组用2位索引

共发生替换3次。

(4) 8路组相联时有2个组用1位索引

共发生替换0次

12. 块大小16Byte 则块内偏移4位

总容量256Byte 则有16个块.

数组是32位的，故一个数据为4Byte. 小一个块可放4个数

缓存A：2路组相联. 小有8个组 由3位物理地址进行索引

缓存中总共可以放 16×4=64个数

一次循环不有96个数需要访存，根据缓存的写回写分配

当一个数发生写缺失，缓存中

会发生块替换，新块中有4个数据，都是与当前数据相邻的，

所以后续3个数据不会缺失.

小 第一次循环缺失16+8次=24次.

第二次循环不缺失也是24次 之后也都是24次

小 缺失率 $\frac{24}{96}=25\%$

缓存B：直接映射，有16个组 由4位物理地址索引

此时缺失率仍然为25%.

即使用2路组相联并没有降低缺失率

13.

代码功能：对所有数组数据做 +1 处理

若忽略循环部件有如下优化：

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

直接映射 }

14. 块大小 32Byte 缓存 4KB

共有 128 个块 而 int 一个数据占 32Bit = 4Byte

一个块可放 32/4 = 8 个数据

缓存中可放 128×8 个数据

优化前：第一次循环缺失 128 次.

第二次循环不缺失，第 3、4、5、6、7、8 次都不缺失.

后面 8 次循环也是第一次缺失 128 次 后 7 次都不缺失.

共缺失：$128 \times \frac{64}{8} = 128 \times 8 = 1024$ 次.

优化后：第一次循环不缺失 16 次，使用 16 个块

总共 128 个块 8 次循环不足填满，每次都缺失 16 次

共 64 次循环 共缺失 $16 \times 8 \times 8 = 1024$ 次.

(2) 全相联. 同样有 128 个块. 一个块放 8 个数据.

优化前：FIFO

前 8 次循环 缺失 128 次. 后面也是如此

缺失次数仍为 1024 次

优化后：缺失率 $\frac{1}{8}$ 共缺失 $\frac{1}{8} \times 128 \times 64 = 1024$ 次.

(3) 优化前：必须有 128 个块，所以 4KB.

优化后：

只需一个块即可所以 32B

15.

input

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

output.

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

16.(1) 共有 $\frac{512}{16}=32$ 个块.

一个块可以有 $\frac{16}{4}=4$ 个数据. 仅第一个缺失后面全中

∴命中率: $\frac{3}{4}=75\%$

(2) 不可以.

无论怎么增加缓存大小, 对于4个数据来说总是第一个缺失, 后面的全命中 即命中率恒为 75%.

(3) 可以.

根据以前分析. 命中率为 $\frac{N-1}{N}$ N为块中数据个数

增大块大小 则块中可存放更多数据 即N个 那么

命中率 $=1-\frac{1}{N}$ 个