5. (1) 页大小4KB → 12位页内偏移 PTE为8B，则64位地址中虚拟页号占 64-12 = 52位

则对单级页表为 $2^{52} \times 8B = 2^{55}B = 2^{15}TB$ 巨大

(2) 使用48位虚拟地址：页号：48-12 = 36位 $2^{36} \times 8B = 2^{39}B = 512GB$

(3) 多级页表可按照进程的实际大小去分配已~~有空间~~相应的页表，多级页表中后面的每级的叶分支并不一定都会对应分配到内存，支持稀疏的地址空间，最终使页表分配到内存与进程的实际内存使用量成正比

要看前一级对应的页表项是否有效

6. 缓存地址具体怎么分还要看它是怎么与实际内存结合的：①缓存地址高位为块地址，低位为块内偏移。~~[illegible]~~而在内存中若按一个一个块来看的话块地址是连续的，无论是采用直接相映射还是组相联，若是模从的形式，那么块地址的低位就会作为索取代表索引，而高位才作为可区分量才能成为tag ②~~[illegible]~~在缓存与页TLB结合使用时，采用tag高位，index中间可实现index索引与tag的TLB转换并行执行，加快速度

7. 由于页内偏移直接是正确的，不需要再经TLB查表转换。所以在此时，译码出来的虚拟地址中tag部分由于在页号位置，需要经表TLB转换，而index在页内偏移中，直接可以选择出对应cache的多路索引，这样一来cache多路索引与tag地址转换可并行加速。如果 index + offset_block > offset_page，那么index有一部分必须经过地址页表转换才能变为正确的物理地址，无法在tag转换时进行cache的索引；如果 index + offset_block < offset_page，可能映射的数字没达到最高（很贵）一定cache的组相联路数）

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

(2) $1GB = 2^{10}MB = 2^{20}KB = 2^{14}\cdot 64KB$ 则命中率：$\frac{1}{2^{14}}$

延时：$1\times\frac{1}{2^{14}} + 110\times(1-\frac{1}{2^{14}}) = 110$ 周期

(3) 当运行的程序具有较好的空间、时间局部性时，cache中的内容可以被有预判的放入，从而大大减少直接访问内存的次数，访存性能提升；而程序没有较好局部性的，cache命中率很低，访存性能很难有效提升

(4) $1\cdot 2 + 110(1-2) = 105 \Rightarrow 2 = 4.59\%$ 平均缓存命中率至少有4.59%

9.

	组数量	组索引位数Bit	标志位数Bit	偏移位数Bit
1	32	5	21	6
2	8	3	23	6
3	1	0	26	6
4	256	8	18	6
5	64	6	19	7
6	256	8	18	6
7	64	6	20	6
8	32	5	20	7

10.11) A: 8KB直接映射 L1 cache $\overline{t_1} = 0.22 + p_1 \cdot 100$

B: 64KB四路组相联 L1 cache $\overline{t_2} = 0.52 + p_2 \cdot 100$

A优于B $\Rightarrow \overline{t_1} < \overline{t_2} \Leftrightarrow 0.22 + p_1 \cdot 100 < 0.52 + p_2 \cdot 100 \Leftrightarrow p_1 - p_2 < \frac{0.3}{100} = 0.3\%$

12) A: $0.22 + 0.22 \cdot k \cdot p_1$ B: $0.52 + 0.52 k \cdot p_2$

$\overline{t_1} < \overline{t_2} \Leftrightarrow 0.22 + 0.22 k p_1 < 0.52 + 0.52 k p_2 \Leftrightarrow 0.22 p_1 - 0.52 p_2 < \frac{0.3}{k}$

11. 16块 每块: 64B 块地址: 0x1001. 0x1005. 0x1021. 0x1045. 0x1305. 0x2ee5. 0xff05

直接映射: 16块 以用看尾位 1 5 1 5 5 5 5 5次

2路组相联: 8组 尾位 1 5 1 5 5 5 3次

4路组相联: 4组 模4尾位 1 1 1 1 1 1 1 3次

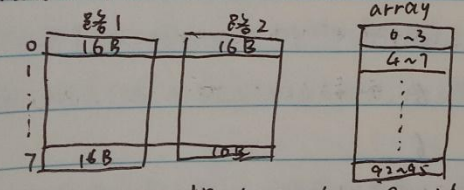
8路组相联: 2组 模2尾位 1 1 1 1 1 1 1 0次

12. 块大小: 16B 共16块 A: 2路组相联 8组 B: 直接映射 16组 LRU

计算cache缺失率 初始为空 问: 寄存器 array起始于地址0

N=100 array为96个 1个元素为4B 数组的总容量为 4B×96, 1个块16B可容纳4个数组元素 将数组元素每4个看成1组, 共24组, 每组存的元素标号 0~3. 4~7, ... 92~95

缓存A情况: 共8组 内存起始地址为0 所以 24模8分配2 共进行N=100次, 计算96×100次



cycle 1时: 缓存为空 0~3的tag进入缓存, miss 8次

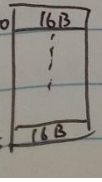
然后32~63的tag进入路2, miss 8次

64~95进入时 由LRU原则, 0~3被替换 miss 8次

所以cycle 1过后, 路1的tag为64~95, 路2 tag为32~63, 且路2为LRU

进入cycle 2: 0~3进入, miss, 路2替换, 然后32~63进入时 2miss, 路1替换, 64~95时仍miss, 路2替换, 上一次替换掉的正好时下一次使用的, 导致后面一直miss 缺失率: $1 - \frac{2}{3 \times 100} = 99.3\%$

缓存B:



24模16分配L 则 0~3. 64~95均对应上半缓存 32~63对应下半缓存

cycle 1: miss 的

2后的cycle: miss 2 hit 1

缺失率: $\frac{3+2\times 99}{3\times 100} = \frac{201}{300} = 67\%$

了初始缓存为空应当作miss吗？→强制缺失✓

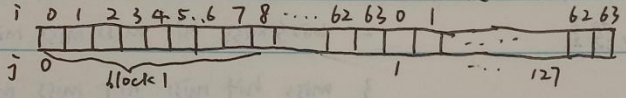
13. 代码实现的功能：对每个a[i][j]自加1 但现在为固定i改变了，每次j改变时A[i][j]与A[i+1][j]可能相隔甚远，导致缓存miss 考虑充分利用空间局部性 i,j互换。

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

14. (1) 4KB 直接映射 块大小32B 块数：$\frac{4KB}{32B}=128$

数组 A[j][i] j:0~127 i:0~64 先i相邻后j 认为是int型1个元素占4B 1个块放8个元素

则数组特点：



共8×128对应块。

直接映射情况下：每块会被8个内存区域映射到 $\frac{128}{64/8}=2^4=16$

比如说，j=0，j=16，j=32，…，的对应i序号块会映射到同一个cache块中 缓存初始为空。

优化前：大块移动→j固定，变i cycle1：j=0,1,2,…,15 hit 从j=16开始miss 之后均miss

在cycle=2及以后一开始也miss 缺失次数：~~$63\times128+128-16=$~~ $64\times128=8192$次。

(一开始时缓存为空，所以tag肯定比对不上，也认为是缓存miss)

优化后：j固定，变i cycle1：block初始会miss1次，之后均hit，共miss 8次

到cycle门时，会出现缓存冲突，不也替换后仍是miss8次，缺失次数：~~64×128~~ $8\times128=1024$次

(2). 4KB 全相联 数组特点没变 FIFO

优化前：j从0移至128 恰好占满所有缓存 下一次i移动了次都hit 缺失：$8\times128=1024$次。

优化后：变i 占满才替换 其实与直接映射相同 $8\times128=1024$次

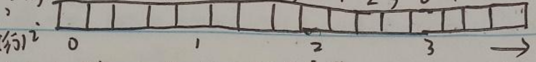
(3) 要求：除了初始为空的强制缺失无其他

优化后：$8\times128\times32B=32KB$ 优化前：$4KB\times8=32KB$ 均需要32KB

15. int 4B input起始于0x0 output起始于0x40

L1 cache: 32B 块大小16B 2块 直接映射 写策略: hit和miss时要写入cache当前段

input数据:



1个块可存4个元素

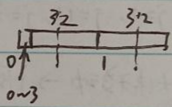
按字节寻址, input共占 4×4×4=64B字节 可见 ~~...~~ 这里input与output地址上是相连的

分析: 此时input和output映射同一块缓存, 可能相互冲突 初始: input行0, 列0, miss, 后面到input和output的行0.行2都映射到同一块缓存, 且为写分配策略, 所以output会行0又会把刚load进入的input行0覆盖

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

16.(1) 512B的cache: 块大小16B, 共 $\frac{512}{16}=32$块, 两路组相联 → 共16组

1个块中可装4个元素 128÷4=32 输入的input共2×32组



故cache每组有4个段内存对应, 存入两路

这里input[0][i]与input[1][i]总是对应同一组

当i=0时, input[0][i] input[1][i] 均进入cache第1组, 在两路保存 总是第1次miss, 后3次hit

当i到[64~127]后半段, 出现缓存冲突, 发生替换, 同理命中率为75%

(2) cache大小增加: 不会改善命中率 因为单纯cache大小增加, 可以使组相联的组数变多, 但一个块中也就只能放4个元素, 所以还是hit 3次 miss 1次, 命中率最多75%

(3) cache块大小增加, 可以改善命中率 在两路组相联情况下1次读取的两个数据正好放入~~两路~~同一组对应的两路cache块中, 所以~~cache块~~只会初始miss 1次, cache块越大后面hit次数越多, 命中率提升