2023-5-18. T5~16.

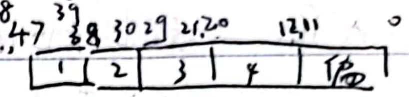
T5. ~~(1) 512个页表条目 → 1个页表 → 2MB地址空间~~

(1) 64位虚拟地址中，有12位用作每个页的页内偏移。

至多有 2^{52} 个页表项，即 2^{55} Byte页表空间

(2) 48位虚拟地址中，至多 2^{36} 个页表项，即 2^{39} Byte页表空间：512GB

(3) 假设48位虚拟地址，则至多 2^{36} 页表项



使用四级页表，每一级有512个页表项。

地址寄存器 ⊕ 一级页表 → 读出第2级内存基址。

2+二级页表 → 第3级内存基址。

⋮

→ 读出物理页号。

对于占用内存较少的进程，只需要4个页表即可。

假设一个进程空间只改变第0~22位，对于一级页表，$2^9 \times 4 = 2048$ 个页表，

对于四级页表，需 $1+1+4+2^9 = 518$ 页表。

实际页表存储开销分别约：8MB和2MB.

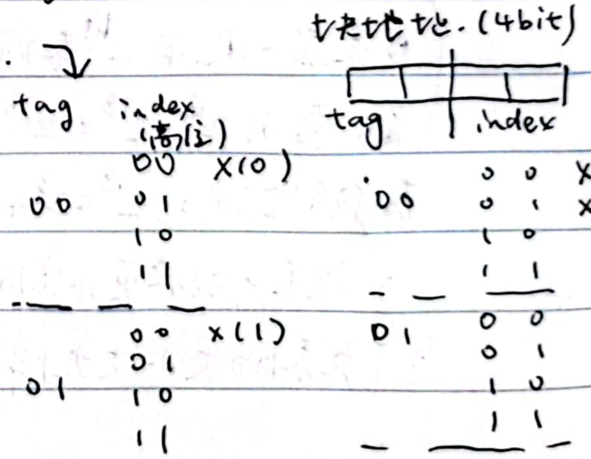
6. 一般程序地址改变中间位的比改变高位的多。~~若中间位为索引~~。

考虑一个直接映射缓存，当反复访问x(0)、x(1)时，

缓存就会因x(0)、x(1)无法同时放在组内

而发生抖动。

一般以中间地址为块索引，抖动概率会大幅降低。



7. 虚拟地址 = 页偏移 + 页位置.

页位置映射标签位.

这样把页偏移映射索引 + 块内偏移.

对虚拟地址可以容易转化成物理地址.

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

2) 假设访问 N 次. 每次 cache hit 概率为: $\frac{64KB}{1GB} = \frac{1}{64}$

$Cycle_{total} = \frac{1}{64} \times 1 + \frac{63}{64} \times 110 \approx 108$ 周期

3) 第2)中程序局部性太弱. 平均 cache miss rate 高. 访存此时高

4) 设 cache hit 概率是 p.

$p \times 1 + (1-p) \times 110 = 105$. 得: $p = 0.046$.

$p \geq 0.046$ 时. 才能让存储系统在使用 L1 时获得收益.

9.

2^S	S	tag bit	offset bit
32	5	21	6
8	3	23	6
0	0	26	6
256	8	18	6
64	6	19	7
256	8	18	6
64	6	20	6
32	5	20	7

10. 1) 系统A的平均访存时间：

t_A: $0.22ns \times (1-P_1) + (100ns + 0.22ns) \times P_1$

系统B：

$t_B = 0.52ns \times (1-P_2) + (100ns + 0.52ns) \times P_2$

$t_A < t_B$ 得：$P_1 < P_2 + 0.003$.

2) $t_A = 0.22ns \times (1-P_1) + (1+k) \times 0.22ns \times P_1$

$t_B = 0.52ns \times (1-P_2) + (1+k) \times 0.52ns \times P_2$

$t_A < t_B$ 得：$22k P_1 < 30 + 52k P_2$

11. 16 lines × 64 Bytes　　0x1001：

① 直接映射：index：4bits.

1000000000;0001	cold miss.	①
1000000000;0101	cold miss.	②
1000000010;0001	冲突	③
1000000100;0101	冲突	④
1001100000;0101	冲突	⑤
1011101110;0101	冲突	⑥
1111111110000;0101	冲突	⑦

② 2路组相联：index 3bits

0x1001	cold miss	①	0xeee5	冲突	⑥
0x1005	cold miss	②	0xff05	冲突	⑦
0x1021	cold miss	③		冲突	
0x1045	cold miss	④			
0x1305	冲突	⑤			

3. 4路组相联：index bits = 2

① cold miss ⑤ 冲突

② cold miss ⑥ 冲突

③ cold miss ⑦ 冲突

④ cold miss ⑧ 冲突

4. 8路组相联 index bits = 1.

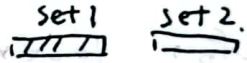
全部cold miss.

12. ① 缓存A. 16 lines. × 16 Bytes. 2set × 8 indexs.

array: 0 ~ 96×4Byte. : 00000,0000,0000 ~ 00011,0000,0000

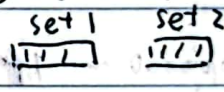
每个line可存4个int型数据.

初始时：每16个Byte缺失一次. 直到 Cache填充8 lines: count = 8.



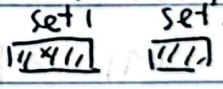
00000,0000 ~ 01000,0000

每 16 Byte 缺失一次. 直到 Cache 填满：count = 16.



01000,0000 ~ 10000,0000

每 16 Byte 一次. 直到 11000,0000 : count = 32. 替换先前用过的.



每次调动32个数组元素. 发生8次缺失. 故总：100 × 3 × 8 = 2400 次

$$\text{miss rate} = \frac{2400}{96\times100} = 0.25$$

② 缓存B：16 lines × 16 Byte.

每 16 Byte 缺失一次：直到 Cache 填满 count = 16

00000,0000 ~ 10000,0000

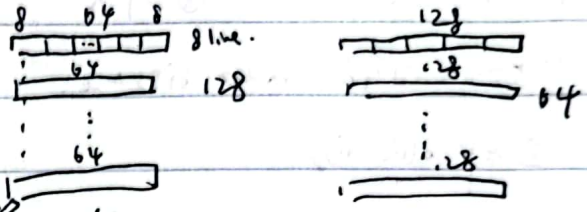
再从头更换缓存. 直到 11000,0000 count = 24.

总共：100 × 24 = 2400 次

$$\text{miss rate} = \frac{2400}{96\times100} = 0.25$$

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

14. 4KB. 32Byte. 128 lines



(1) 优化前：每个元素占4个字节.

故每一个line可存8个元素.

$128 \times 64 = 8192$ 次. 每一个元素都替换 cache.

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

(2). 优化前：1024次.

优化后：1024次.

(3). 优化前：$8 \times 128 \times 32\text{Byte} = 1\text{TB}$.

优化后：$1024 \times 32\text{Byte} = 1\text{TB}$.

15. input: 00[0]0000 ~ 11[1]0000 output: 10[0]0000 ~ 01[1]0000 (4位元素.)

每个line存4位元素.

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