

5. 回答以下问题:

- 1) 如果页大小为 4KB, 每个页表条目使用 8 字节空间, 内存系统按字节寻址。则使用完整的 64 位虚拟地址时, 一个单级页表系统需要多大的空间用于存储页表?
- 2) 实际上, 多数真实系统仅限制使用 64 位系统的一部分位作为有效的访存空间, 例如 Sv48 即仅使用 48 位的虚拟地址空间, 则保持其他假设不变时, 一个单级页表系统存储页表所需要的空间被降低到多少?
- 3) 多级页表为什么可以降低虚拟内存系统的实际页表存储开销?

1) 一个页大小为 4KB, 即  $2^{12}$  位, 故需要 12 位来表示页内偏移量  
故需要用  $64 - 12 = 52$  位来表示页号, 故一个页表可存储  $2^{52}$  个页表条目  
则一个单级页表需要空间:  $2^{52} \times 8B = 32PB$

2)  $2^{48} / 2^{12} \times 8 = 2^{36}B = 512GB$

3) 多级页表能够随着进程占用内存空间的增大对应地增多属于该进程的页表数目, 进程占用空间小时页表数目也少。

6. 试简要分析为什么缓存一般使用地址的中间位作为组索引、高位作为标签, 而不是用高位作为组索引, 中间位作为标签?

高位的标签可以提供足够的位数区分不同的数据块,  
使用高位作为标签, 可以充分利用缓存的容量存储更多数据块。

使用中间位作为组索引虽然会限制缓存可以容纳的组数, 但可以快速寻到需要查询的组, 访问效率高。

7. 一些真实的缓存系统在设计时会有意让地址的组索引和块内偏移的总位数与虚拟内存系统的页偏移位数相同, 试分析这样做有什么好处。

当处理器发出的虚拟地址中的索引位和块内偏移两者的位数之和不大于页内偏移的位数时, 可以使用 VIPT (virtually indexed, physically tagged) 缓存, 令地址翻译和缓存索引并行, 减少命中时间。

8. 缓存作为加速处理器访存的手段，依赖于处理器访存的局部性原理。假设一个由 L1 缓存和主存构成的存储系统。不同情况下的访问延时为：缓存命中时 1 周期、缓存缺失时 110 周期、禁用缓存总是直接访问主存时 105 周期。则：
- 1) 如果使用 L1 缓存，且程序的平均缓存缺失率为 3%。则存储系统平均访问延时为多少？
  - 2) 如果一个具有完全随机数据访问性质的程序，该程序持续地访问一个 1GB 数组中的随机位置。假设主存足以放下完整数组，L1 大小为 64KB，则存储系统平均访问延时为多少？
  - 3) 利用 a 和 b 结果，说明局部性原理如何影响处理器的访存性能。
  - 4) 程序的平均缓存命中率需要满足什么条件，才能让本例中的存储系统在使用 L1 时获得性能收益？

1) 平均访问延时:  $N_1 = 1 \times 97\% + 110 \times 3\% = 4.27$  周期

2)  $1GB = 2^{30} KB$ ，缓存命中率:  $\frac{2^6}{2^{30}} = \frac{1}{2^{24}}$

平均访问延时:  $N_2 = 1 \times \frac{1}{2^{24}} + 110 \times \frac{2^{24}-1}{2^{24}} \approx 110$  周期

3) 缓存系统的设计是基于处理器访存的局部性原理的，包括时间局部性和空间局部性。满足缓存较高的命中率，才能够显著优化平均访存时间。

但若 2) 中条件数据完全随机访存，不再成立局部性原理，导致缓存的命中率很低，对性能几乎没有影响。

4) 设缓存的命中率为  $K$ ，则有：

$$1 \times K + 110 \times (1-K) < 105$$

$$\Rightarrow K > \frac{5}{107}$$

故命中率至少要达到  $\frac{5}{107}$ ，采用 L1 缓存才会有优势。

9. 根据给出的不同缓存配置，补全下表中缺失的字段。

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

10. 虽然较大的缓存通常有较低的缺失率，但更大的缓存一般也带来更高的命中延时。最终的性能变化由两者综合决定。假设两个具有 L1 缓存的存储系统：系统 A 使用 8KB 直接映射缓存，命中延时为 0.22ns，缓存缺失率为  $p_1$ ；系统 B 使用 64KB 四路组相联缓存，命中延时为 0.52ns，缓存缺失率为  $p_2$ 。则：

- 1) 如果缓存的缺失代价均为额外 100ns，则  $p_1$  和  $p_2$  满足什么条件时，系统 A 的平均内存访问时间优于系统 B？
- 2) 如果缓存的缺失代价分别是各系统命中延时的  $k$  倍，则  $p_1$  和  $p_2$  满足什么条件时，系统 A 的平均内存访问时间优于系统 B？

1)  $(1-p_1) \times 0.22 + p_1 \times 100.22 = 0.22 + 100p_1$   
 $(1-p_2) \times 0.52 + p_2 \times 100.52 = 0.52 + 100p_2$   
 $0.22 + 100p_1 < 0.52 + 100p_2$   
 $\Rightarrow p_1 < 0.003 + p_2$

2)  $(1-p_1) \times 0.22 + p_1 \times (k+1)0.22 = 0.22 + 0.22kp_1$   
 $(1-p_2) \times 0.52 + p_2 \times (k+1)0.52 = 0.52 + 0.52kp_2$   
 $0.22 + 0.22kp_1 < 0.52 + 0.52kp_2$   
 $\Rightarrow p_1 < \frac{15}{11k} + \frac{26p_2}{11}$

11. 假定数据缓存共有 16 个块, 块大小为 64 字节。若有一系列发往数据缓存的请求, 它们的块地址 (块地址是已经去除块内偏移后的内存地址) 为: 0x1001, 0x1005, 0x1021, 0x1045, 0x1305, 0x20e5, 0x0f05。假设数据缓存初始为空, 讨论当此数据缓存为直接映射、2 路组相联、4 路组相联和 8 路组相联的情况下, 缓存发生块替换的次数。

数据 0x 开头是 16 进制

$0x1001 \bmod 16 = 1$       直接映射: 5 次缓存块替换  
 $0x1005 \bmod 16 = 5$       2 路组相联: 3 次缓存块替换  
 $0x1021 \bmod 16 = 1$       4 路组相联: 3 次缓存块替换  
 $0x1045 \bmod 16 = 5$       8 路组相联: 0 次缓存块替换  
 $0x1305 \bmod 16 = 5$   
 $0x20e5 \bmod 16 = 5$   
 $0x0f05 \bmod 16 = 5$

12. 增加缓存的相联度通常可以降低缺失率, 从而提高性能。但对于特殊的程序, 情况可能相反。假设两个块大小均为 16 字节, 总容量均为 256 字节的缓存; 缓存 A 为 2 路组相联, 缓存 B 为直接映射。两个缓存均使用 LRU 替换策略。分别计算运行下述程序时, 使用上述两种缓存的缺失率。(初始时缓存为空, i 和 j 均缓存于寄存器中, 数组 array 在内存中的起始地址为 0)。

```

#define N 100
int32_t array[96];
for(int32_t i = 0; i < N; i++) {
    for(int32_t j = 0; j < 96; j++) {
        array[j] = i*j;
    }
}

```

LRU 策略 (least recently used): 数组 array [96] 中每个数 32 位, 占 4 个字节, 1 个块中可放 4 个数

如 j 的存储于寄存器中, 访问延迟事件, 不会进入缓存。

2 种缓存都有 16 个块, 每个块的大小都是 16 字节, 将 array [96] 分成 3 部分, 每部分在内存中占 8 个块, 分别编号 ① ② ③

缓存 A: 2 路 2 路:

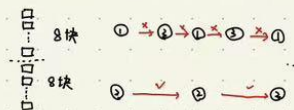
第 1 路: □ □ □ □ □ □ □ □ ① → ② → ③ → ①

第 2 路: □ □ □ □ □ □ □ □ ③ → ① → ② → ③

可知每路缓存中写 4 个数 (写满一个块) 后就会发生块替换。

即缺失次数每 3 次, 则缓存的缺失率为 25%

缓存 B: 直接映射。将缓存分成 2 部分。



i=0 时, 缓存缺失 24 次, 命中 72 次。i=1~99 时, 缓存缺失 16 次命中 80 次。

故缓存的缺失率为:  $\frac{24 + 99 \times 16}{99 \times 100} = 16.75\%$

13. 考虑如下所示的 C 语言代码片段:

```

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

```

在内存中数据的组织方式为 A[p][q] 与 A[p][q+1] 相邻。请从利用缓存空间局部性的角度, 优化以上代码, 不可更改代码实现的功能。

```

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

```

14. 仍然考虑题 13 中的代码, 假设系统仅在对数组 A 发生读写时访问缓存, 其他变量全部位于寄存器中。缓存初始为空。回答以下问题:

- 假设存在一个 4KB 大小的直接映射缓存, 块大小 32 字节, 则优化前和优化后代码的缓存缺失次数分别为多少?
- 假设存在一个 4KB 大小的全相联缓存, 块大小 32 字节, 使用 FIFO 替换策略。则优化前和优化后代码的缓存缺失次数分别为多少?
- 对于块大小 32 字节的直接映射缓存, 如果希望系统除了必要的强制缺失外不出现任何其他缺失, 则优化前和优化后代码分别至少需要多大的缓存容量?

1) 4KB =  $2^{12}$ B    32B =  $2^5$ B    故缓存中共有 128 个块, 数组的大小是 A[128][64], 假设 int 类型占 32 位, 即 4 个字节, 则每个块中可以存放 8 个数  
内存中共有 128 × 64 个数, 需要分配 1024 个块, 64 个数占据 8 个块, 在使用直接映射缓存的情况下:

优化前缓存缺失次数:  $N_1 = 128 \times 64 = 8192$  次

优化后缓存缺失次数:  $N_2 = 8 \times 128 = 1024$  次

2) 4KB 大小的全相联缓存, 块大小 32B, 故一组 128 块, 且采用 FIFO 替换策略。

优化前缓存缺失次数:  $n_1 = 128 \times 8 = 1024$  次

优化后缓存缺失次数:  $n_2 = 8 \times 128 = 1024$  次

3) 强制缺失指初始时缓存为空不可避免的快速替换, 基块大小仍是32字节

优化前: 至少需要 128 个块, 即  $2^7 \times 2^5 B = 2^{12} B = 4KB$

优化后: 至少需要 1 个块, 即  $2^5 B$

15. 考虑如下所示的实现矩阵转置功能的 C 语言代码片段:

```
int input[4][4], output[4][4];
for(int i = 0; i < 4; ++i){
    for(int j = 0; j < 4; ++j){
        output[j][i] = input[i][j];
    }
}
```

假设系统中 int 变量为 4 字节, input 数组的起始地址为 0x0, output 数组的起始地址为 0x40。假设上述程序运行在一个存在 L1 缓存的系统中, 该缓存总大小 32 字节, 块大小 16 字节, 直接映射, 缓存策略为直写、写分配。若初始缓存为空, 系统仅在对 input 和 output 的读写时会访问缓存, 请在下表中填写执行上述程序时数组中每个元素的缓存命中情况。

	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	miss	hit	miss	miss	miss	miss
行 1	miss	hit	miss	hit	miss	miss	miss	miss
行 2	miss	hit	miss	hit	miss	miss	miss	miss
行 3	miss	hit	miss	hit	miss	miss	miss	miss

直写即写通, 数据不仅被写入缓存, 而且被写入下一级存储, 即缓存中内存中的数据始终保持一致。

假设处理器是 32 位, 则 input 数组在内存中占据的块编号为 1, 2, 3, 4, output 数组在内存中占据的块编号为 11, 12, 13, 14。

缓存中有 2 个块, 每个块中可容纳 4 个数。

假设在内存中 input[i][j] 与 input[i][j+1] 相邻, output[j][i] 与 output[j][i+1] 相邻。

16. 考虑如下所示的 C 语言代码片段:

```
int input[2][128];
int sum = 0;
for(int i=0; i<128; ++i){
    sum += input[0][i] * input[1][i];
}
```

假设系统中 int 变量为 4 字节, input 数组的起始地址为 0x0。系统仅在对 input 数组的读写时会访问缓存, 其他变量全部位于寄存器中。则:

- 1) 如果存在一个 512 字节大小的缓存, 该缓存两路组相联, 块大小为 16 字节, 使用 LRU 替换策略, 缓存初始为空。则执行上述程序时缓存的命中率为多少?
- 2) 在其他条件不变的情况下, 增加缓存的总大小可以改善该程序的命中率吗? 请说明理由。
- 3) 在其他条件不变的情况下, 增加缓存的块大小可以改善该程序的命中率吗? 请说明理由。

假设在内存中数据 input[i][j] 与 input[i][j+1] 相邻。

- 1) 缓存大小为 128B, 块大小 16B, 故有 32 个块, 2 路 16 组, 可以容纳 128 个数。

数组 input[2][128] 中有 256 个数, 在内存中会占据 64 个块。

且采用 LRU 替换策略, 缓存初始为空。

内存每个块中的数据始终是 1 缺失 3 命中, 故缓存的命中率是 75%。

- 2) 增加缓存的总大小不会改善程序的命中率, 因为对缓存中占有块的替换与缓存中空白的替换效果相同。

- 3) 其他条件不变, 增加块的大小可以改善程序的命中率, 因为一般来说一个块中只有块首的数据会出现缓存有缺失, 随着块容量的增大, 块的数量减少, 缓存缺失的次数也会相应减少。