

5/23. che 4

17. 解: 虚拟地址 14 bit . 物理地址 12 bit  
页大小 64 B.

TLB 16 个条目. 4 路组相联. 一组 4 个条目.

L1 Cache 物理寻址. 块大小 4 字节. 共 16 个组.

虚拟地址  $0x05a4 \Rightarrow 00\ 0101\ 1010\ 0100$

1) 页大小  $64B = 2^6B$ . 用 6 bit 作页内偏移 offset.

TLB 4 组. 用 2 bit 作检索组号的 index

还剩  $14 - 6 - 2 = 6$  位虚地址  $00\ 0101$  对应  $0x05$  为 tag

index = ~~0~~ 10 对应组号 2. 命中 TLB. 物理地址  $0x1c$ .

$\Rightarrow$  物理地址 由  $0x1c$  和  $100100$  拼接:  $0111\ 0010\ 0100$ .  
 $= 0x724$ .

2)  $2^{12}/64 = 64$  个条目.

3) 使用  $0x724$  访问内存.

Cache 分割地址:  $0111\ 0010\ 0100$

块大小 offset ~~4B~~, 16 个组用 4 bit 检索 (index).  
4B  
25 bit

$\Rightarrow$   $\frac{0111\ 00}{\text{tag}} \quad \frac{1001}{\text{index}} \quad \frac{00}{\text{offset}}$ . 组号为 9. tag 为  $0x1c$ . offset 为  $0x00$  命中.

$\Rightarrow$  命中 ~~Cache line 9~~ Cache line 9. tag  $0x1c$ . index ~~0x1001~~  $0x1001$ . offset  $0x00$ .



18. D. LRU.

访存地址

A B C D A B C D  
- A A C C A A C

way0

- B B D D B B

way1

- N N N N N N N N

命中?

N N N N N N N N

命中率为0%

2). 采用MRU替换策略. most recently used.

则上图表变为:

访存地址

A B C D A B C D A B C D A B  
- A A A A A B C C C C C D A

way0

- B C D D D D D A B B B B

way1

- N N N N Y N N Y N N Y N N Y

命中?

N N N N Y N N Y N N Y N N Y

长时间运行后, 程序缓存命中率约为50%

19. 1). 微位标, 在判断缓存命中与否时, 控制器仅取低标签位进行比较, 如果在同一缓存组内有两个或更多的缓存行具有相同的低标签位, 那么控制器将无法准确判断哪一个缓存行应该被选中。因此, 低标签位唯一是为了确保能准确地匹配缓存行。

2). 通常的缓存替换策略中, 主要依赖于标签和索引的比较结果。但在使用了微标签技术的缓存系统中, 因为部分标签信息(低标签位)已经在判断缓存命中与否的早期阶段被取出并比较使用, 所以在后端的替换决策中, 只能使用高位标签, 可能会影响替换策略的复杂性, 使之更复杂, 使效率降低, 但整体而言, 微标签技术可以用命中率和性能提升来弥补。



3) 虚拟页偏移 = 物理页偏移. 无需翻译 (无需经过TLB).

页大小: 16KB  $\Rightarrow$  页 offset 位数 14 bit.

8KB 大小的回路组互联: 每个 set  $8KB/4 = 2KB$ .

题目未指明块大小, 这里使用常见的 64B. 块内偏移 6 位.

则每个 set 有  $2KB/64B = 32$  块.

组索引位数:  $8KB/64B \times \frac{1}{4} = 2^3 \div 2^6 \div 2^2 = 2^5 \Rightarrow$  组索引 5 位.

$\Rightarrow$  低位标签 LTag = 页偏移 - 组索引位 - 块内偏移.

$= 14 - 5 - 6 = 3$  位. 至多 3 bit LTag.

20. 目录式一致性 } 优点: 单对单传播, 扩展性好.

缺点: 一致性事务处理时间长, 延迟高.

监听一致性 } 优点: 单对多传播在带宽足够时用总线广播, 延迟低.

缺点: 当处理器核数量增加, 一致性事务的流量剧增, 扩展性差, 对带宽要求高, 难以在大规模多核处理器中实现.

缓存一致性的代价:

监听 & 发送

① 硬件复杂性: (一致性协议, 一致性目录, 一致性消息的接口).

② 性能开销: 发送 invalid 信号, 确认其余副本有无修改等.

③ 能耗增加: 一致性协议的实现导致需要更多能量.

④ 带宽消耗: 占用通信带宽可能降低高性能计算的最大性能.

⑤ 编程复杂性: maybe 要通过控制数据的分布和访问模式来减少一致性冲突 & 通信开销. 程序优化复杂.