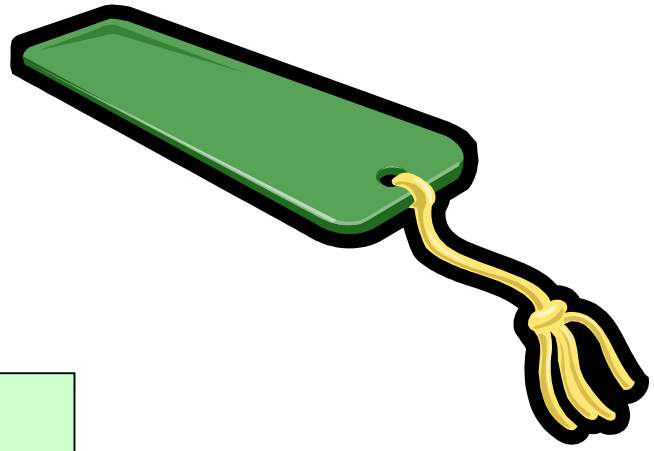
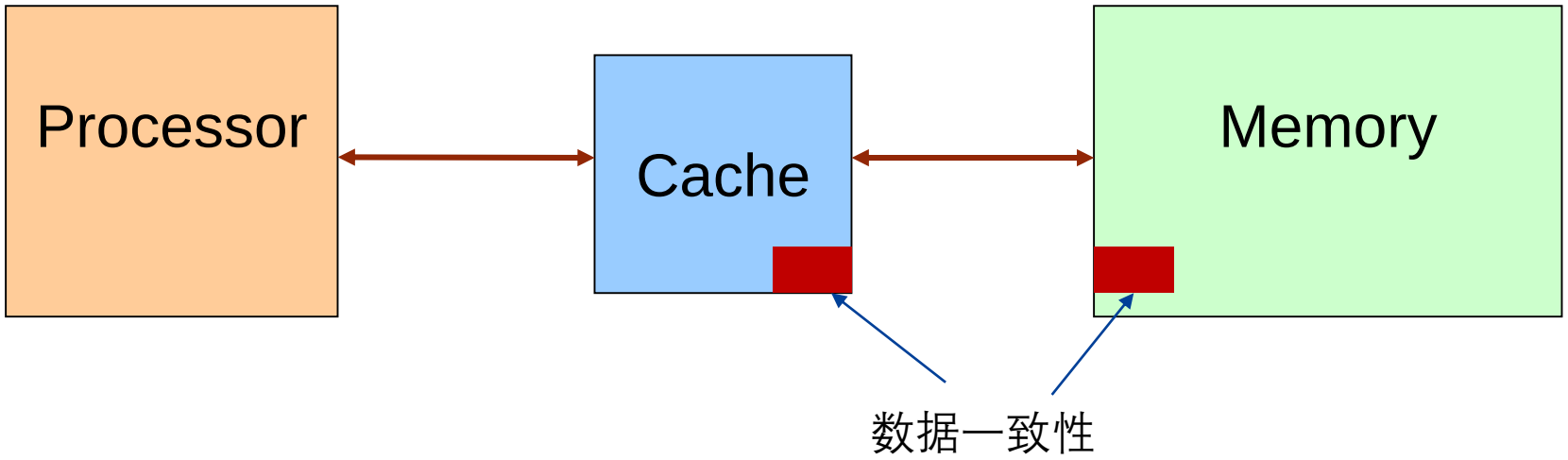
Cache 更新策略

缓存 (Cache) 的关键问题之一

- 如何保证缓存 (cache) 与主存 (memory) 的一致性
  - 更新策略 Write Policy



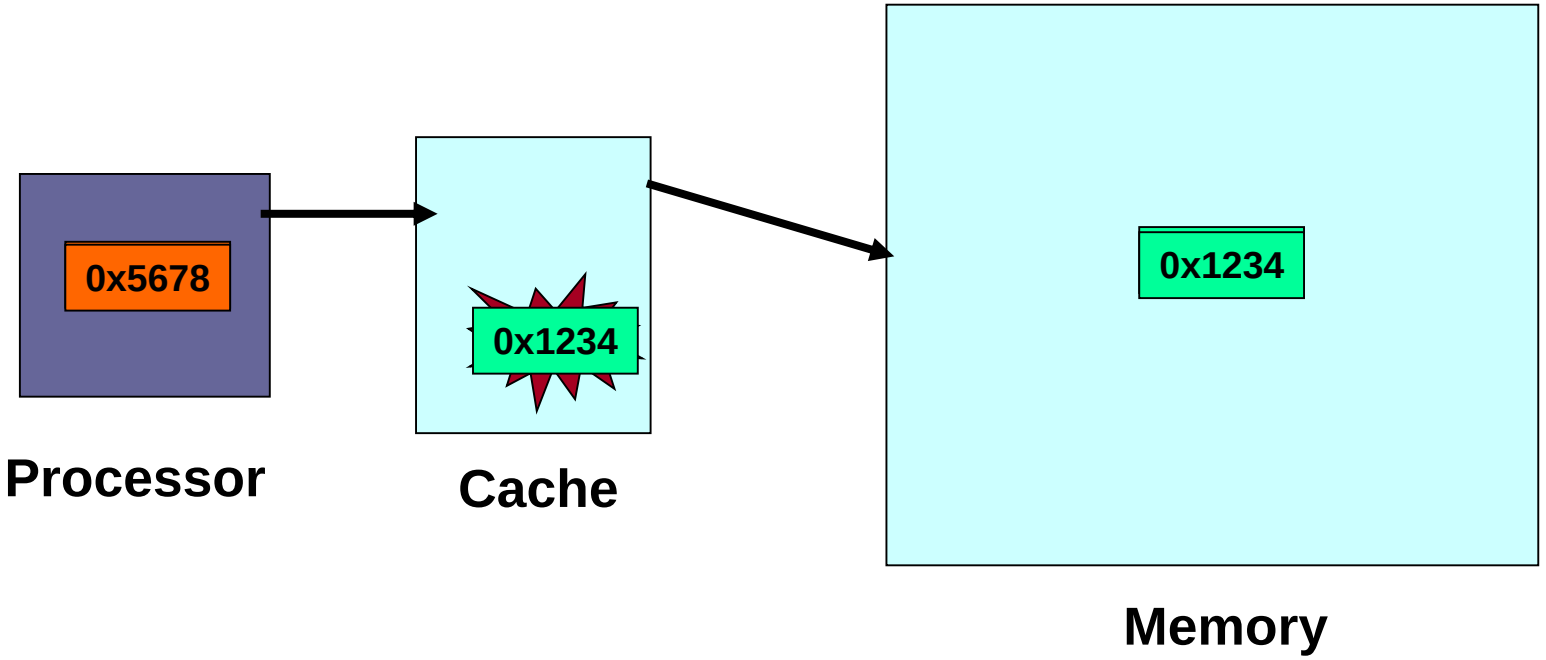


Cache命中 (hit) 时的写策略

- 写直达法 Write-through (直接向内存中写数据)

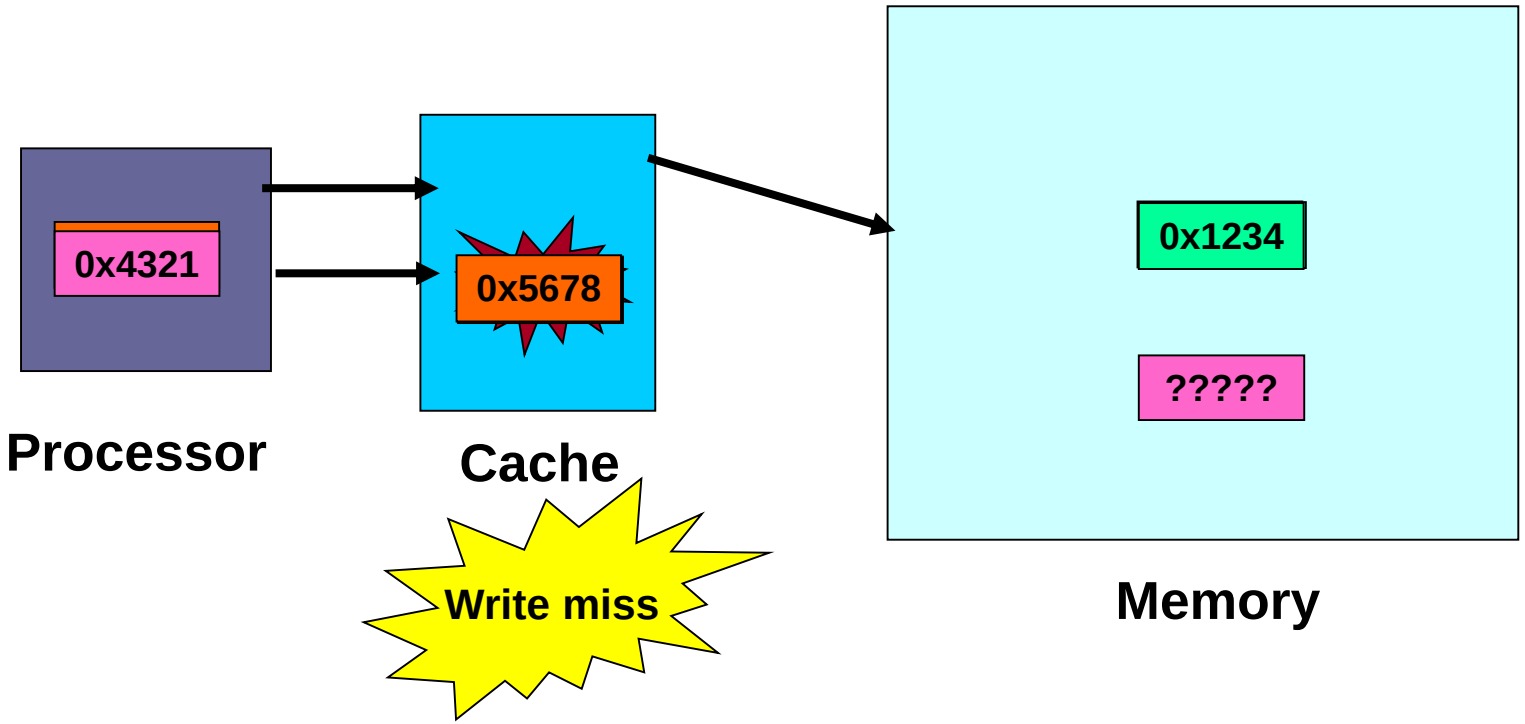
- 写回法 Write-back (推迟对内存的写入直到数据所在缓存的行被替换)
  - 需要设置一个页面重写标志位 (dirty bit) 用于标记缓存中的行是否与内存中的对应行相同

写直达 Write-through



1. 读Cache 不命中
2. 将该块从主存调入Cache, 并读入处理器
3. 处理器将该块从0x1234 改为0x5678
4. 修改的内容， 写入Cache,同时写入Memory

写回法 Write-back

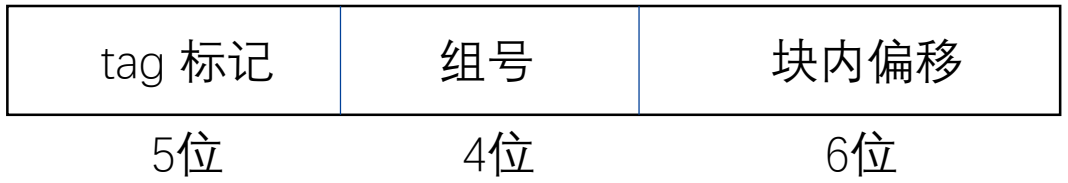


1. 假设读x, 读Cache 不命中
2. 将该块从主存调入Cache, 并读入处理器
3. 处理器将该块从0x1234 改为0x5678
4. 修改的内容，只写入Cache, 不写入Memory
5. 处理器只和Cache打交道，Memory中的内容和Cache中的不一致也没关系
6. 当处理器要写的数据,假设为y, 在另外一块，发生miss, 需要将新块替换进来，5678所在的块替换出去，首先会将0x5678所在的Cache块，写入内存
7. 读入新的块，将它替换0x5678数据所在的块
8. Cache 新块的中y内容修改为0x4321

举例

- 某计算机的主存地址空间大小为32KB， 按字节编址
- 缓存 (Cache) 采用4路组相联映射，LRU替换算法和写回（write-back）策略，能存放4KB数据
- 主存与缓存之间交换的主存块大小为64B

（1）主存地址字段如何划分?



块大小 64B=2^6 B
Cache 行数 4KB/64B=64
Cache 4路组相联
Cache 组数 64/4=16=2^4
主存大小32KB=2^{15} B

（2）Cache 的总容量有多少位?

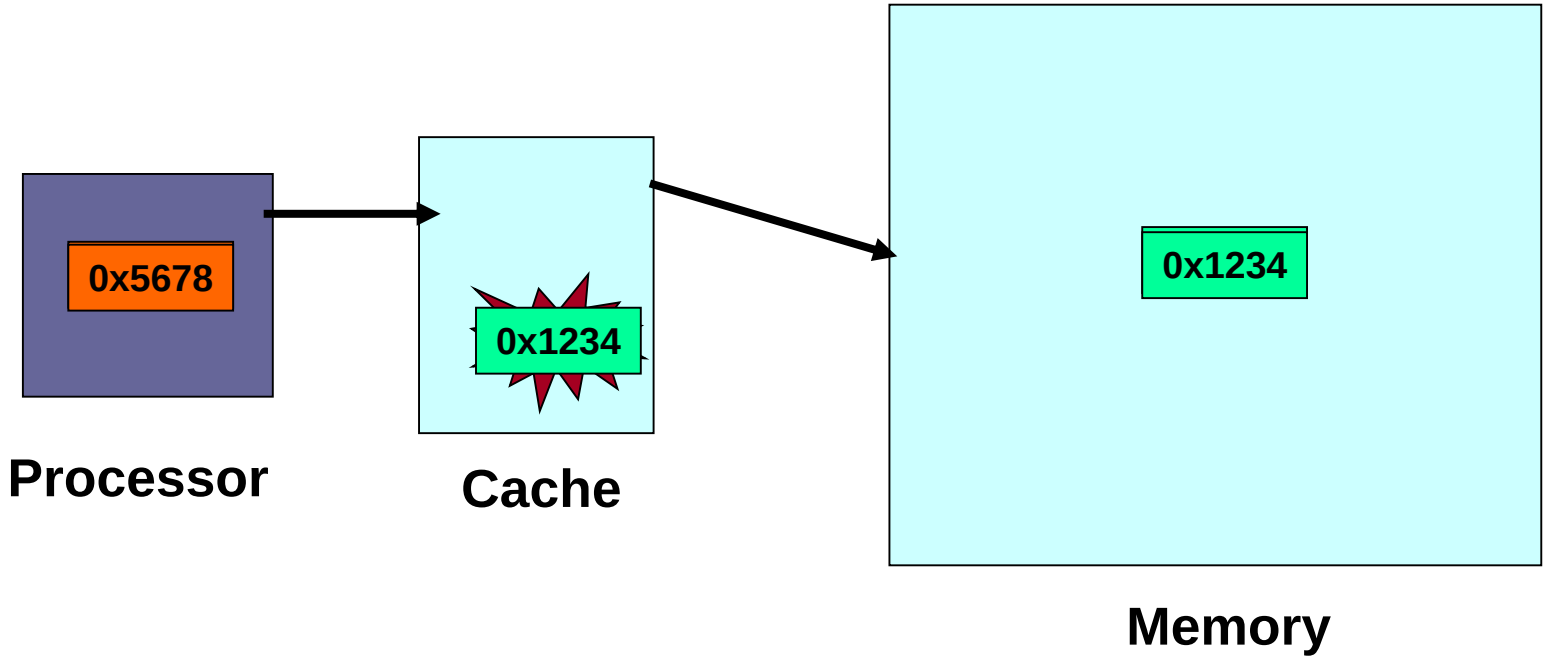
写回策略：每一行有：1位修改位（dirty bit)
LRU替换： 每组有4行，所以每行有：2位 LRU位
每行还有：5位tag 标记位，1位有效位和64B 数据
总容量： 64*（5+1+1+2+64*8）= 33344 位

Cache失效 (miss) 时的写策略

- 按写分配法 Write-allocate (将数据读入缓存，在缓存中更新内容)
  - 如果要连续访问同一位置，这种策略较优
  - 出现写失效时，首先会出现读失效
  - 在缓存中先为写失效分配一行 (line)，然后在这一行发生一次写命中
  - **常与写回法 (write-back) 搭配使用**

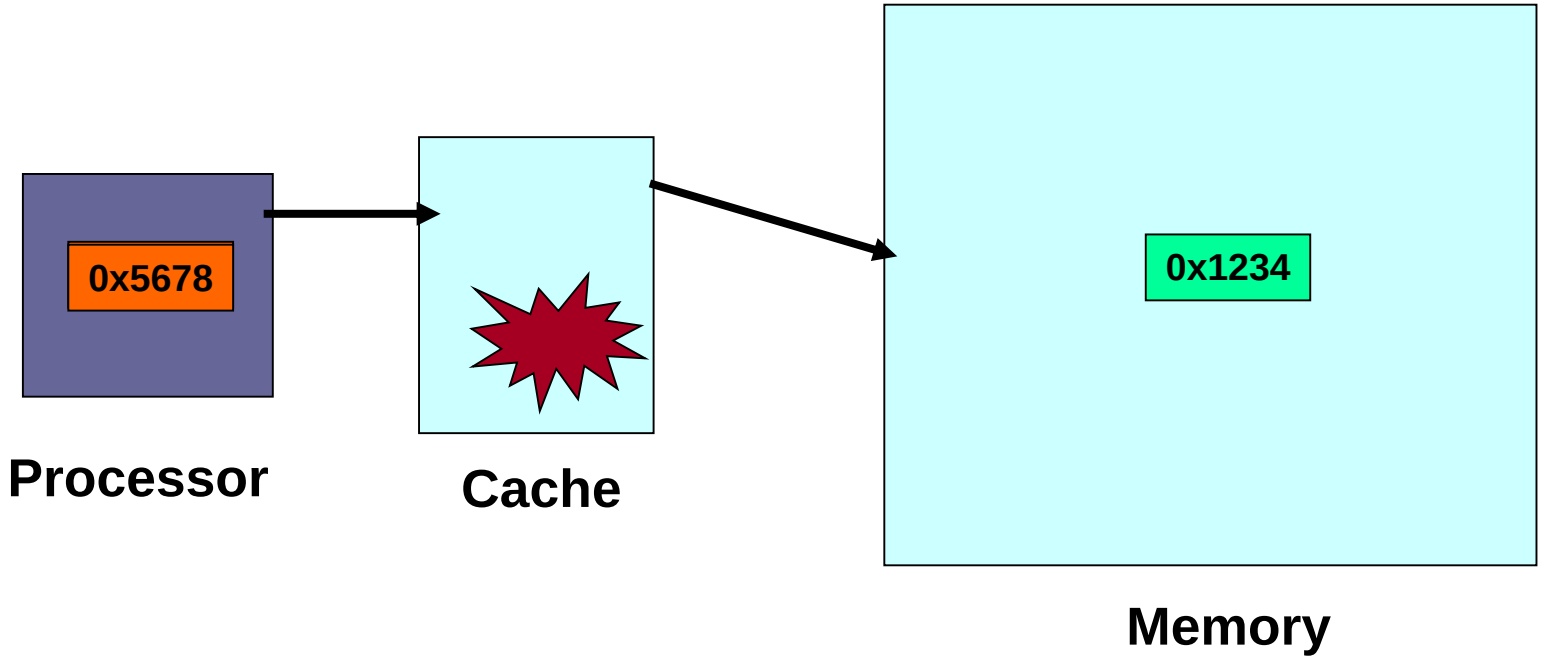
- 不按写分配法 No-write-allocate (直接修改内存中的内容，无需读入缓存)
  - 写失效不影响缓存中的数据
  - 数据仅在低级的存储器中被修改
  - **常与写直达 (write-through) 搭配使用**

按写分配法 Write-allocate



1. 写Cache 不命中
2. 将该块从主存调入Cache, 并读入处理器
3. 处理器将该块从0x1234 改为0x5678
4. 修改的内容写入Cache

不按写分配法 No-write-allocate



1. 写Cache 不命中
2. 处理器在内存中将该块从0x1234 改为0x5678

典型策略

- 写直达+不按写分配法 (Write-through + No-write-allocate)
- 写回法+按写分配法 (Write-back + Write-allocate)

举例

假定某处理器可通过软件对高速缓存设置不同的写策略，在下列两种情况下，应分别设置成什么写策略?

（1）处理器主要运行包含大量存储器写操作的数据访问密集型应用。

采用写回法 (write-back) 策略较好，可减少访存次数。

（2）处理器运行程序的性质与（1）相同，但安全性要求高，不允许有任何数据不一致的情况发生。

采用写直达 (write-through) 策略较好，能保证数据的一致性。

小结

- 缓存 (cache) 命中时
  - 写直达法
  - 写回法
- 缓存 (cache) 失效时
  - 按写分配法
  - 不按写分配法

- 根据应用类型和需求，选择不同的策略组合