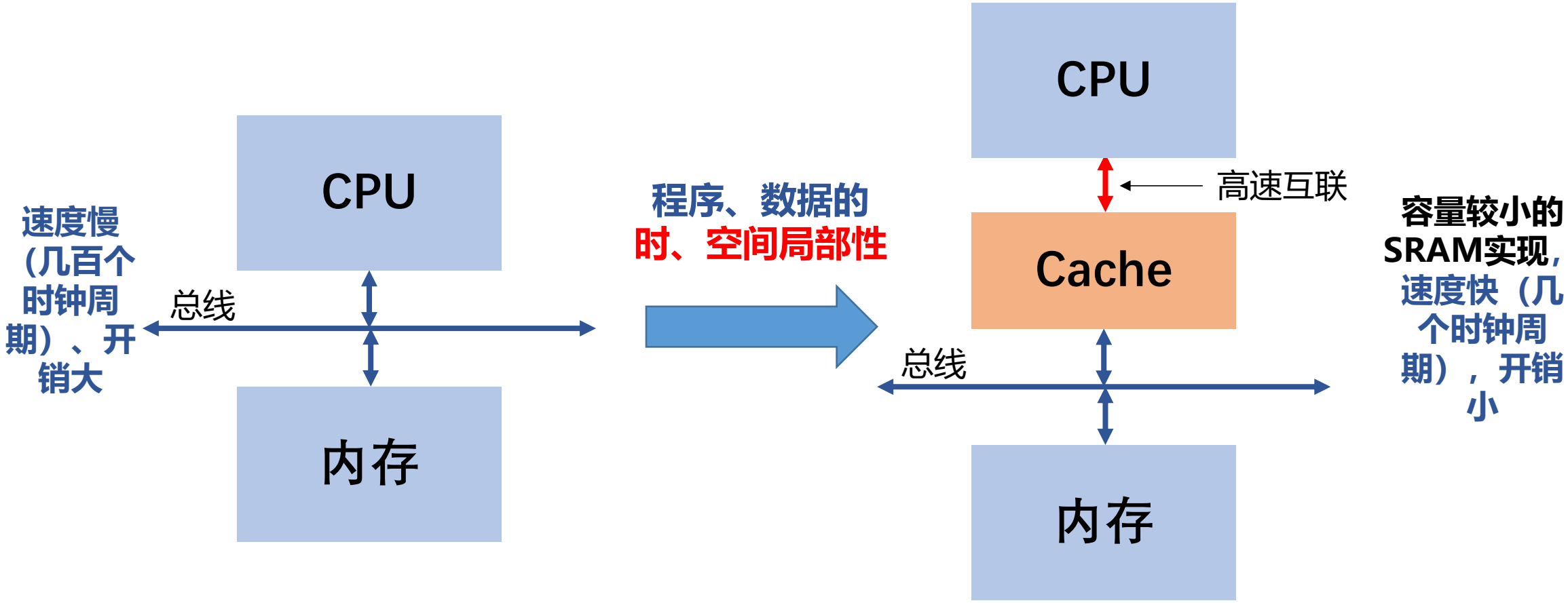
一、高速缓存（Cache）设置意义

减少对内存的频繁访问带来的延时、功耗开销



CPU 直接从内存搬运数据

Cache 暂存常用数据

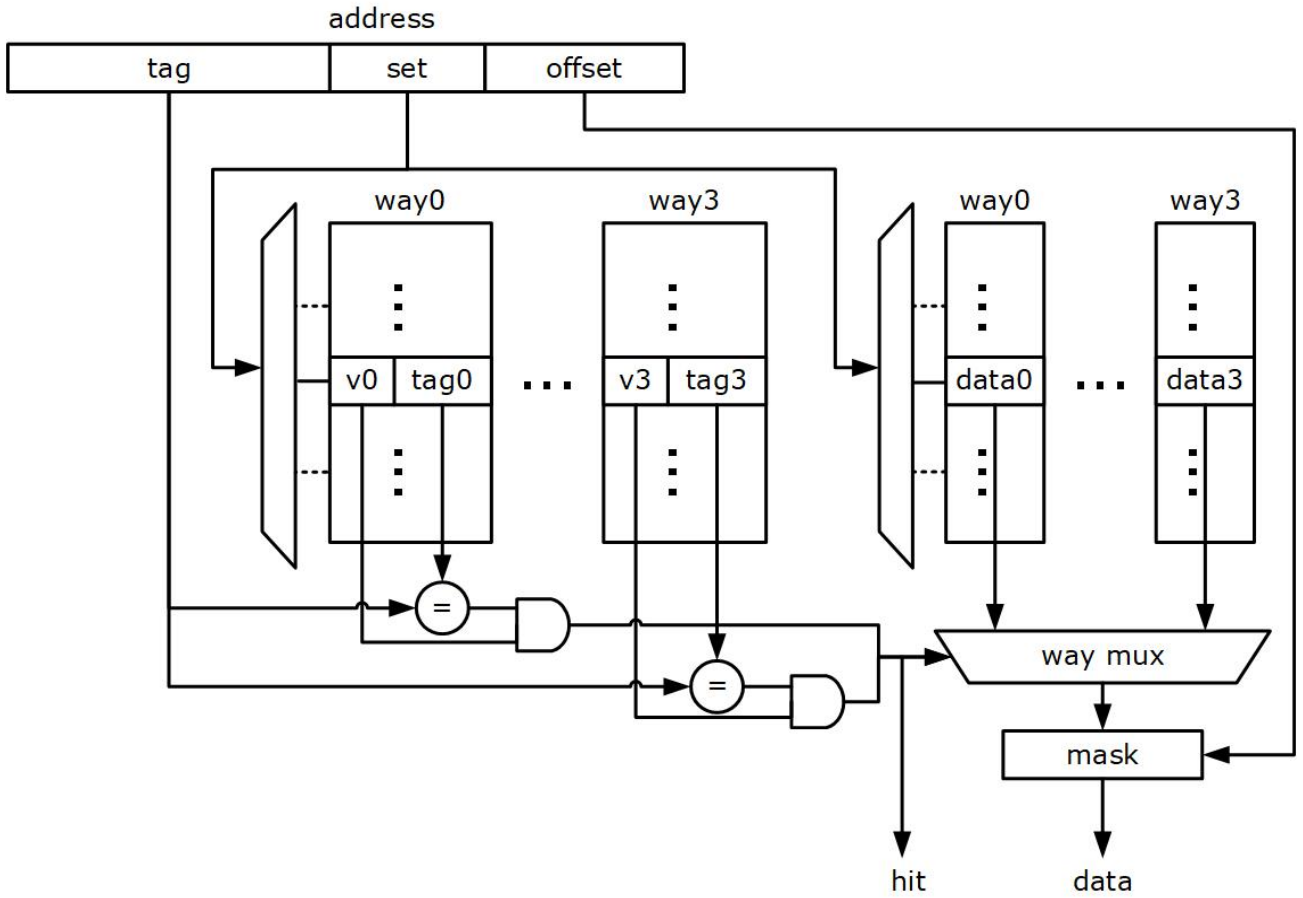
二、Cache设计基本点

- 结构组成
- 地址映射方式
- 写策略
- 置换算法

三、Cache的基本结构

标签（tag）阵列+数据（data）阵列+控制逻辑

- Tag: 地址的一部分，用于标识cache行
  - Cache 容量小于内存，暂存内存数据/程序副本。多个内存块可以映射到cache的统一行上，于是需要tag字段进行标识。
- 控制逻辑：查找、比较、置换等



Cache简单示意图

四、内存到Cache的映射方式

◆ 直接相联映射

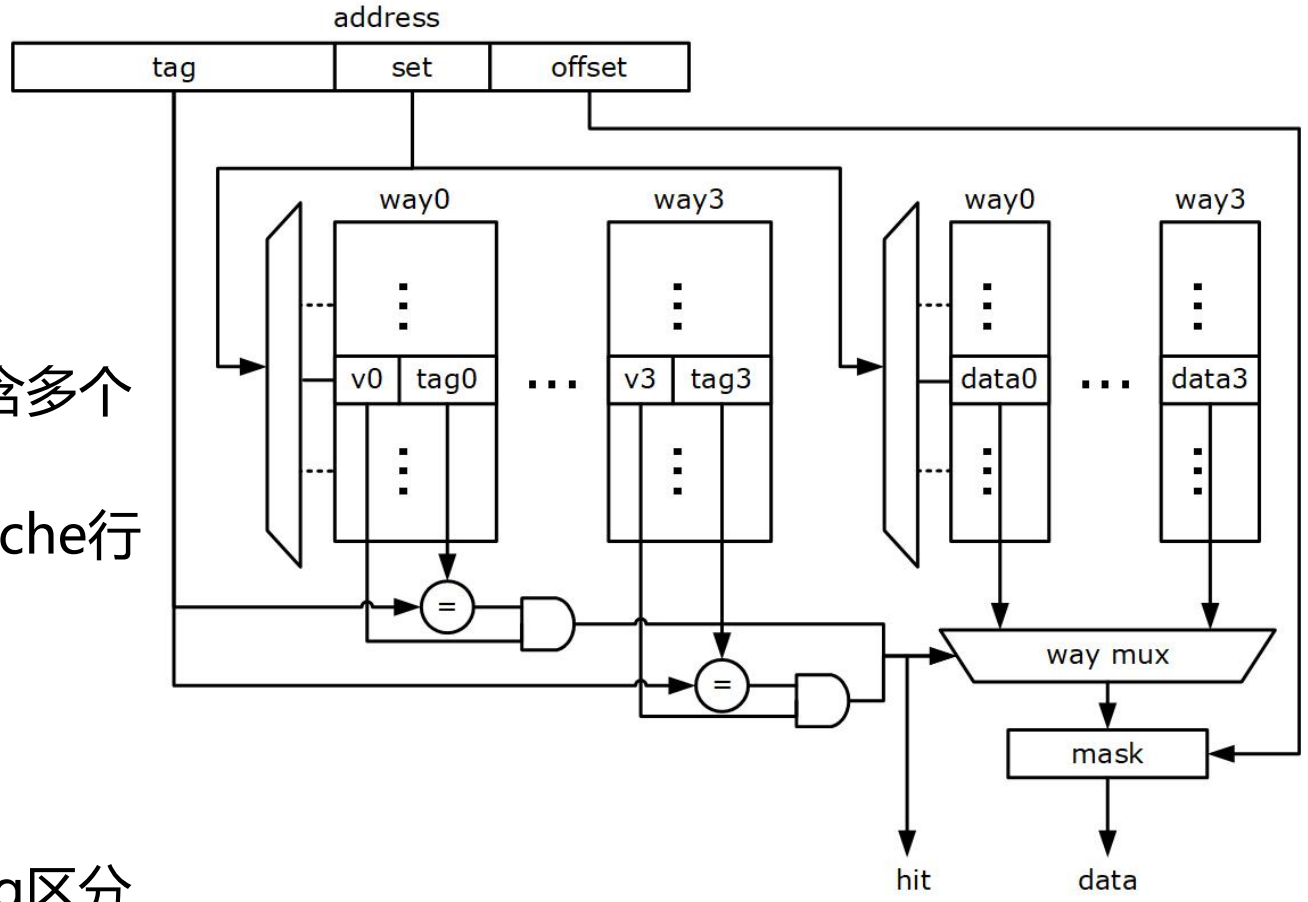
- Cache包含多个cache行，由set标记
- 每个内存块到cache行的映射固定

◆ 组相联映射

- Cache包含多个组，由set标记，每组包含多个cache行（多路-way）
- 内存块到cache组的映射固定，到组内cache行的映射不固定，靠tag区分

◆ 全相联映射

- Cache包含多个cache行
- 内存块到cache行的映射不固定，全靠tag区分



Cache简单示意图

五、Cache的写策略

University of Chinese Academy of Sciences

1. 写数据被写的数据行在cache中

a. 直写：写入cache和主存

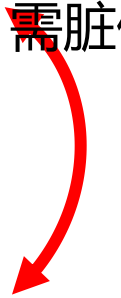
b. 写回：写入cache；cache行被替换出去时，才写入主存；需脏位标记；会产生一致性问题

2. 被写的数据行不在cache中

a. 写分配：先把内存块读入cache中，再写到该cache行中

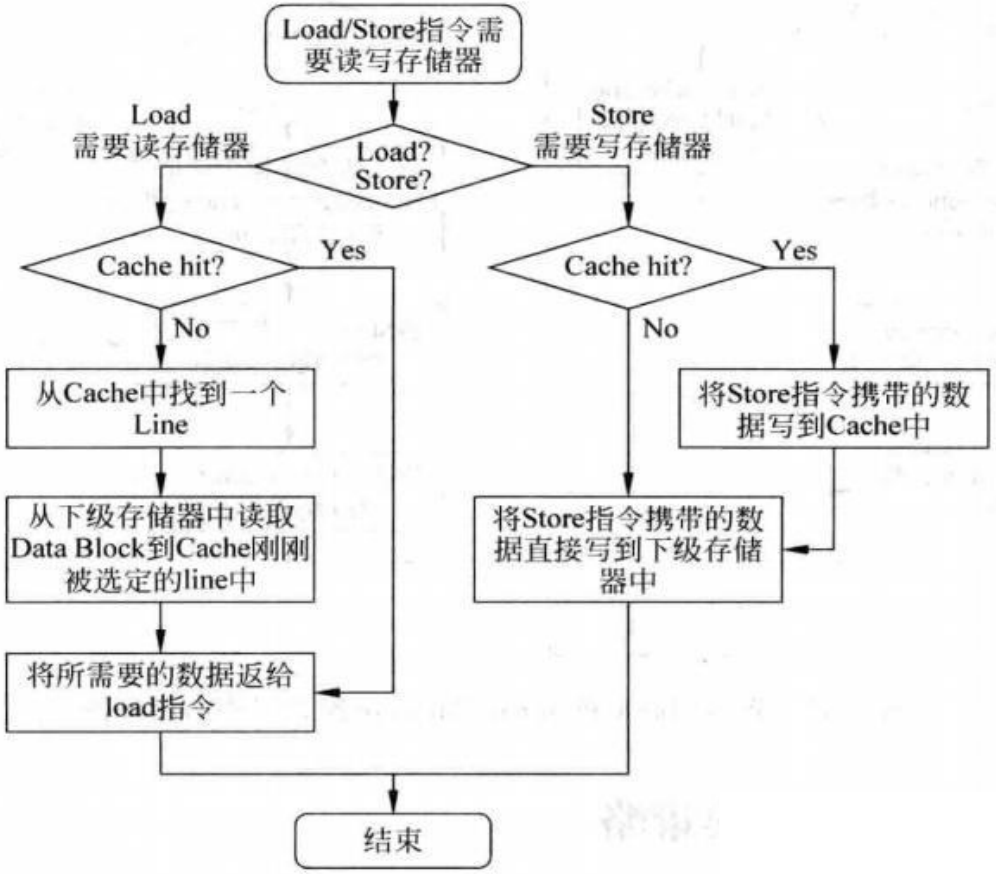
b. 写不分配：直接写入主存

直写+写不分配

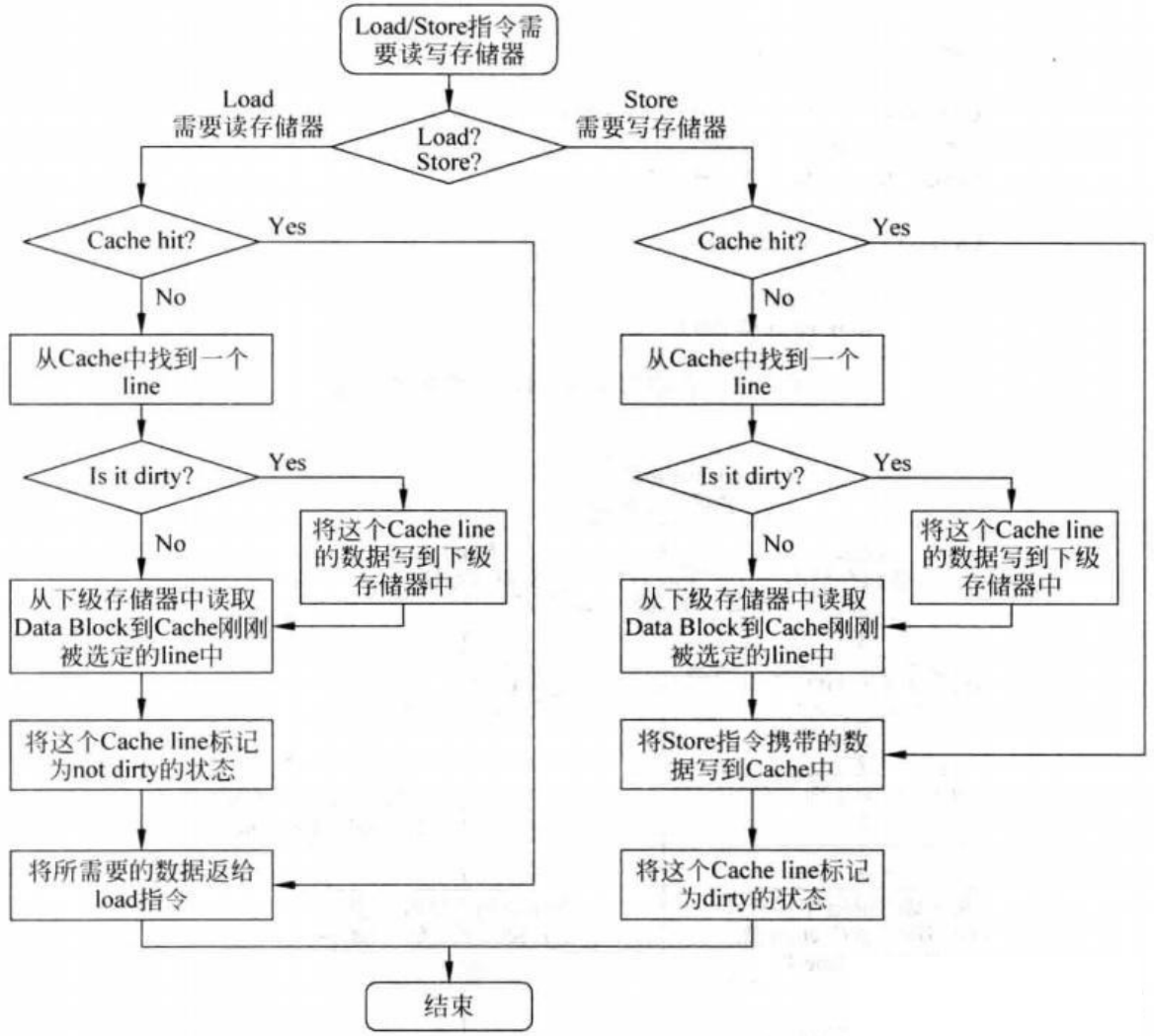


写回+写分配

五、Cache的写策略



直写+写不分配



写回+写分配

六、常见Cache的置换算法

组相联/全相联映射，当组内cache行已满，但仍需加入新的cache行时，决定替换哪路

a. 随机替换

随机替换一路；由一个内置的时钟计数器实现，需要替换时根据计数器内容决定置换哪路

b. 先入先出

先进入该组的路优先被替换；队列实现

c. 最近最少使用（LRU）

替换最近最早访问的路；每个cache行对应一个log2(路数量way)位的计数器，取值0~(way-1)，数值小的表示最近被访问的，数值最大的优先被替换

d. 最近最不经常用（LFU）

替换访问次数最小的路；计数器记录访问次数

七、其他优化

- Cache置换策略优化——降低miss率，减小面积开销
- 多级缓存
- 流水线设计
- 无阻塞缓存——MSHR（缺失状态寄存器）
- 采用多种缓存以提高缓存带宽——分成多个bank（地址中取几位标记bank）
- 合并写、路预测、关键字优先和提前重启动、编译器优化、预取、编译器控制预取 ……

口 参考《计算机体系结构—量化研究方法》

八、Cache部分实验要求

◆ 基本要求：

设计一个高速缓存，满足：

- 采用多路组相联设计；
- Tag 和 data 阵列分开，且并行访问；
- 写回+写分配的写策略；
- 实现一种cache置换算法。

◆ 进阶要求：

- 选择并实现至少1种cache优化手段，可以不在实验文档列举的优化手段中。

◆ 评估指标：

- 主要包括访问延时、带宽、吞吐量、cache命中率、硬件开销。将根据各指标表现进行综合评定。