存储器

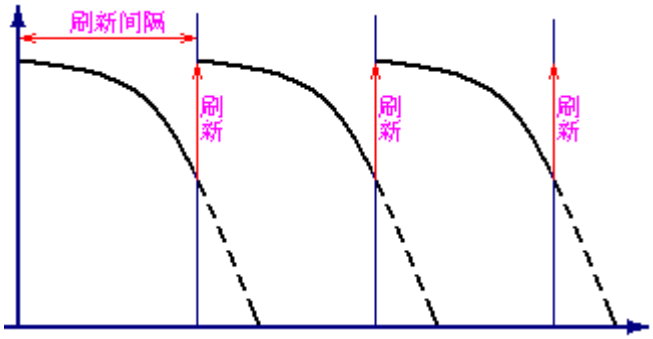
按存储介质：

按存取方式：

- 随机：半导体介质各种存储器
- 顺序：磁带
- 半顺序：磁盘，CDROM

存储器原理：

- 六管静态存储单元：存1位，但是很浪费，经常做Cache
- 动态存储单元：没有电源供电，存在的反向泄露电流会使保存的信息丢失，所以必须不断地刷新。经常被用于做主存
  - 刷新周期：连续两次对整个存储器全部刷新的间隔，取决于电平能够维持的时间。
  - 刷新时间：对整个存储器刷新一次所需要的时间。
  - 集中刷新：把一个刷新周期所有刷新集中在一起进行，存在不能进行存储器读写的“死时间”
  - 分散刷新：每个读写操作的同时进行存储器刷新，读写速度变慢
  - 异步刷新：把所有刷新平均分散在整个周期进行，控制较为复杂



- 动态存储器（DRAM）
- 二维地址译码：以二维存储单元阵列组成一个整体。
  - 设存储器阵列为$M \times N$，则所需要的“钥匙”为$M+N$，每把钥匙分别对应一行或一列的存储单元。只有两个译码器同时选中，该单元才被选中。
- 位扩展：将若干的存储单元阵列，则相同地址译码在不同二维阵列上对应的单元合并为存储字。
- 存储芯片：
  - 地址线Adr：决定可寻址的范围
  - 数据线Dat：决定数据的位数
  - 控制线：包括电源、读写控制、片选
- $1MB = 1024 \times 1024 \times 8\ bit = 2^{20} \times 8\ bit$ 数据需要20根地址线，数据线8根
- 地址线A0~A10 数据线D0~D7：$2K \times 8$ 位
- 字扩展：
  - $32KB$ 有15个地址线 8个数据线

    - $8K \times 1$ 有13个地址线 1个数据线
- 大头BigEndian：一个字或半字的高位字节位于地址低的字节，如Java
- 小头little-endian：一个字或半字的低位字节位于地址低的字节，如Windows
    - 例: 0x12345678
    - 大头：0x12 0x34 0x56 0x78
    - 小头：0x78 0x56 0x34 0x12

磁盘

- 相应时间计算：$T = QueuingDelay + Controller + Seek + Rotation + Transfer$
- 假定每个扇区为512字节，磁盘转速为5400RPM，寻道时间为12ms，数据传输速率为4MB/s，磁盘控制器开销为1ms，不考虑排队时间，则磁盘的相应时间时多少?
    - $QueuingDelay = 0$
    - $Controller = 1ms$
    - $Seek = 12ms$
    - $Rotation = 0.5/5400RPM = 5.5ms$
    - $Transfer = 0.5KB/(4MB/s) = 0.1ms$

Cache（高速缓存）

定义

- Cache是一种小容量高速缓冲存储器，由SRAM组成。
- Cache直接制作在CPU芯片内，几乎与CPU芯片一样快。
- Cache中存放一个主存块的对应单位称为“槽（Slot）”或“行（Line）”

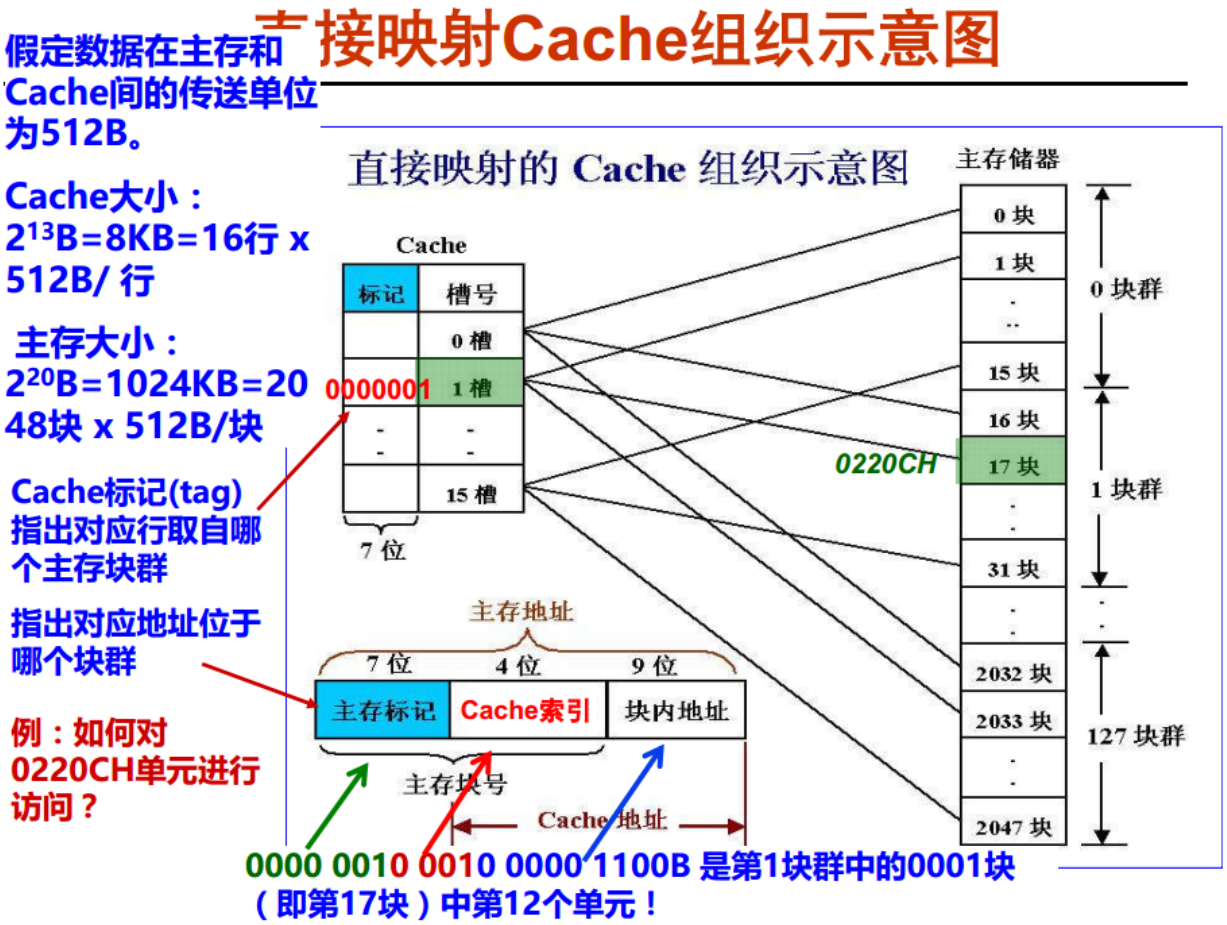
映射方式

- 直接映射（Direct）：每个主存块映射到Cache固定行。
- 全相联（Full Associate）：每个主存块映射到Cache的任一行。
- 组相联（Set Associate）：每个主存块映射到Cache固定组的任一行。

直接映射（1-Way组相联）

- 假设主存大小为1MB，Cache有16个槽。
    - 主存即$2048Block \times 512B/Block$，块内地址为9位（512B）。
    - Cache的大小为$2^{13}B = 8KB$，Cache槽号为4位（16槽）。
    - 主存块数/Cache块数=主存大小/Cache大小=$1MB/8KB = 2^7 = 128$块群，则主存标记为7位。
- 特点:
    - 容易实现，命中时间短。
    - 无需考虑淘汰（替换）问题。

- 但不够灵活，Cache存储空间得不到充分利用，命中率低。例如，需将主存第0块与第16块同时复制到Cache中时，由于它们都只能复制到Cache第0行，即使Cache其它行空闲，也有一个主存块不能写入Cache。这样就会产生频繁的 Cache装入。

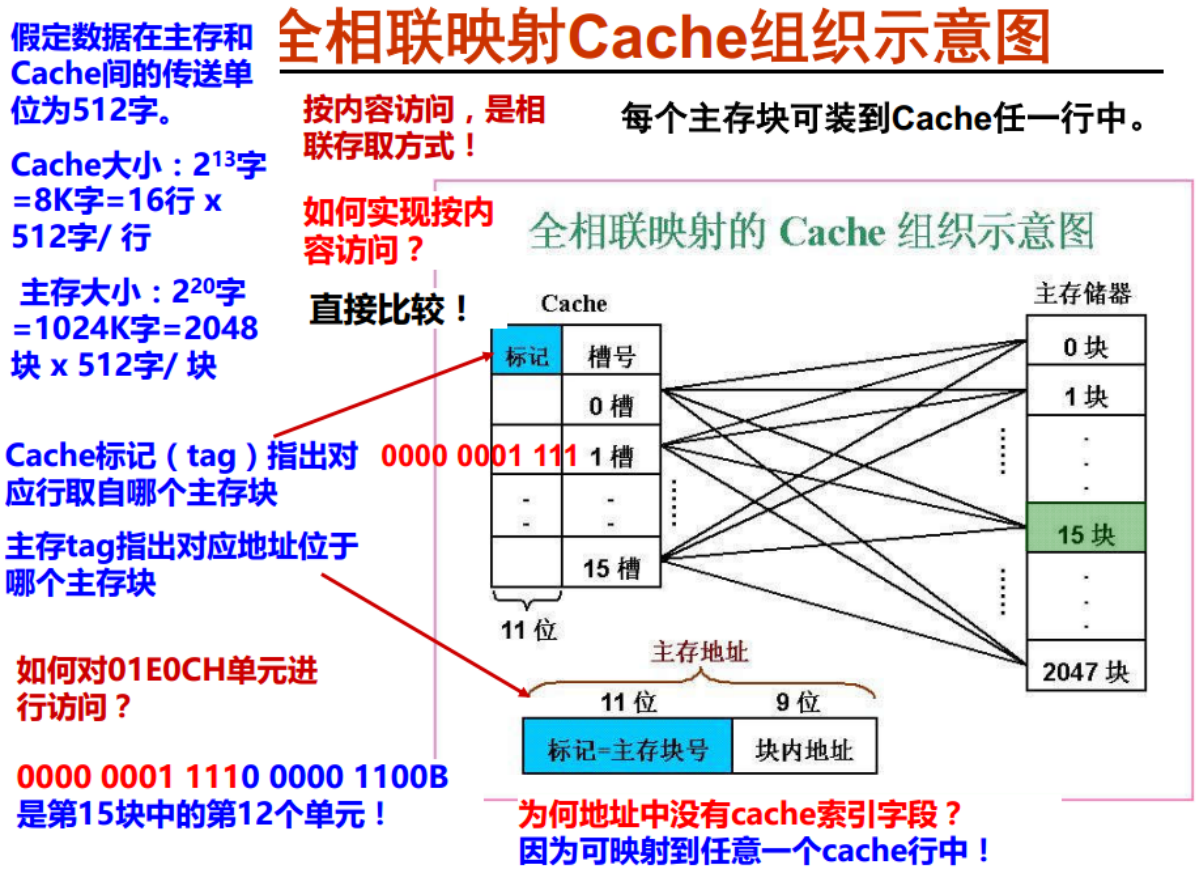


有效位

- 有效位1位，1表示信息有效，0表示信息无效。

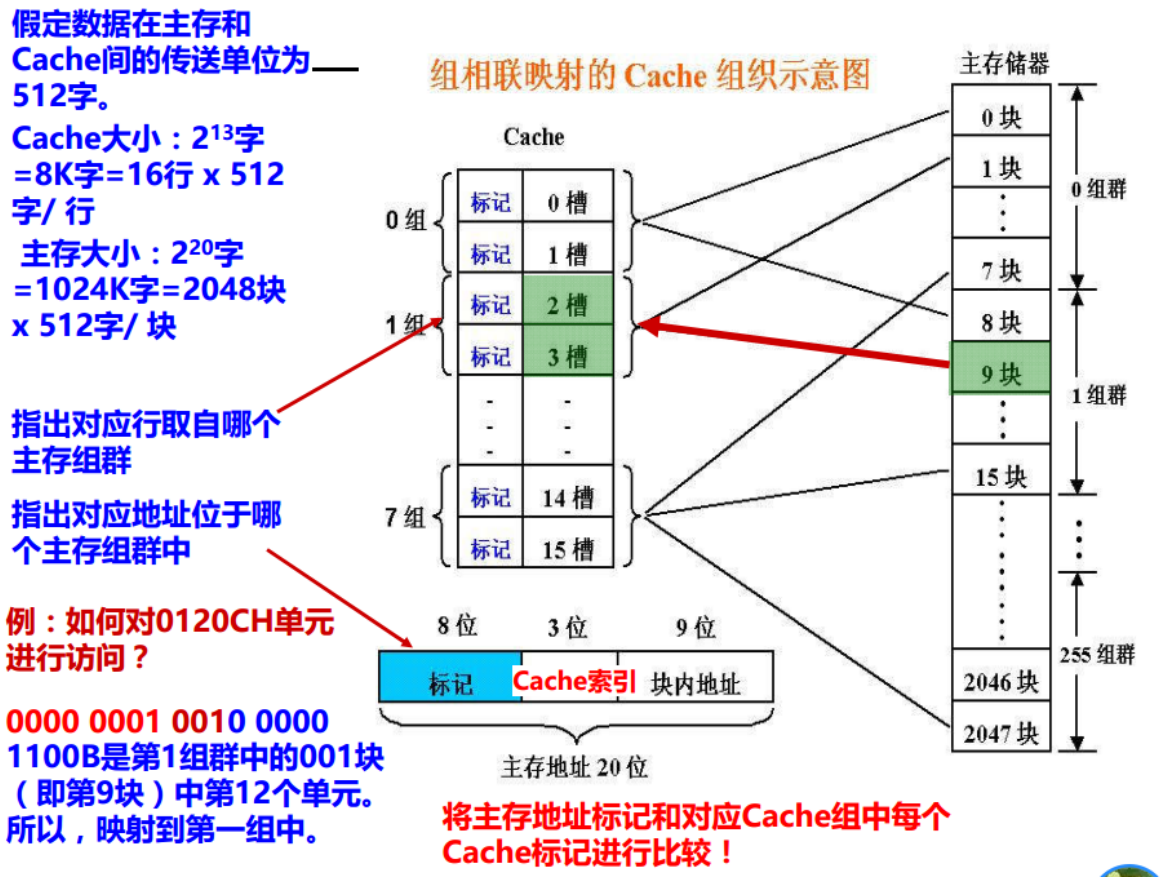
全相联（BlockNums-Way组相联）

- 每个主存块可以装到Cache中的
- *tag*标记需要标记对应的主存块号
- 缺点：命中时间长



组相联

- 组间模映射，组内全映射
- Cache组号 = 主存块号 mod Cache组数
- *tag*应指出是来自于哪个主存组群
- 例：假设Cache划分为：8K字=8组×2行/组×512字/行。则主存第100块被映射到100 mod 8 = 4，即第4组的任意行中。



容量计算

【例1】主存与Cache之间直接映射，块大小为16B，Cache容量为64KB，主存地址32位。

【解】

- 主存地址划分成3部分，块内地址4位（16B），索引（Cache的行号）占12位，剩下的16位为tag
- $tag = 32bits - 4bits - 12bits = 16bits$
- 由于是tag是对应的是取自于哪个主存块群，因此可以理解为，一共有$2^{32}(4GB)/2^{16}(64KB)$个块群，因此需要16位来进行编码。
- Cache行数$BlockNum = 64KB/16B = 4K$块，每一行具有1位有效位，加上16位的tag，以及64KB的数据区，每字节8位。
- $Total = (1bits + 16bits) \times 4K + 64KB \times 8bits/B$

【例2】主存与Cache之间全相联映射，块大小为16B，Cache容量为64KB，主存地址32位。

【解】

- tag需要指出是位于哪一块，因此需要$2^{32}/(16B/block) = 2^{28}$，即28位
- tag也可以通过$log\frac{4GB}{16}/1$ 来计算。
- $Total = (28bits + 1bits) \times 4K + 64KB$

【例3】主存与Cache之间4-Way组相联映射，块大小为16B，Cache容量为64KB，主存地址32位。

【解】

- tag需要指出位于哪一组（类似于直接映射），$tag = 32 - 4 - (12 - 2) = 18$
- 可以理解为：主存一共2^{28}块，Cache共2^{12}块，每4块一组，且组内等价，相当于主存映射到Cache的比例变为$2^{28}/2^{10} = 2^{18}$，即需要18位

- $Total = 64KB + (18 + 1) \times 4K/8$

【整理】TAG的计算公式为$log\frac{主存大小}{块大小 \times Cache组数}$