第三节 高速缓冲存储器

一、概述

1.为什么用 Cache

避免 CPU“空等”现象等。因为主存容量大、但是速度很低，CPU 速度很快，所以为了避免这种速度差过大的情况，引入了缓存（Cache）。Cache 是一种容量小，速度高的存储器，用静态 RAM 做成。

要保证系统的性能，必须保证 CPU 要访问的指令等都能在 Cache 中找到。要依靠程序访问的局部性原理。

程序访问的局部性原理分两部分，第一部分是时间的局部性，指当前正在使用的指令和数据在不久的将来还会被使用到，应该放到 Cache 中；第二部分是空间的局部性，指的是当前正在使用的指令或数据在不久的将来相邻的指令和数据可能会被用到。

所以将**当前使用**的指令和数据以及**相邻的**指令和数据放在 Cache 中。

2.Cache 的工作原理

（1）主存与缓存的编址

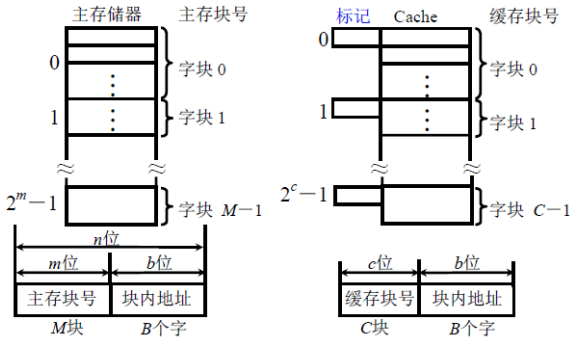

图 3.1 Cache 的主存和缓存编址

如图 3.1 所示，左侧是主存储器结构，右侧是 Cache 的结构。我们把主存储器和 Cache 分成大小相等的块，主存中有 M 块，Cache 中有 C 块，C 远远小于 M。如果把主存或 Cache 分成若干块，CPU 给出内存的地址就可以分为两部分，一部分为块内偏移地址，位数决定块的大小，一部分为块号；Cache 也分为块内地址和块号，但是一般情况下 Cache 块号意义不大。

然后我们来看以下块内部分。内存块和 Cache 块大小相同，所以块内地址位数完全相同，另外一块在内存和 cache 之间进行传送是整体传送，块内字节顺序不会发生任何变化，所以内存块内地址和 Cache 块内地址完全相同。Cache 有一个标记，它标记了主存块和 cache 块的对应关系。如果我们把主存中一个块调入 cache 中，那么可以将主存块号写到标记当中，将来 CPU 给出一个内存地址，希望在 Cache 访问到这个数据，它首先要确定该块是否送到 Cache 中，就拿主存块号和 Cache 中标记进行对比。

主存和 Cache 在 Cache-主存模块中**是按块存储、按块传输，块大小相同，块内地址相同**。

（2）命中与未命中

假设缓存共有 C 块，主存共有 M 块，且 M 远远大于 C。

命中就是主存块**调入**缓存，主存块与缓存块**建立**了对应关系，用**标记记录**与某缓存块**建立**了对应关系的**主存块号**。

如果主存块未调入缓存，主存块与缓存块未建立对应关系，称为**未命中**。

（3）Cache 的命中率

CPU 与访问的信息在 Cache 中的**比率**称为 Cache 的命中率。命中率与 Cache 的**容量**和**块长**有关。容量太小，保存在 Cache 的块数目少，降低命中率；块太大，没有充分运用程序的局部性原理，一次性存相邻内容太多，利用的信息有可能会很少，大部分内容被浪费，降低命中率。一般每块可取 4~8 个字。

块长取一个存取周期内从主存调出的信息长度。

（4）Cache-主存系统的效率

效率 e 与命中率有关，如式（3.1）所示。

$$e=\frac{\text{访问}Cache\text{的时间}}{\text{平均访问时间}}\times 100\% \tag{3.1}$$

设 Cache 的命中率为 h，访问 Cache 的时间为 tc，访问主存的时间为 tm，则

$$e=\frac{t_c}{h\times t_c+(1-h)\times t_m}\times 100\%$$

3.Cache 的基本结构

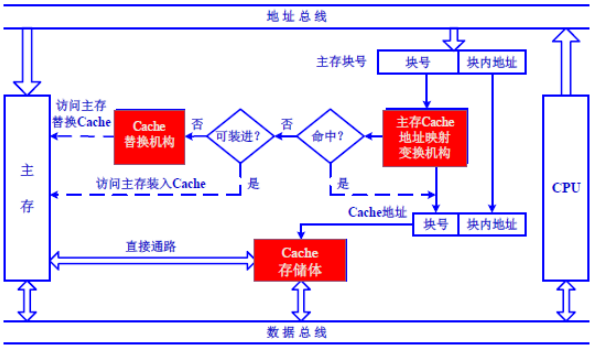

图 3.2 Cache 的基本结构

如图所示，左侧是主存，右侧是 CPU，中间就是我们的 Cache 主体。CPU 要访问内存的话，会给出地址，这个地址分为块号和块内地址，由于前文所述，主存和 Cache 的块内地址完全相同，所以直接把块内地址往下传输，而块号在变换机构中确认是否发生命中，若命中则给出当前内存块保存在 Cache 中的块号，若没有命中，则判断 Cache 空间是否可以装进，可以的话则装入，若没有，则启用 Cache 替换机构，由其根据替换算法决定 Cache 中哪个块写入主存或者作废，然后把要用的这个块写入 Cache 存储体。

在这里面有一个地址的映射和变换，这其实是两个功能。第一个功能是映射，实际上给出一个规则，主存当中的一个块如果要放到 Cache 中，可以被放到 Cache 中那些或者哪个块，这就是映射规则。第一个功能是变换，就是把主存的块号转换成相应的 Cache 块号，把主存地址转换成 Cache 地址。

另外，在主存和 Cache 存储体有一个直接通路，完成了主存和 Cache 的信息交换，另外有些计算机为了提高速度，若检测到发生不命中的情况，那么通过下面数据线先把主存中 CPU 需要的数据送到 CPU 中，同时这个块在 Cache 和主存间进行传送。

基本结构中最核心的三大部分是：主存 **Cache 地址映射变换机构**，**Cache 替换机构**和**Cache 存储体**。

4.Cache 的读写操作

（1）读操作

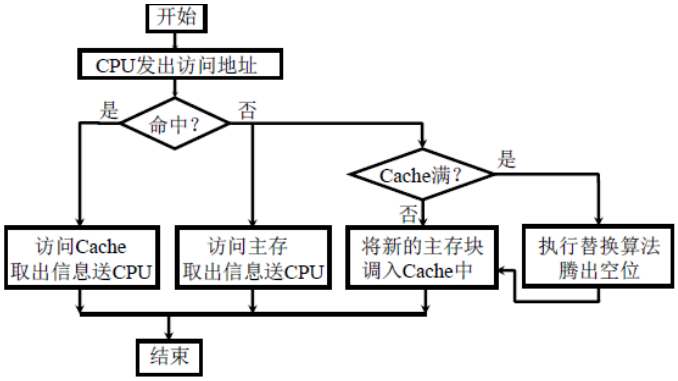

图 3.3 Cache 的读操作

CPU 发出访问地址，由 Cache 的映射变换机构判断是否命中：若命中，访问 Cache 取出信息送 CPU；若没有命中，有两条路都要走，一条访问主存，取出信息送给 CPU，另一条判断 Cache 是否已满（其实是**判断允许存放的那些 Cache 块是否满了**），若没有则将新的主存块调入 Cache 中，否则执行替换算法腾出空位，将新的主存块调入 Cache 中。

（2）写操作——**Cache 和主存的一致性**

读操作不会修改 Cache 和内存的关系，但是写操作可能会使 Cache 和主存不一致。

写有两种规则：

①写直达/写通过法（Write-through）

写操作时数据既写入 Cache 又写入主存。写操作时间就是**访问主存的时间**，Cache 块退出时，不需要对主存执行写操作，更新策略比较容易实现。**时刻保证主存和 Cache 内容一致**。缺点是可能会造成 CPU 对同一个内存单元反复的写。

②写回法（Write-back）

写回法允许一段时间内主存和 Cache 内容不一致。写操作时只把数据写入 Cache 而不写入主存，Cache 数据被替换出去时才写回主存。写操作时间就是**访问 Cache 的时间**。Cache 块退出时，被替换的块需写回主存，增加了 Cache 的复杂性。

5.Cache 的改进

（1）增加 Cache 的级数

片载（片内）Cache：直接在 CPU 内做 Cache，甚至了两级 Cache。

片外 Cache：片外做大容量 Cache。

现在计算机至少有 3 级 Cache。

（2）统一缓存和分立缓存

可以分为指令 Cche、数据 Cache。与指令执行的控制方式有关，是否流水。

例如：Pentium 是 8K 指令 Cache，8K 数据 Cache；PowerPC620 是 32K 指令 Cache，32K 数据 Cache。

二、Cache-主存的地址映射（重点和难点）

1.直接映射

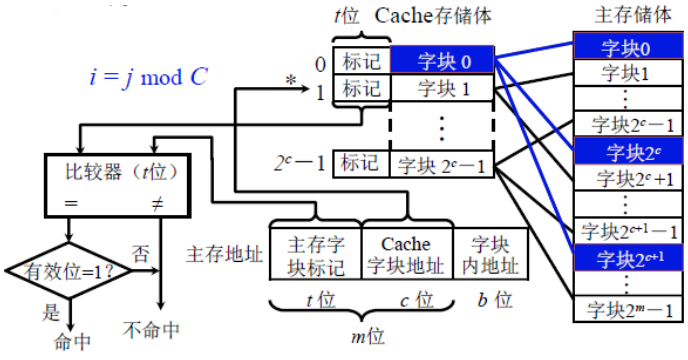

图 3.4 直接映射

所谓直接映射，就是主存当中任意一个给定的块只能映射到某一个指定的 Cache 块中。具体做法如下：以 Cache 块作为度量来衡量主存，以 Cache 块分隔主存分为多个区，每个区和 Cache 块的长度一致，可以直接进行编号，每个字块一一对应，不可修改。由于 Cache 里面的第 0 块装载可能是主存任何一个区的第 0 块，所以我们要把主存中哪个区写到标记里面，也就是主存字块标记（t 位），然后进入比较器判断是否在 cache 中有，若有，则就根据 Cache 字块地址直接找到 Cache 中的这个块。

这种结构 Cache 块利用率可能会非常低，冲突概率非常大。

每个缓存块 i 可以和**若干个主存块**对应，**每个主存块 j** 只能和**一个缓存块**对应。

2.全相联映射

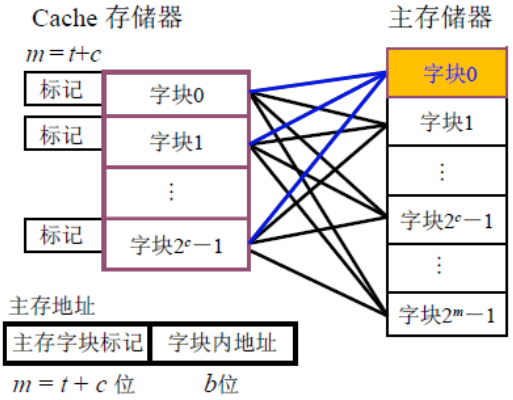

图 3.5 全相联映射

它的最大特点是**主存**中的**任一块**可以映射到**缓存**中的**任一块**。Cache 利用率提高，只要 Cache 中还有空闲，主存中就可以直接调入。

但是在这里如果我们给出一个主存地址，CPU 要访问这个地址，首先要确认地址所在的块是否被调入到 Cache 中，因为是全相联，这个块被调入的话可能在任何一个块，所以这个主存字块标记要和 Cache 所有的块比较，这个比较同时进行，电路较复杂，速度较慢。

第二个问题，参加比较的位数也会比较长。由于任何一个块都可以放到 Cache 中，所以 t+c 都需要放在 Cache 存储器的标记中，所以要求比较器比较长。

3.组相联映射

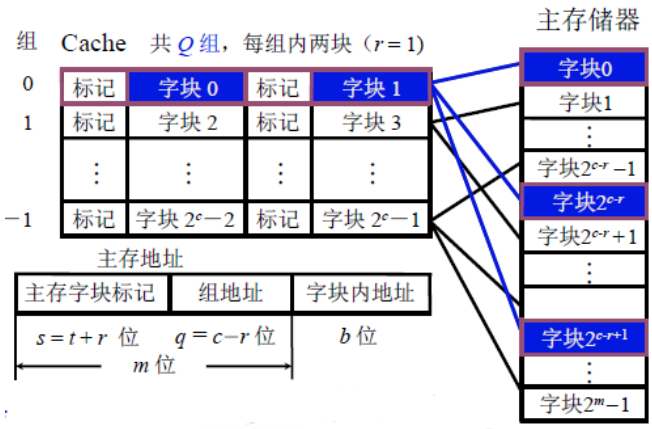

图 3.6 组相联映射

它是全相联映射和直接映射的这种。它是先把 Cache 分为 Q 组，每组有 R 块，并且有以下联系：i=j mod Q。其中，i 为缓存的组好，j 为主存的块号。**某一主存块按模 Q 将其映射到缓存的第 i 组内的任一块**，如图 3.6 所示。**Cache 分为了多少组，主存中一个组就包含多少块**。

映像的时候，每个区的第 0 块可以放到 Cache 中第 0 组任何一个位置，也就是说，主存当中每个区内的块在区里的编号直接决定了放入到 Cache 当中可以在哪个组。

这种方法和直接相联相比，一个块有多个位置，即使其中一个位置被占用，内存块也可以被调入到另外一个块的位置。

和全相联相比，如果去找某一个内存块是不是被调入到 Cache 中，只需要确定它在某个区的块标号就可以，给出标号后找到给定的组，比较的时候给定组的标号和区号进行比较即可，不需要和每一个 Cache 块进行标记。

某一主存块按模 Q 将其映射到缓存的第 i 组内的任一块这句话中，**某一主存块按模 Q 将其映射到缓存的第 i 组**体现出来了直接映射的特点，而**任一块**体现出来了全相联映射。

这种方法电路相对简单，进行比较的时候每一组需要的比较器少，速度比较快，Cache 了利用率较高。这种方法是现代计算机 Cache-主存映射常用方法。

如果组相联映射 Cache 只有一个组，那么组相联映射就变为了全相联映射；若 Cache 共 Q 组，每个组内一块，那就变成了直接映射。

这三种映射方法在缓存-主存映射中的利用方式不同。靠近 CPU 的 Cache 层次要求高速度，在这些层次采用直接映射或者路数比较少的组相连，中间的层次可以采用组相连方式，例如两路组相连等，距离 CPU 最远的采用全相联映射方式，距离越远对速度强调越低，并且对 Cache 利用率的强调越高。

三、替换算法

1.先进先出（FIFO）算法。FIFO 算法选择最早调入 Cache 的字块进行替换，不需要记录各字块的使用情况，比较容易实现，开销小，但没有根据访存的局部性原理，故不能提高 Cache 的命中率。因为最早调入的信息可能以后还会用到或者经常用到。

2.近期最少使用（LRU）算法。比较好利用了访存局部性原理，替换出近期用得最少的字块。它需要随时记录 Cache 中各字块的使用情况，以便确定哪个字块是近期最少使用的字块。它实际是一种推测方法，比较复杂，一般采用简化方法，只记录每个块最近一次使用的时间。LRU 算法的平均命中率比 FIFO 高。

3.随机法。随机的确定被替换的块，比较简单，可采用一个随机数产生器产生一个随机的被替换的块，但它也没有根据访存的局部性原理，故不能提高 Cache 命中率。

小结

直接映射某一主存块只能固定映射到某一缓存块，不灵活，利用率低，速度快；全相联某一主存块能映射到任一缓存块，成本高，速度慢，Cache 利用率高；组相连某一主存块只能映射到某一缓存组中的任一块，速度较快，Cache 利用率较高。

第四节 辅助存储器（非重点）

一、概述

1.特点：不直接与 CPU 交换信息。

2.磁表面存储器的技术指标：

（1）记录密度：道密度 D_t，位密度 D_b。

（2）存储容量：C=n×k×s，n 表示盘面数，k 表示一个盘面上的磁道数，s 表示一个磁道的二进制信息。

（3）平均寻址时间：寻道时间+等待时间

辅存的速度：寻址时间、磁头读写时间。

（4）数据传输率：Dr=Dv×V。Dv 为位密度，V 为旋转速度。

（5）误码率：出错信息位数与读出信息的总位数之比。

二、磁记录原理

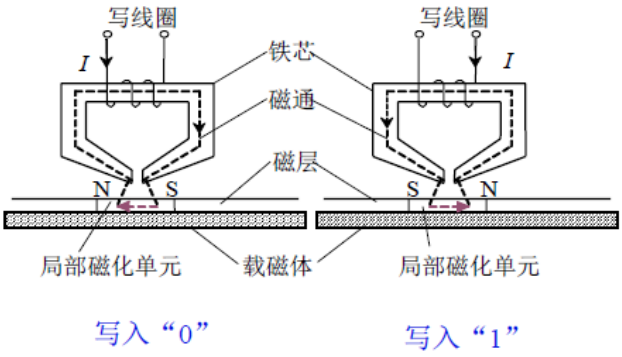

图 4.1 磁记录“写”操作

利用磁化方向不同来区分“0”和“1”。

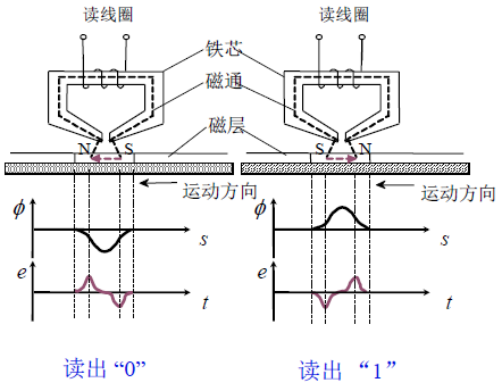

图 4.2 磁记录“读”操作

根据电势变化判断是“0”还是“1”。

三、硬磁盘存储器

1.分类：

可以从以下角度分：

（1）固定磁头（磁盘旋转，需要磁头数量庞大，速度快）和移动磁头（磁盘固定，需要磁头数量少，但是速度相对较慢）。

（2）可换盘和固定盘。

2.硬盘存储器结构

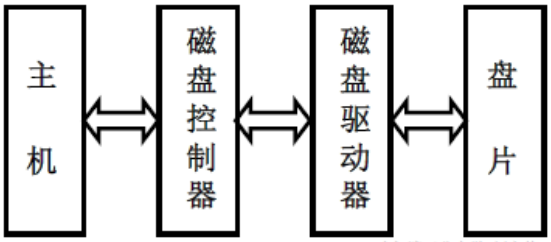

图 4.3 硬磁盘存储器结构

（1）磁盘驱动器（不做介绍）

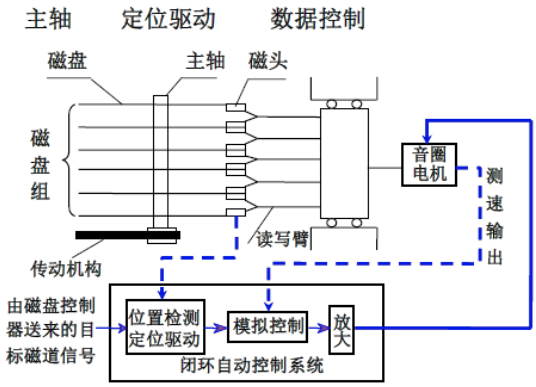

图 4.4 磁盘驱动器

（2）磁盘控制器

功能：

•接收主机发来的命令，转换成磁盘驱动器的控制命令；

•实现主机和驱动器之间的数据格式转换；

•控制磁盘驱动器读写。

磁盘控制器是主机与磁盘驱动器之间的**接口**，对主机通过总线来进行操作，对硬盘（设备）直接操作。

（3）盘片：由硬质铝合金材料制成。

四、软磁盘存储器

1.概述

	硬盘	软盘
速度	高	低
磁头	固定、活动、浮动	活动，接触盘片
盘片	固定盘，盘组大部分不可换	可换盘片
价格	高	低
环境	苛刻	较不苛刻

五、光盘存储器

1.概述

采用光存储技术：**利用激光写入和读出**。

第一代光存储技术：采用非磁性介质，不可擦写。

第二代光存储技术：采用磁性介质，可擦写。

2.光盘的存储原理

只读型和只写一次型：热作用（物理或化学变化）

可擦写光盘：热磁效应