# 第四章 存储体系

## 4.1

**存储器的主要性能**：速度、容量、价格

**主要目标**：高速度，低价格，大容量。

——————存储体的字长（字节或位）

——————每个存储体的字数

——————并行工作的存储体的个数。

**速度**：访问时间（）、贮存周期（）、频宽（）

：存储器从接到访存申请，到数据被读到数据总线的时间，它是启动一个访存读操作后，必须等待的时间。

：连续启动一个存储体所需的时间，即存储器进行一次存/取所需要要的时间，一般比大。

：每秒传输的位数或字节数。

单体存贮器：

多体存贮器：

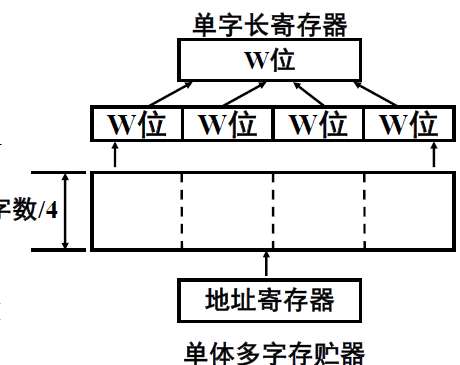
**容量**

**价格**

位价格

**并行访问存储器（感觉复习PPT上有问题）**

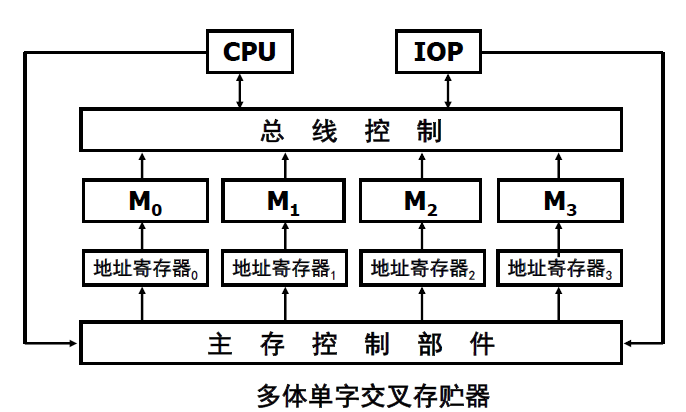
单体多字：



把字位的存储器变为了字位的存储器。

缺点：需要位数足够多的寄存器；多次访问总线

多体单字：



将地址码分为两个部分，一部分作为存储器地址，另一部分负责选择数据。

缺点：访问冲突大（取指、读操作数、写数据、读写）

高位交叉：用地址码的高位部分区分存储体号，主要用于扩大存储器容量

低位交叉：用地址码的低位部分区分存储体号，主要用于提高存储器访存速度

**存储体系的定义：**两个或两个以上速度、容量和价格各不相同的存储器用硬件、软件或软件与硬件相结合的方法连接起来成为一个存储系统（对应用程序员透明。速度、价格、容量都接近最好的那个存储器）

**主存-辅存存储层次（二级存储层次）**

又称虚拟存贮器。由主存储器和磁盘存储器构成，对应用程序员透明，系统程序员可见。主要用于扩大存储容量，弥补主存容量不足

**Cache-主存存贮层次（耳机存储层次）**

由主存和Cache组成，对系统程序员和应用程序员都透明。用于提高存储器速度，弥补主存的速度不足。

**程序局部性原理：**

时间局部性：最近的未来要用到的信息可能就是当前正在使用的信息——这是由程序的循环造成的。

空间局部性：最近的未来要用到的信息可能就是当前信息的相邻信息——这是由程序的顺序执行造成的。

**存储系统的单位容量平均价格：**

**存储系统的速度**：

表示方法：访存时间、存取周期、存储周期、存取时间、读出时间等

命中率：产生的逻辑地址能在中访问到的概率。

若逻辑地址流在中范文到的次数位，在中访问次数位，则：

等效访问时间

假设访问和访问时同时启动的

假设访问和访问不是同时启动

访问效率，越接近1越好。

**采用预取技术提高命中率：**

程序局部性原理

————原来的命中率

————数据块的大小数据重复使用次数

**提高存储系统速度的途径：**

提高命中率

两个存储器的速度不要相差太大

加快内部地址映像及变换

## 4.2

**虚拟存储器工作原理：**

虚拟存储器是一个大容量存储器的逻辑模型，指的是主存-辅存层次

**（复习PPT只提了页式管理的原理，段式管理没有提及）**

把主存储器和虚拟存储器分为固定大小的页，主存储器的页称为实页，虚拟存储器中的页称为虚页。

一个主存地址由两部分组成，实页号和页内偏移：



一个虚地址由三部分组成，用户号，虚页号和页内偏移.



**页表的内部地址变换：**

根据用户号查找相应的页表，根据页表的映射关系找到虚页号对应的实页号，将实页号和页内偏移直接拼接得到主存实地址。

**页表的外部地址变换：**

首先查外页表得到磁盘存储器的实地址。把磁盘存储器的实地址和主存储器实页号送入输入输出处理及机。把要访问数据所在的一整页都从磁盘存储器调入到主存储器中。

**三种地址空间：**

程序地址和空间（虚拟地址空间）、主存地址空间、辅存地址空间。

**地址映像：**把虚拟地址空间映像到主存地址空间

**地址变换：**在程序运行时，把虚地址变换成主存实地址。

**三种虚拟存储器：**页式虚拟存储器、段式虚拟存储器、段页式虚拟存储器

**造成虚拟存储器速度降低的主要原因：**

要访问主存储器须先查段表或页表

可能需要多级页表：

————页面的大小

————虚拟存储空间的大小

————一个页表存储字的大小

**页面替换算法的使用场合：**

虚拟存储器中，主存页面的替换，一般用软件实现

Cache块替换一般用硬件实现

虚拟存储器的快慢表中，块表存储字的替换，用硬件实现

虚拟存储器中，用户基地址寄存器的替换，用硬件实现

在有些虚拟存储器中目录表的替换。

**页面替换算法：**随机算法、先进先出算法（FIFO）、最近最少使用算法（LFU），最久没有使用算法（LRU），最优替换算法（OPT）

**提高等效访问速度的方法：**

**目录表（相联目录法）：**用小容量高速存储器存放页表，和页表法有区别。

**快慢表：**

快表：用快速硬件构成小容量的“相联目录表”，存放当前正在使用的虚实地址映像关系

慢表：将原先存放全部映像关系的页表。用软件实现。

快表时慢表中很小的一部分副本

快表与慢表页构成了一个两级存储系统

**散列函数：**

目的：把相联访问变成地址访问，从而加大快表容量。

采用散列变换实现快表按地址访问。

避免散列冲突：采用相等比较器

地址变换过程：相等比较与访问存储器同时进行。

## 4.3

**地址映像：**

把存放到主存中的程序按照某种规则装入到Cache中，并建立主存地址与Cache地址之间的对应关系

**地址变换：**

当程序已经装入到Cache之后，在实际运行过程中，把主存地址变换到Cache地址。

**在选取地址映像方法要考虑的主要因素：**

地址映像和变换硬件的速度是否高，价格是否低，实现是否容易

Cache空间的利用率是否高

块冲突概率是否低。

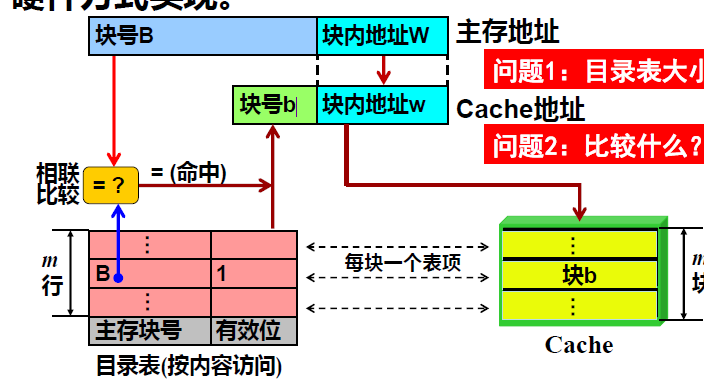
**全相联映像及其变换：**

映像规则：主存中任意一块都可以映像到Cache中的任意一块。

地址变换：采用相联存储器构成的目录表，以硬件方式实现

优点：块冲突概率最低；Cache空间利用率最高

缺点：所需容量的相联存储器代价较高；Cache容量很大，速度难以提高



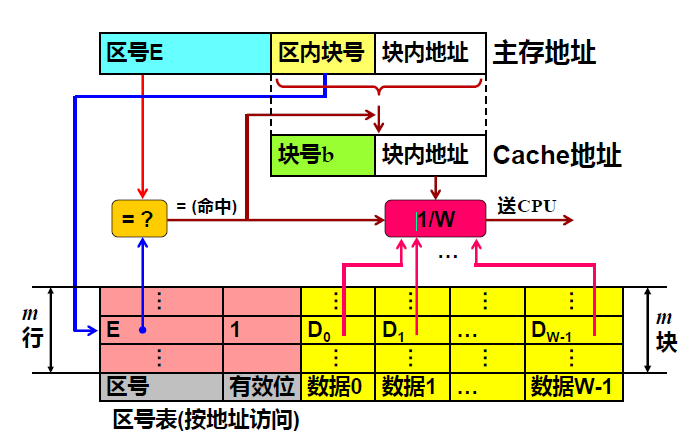
**直接映像及其变换：**

映像规则：主存中一块只能映像到Cache的一个特定快中。

计算公式为：

优点：所需硬件简单，成本较低。访问Cache可与访问区号表、比较区号等操作同时进行，节省变换时间。

缺点：块冲突概率很高；Cache空间利用率很低。



**组相联映像及其变换**

映像规则（位选择映像算法）：

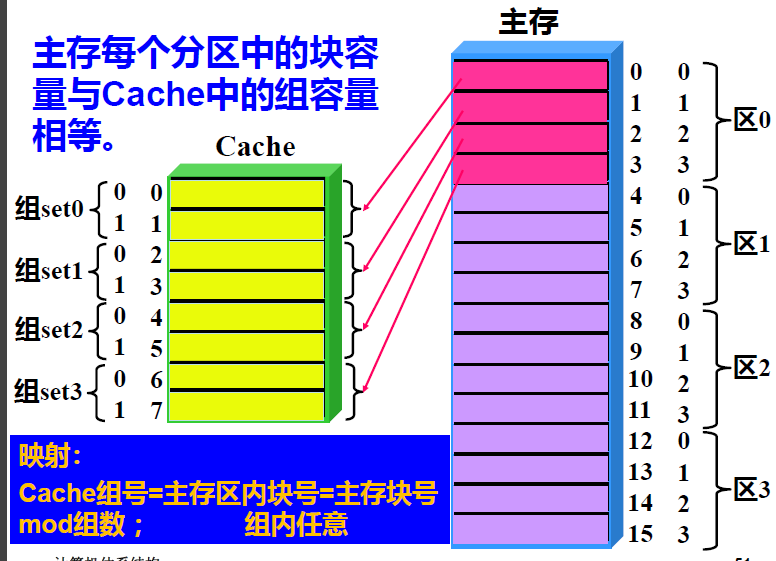
主存和Cache按照相同大小划分成块；

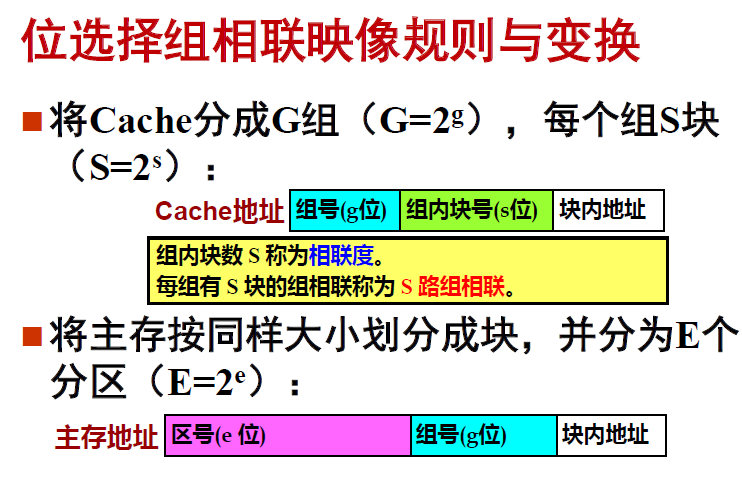
Cache划分成大小相同的组，主存按照Cache组容量分区。

主存到Cache，组间采用直接映像，组内全相联。

优点：块冲突概率比直接映像低得多。Cache空间利用率也比直接映像提高。实现成本比全相联映像要低的多。性能接近于全相联映像。

缺点：实现难度和造价要比直接映像方式高。





**Cache替换算法：**随机算法、FIFO算法、LRU算法。

**Cache替换算法实现：全部用硬件**

堆栈法：使用硬堆栈

比较对法：使用逻辑电路、触发器等

**Cache替换算法的使用时机：**

当所要房访问的块不在Cache中时，则发生块失效。

当所能装入的Cache块都被装满时，则出现块冲突，必须进行块替换。

**直接映像方式实际上不需要替换算法**

**Cache一致性问题：**

仅讨论单处理机、单存储器。

造成Cache与主存的不一致的原因：

CPU更新（写）了Cache而未更新主存

I/O更新了主存而未更新Cache

**Cache更新算法：**

写直达法：CPU执行写操作时，利用直接通路，把数据同时写入Cache和主存

写回法：页成为抵触修改法，CPU只写入Cache，并将Cache块设置修改位，仅当替换时，才把修改过的Cache块写回到主存。

**写Cache的两种方法：**

不按写分配法：在写Cache不命中时，只把索要写的字写入主存

按写分配法：在写Cache不命中时，还把一个块从主存读入Cache.

目前，在写回法中采用按写分配法，在写直达法中采用不按写分配法。

**Cache的预取算法**

按需取：在出现Cache不命中时，把一个块渠道Cache中来

恒预取：无论Cache是否命中，都把下一块取到Cache中

不命中预取：当Cache不命中，把本块和下一块一起渠道Cache中。

主要考虑因素：

命中率的提高

Cache与主存之间通信量的增加