存储系统

吕熠娜
厦门大学信息学院

存储系统

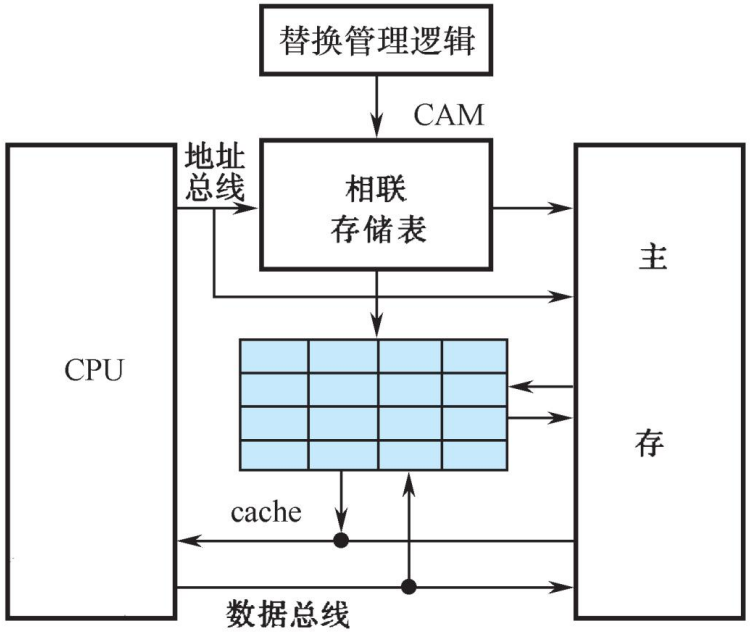
- 随机读写存储器
- 只读存储器和闪速存储器
- Cache存储器
- 虚拟存储器
- 存储保护

高速缓冲存储器Cache

- Cache是为了解决CPU与主存速度不匹配而采取的技术，其有效性利用程序的局部性原理。
  - 程序的局部性有两个方面的含义：时间局部性和空间局部性。
    - 时间局部性是指如果一个存储单元被访问，则可能该单元会很快被再次访问。这是因为程序存在着循环。
    - 空间局部性是指如果一个存储单元被访问，则该单元邻近的单元也可能很快被访问。这是因为程序中大部分指令是顺序存储、顺序执行的，数据也是以向量、数组、树、表等形式簇聚地存储在一起的。
  - 高速缓冲技术就是利用程序的局部性原理，把程序中正在使用的部分存放在一个高速的容量较小的Cache中，使CPU的访存操作大多数针对Cache进行，从而使程序的执行速度大大提高。

高速缓冲存储器Cache

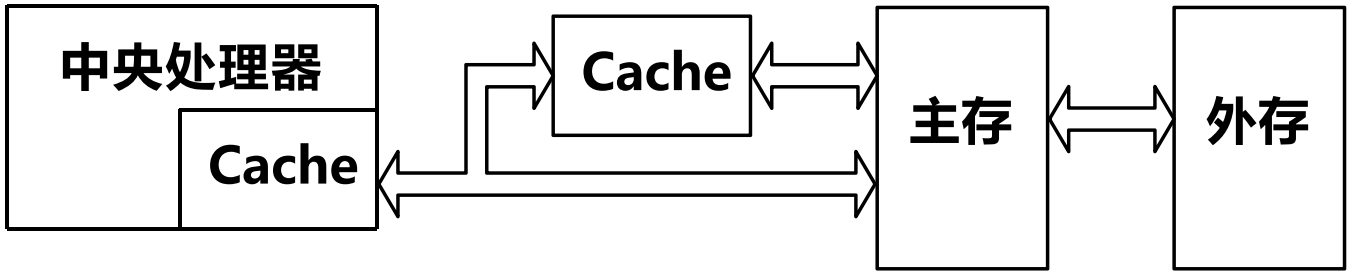
- 当CPU读取内存中一个字时，便发出此字的内存地址到Cache和主存。此时Cache控制逻辑依据地址判断此字当前是否在Cache中：
  - 若是，则Cache命中，此字立即传送给CPU；
  - 若非，则Cache缺失(未命中)，用主存读周期把此字从主存读出送到CPU，与此同时，把含有这个字的整个数据块从主存读出送到Cache中。



- 假设Cache读出时间为50ns，主存读出时间为250ns。存储系统是模块化的，主存中每个8K模块和容量16字的Cache相联系。Cache分为4行，每行4个字(W)。分配给Cache的地址存放在一个相联存储器CAM中，它是按内容寻址的存储器。当CPU执行访存指令时，就把所要访问的字的地址送到CAM；如果W不在Cache 中，则将W从主存传送到CPU。与此同时，把包含W的由前后相继的4个字所组成的一行数据送入Cache，替换原来Cache中的一行数据。

高速缓冲存储器Cache

- Cache采用SRAM器件，其存取速度接近CPU的速度。目前集成在CPU中的Cache有三级（L1 L2 L3 Cache），早些年安装在主板上的Cache称为二级Cache（L2 Cache）。



Cache的命中率

增加Cache的目的，就是在性能上使主存的平均读出时间尽可能接近Cache的读出时间。因此，Cache的命中率应接近于1。

在一个程序执行期间，设 N_c 表示Cache完成存取的总次数，N_m 表示主存完成存取的总次数，命中率 h 定义为：

$$h = \frac{N_c}{N_c + N_m}$$

若 t_c 表示命中时的Cache访问时间，t_m 表示未命中时的主存访问时间，$1 - h$ 表示未命中率，则Cache/主存系统的平均访问时间 t_a 为：

$$t_a = ht_c + (1-h)t_m$$

设 $r = t_m/t_c$ 表示主存慢于Cache的倍率，e 表示访问效率，则有：

$$e = \frac{t_c}{t_a} = \frac{t_c}{ht_c + (1-h)t_m} = \frac{1}{h + (1-h)r} = \frac{1}{1 + (1-h)(r-1)}$$

为提高访问效率，命中率 h 越接近1越好。命中率 h 与程序的行为、Cache的容量、组织方式、块的大小有关。

Cache的命中率

- CPU执行一段程序时，Cache完成存取的次数为1900次，主存完成存取的次数为100次，已知Cache存取周期为50ns，主存存取周期为250ns，求Cache/主存系统的效率和平均访问时间。

解：$h=N_c/(N_c+N_m)=1900/(1900+100)=0.95$

$r=t_m/t_c=250ns/50ns=5$

$e=1/(r+(1-r)h)=1/(5+(1-5)\times0.95)=83.3\%$

$t_a=t_c/e=50ns/0.833=60ns$

什么是平均访问时间?

平均访问时间 = 访问的总时间/访问的总次数

访问的总时间 = Cache访问时间 + 主存访问的时间

= Cache访问的平均时间×Cache访问的次数 + 主存访问的平均时间×主存访问的次数

$t_a = (250\times100+50\times1900) \div (1900+100) = 60$

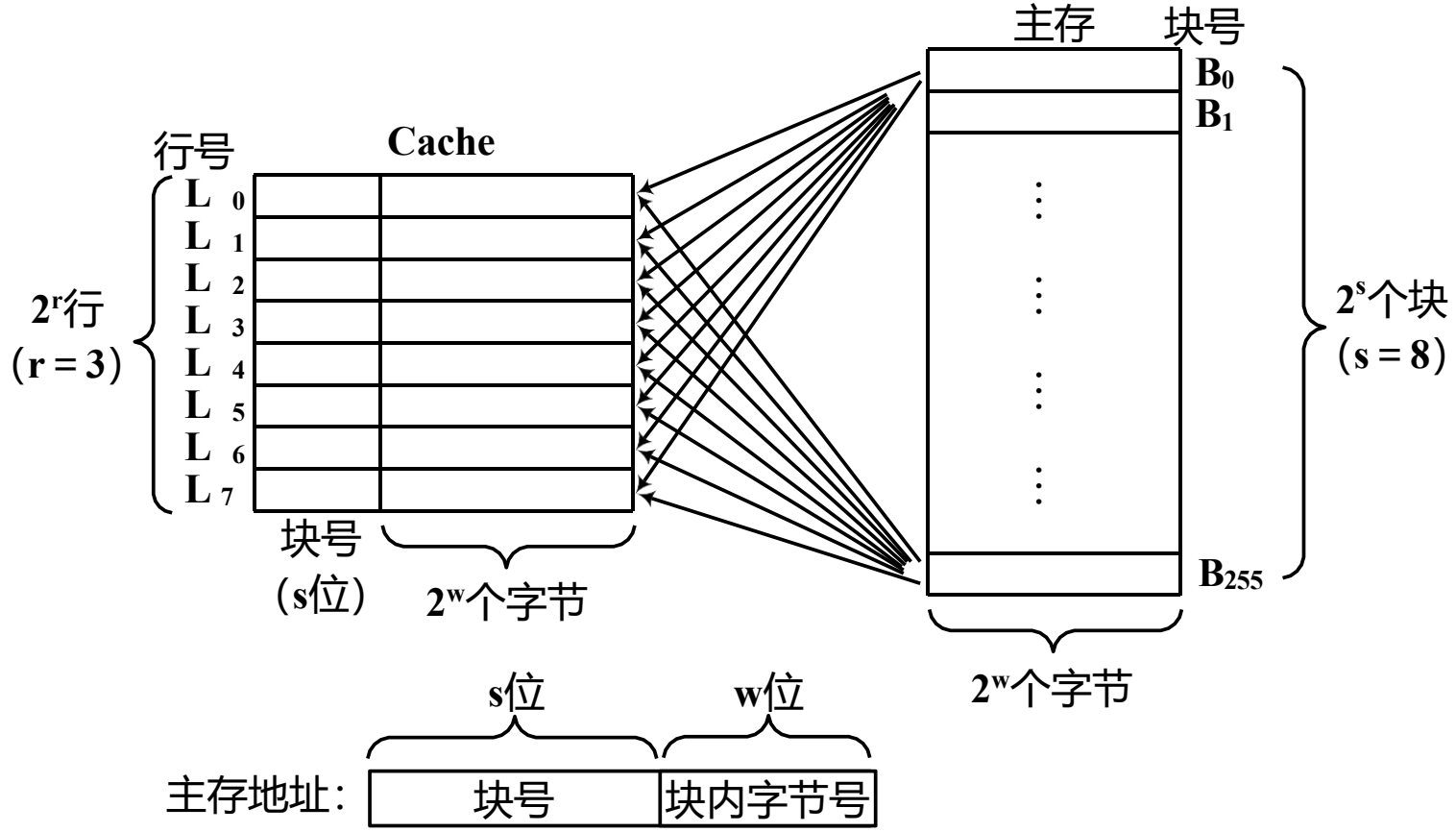
主存与Cache的地址映射

- 与主存相比，Cache的容量很小。它保存的只是主存内容的一个子集。为了把主存中的数据放到Cache中，必须应用某种方法把主存地址定位到 Cache中，称为地址映射。
- 当CPU访问存储器时，它给出的主存地址会自动变换为Cache地址。这个变换的过程是由硬件实现的，对程序员是透明的。
- Cache的数据以“行”为单位，用 L_i 表示，其中 $i = 0, 1, \ldots, 2^r - 1$，共 2^r 行；
- 主存的数据以“块”为单元，用 B_j 表示，其中 $j = 0, 1, \ldots, 2^s - 1$，共 2^s 块。
- Cache的行和主存的块是等长的，每行（块）由 2^w 个连续字节组成。
- 地址映射方式有全相联方式、直接方式和组相联方式三种。

主存与Cache的地址映射

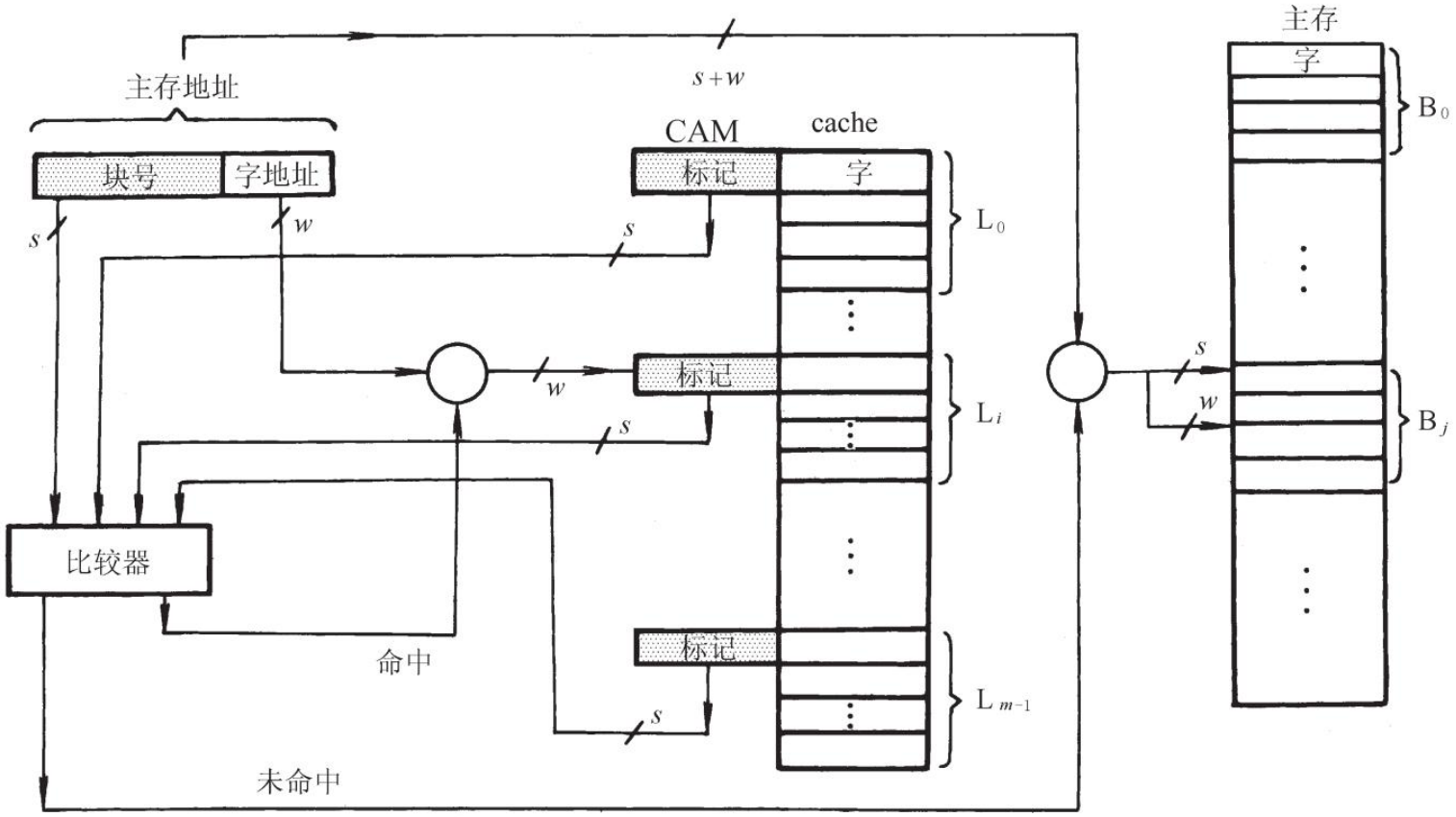
1. 全相联映射方式：

- **主存的任意一块可以存放在Cache的任意一行中。**
- **Cache的每一行有s位的标记，用于保存该行所存放的主存块的块号。**
- **CPU给出的主存地址，高s位作为主存块的块号与Cache所有行的标记进行比较，若与某行的标记相同（命中），则利用低w位从该行读出相应字节；若不命中，则从主存中读出相应字节。**



主存与Cache的地址映射

- **全相联映射的检索过程：为了快速检索，指令中的块号与cache中所有行的标记同时在比较器中进行比较。如果块号命中，则按字地址从cache中读取一个字；如果块号未命中，则按主存地址从主存中读取这个字。**



主存与Cache的地址映射

- **有一个处理器，主存容量 1MB，字长 1B，块大小 16B， cache 容量 64KB。若 cache 采用全相联映射，对内存地址$(B0010)_{16}$给出相应的标记和字地址。**

解　块大小=行大小=2^4字节=2^w字节，　所以 w=4 位

主存寻址单元数=2^{s+w}=1M=2^{20}，　所以 $s+w$=20，s=16 位

主存的块数=2^s=2^{16}

标记大小=s=16 位

内存地址格式如下所示：

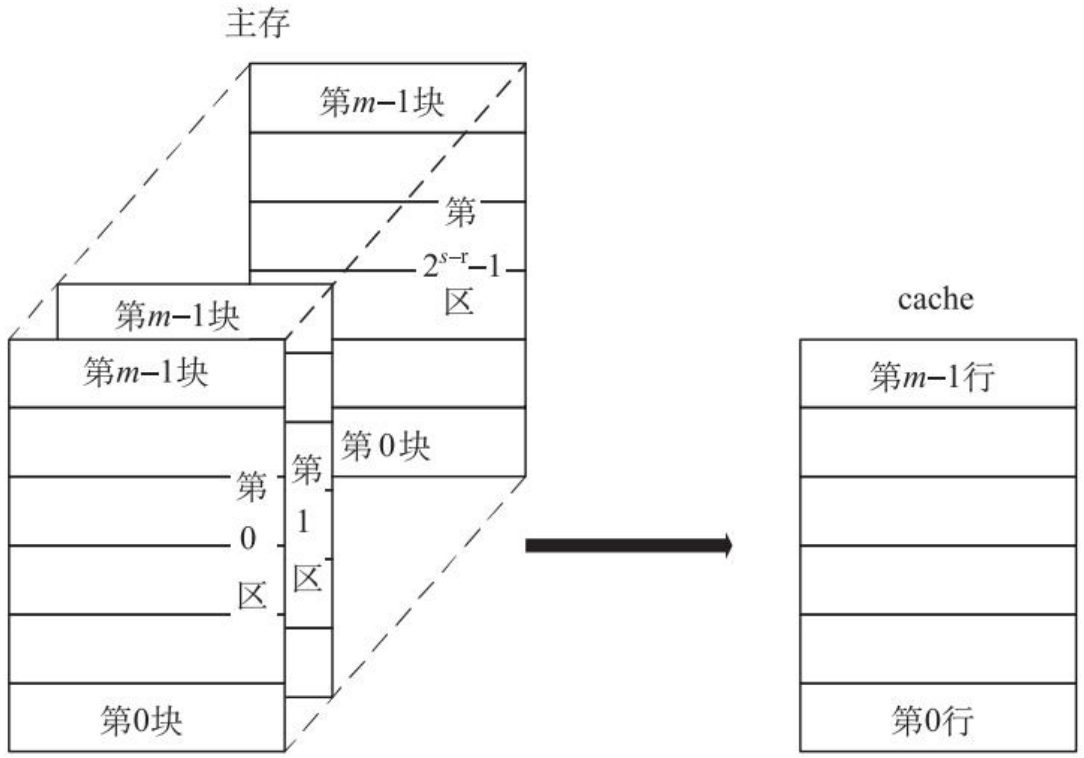
标记 s	字地址 w
16 位	4 位

由于内存地址$(B0010)_{16}=(1011\ 0000\ 0000\ 0001\ 0000)_2$

故对应的标记 $s=(1011\ 0000\ 0000\ 0001)_2$　字地址 $w=(0000)_2$

主存与Cache的地址映射

2. 直接映射方式：每个主存块只能存放在Cache的一个特定行中。可以把主存首先分区/组，每个区/组的块数与 Cache 的行数m相等。

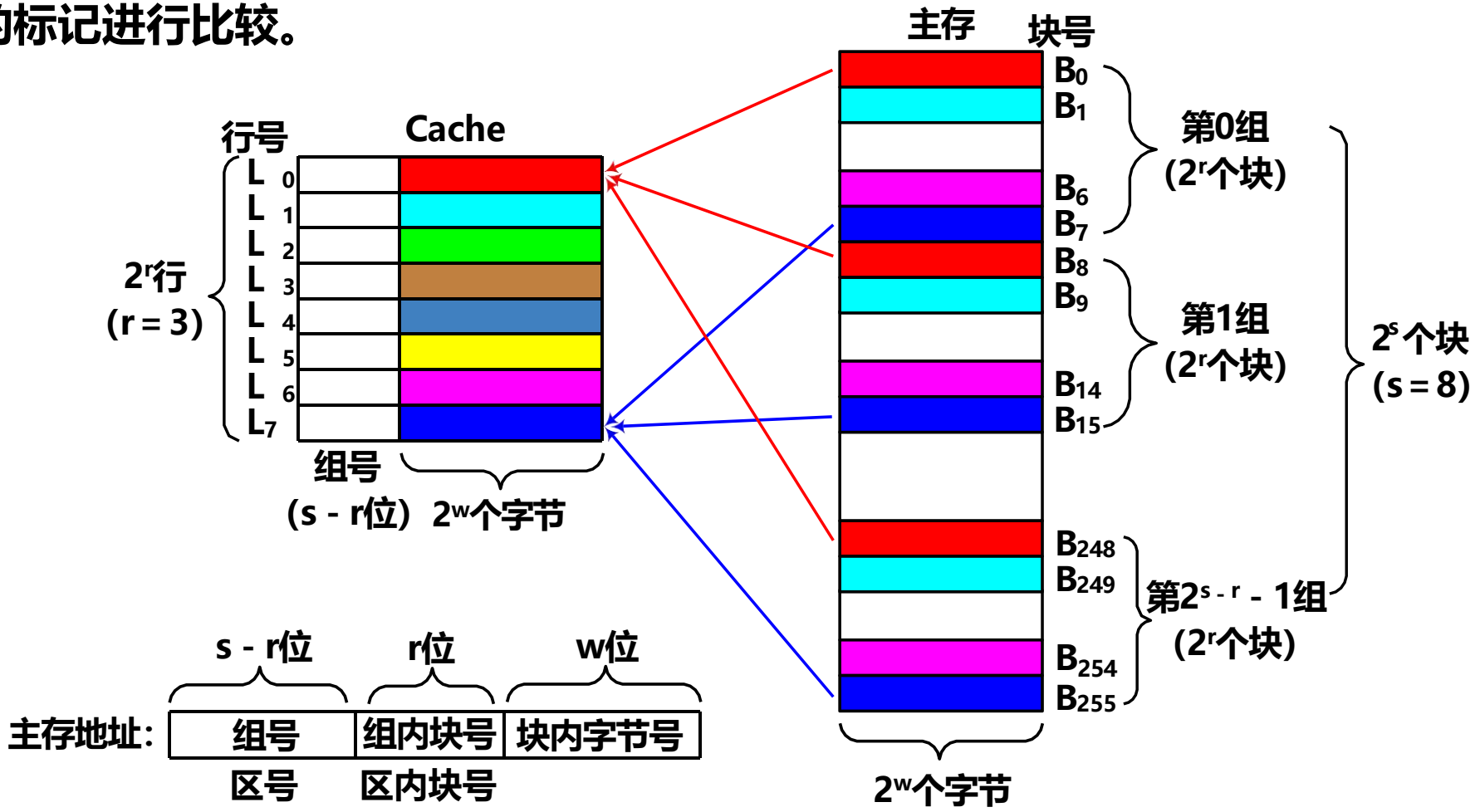


主存与Cache的地址映射

2. 直接映射方式：Cache行号i与主存块号j关系为：

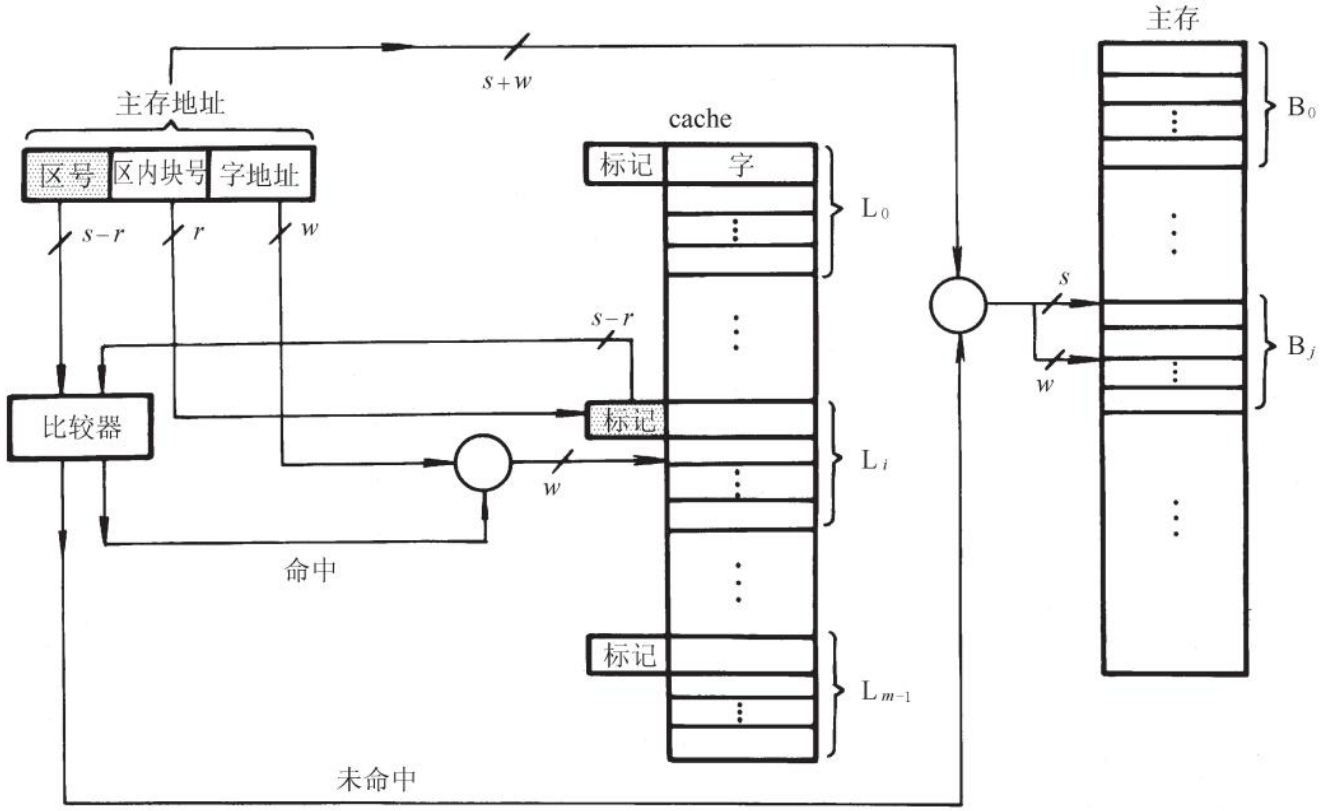
$$i = j \bmod 2^r$$

利用主存地址的中间r位确定可以保存该主存块的Cache行，再将主存地址的高（s - r）位与该行的标记进行比较。



主存与Cache的地址映射

- **直接映射方式的检索过程：当 CPU 以一个给定的内存地址访问 Cache 时，首先用 r 位区内块号找到 Cache 中的特定一行，然后用地址中的 s–r 位区号部分与此行的标记在比较器中做比较。若相符即命中，在 Cache 中找到了所要求的块， 而后用地址中最低的 w 位读取所需求的字。若不符，则未命中，由主存读取所要求的字。**



主存与Cache的地址映射

- **直接映射方式的内存地址格式如下所示，若主存地址用十六进制表示为 BBBBBB，请用十六进制格式表示直接映射方法 Cache的标记、行、字地址的值。**

标记 $s-r$	行 r	字地址 w
8 位	14 位	2 位

解 $(\mathrm{BBBBBB})_{16}=(1011\ 1011\ 1011\ 1011\ 1011\ 1011)_2$

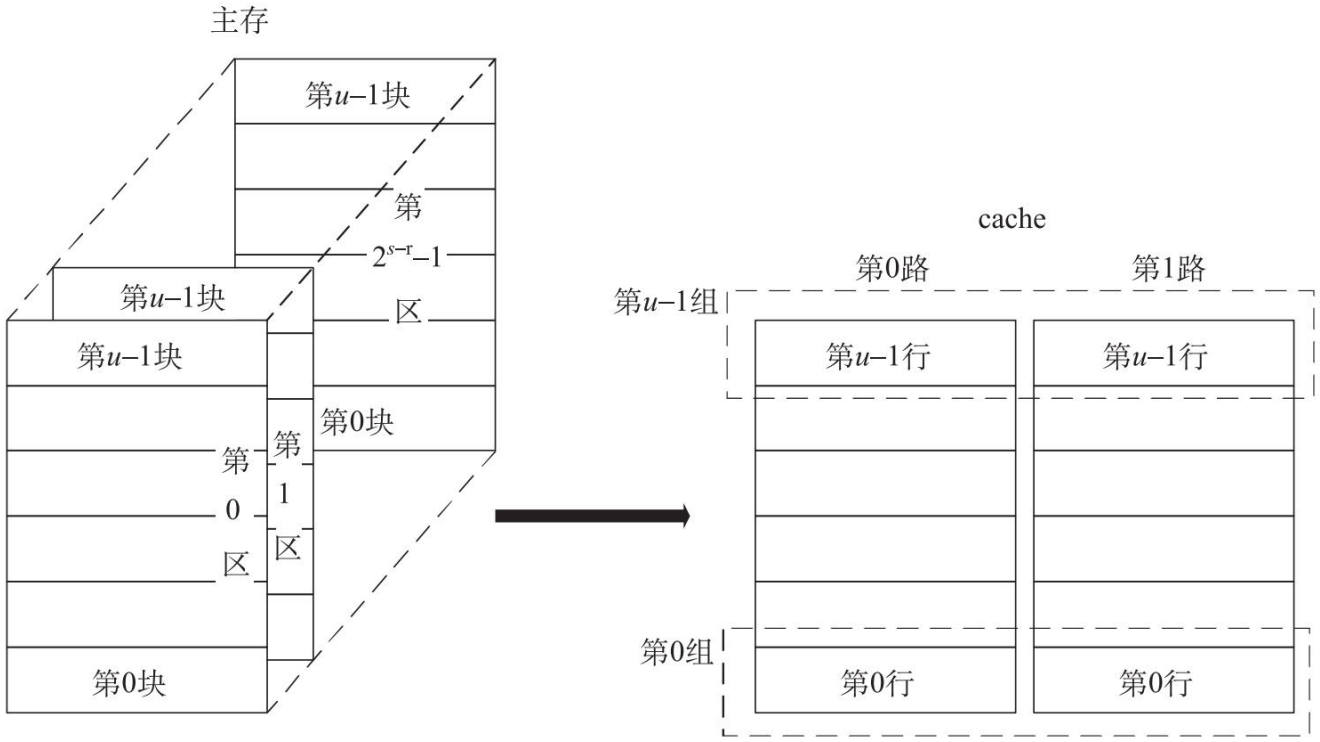
标记 $s-r=(1011\ 1011)_2=(\mathrm{BB})_{16}$

行 $r=(1011\ 1011\ 1011\ 10)_2=(2\mathrm{EEE})_{16}$

字地址 $w=(11)_2=(3)_{16}$

主存与Cache的地址映射

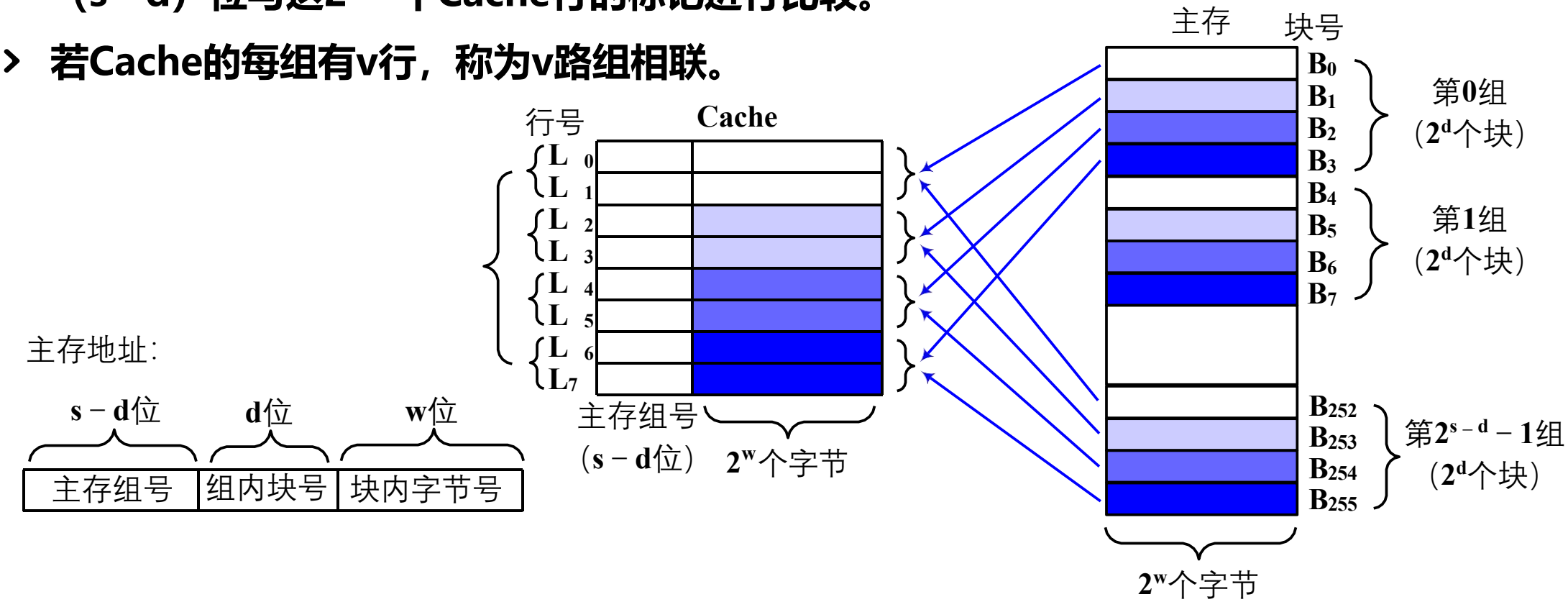
3. 组相联映射方式：将Cache划分为2^d组，每组2^{r-d}行；主存每2^d块作为一组，组中的每一块可以存放到Cache的2^{r-d}行中。



主存与Cache的地址映射

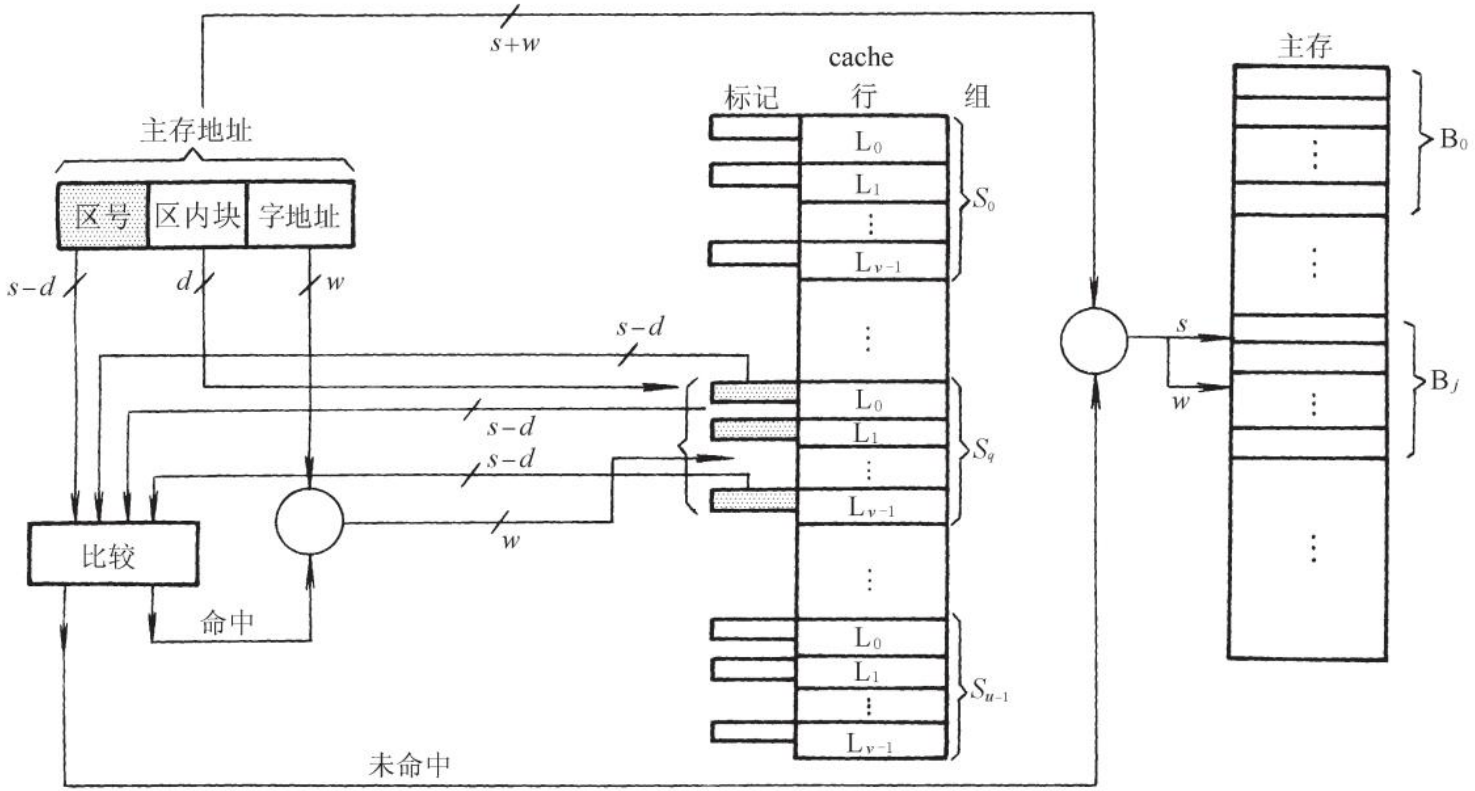
3. 组相联映射方式：将Cache划分为2^d组，每组2^{r-d}行；主存每2^d块作为一组，组中的每一块可以存放到Cache的2^{r-d}行中。

- **利用主存地址的中间d位确定可以保存该主存块的2^{r-d}个Cache行，再将主存地址的高（s - d）位与这2^{r-d}个Cache行的标记进行比较。**
- **若Cache的每组有v行，称为v路组相联。**



主存与Cache的地址映射

- **组相联映射方式的检索过程：当 CPU 给定一个内存地址访问 Cache 时，首先用 d 位区内块号找到 Cache 的相应组，然后将主存地址高 s–d 位区号部分与该组 v 行中的所有标记同时进行比较。哪行的标记与之相符，哪行即命中。此后再以内存地址的 w 位字地址部分检索此行的具体字，并完成所需要求的存取操作。如果此组没有一行的标记与之相符，即 Cache 未命中，此时需按主存地址访问主存。**



主存与Cache的地址映射

一个组相联 cache 由 64 个行组成，每组 4 行。主存储器包含 4K 个块，每块 128 字。请表示内存地址的格式。

解　块大小=行大小=2^w 个字，　$2^w=128=2^7$，　所以 $w=7$

每组的行数 $=v=4$

cache 的行数 $=uv=2^d\times v=2^d\times 4=64$，　所以 $d=4$

组数 $u=2^d=2^4=16$

主存的块数 $=2^s=4\text{K}=2^2\times 2^{10}=2^{12}$，　所以 $s=12$

标记大小 $=s-d=12-4=8$（位）

主存地址长度 $=s+w=12+7=19$（位）

主存寻址单元数 $=2^{s+w}=2^{19}$

故 $v=4$ 路组相联的内存地址格式如下所示：

标记 $s-d$	组号 d	字地址 w
8 位	4 位	7 位

主存与Cache的地址映射

三种映射方式的比较：

全相联映射方式最为灵活，因为主存的任意一块可以保存在Cache的任意一行中，Cache利用率高。但Cache的标记位太长，而且需要将主存地址的高s位与所有Cache行的标记进行比较，硬件成本高，比较时间长。

直接映射方式的灵活性差，每个主存块只能存放在Cache的特定行中，如果相距为2^r的整数倍的两个主存块都需要存放在Cache中，就会发生冲突。这种方式的优点是硬件简单，只需要将主存地址的高（s - r）位与一个Cache行的标记进行比较。

组相联映射方式是前两种方式的折中。每个主存块可以存放在几个Cache行中，具有一定的灵活性，降低了冲突的可能。主存地址的高（s - d）位只需要和几个Cache行的标记进行比较，硬件成本相对较低。

总结

- ✓ Cache命中率、效率及其相关计算
- ✓ 地址映射的方式及其特点
- ✓ 行号、标记（Tag）、页内地址、组号等位数的计算
- ✓ 地址映射、比较过程

替换策略

Cache应尽量保存最新的数据，必然产生替换。

- 对于直接映射方式，每个主存块只有一个Cache行可以存放，只需把该行的原主存块换出即可。
  - 对于全相联和组相联映射方式，需要一定的替换策略。
- 最不经常使用算法（Least Frequently Use, LFU）：每行设置一个计数器，新行建立后从0开始计数。每命中一次，命中行的计数器加1。需要替换时，将计数值最小的行换出，其它行计数器清0。
- 近期最少使用算法（Least Recently Use, LRU）：每行设置一个计数器，新行建立时计数器为0。每命中一次，命中行计数器清0，其它行计数器加1。需要替换时，将计数值最大的行换出。
  - LRU保护了刚建立的行和刚刚命中的行，符合程序的局部性原理。

替换策略

- 对于2路组相联的Cache，每个主存块只能存放在Cache中一个特定组的两行中，只需用一个二进制位作为标记。若一组中A行刚建立或刚命中，则将此位置1；B行刚建立或刚命中，则将此位置0。需要替换时，检查此二进制位的值，为0则替换A行，为1则替换B行。奔腾CPU的数据Cache是2路组相联结构，就是采用这种简化的LRU算法。
- 先进先出算法（First In First Out, FIFO）：按调入Cache的先后顺序决定淘汰的顺序，将最先进入Cache的块换出。这种方法要求记录每块进入Cache的先后次序。这种方法容易实现，其缺点是可能会把一些需要经常使用的程序块（如循环程序）也作为最早进入Cache的块替换掉。
- 随机替换：从Cache中随机选择一行换出。

Cache的写策略

- Cache的内容只是主存内容的副本，应当和主存内容保持一致。CPU对Cache的写入改变了Cache的内容，应设法使Cache和主存保持一致。常用的写策略有两种：
  - **写回法（Write Back）**：CPU写Cache命中时，只修改Cache内容，而不立即写入主存。只有当此行被换出时才写回主存。这种方法使Cache对CPU - 主存之间的读写操作都能起到高速缓冲作用。每个Cache需配置一个修改标志位。
  - **全写法（Write Through）**：写Cache命中时，Cache和主存同时进行写操作。Cache在写操作时无高速缓冲功能。
- 现代计算机中，能够访问主存的设备不止一个。一个计算机可以有多个 CPU，它们有各自的Cache和公用的主存；有些I/O设备也可以独立进行主存的读写。为了保证各个CPU的Cache及主存之间数据的一致性，需要更复杂的技术。

总线监听协议

- 每个Cache不断地监视地址总线，检测总线上其它CPU或I/O设备对存储器的写操作，并把对自身的写操作发布到总线上。
- 监听协议有两种：写 - 无效协议和写 - 更新协议。
  - 写 - 无效协议是指某Cache数据被修改后使所有其它 Cache中的相应数据副本失效；
  - 写 - 更新协议是指某Cache数据被修改后，广播修改后的数据以更新所有Cache中的相应数据副本。
- 奔腾CPU中采用的MESI协议属于写 - 无效协议。每个Cache行有4种状态，用两位进行标记，并根据监听结果和本地CPU的动作进行状态转换。
  - M (Modified): 修改态。此Cache行已被本地CPU修改，且没有写回主存。
  - E (Exclusive): 专有态。此Cache行和主存内容相同，其它Cache中没有副本。
  - S (Shared): 共享态。此Cache行有效，其它Cache中可能有该行的有效副本。
  - I (Invalid): 无效态。此Cache行无效。

作业

1. 已知 cache 存储周期 40ns，主存存储周期 200ns， cache/主存系统平均访问时间为 50ns，求 cache的命中率是多少?

2. 一个组相联 cache 由 64 个行组成，每组 4 行。主存储器包含 4K 个块，每块 128 字。请表示内存地址的格式。

虚拟存储器

- **虚拟存储器是指主存 - 辅存层次在操作系统的调度下，以透明的方式给用户提供一个比实际主存空间大得多的程序空间。**
- **程序员编程时所用的地址称为逻辑地址或虚拟地址（虚地址），主存的实际地址称为物理地址（实地址）。程序每次访问主存时，要进行虚、实地址的转换，通常用软、硬件结合的办法实现。**
- **主存 - 辅存层次和Cache - 主存层次有很多相似之处，它们都包括一个容量较小速度较高的存储器和一个容量较大速度较低的存储器，都采用地址变换及映射方法和替换策略，都基于程序局部性原理。它们遵循的原则是：把程序中最近常用的部分驻留在高速存储器中，一旦这部分变得不常用了，就把它们送回低速存储器；这种换入换出对应用程序是透明的；目的是使存储系统的性能接近高速存储器，容量和价格接近低速存储器。**
- **Cache - 主存层次主要是为了提高存储系统的速度，而主存 - 辅存层次主要是为了扩大程序可用的地址空间。**

虚拟存储器

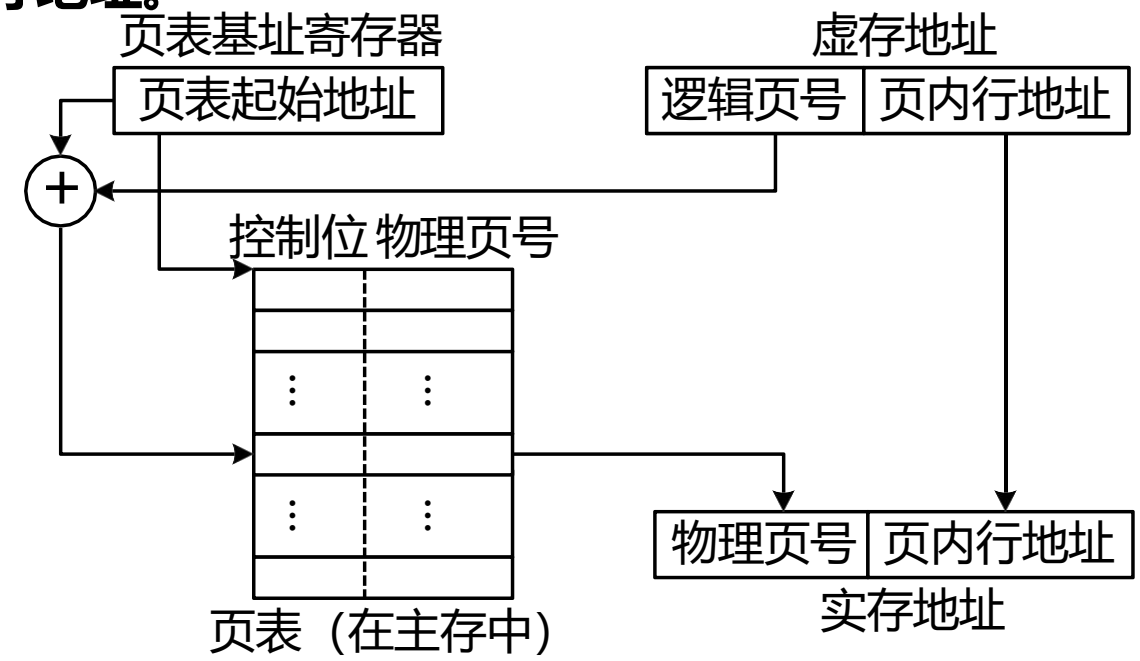
虚拟存储器的管理方式有三种：段式、页式和段页式。

- **段式管理：**程序可以按照逻辑结构划分为多个相对独立的段。将主存按段分配的管理方式称为段式管理。段的大小取决于程序的逻辑结构，可长可短。用段表来指明各段在主存中的位置，段表本身也是主存中一个可再定位段。段的逻辑独立性使它易于管理、修改和保护。由于段长不固定，给主存空间的分配带来麻烦，容易形成碎片。
- **页式管理：**是把主存空间划分成长度固定的页。优点是页的起点和终点地址固定，给造页表带来方便，对主存空间的浪费小。由于页不是逻辑上独立的实体，处理、保护和共享都不如段式管理方便。
- **段页式管理：**是将分段和分页结合起来，程序按模块分段，段内再分页。用段表和页表进行两级定位管理。

页式管理

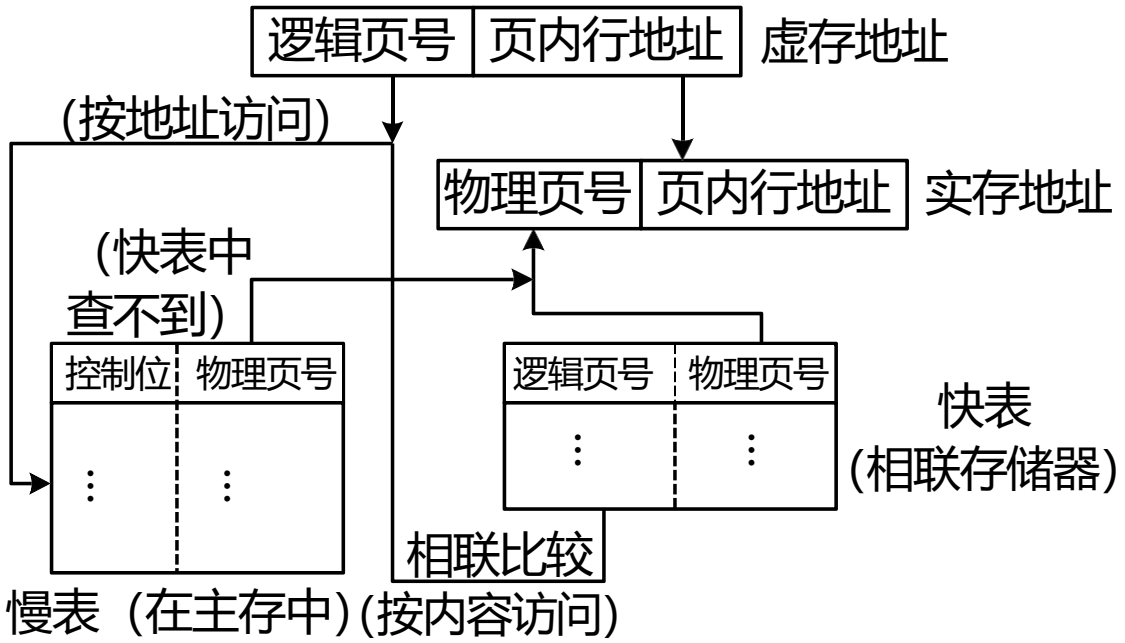
- 虚拟空间分成逻辑页，主存空间也分成同样大小的物理页。
- 虚存地址分为两个字段：高字段为逻辑页号，低字段为页内行地址。

- 实存地址也分两个字段：高字段为物理页号，低字段为页内行地址。
- 两者的页内行地址是相同的。虚实地址的转换通过页表实现。页表中每一个逻辑页号有一个表项，表项内容包含该逻辑页的物理页号，用它作为实存地址的高字段，与虚存地址的页内行地址字段相拼接，产生完整的实主存地址。
- 页表的表项中通常还有装入位、修改位、保护位、控制位等。装入位表明该逻辑页是否调入主存，访问没有装入的页会启动输入输出系统，从辅存装入该页。修改位指出主存中页面是否被修改。



页式管理

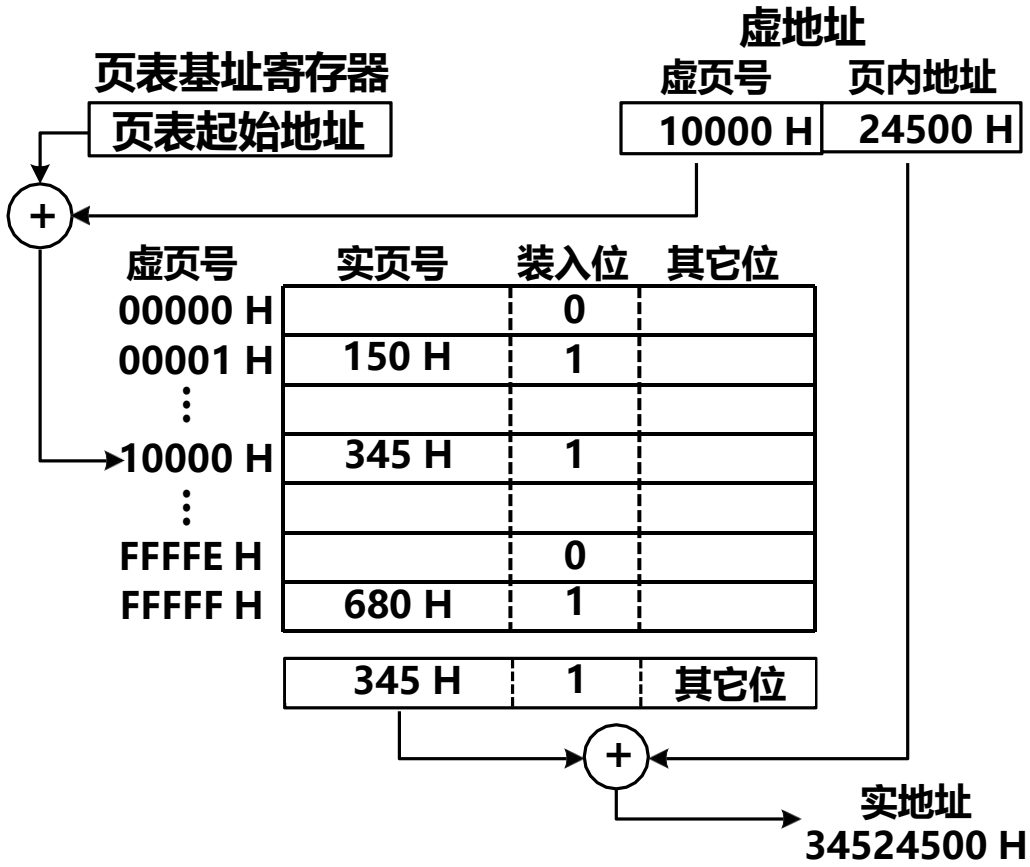
- 访问存储器时，首先要查找页表，根据查表获得实存地址再次访问主存。因此需要两次访问主存。可以把页表中最活跃的部分存放在高速存储器中组成快表。为了提高查找速度，可以引入硬件支持，如采用相联存储器。
- 快表是慢表中部分内容的副本。查表时，用逻辑页号同时查找快表和慢表。快表命中则使慢表的查找作废。快表查不到则等待慢表的查找结果，并将逻辑页号和物理页号送入快表。



页式管理

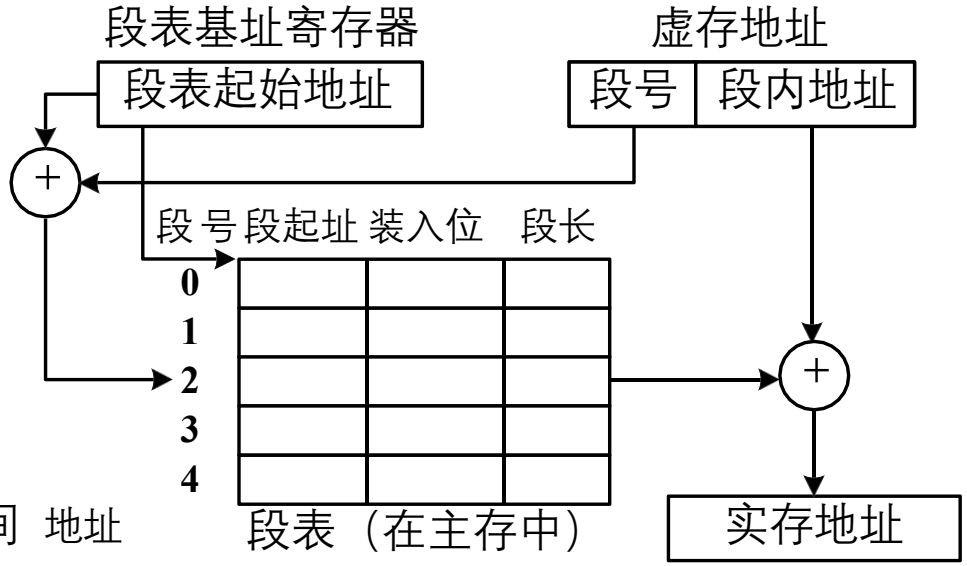
例：某计算机的虚拟存储系统有40位虚地址，32位实地址。页的大小为1MB，有装入位、修改位、保护位和使用位四个控制位。所有虚页都在使用中。

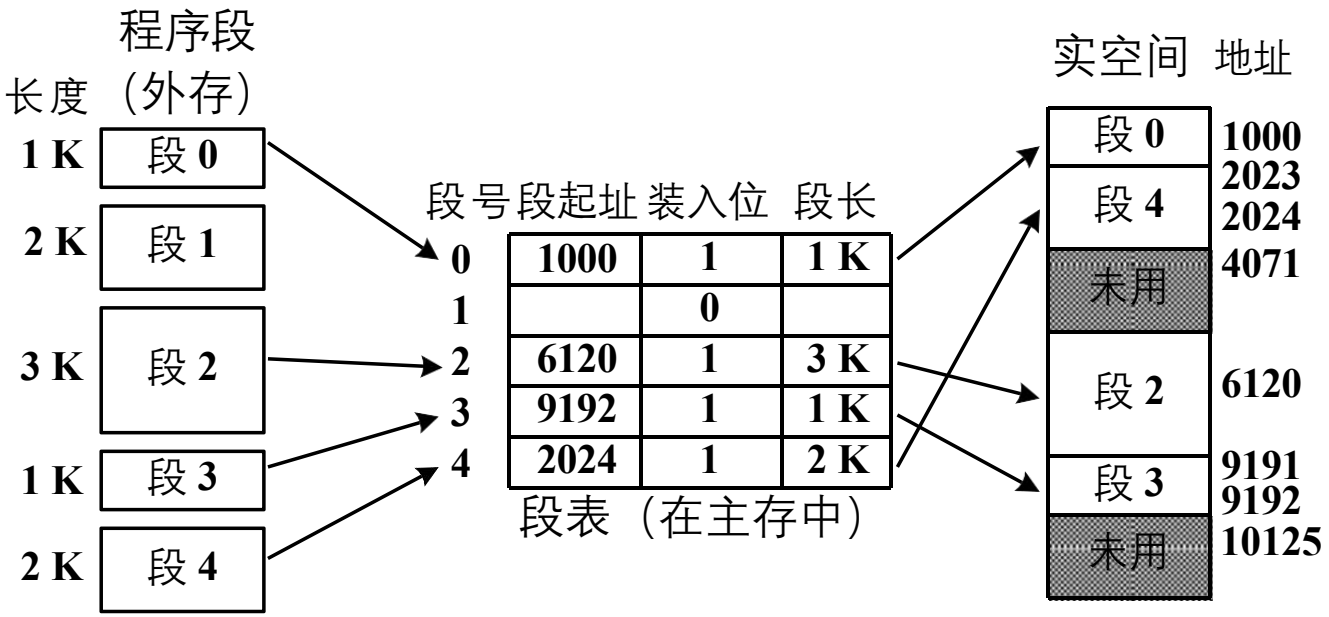
- 每个页表项的长度为实页号位数（12位）＋控制位（4位）＝16位。
- 虚页号20位，因此有2^{20}＝ 1M个虚页，页表大小为1M×16位。



段式管理

- **段是按照程序的逻辑结构划分的，各个段的长度因程序而异。虚地址由段号和段内地址组成。**





为了把虚地址变换成实主存地址，需要一个段表。由于段的长度不固定，段表中需要有长度指。段表也是一个段。

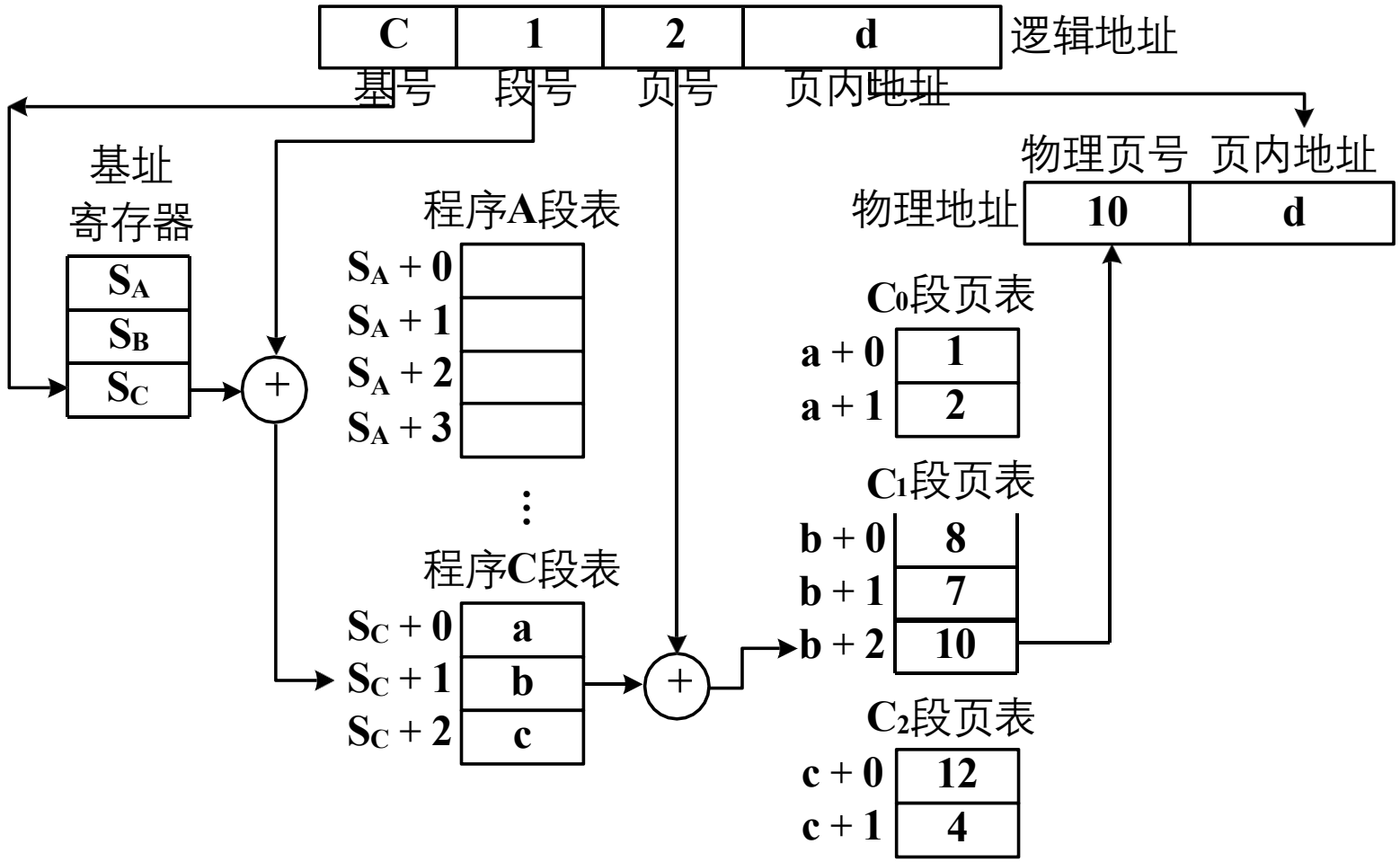
段页式管理

- 把程序按逻辑单位分段以后，再把每段分成固定大小的页。程序对主存的调入调出是按页面进行的，但它又可以按段实现共享和保护，兼备页式和段式的优点。
- 每道程序是通过一个段表和一组页表来进行定位的。段表中的每个表项对应一个段，每个表项包括该段的页表起始地址及该段的控制保护信息。由页表指明该段各页在主存中的位置以及是否已装入、已修改等状态信息。
  - 缺点是在映象过程中需要多次查表。
  - 如果有多个用户在机器上运行，多道程序的每一道需要一个基号，由它
- 指明该道程序的段表起始地址。
  - 虚拟地址格式如下：

基号	段号	页号	页内地址

段页式管理

例：有三道程序（A、B、C），每道程序有一张段表，段表起始地址分别为 S_A、S_B、S_C，每段有一张页表。段表的每行就是相应页表的起始地址，页表的每行就是相应的物理页号。



替换算法

- 虚拟存储器中的替换策略一般采用LRU算法、LFU算法 、FIFO算法，或将两种算法结合起来使用。对于将被替换出去的页面，假如该页调入主存后没有被修改，就不必进行处理，否则就把该页重新写入外存。为此，在页表的每一行应设置一修改位。

例：主存只有a、b、c三个页，组成a进c出的FIFO队列，进程访问页面的序列是0，1，2，4，2，3，0，2，1，3，2号。若采用①FIFO算法，②FIFO算法+LRU算法，求两种替换策略情况下的命中率。

页面访问序列		0	1	2	4	2	3	0	2	1	3	2	命中率
FIFO	a	0	1	2	4	4	3	0	2	1	3	3	2/11=18.2%
	b		0	1	2	2	4	3	0	2	1	1	
	c			0	1	1	2	4	3	0	2	2	
FIFO + LRU	a	0	1	2	4	2	3	0	2	1	3	2	3/11=27.3%
	b		0	1	2	4	2	3	0	2	1	3	
	c			0	1	1	4	2	3	0	2	1	

虚拟存储器实例

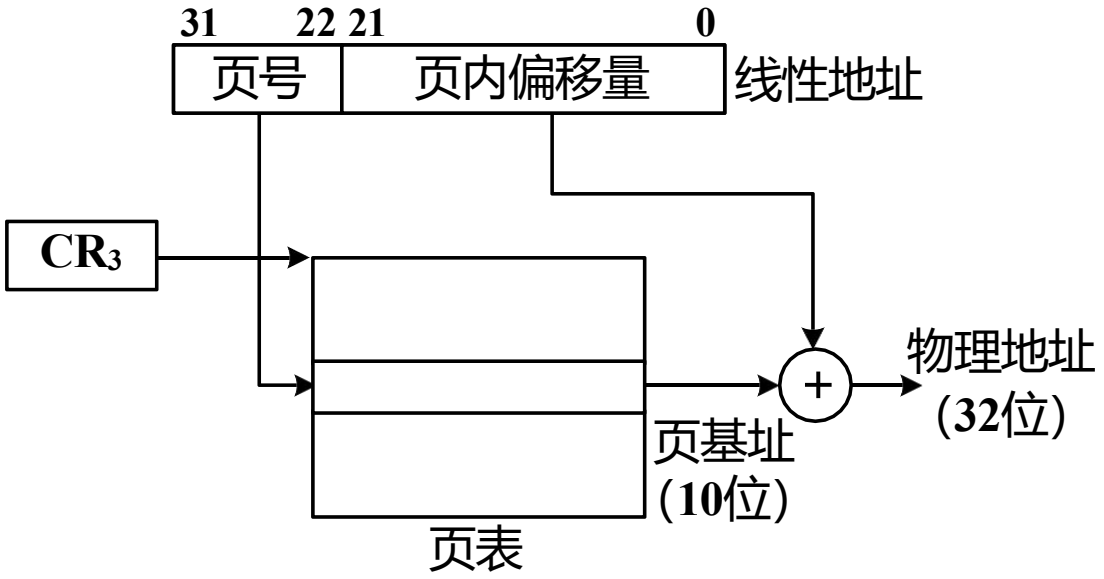
奔腾CPU的存储管理部件MMU包括分段部件SU和分页部件PU两部份，可单独或同时工作。

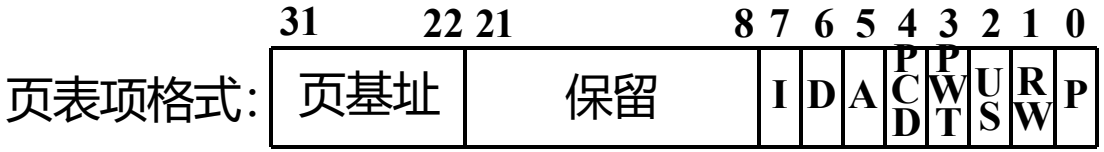
有三种虚拟地址模式。

- **分段不分页模式：** 虚拟地址由一个16位的段选择符和一个32位的偏移量组成。段选择符的最低两位是特权级，高14位指定具体的段。一个进程可拥有的最大虚拟地址空间为$2^{14+32}=2^{46}=64TB$。
- **分段分页模式：** 在分段基础上增加分页存储管理。分页管理包括两级页表，分别称为页目录表和页表。SU部件转换后的32位地址称为线性地址，包括10位页目录索引、10位页表索引和12位页内偏移量。再由PU部件完成两级页表的查找，得到20位的实页号，和12位页内偏移量拼接，得到32位物理地址。进程的最大虚拟地址空间也是64TB。
- **不分段分页模式：** 这种模式下SU不工作，只是分页部件PU工作。32位虚拟地址被看成是由页目录、页表、页内偏移量三个字段组成。由PU完成虚拟地址到物理地址的转换。进程的最大虚拟地址空间是4GB。

分页地址转换

- 奔腾CPU有两种分页方式：一种是4 KB的页，使用二级页表（页目录表、页表）进行地址转换；另一种是4 MB的页，用单级页表进行转换。后一种方式下，32位线性地址分为高10位的页号和低22位的页内偏移量。全系统只有一张页表，控制寄存器CR_3指向页表的起始地址。页表有1 K个表项，每项32位。



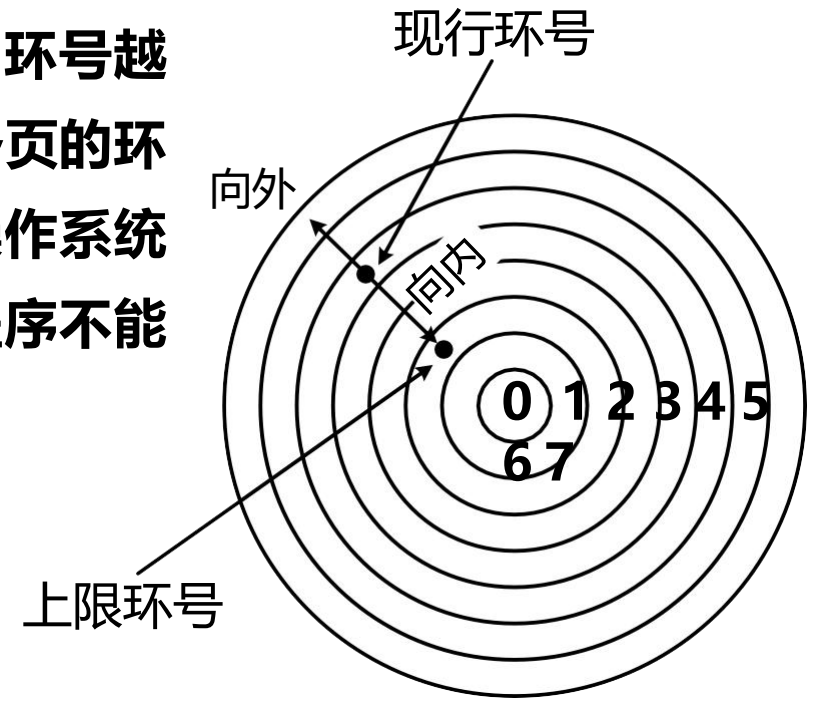


存储保护

- 存储保护的目的是防止一个用户程序出错而破坏其他用户的程序和系统软件，还要防止一个用户程序不合法地访问不是分配给它的主存区域。包括存储区域保护和访问方式保护。
- **页表和段表保护：**每个程序的段表和页表都有自身的保护功能。每个程序的虚页号是固定的，如果虚页号出错，则在页表中找不到，也就访问不了主存。段表和页表的保护功能相同。另外，段表项中包含段长，段长通常就是该段包含的页数。如果页号大于段长，说明该页号非法。
- **键保护：**为主存的每一页配一个键，称为存储键。访问键赋予每道程序。当数据要写入主存的某一页时，访问键和存储键比较。二者相符则允许访问该页，否则拒绝访问。
- 另外还有取数保护。为每页设置一个一位的取数键。取数键为0则该页只受存数保护；为1则受到存取保护，只有访问键和存储键相同的程序才能存取这些页。

存储保护

环保护：将系统程序和用户程序分层，每层叫做一个环。环号越大，等级越低。在现行程序运行前由操作系统定好程序各页的环号。程序可以访问任何外层空间；访问内层空间则需由操作系统来判断这个向内访问是否合法。每个程序有上限环号，程序不能访问低于上限环号的存储区域。



访问方式保护：对主存信息的使用有三种方式：读(R)、写(W)和执行(E)。相应的访问保护方式就是R、W、E三种方式的逻辑组合。

逻辑组合	含义	逻辑组合	含义
R+W+E	不允许任何访问	(R+E)•W	只能写访问
R+W+E	可进行任何访问	(R+E)•W	不准写访问
(R+W)•E	只能读写，不可执行	R•(W+E)	只能读访问
(R+W)•E	只能执行，不可读写	R•(W+E)	不准读访问

谢谢