# 一、基本概念

1、名词解释

- 存储位元（存储元件）
- 存储单元：若干存储位元组成一个存储单元
- RAM（SRAM、DRAM）
- ROM（掩模ROM、PROM、 EPROM、$E^2$PROM）、闪存
- 易失性存储器
- Cache
- 虚拟存储器
- 存储器带宽
- 存取时间和存储周期

# 2、DRAM的三种刷新方式

- **集中式刷新**：在刷新周期内，对DRAM所有行进行集中刷新。
- **分散式刷新**：每一行的刷新插入到正常读/写周期之中。存储周期的前半段时间用来读/写操作或维持信息，后半段时间作为刷新操作时间， 加长了系统周期，刷新过于频繁
- **异步刷新**：前两种方式的结合，把刷新操作平均分散到整个刷新周期（PC机采用的刷新方式）

# 3、主存储器的性能指标

- **存储容量**：存储器中可以容纳的存储单元总数。存储容量越大，能存储的信息就越多。
- **存取时间（存储器访问时间）**：指一次读操作命令发出到该操作完成，将数据读出到数据总线上所经历的时间。通常，写操作时间等于读操作时间，故称为存储器存取时间。
- **存储周期**：指连续启动两次读操作所需间隔的最小时间，时间单位为ns。通常，存储周期略大于存取时间。
- **存储器带宽**：单位时间里存储器所存取的信息量，通常以位/秒或字节/秒做度量单位。

## 4、高级的DRAM结构

（1）FPM DRAM：快速页模式动态存储器，根据程序局部性原理实现。

- 允许在选定行中，对每个列地址（同一页）进行连续快速访问。

(2) CDRAM：带高速缓冲存储器（cache）的动态存储器

- 在DRAM内集成一个小容量SRAM，显著改进DRAM芯片性能

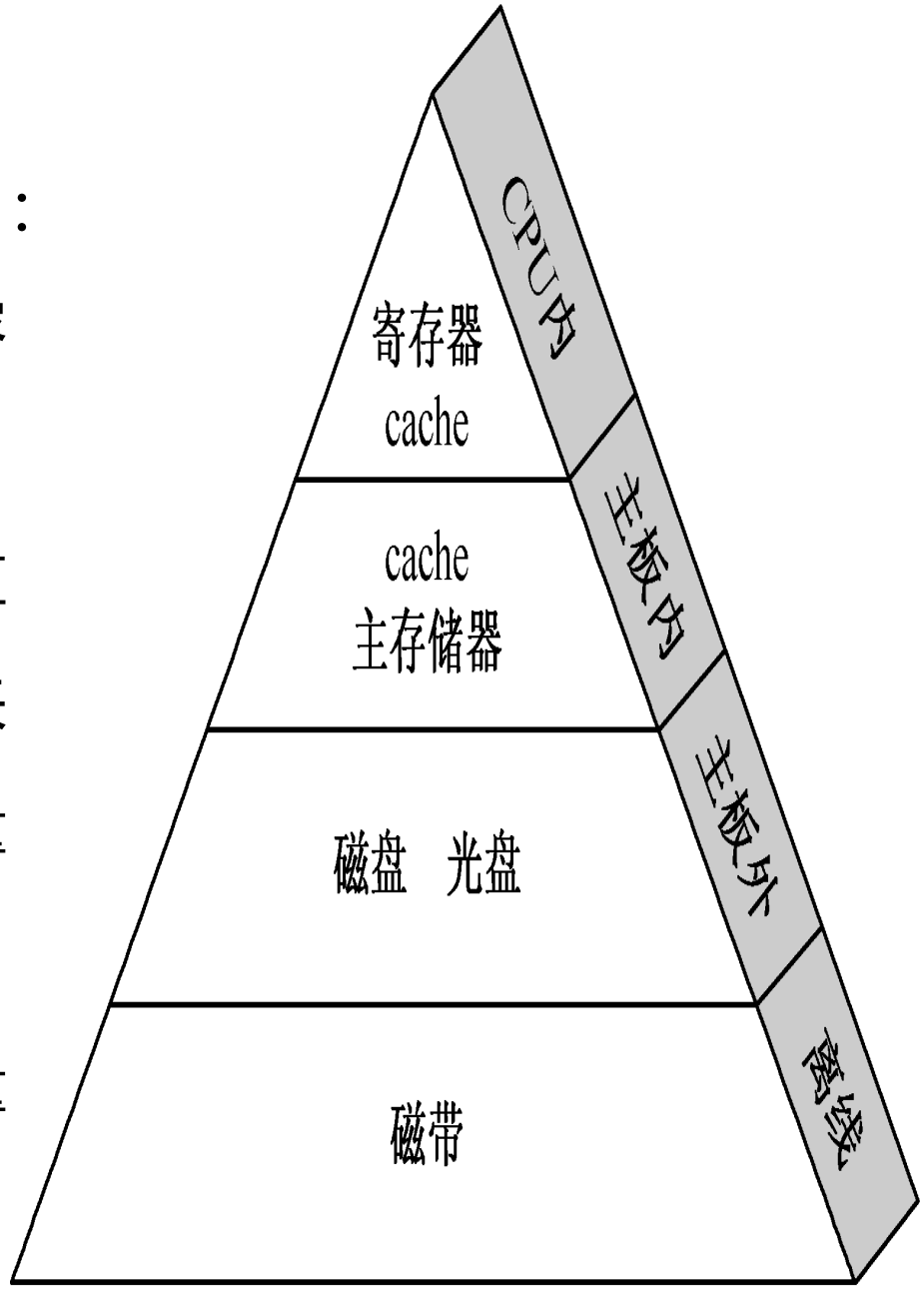
（3）SDRAM（synchronous DRAM）：同步DRAM，内存工作速度与系统总线速度同步，避免了不必要的等待周期。

（4）DDR SDRAM (double data rate SDRAM）：双数据传输率同步动态内存 ，在SDRAM基础上对SDRAM设备进行改进形成的DDR。

- DDR2/DDR II：拥有两倍于DDR 1的内存预读取能力，使DDR2每个时钟周期能够以4倍外部总线的速度读/写数据。
- DDR3/DDR III：预取能力是DDR2的2倍，是DDR2 SDRAM（四倍数据传输率SDRAM）的后继者【增加至八倍】。

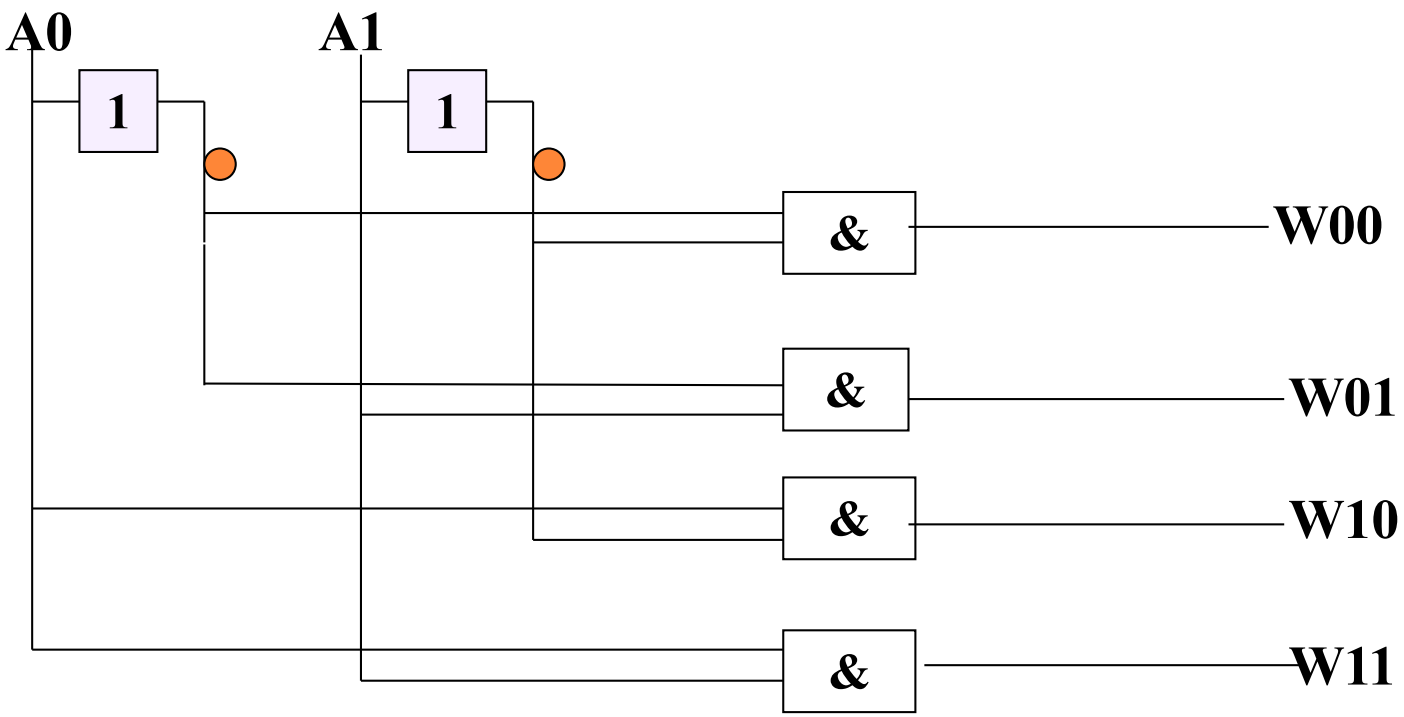
## 5、分级结构

- 高速缓冲存储器（cache）：计算机系统中的高速、小容量半导体存储器。
- 主存储器（主存）：是计算机系统的主要存储器，用来存放计算机运行期间的大量程序和数据。
- 外存储器（外存）：大容量辅助存储器。

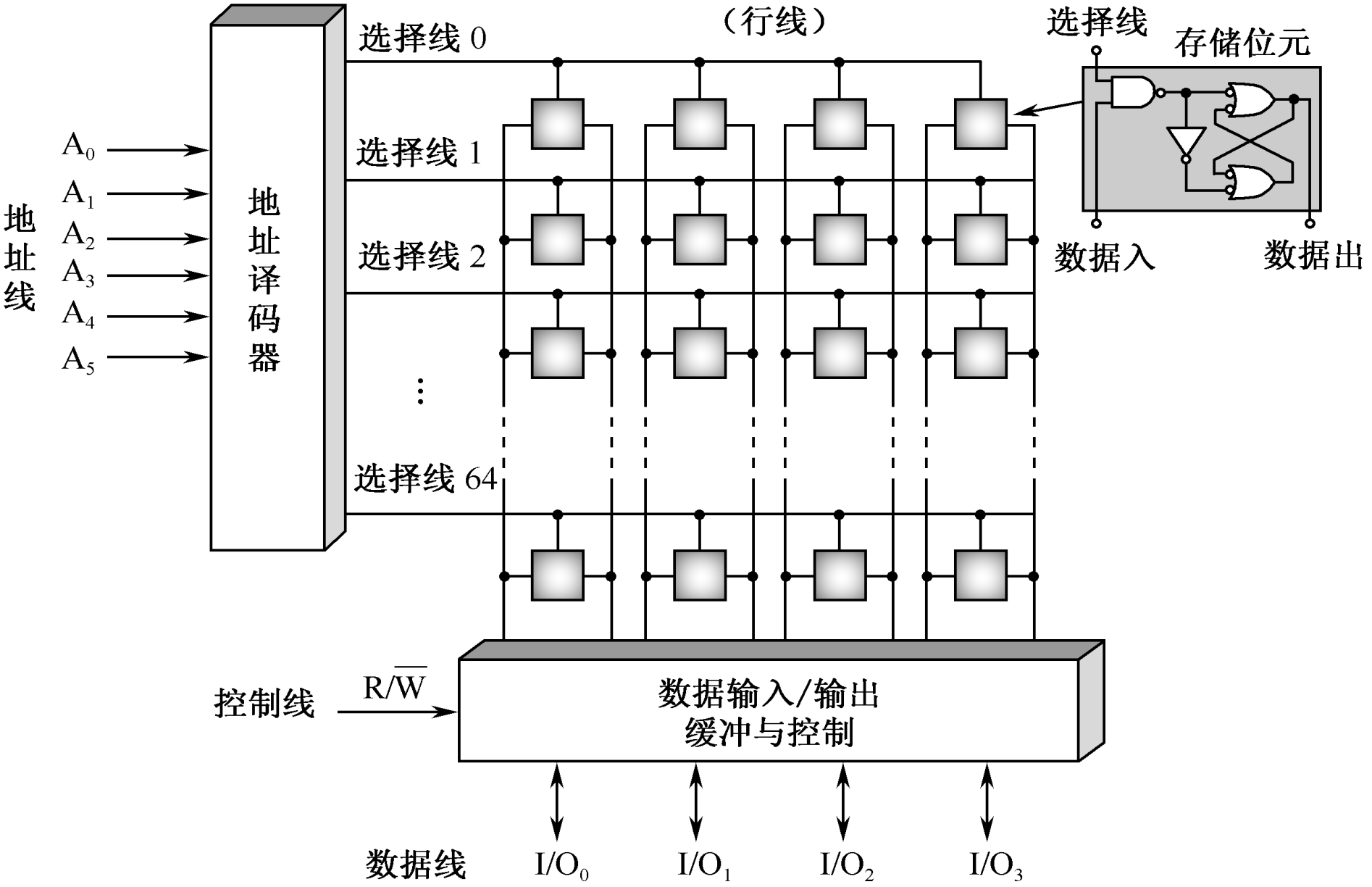


## 二、地址译码逻辑

1、地址译码器：将地址码变换为驱动字线的电位，$2^n$中选1的逻辑电路。



# 2、单译码（一维译码）方式


地址线
A0
A1
A2
A3
A4
A5
地址译码器
选择线 0
选择线 1
选择线 2
选择线 64
（行线）
选择线
存储位元
数据入
数据出
控制线
R/W
数据输入/输出
缓冲与控制
数据线
I/O0
I/O1
I/O2
I/O3

## 3、双译码（二维译码）方式

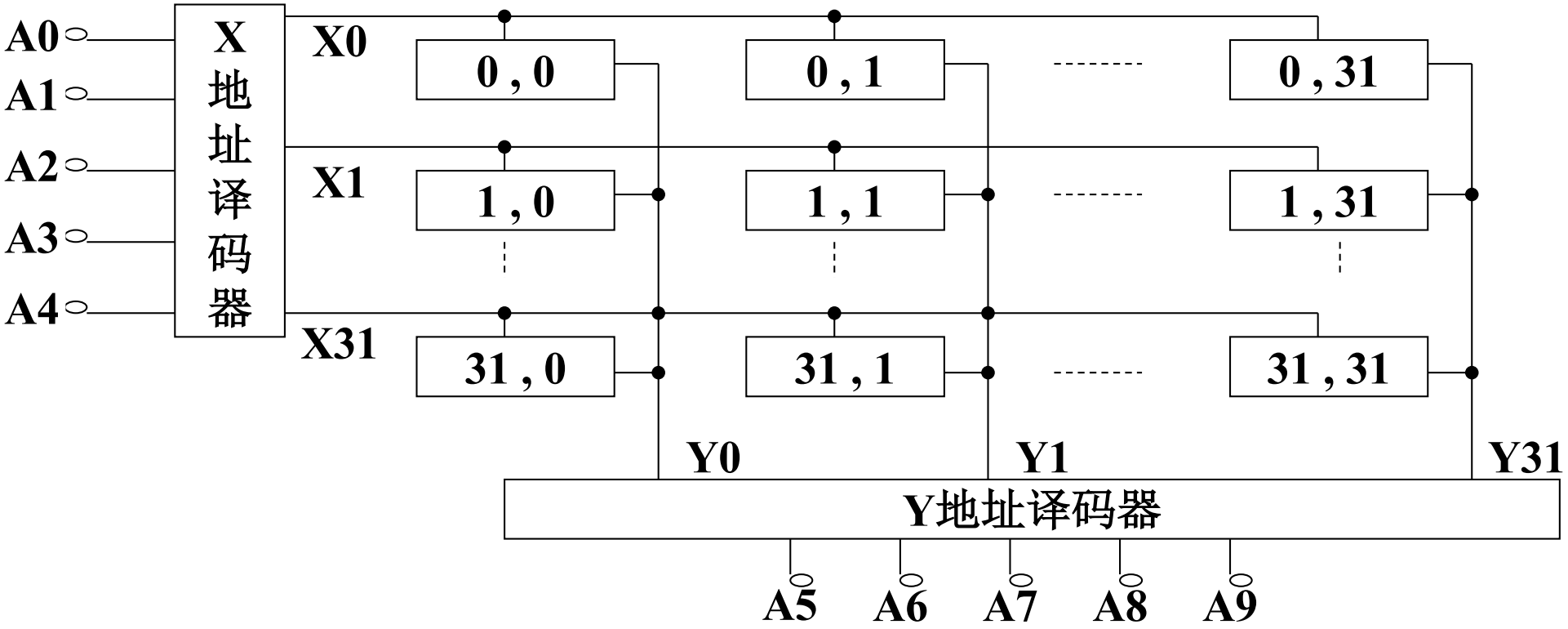
(1) M×N个存储单元：排成M行，N列的二维存储体阵列

(2) 地址分两部分：Nx，Ny

(3)译码过程：Nx经译码，驱动一根字线，Ny经译码，驱动一根字线;

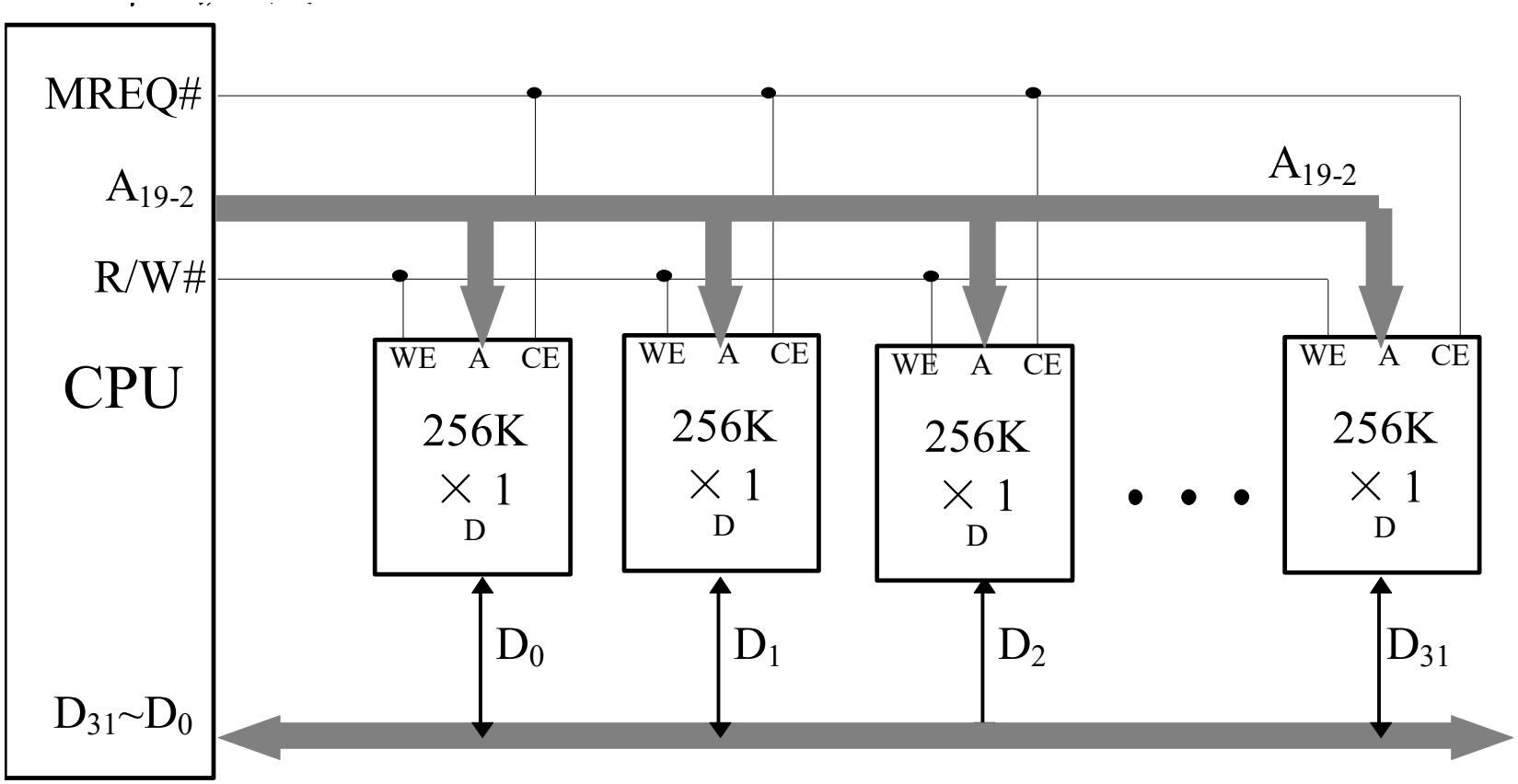
只有行、列交叉点上的存储单元被真正驱动

(4) $M = 2^{N_x}$，$N = 2^{N_y}$，$M \times N = 2^{N_x+N_y}$



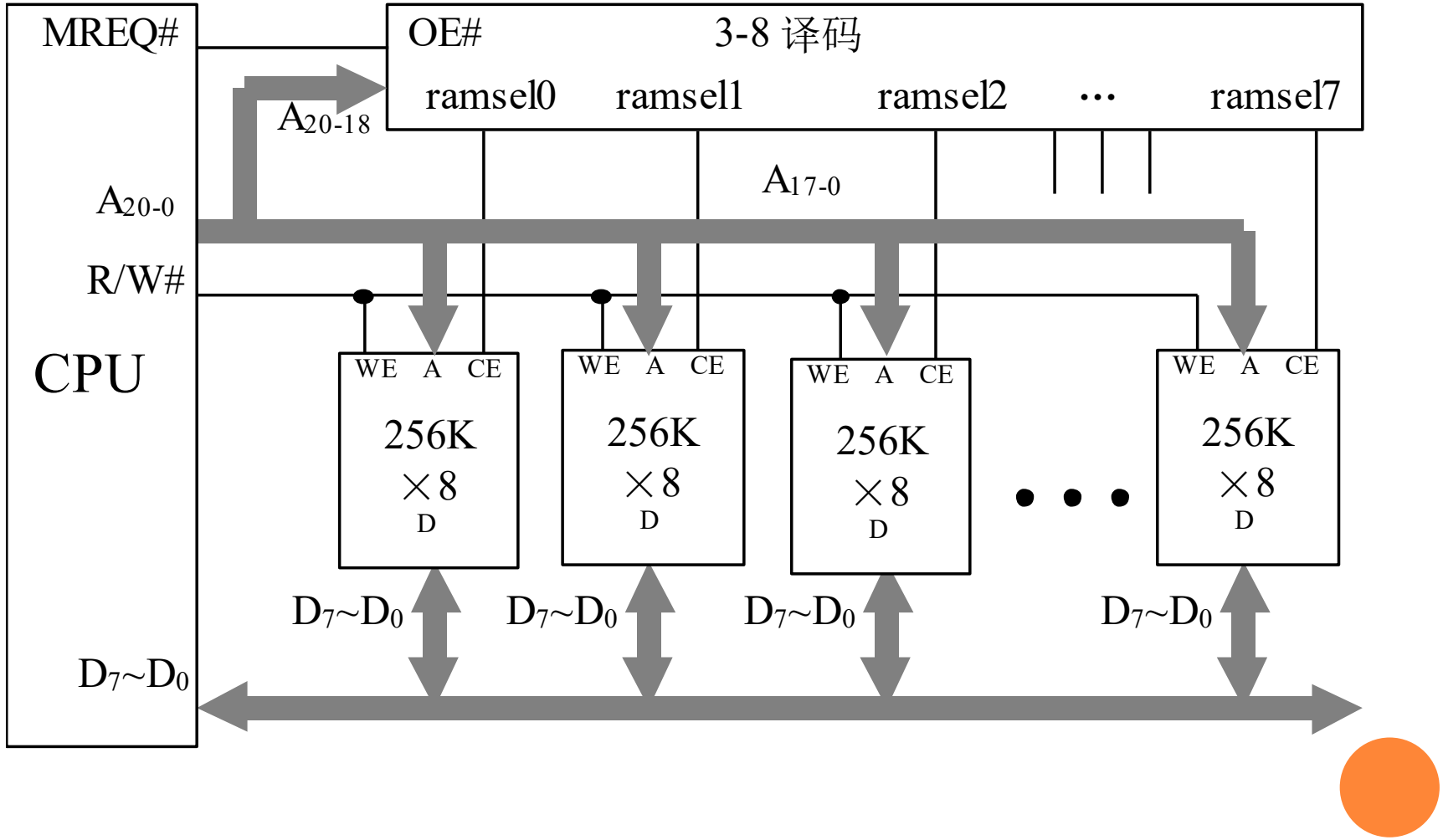
# 三、存储器容量的扩充

## 1、位扩展



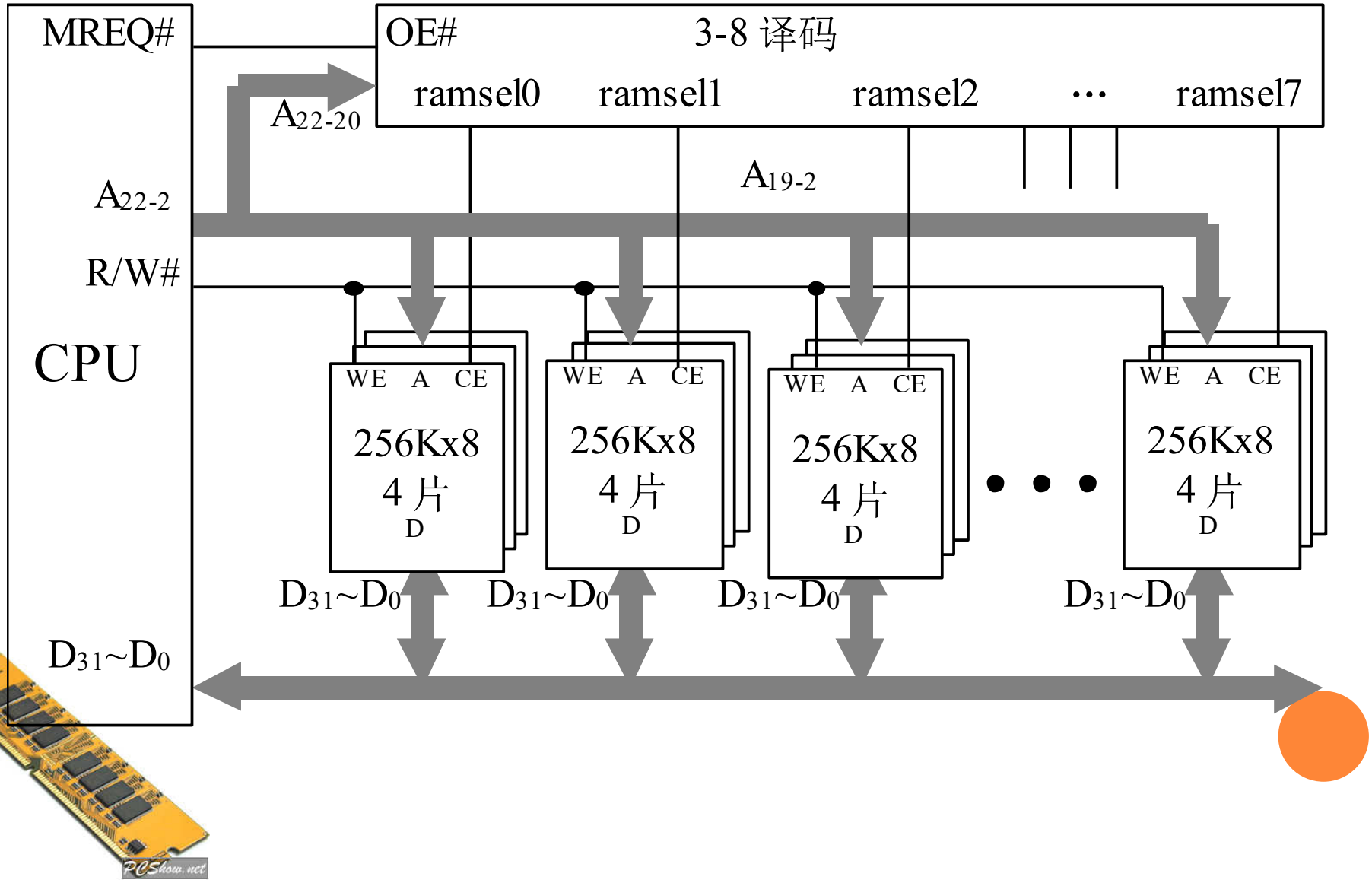
# 2、字扩展

- 地址、数据端、读写控制并联，片选区分各片地址



# 3、字位扩展：同时扩展位数和字数

## 例子：利用256K×8位的存储器芯片设计2M×32位的存储器



# 四、多模块交叉存储器

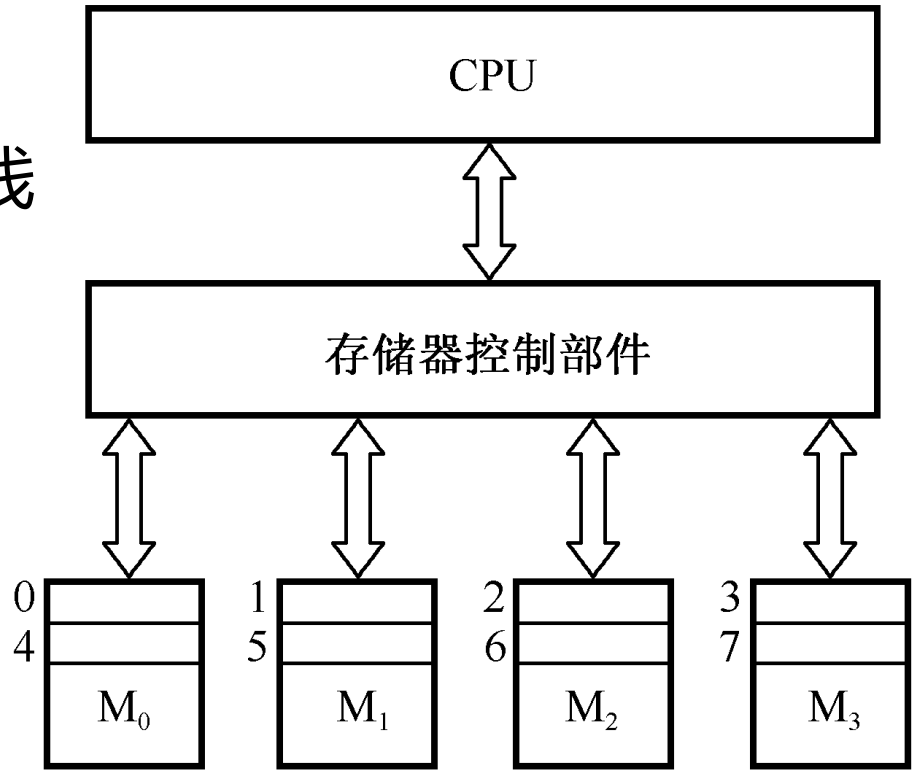
**1、设计思想：**由若干个存储模块组成的并行主存储器，各模块拥有独立的地址寄存器、译码电路、读写控制电路，各模块以同等方式与CPU交换信息。

**2、访问方式：**

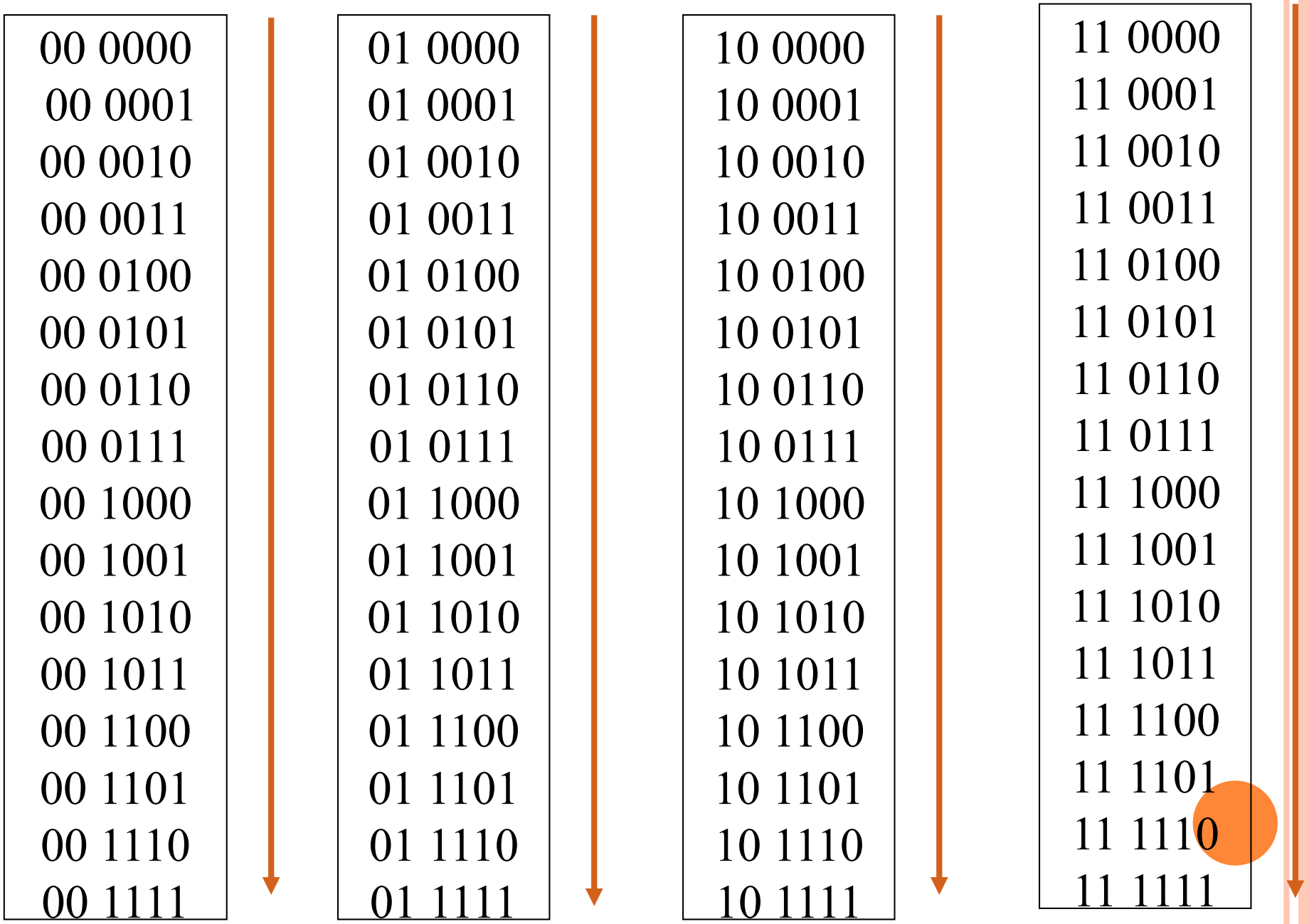
- 分时启动，分时使用总线

**3、编制发式：**

- 顺序编址
- 交叉编址



# 顺序编址：模块号 + 块内地址



# 交叉编址：块内地址 + 模块号

| M0 | M1 | M2 | M4 |
|---|---|---|---|
| 0000 00 00 | 0000 00 01 | 0000 00 10 | 0000 00 11 |
| 0000 01 00 | 0000 01 01 | 0000 01 10 | 0000 01 11 |
| 0000 10 00 | 0000 10 01 | 0000 10 10 | 0000 10 11 |
| 0000 11 00 | 0000 11 01 | 0000 11 10 | 0000 11 11 |
| 0001 00 00 | 0001 00 01 | 0001 00 10 | 0001 00 11 |
| 0001 01 00 | 0001 01 01 | 0001 01 10 | 0001 01 11 |
| 0001 10 00 | 0001 10 01 | 0001 10 10 | 0001 10 11 |
| 0001 11 00 | 0001 11 01 | 0001 11 10 | 0001 11 11 |
| 0010 00 00 | 0010 00 01 | 0010 00 10 | 0010 00 11 |
| 0010 01 00 | 0010 01 01 | 0010 01 10 | 001001 11 |
| 0010 10 00 | 0010 10 01 | 0010 10 10 | 001010 11 |
| 0010 11 00 | 0010 11 01 | 0010 11 10 | 001011 11 |

# 顺序编址和交叉编址的比较

- **有效并行条件：**连续访问的存储单元不在一个模块内。
- **顺序编址特点**
  - 便于扩充；
  - 可靠性高，某一模块故障，其他模块仍可继续工作；
  - 冲突严重：连续地址在同一模块内。
- **交叉编址特点**
  - 冲突下降；
  - 可靠性差，一个存储体故障将使所有程序无法工作。

    【很多程序分布在多个体内】
  - 扩充性差

# 五、CACHE存储器

## 1、Cache读写过程

- CPU发出读写请求，给出内存地址，打入内存地址寄存器；
- 主存 - Cache地址变换机构：判断Cache中是否存有副本；
- **读操作处理：**
  - 命　中：从Cache中读出送给CPU
  - 不命中：从内存读数据，替换缓存内容
- **写操作处理**
  - 全写法
  - 写回法：
  - 写一次法
- **CPU与Cache之间的数据传送是以字为单位**
- **主存与Cache之间的数据传送是以块为单位**

- **全写法**
  - 命　中：缓存、内存同时修改；
  - 不命中：直接写内存，缓存内容替换或不变
  - **特点：**产生不必要的内存写操作；简单，一致性好，适用于多处理机；
- **写回法：**
  - 命　中：只改缓存并加修改标记；撤出时写入内存；
  - 不命中：替换缓存内容，修改缓存并加修改标记；撤出时写入内存；
  - **特点：**复杂,高效,不适合多处理机。
- **写一次法：奔腾的cache写策略**
  - 第一次写命中：采用全写法策略，供其他cache监听处理；
  - 其他写操作：采用写回法策略

# 2、主存与CACHE的地址映射

**三种映射方法**

- **全相联映射(fully associative mapping)**

  可以将一个主存块映射到任意Cache行

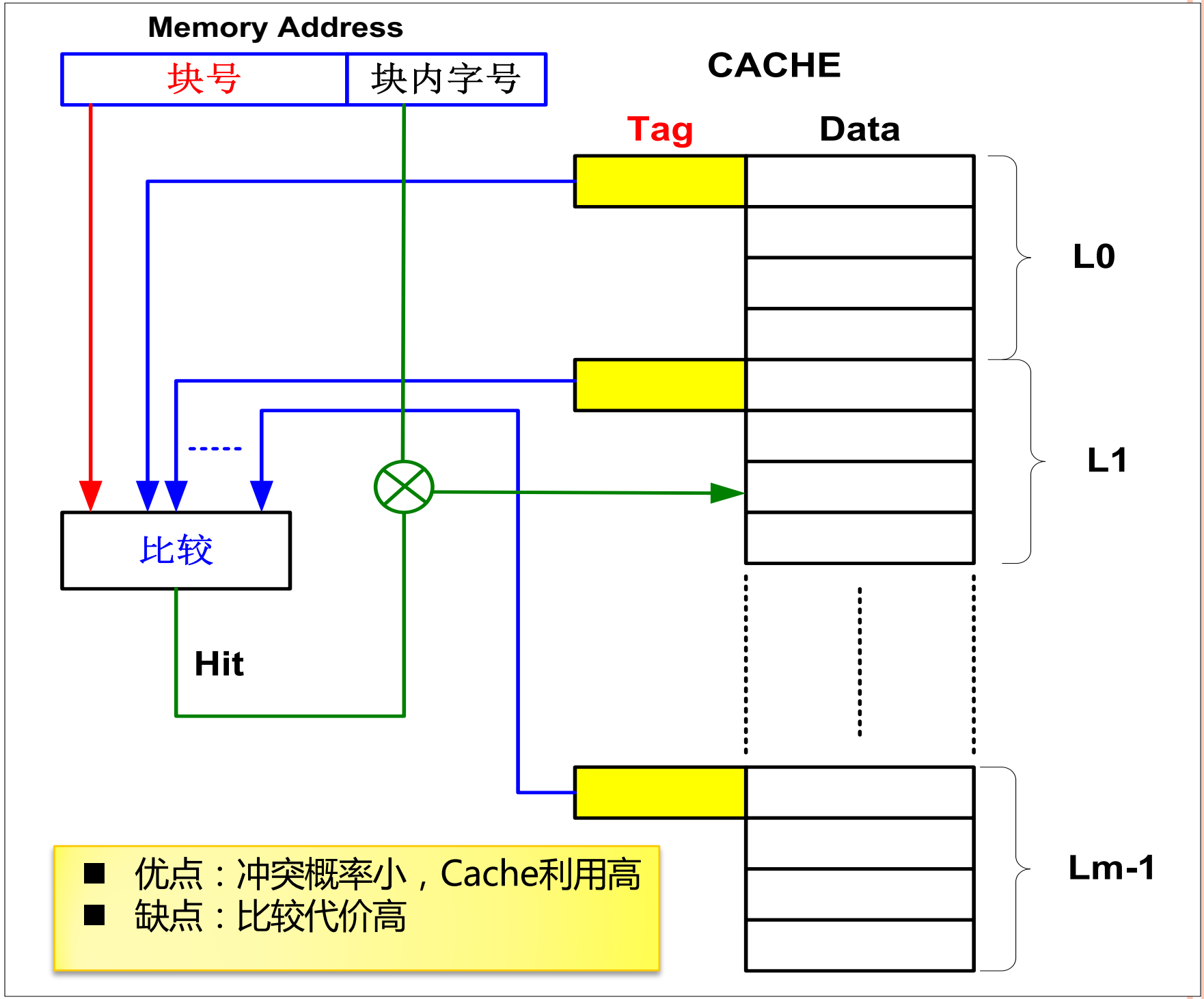
- **直接映射(direct mapping)**

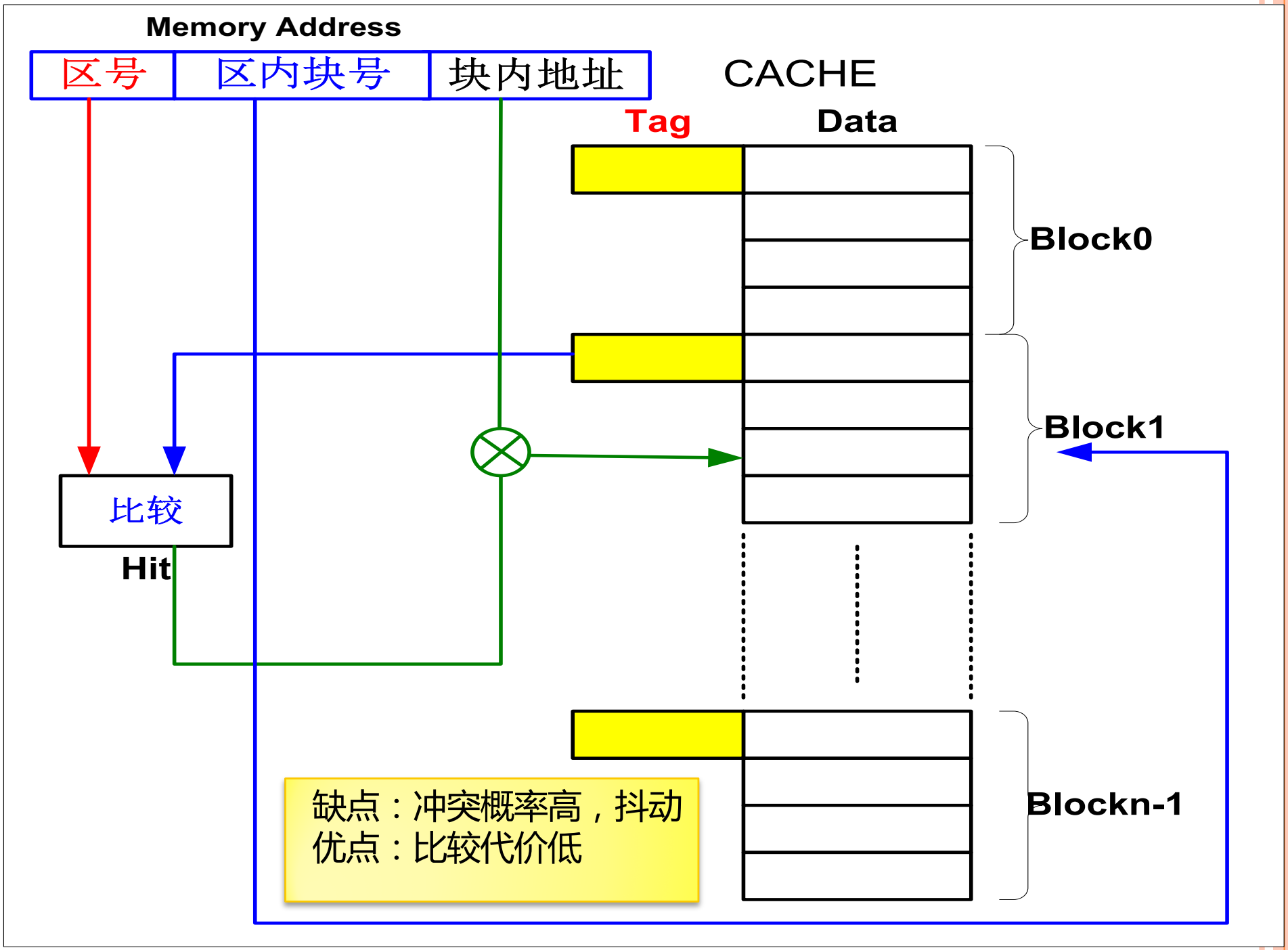
  只能将一个主存块映射到唯一Cache行

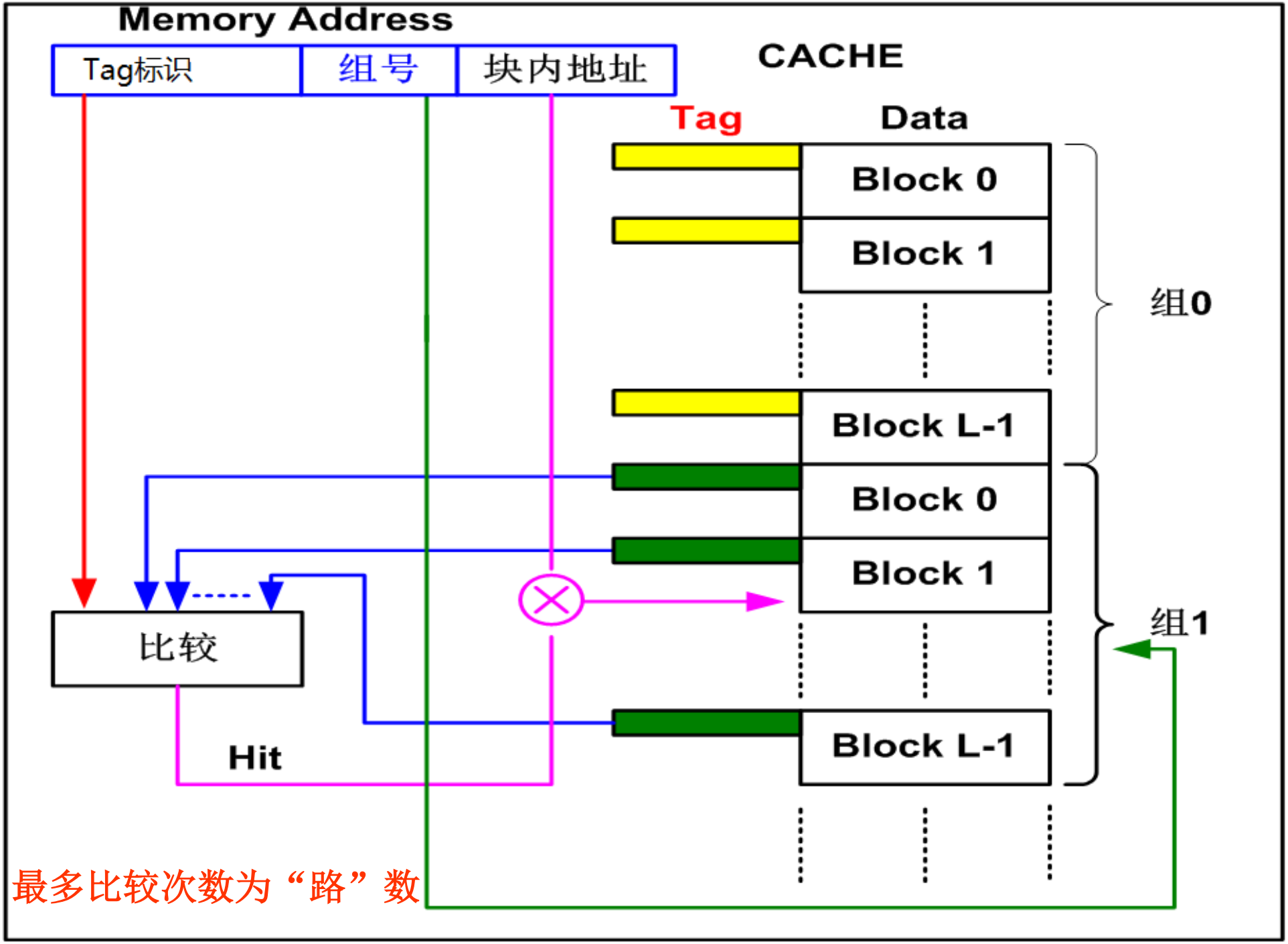
- **组相联映射(set associative mapping)**

  可以将一个主存块映射到唯一Cache组中的任意行

全相联映射过程


Memory Address
块号
块内字号
CACHE
Tag
Data
L0
L1
Lm-1
比较
Hit
■ 优点：冲突概率小，Cache利用高
■ 缺点：比较代价高


Memory Address
区号
区内块号
块内地址
CACHE
Tag
Data
Block0
Block1
比较
Hit
Blockn-1
缺点：冲突概率高，抖动
优点：比较代价低


Memory Address
Tag标识
组号
块内地址
CACHE
Tag
Data
Block 0
Block 1
Block L-1
Block 0
Block 1
Block L-1
组0
组1
比较
Hit
最多比较次数为“路”数

# 3、替换策略

## 1、LFU（最不经常使用 ）

- 访问Cache的某一行时，对应计数器增1；
- 替换策略：选择计数值最小的特定行，其他特定行计数器清零。
- 缺点：不能反映cache的近期访问情况。

## 2、LRU（近期最少使用）

- 访问Cache的某一行时，对应计数器置0，其他行的计数器增加1；
- 替换策略：选择计数值最大的特定行；
- 优点：符合cache的工作原理

## 3、随机替换：从特定行位置随机选取一行换出。

- 优点：硬件上实现容易，速度比前两种策略快。
- 缺点：随意换出的数据可能马上要使用，从而降低命中率和cache工作效率。但这个不足随着cache容量增大而减小。随机替换策略的功效只是稍逊于前两种策略。

# 五、虚拟存储器

## 1、页式虚拟存储器

- 逻辑页和物理页
- 页表机制：一个程序对应一张页表，由OS创建和维护。
- 长页表问题
  - 分页，一部分在主存，一部分在辅存；
  - 二级页表结构：页目录表、页表
  - 反向页表：每个物理页对应一个表项（包括逻辑页号），需要遍历反向页表以判断某虚页是否在内存。
- 快页表：加快地址映射
- 外页表问题：存在辅存中，用于“缺页”处理

## 页表中的其他信息

**① 装入位(有效位)P**

- P=1，该虚页已装入内存，CPU访问有效；
- P=0，该虚页未装入内存，CPU访问无效；检查有无空页；
  - 有空页：启动I/O系统，把该虚页调入主存，P置“1”；
  - 无空页：按替换策略将内存某页变为空页，再做上述操作。

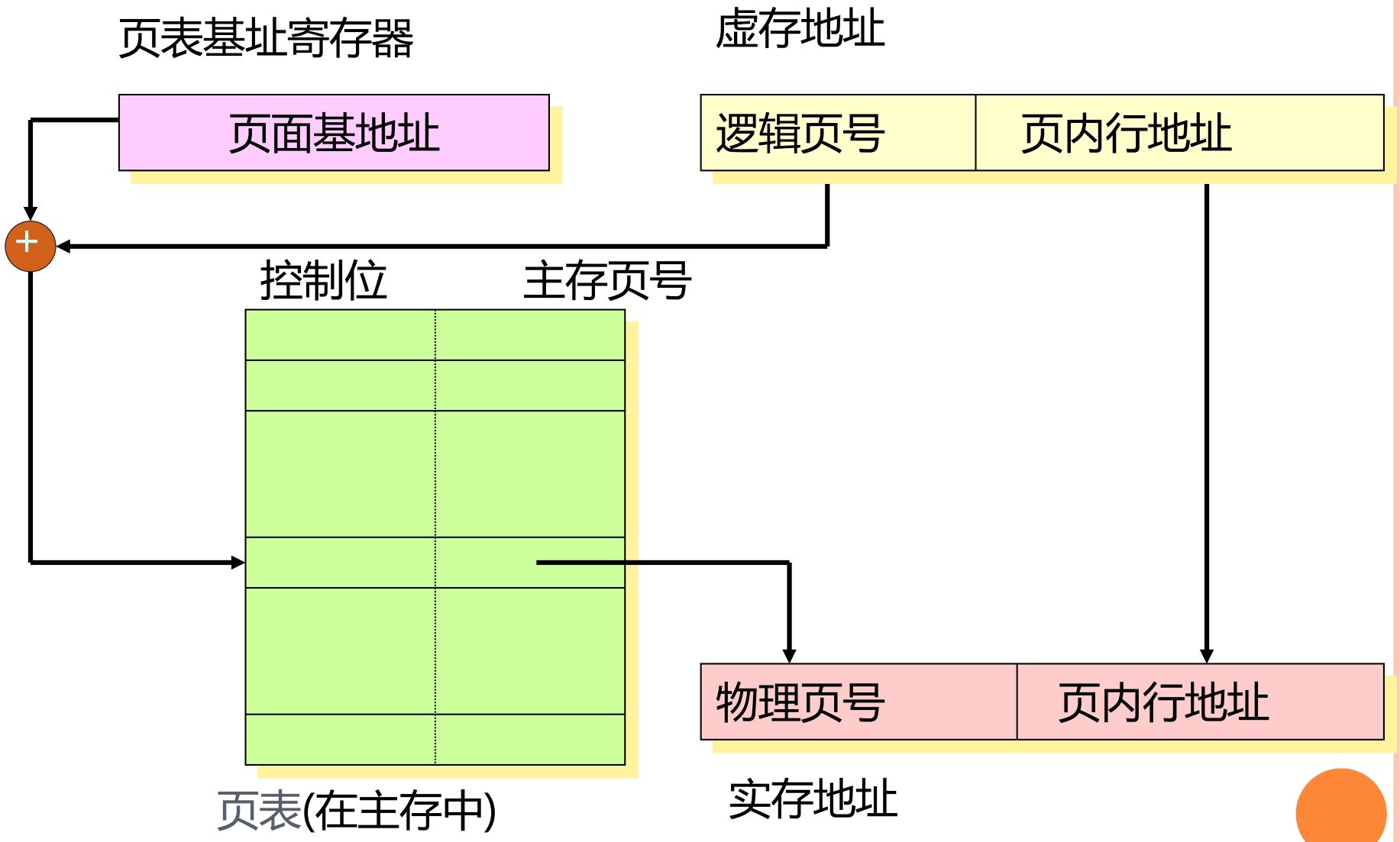
**② 修改位C**：记录虚页内容在主存中是否被修改过。

- 若修改过，则该虚页从内存撤出时，要把修改部分写回辅存；

**③ 替换控制位**：反映该虚页在内存中的活跃程度。

**④ 其他**：如访问权限控制等。

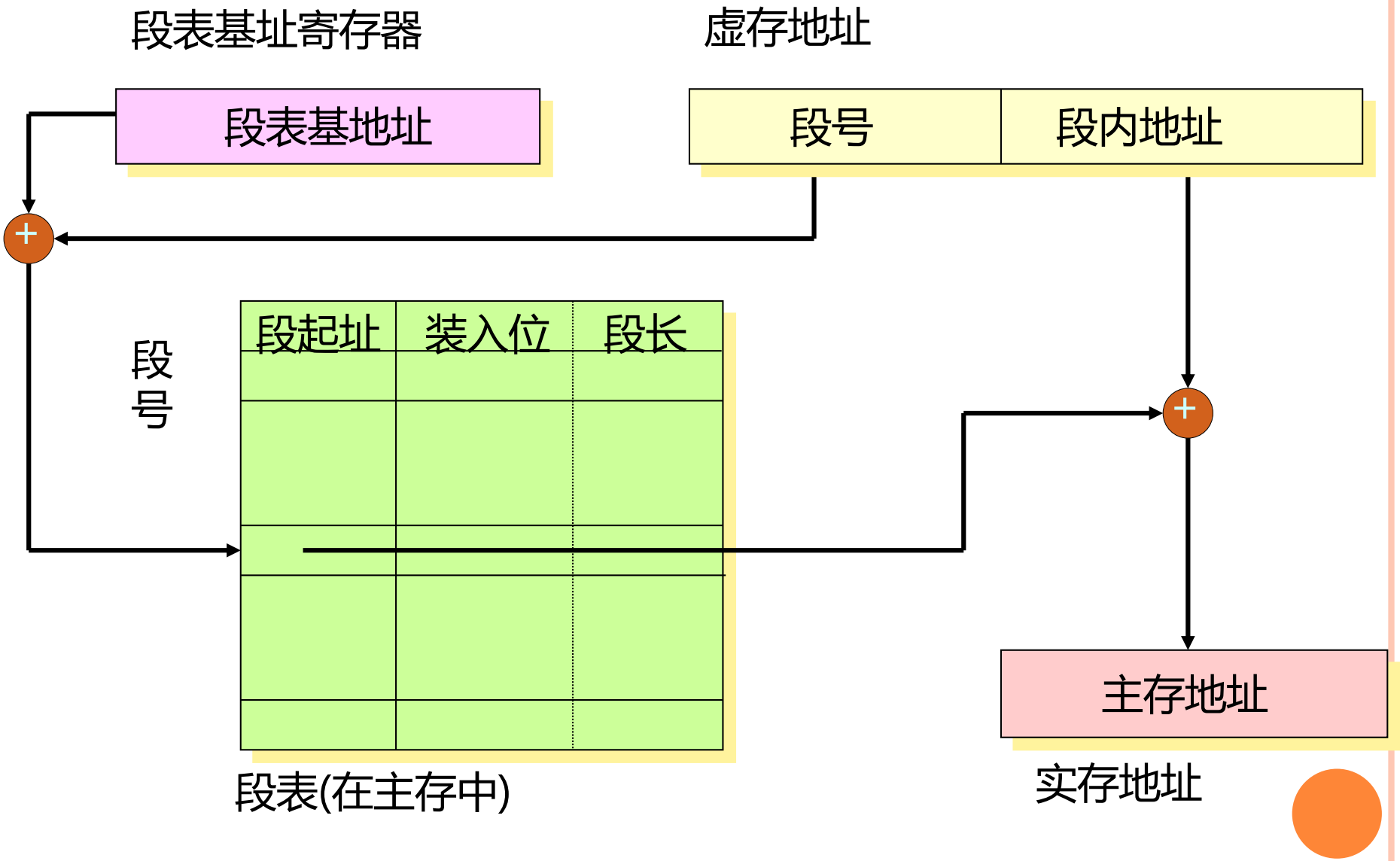
# 地址映射机制


页表基址寄存器
虚存地址
页面基地址
逻辑页号
页内行地址
+
控制位
主存页号
物理页号
页内行地址
页表(在主存中)
实存地址

# 2、段式虚拟存储器

- **适应模块化编程技术**
- **方法**
  - 程序不再机械的按固定长度分页，而是随程序的逻辑结构分段；
  - 一个段可以是一个模块，一个子程序，一个数组，或一张表格；
  - 各段长度不同，段内独立编址（相对地址）
  - 每段程序装入内存的一段连续单元内；
- **段表**：用于虚 - 实地址变换
- **优点**：命中率更高，便于模块化编程；

  **缺点**：内存分配策略复杂，易产生内存碎片，加重OS系统负担。


段表基址寄存器
虚存地址
段表基地址
段号
段内地址
+
段起址
装入位
段长
段号
+
主存地址
段表(在主存中)
实存地址

# 3、段页式虚拟存储器

- 结合了页式、段式优点的虚拟存储技术；
  - 将内存空间等分为页；
  - 将程序分成段，每一个段由若干页组成；
- 段表及页表
  - 每个程序一个段表
  - 每个段对应一个页表。