河海大学

计算机组成与体系结构

黄倩

huangqian@hhu.edu.cn

勤学楼4203

第3章 多层次的存储器

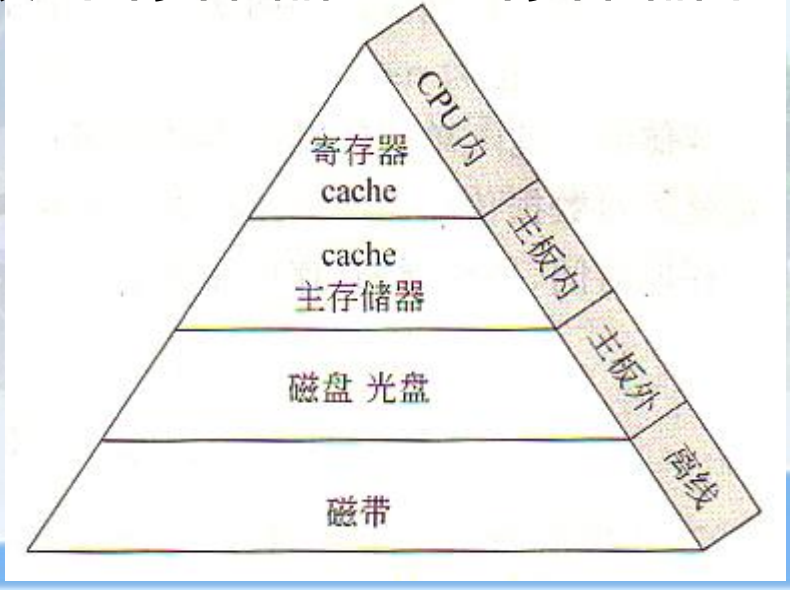
- 存储器概述
- SRAM存储器
- DRAM存储器
- 只读存储器和闪速存储器
- 并行存储器
- CACHE存储器
- 虚拟存储器
- 奔腾系列机的虚存组织

3.1 存储器概述

- 3.1.1 存储器的分类
  - 按存储介质分类：半导体存储器、磁表面存储器
  - 按存取方式分类：
    - 随机存储器、顺序存储器、半顺序存储器
  - 按存储内容的可变性分类
    - 随机读写存储器（RAM）、只读存储器（ROM）
  - 按信息的易失性分类：
    - 易失性存储器、非易失性存储器
  - 按系统中的作用分类
    - 内部存储器、外部存储器；
      主存储器、高速缓冲存储器、辅助存储器、控制存储器

3.1 存储器概述

- 3.1.2 存储器的分级
  - 对存储器的要求：容量大、速度快、成本低
  - 问题：在一个存储器中，难以兼顾这三个方面的要求
  - 解决方案：计算机系统中通常采用多级存储器体系结构，即高速缓冲存储器、主存储器和外存储器

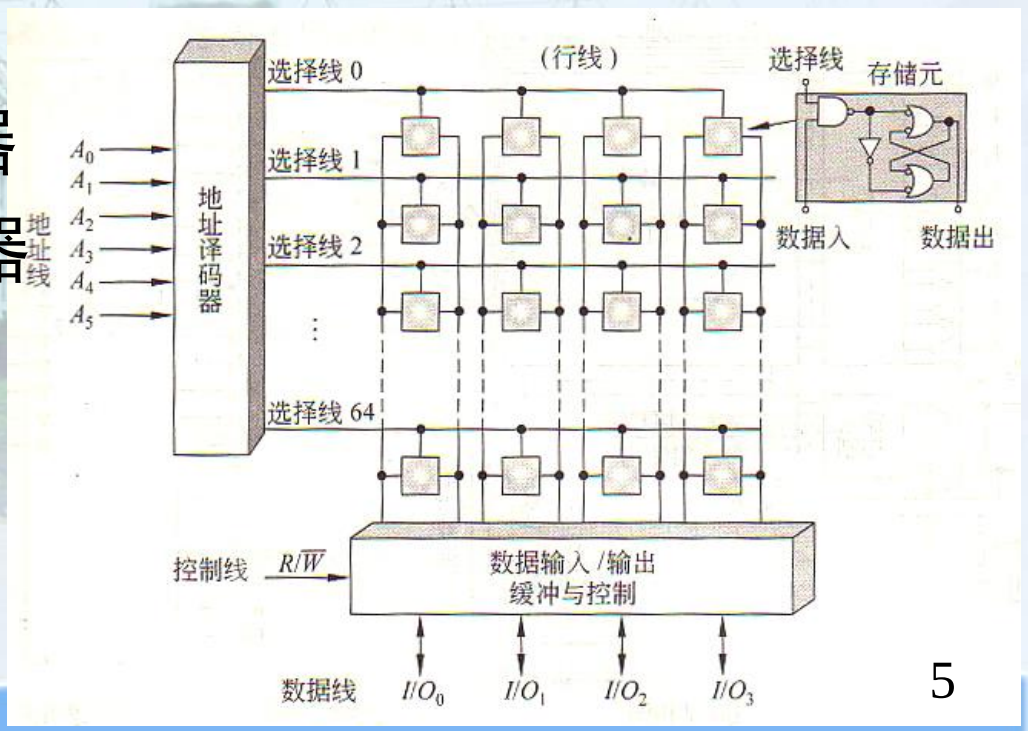


3.1 存储器概述

- 3.1.3 主存储器的技术指标
  - 存储容量：存储器中可以容纳的存储单元总数
    - MB、GB、TB、PB、EB、ZB、YB、BB、NB、DB…
  - 存取时间
    - 读操作时间：一次读操作命令发出到该操作完成、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，故称为存取时间
  - 存取周期/读写周期
    - 连续启动两次读/写操作所需间隔的最小时间
  - 存储器带宽：单位时间里存储器所能存取的信息量

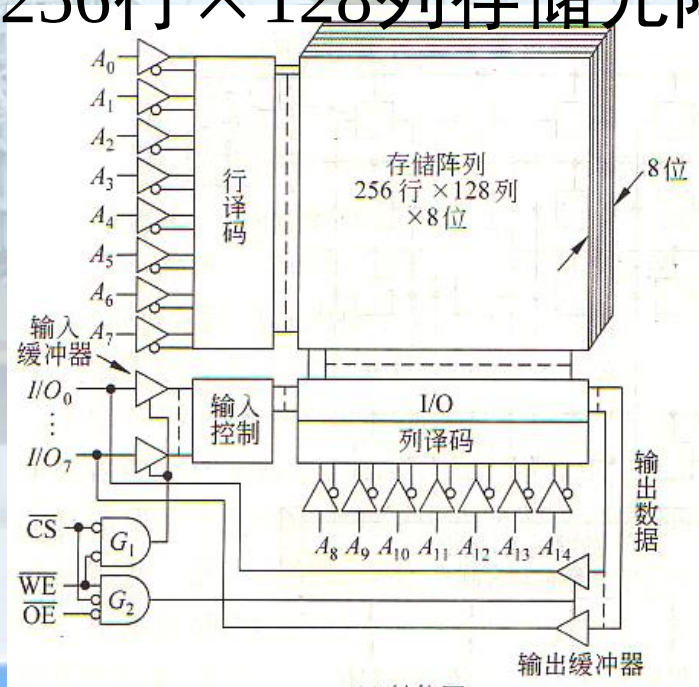
3.2 SRAM

- 3.2.1 基本的静态存储元阵列
  - 存储元：锁存器，存放一个二进制位；断电丢数据
  - 存储元阵列：64个存储单元，每个存储单元可存放4个二进制位
  - 地址线：CPU⇨存储器
  - 数据线：CPU⇔存储器
  - 控制线：R/$\overline{W}$



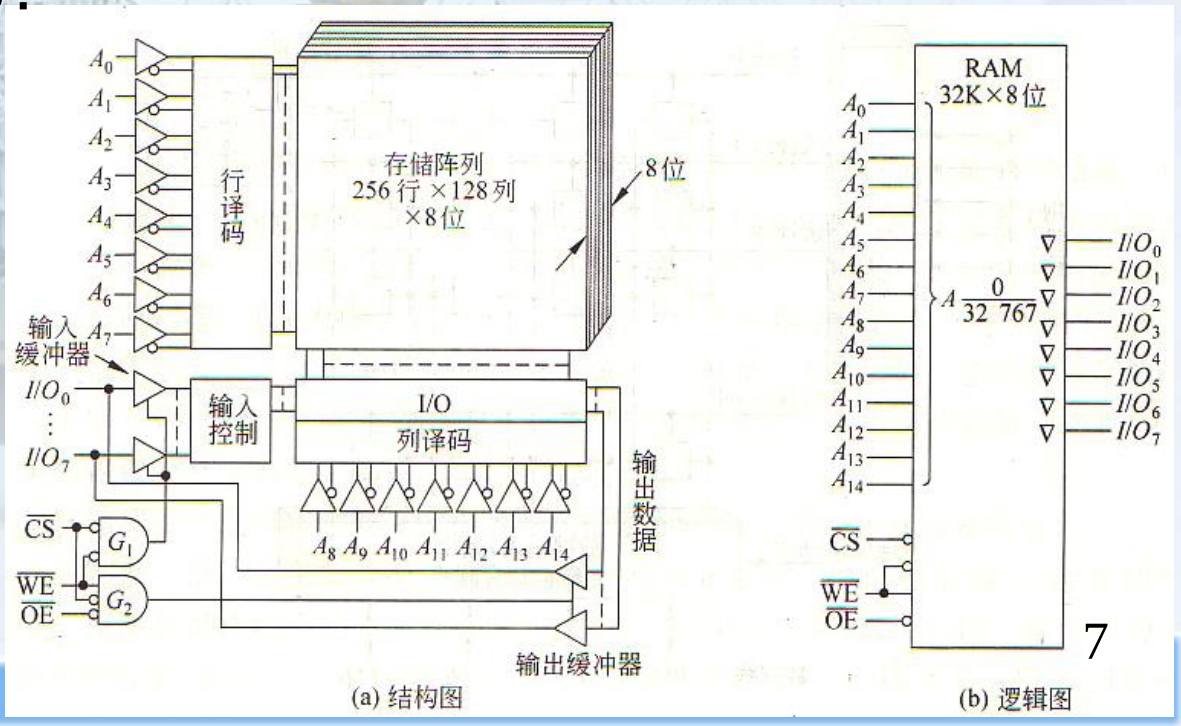
3.2 SRAM

- 3.2.2 基本的SRAM逻辑结构
  - 存储器容量：32K个存储单元，每个存储单元可存放8个二进制位
  - 存储元阵列：8个256行×128列存储元阵列面，相同位置点元同时I/O
  - 地址线：双译码
  - 数据线：$I/O_{0\text{-}7}$
  - 控制线：
    - $\overline{CS}$、$\overline{WE}$、$\overline{OE}$



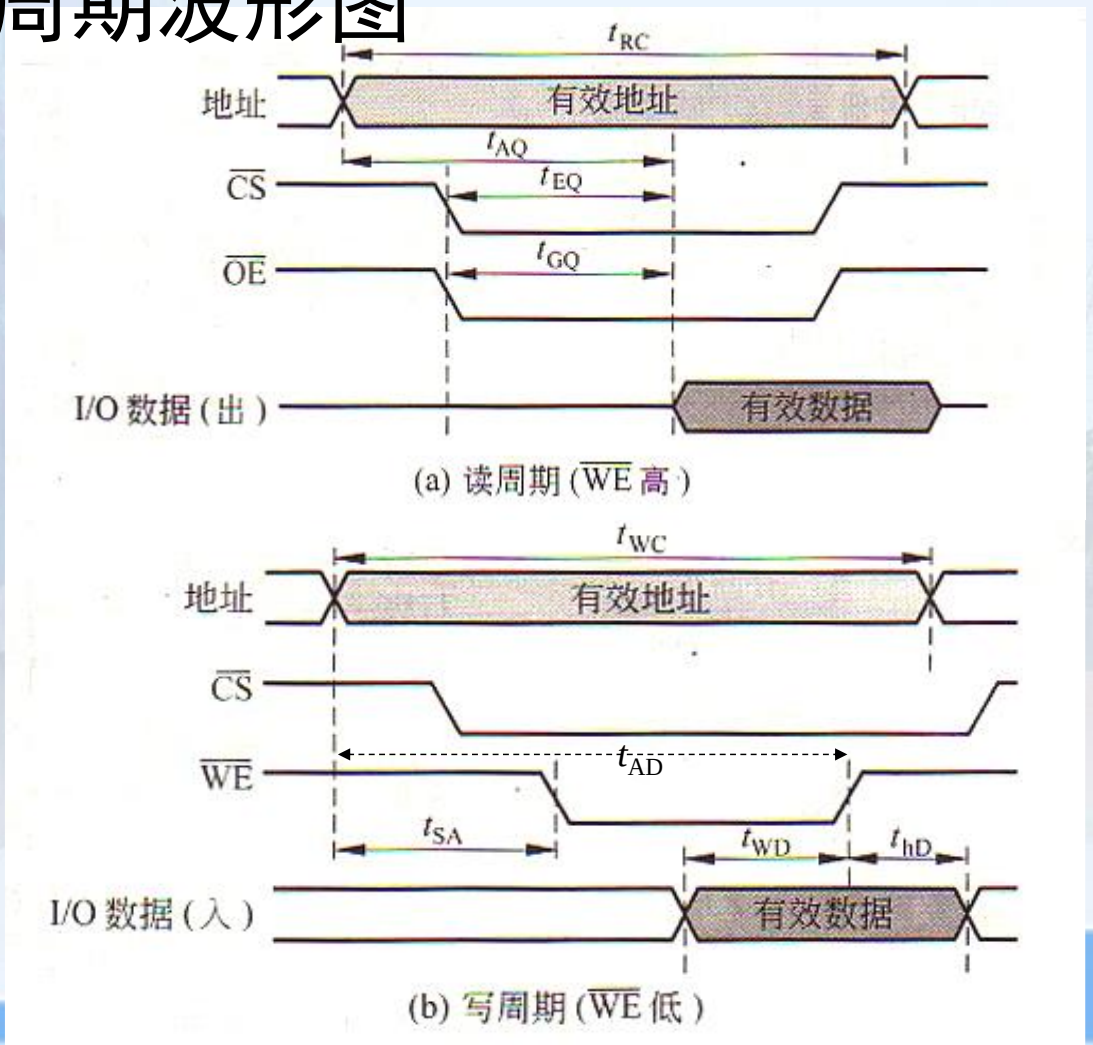
3.2 SRAM

- 3.2.2 基本的SRAM逻辑结构
  - 输出线的根数：15根输入线，单译码方式下2^{15}=32K根，双译码方式下2^8+2^7=384根。
    - 思考：为何取7+8？
    - 类比：数字图像
      - 单译码⇔一维
      - 双译码⇔二维



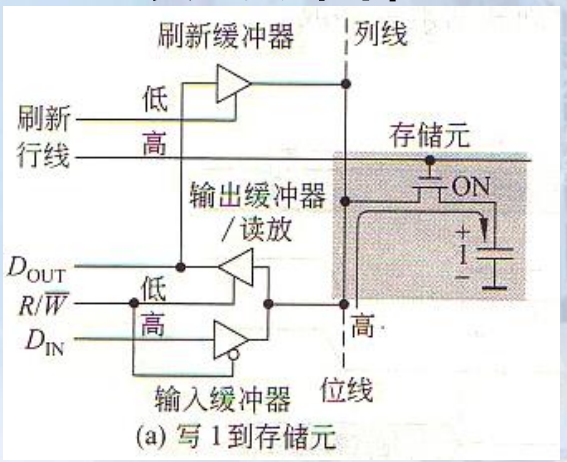
3.2 SRAM

- 3.2.3 读/写周期波形图
  - 读周期
    - t_{RC}
  - 读出时间
    - t_{AQ}
  - 写周期
    - t_{WC}
  - 写数时间
    - t_{AD}

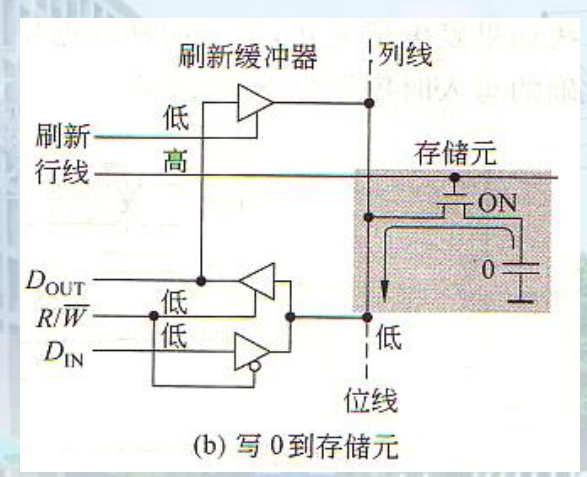


3.3 DRAM

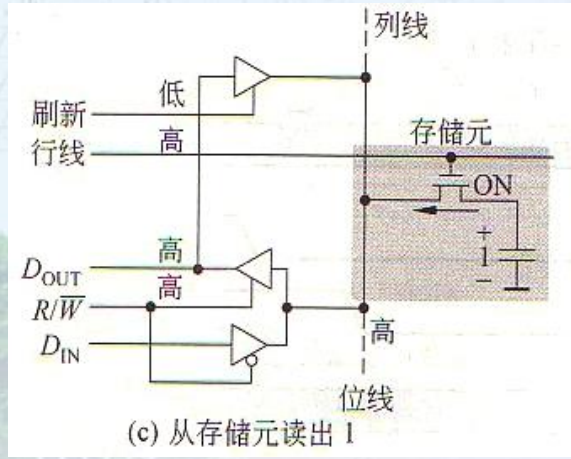
- 3.3.1 DRAM存储元的记忆原理
  - MOS管+电容器：电容器充满=1，电容器无电荷=0
  - 读写操作



(a) 写1到存储元



(b) 写0到存储元

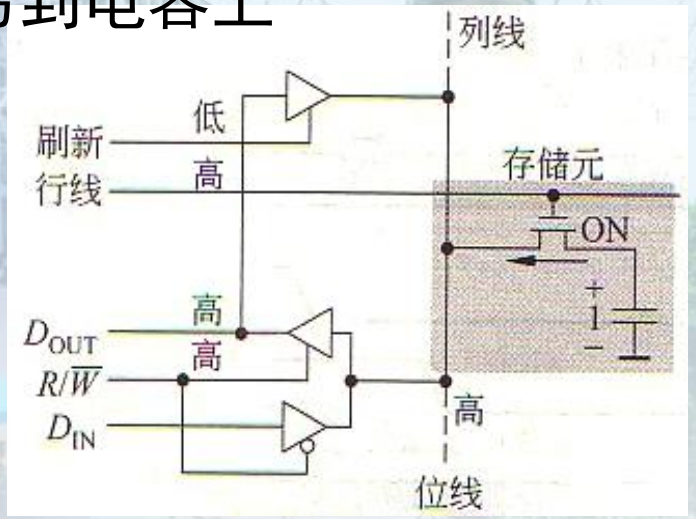


(c) 从存储元读出1

  - 为什么要刷新
    - MOS管、电容会有电荷泄漏
    - 读“1”是破坏性读出，会释放电荷

3.3 DRAM

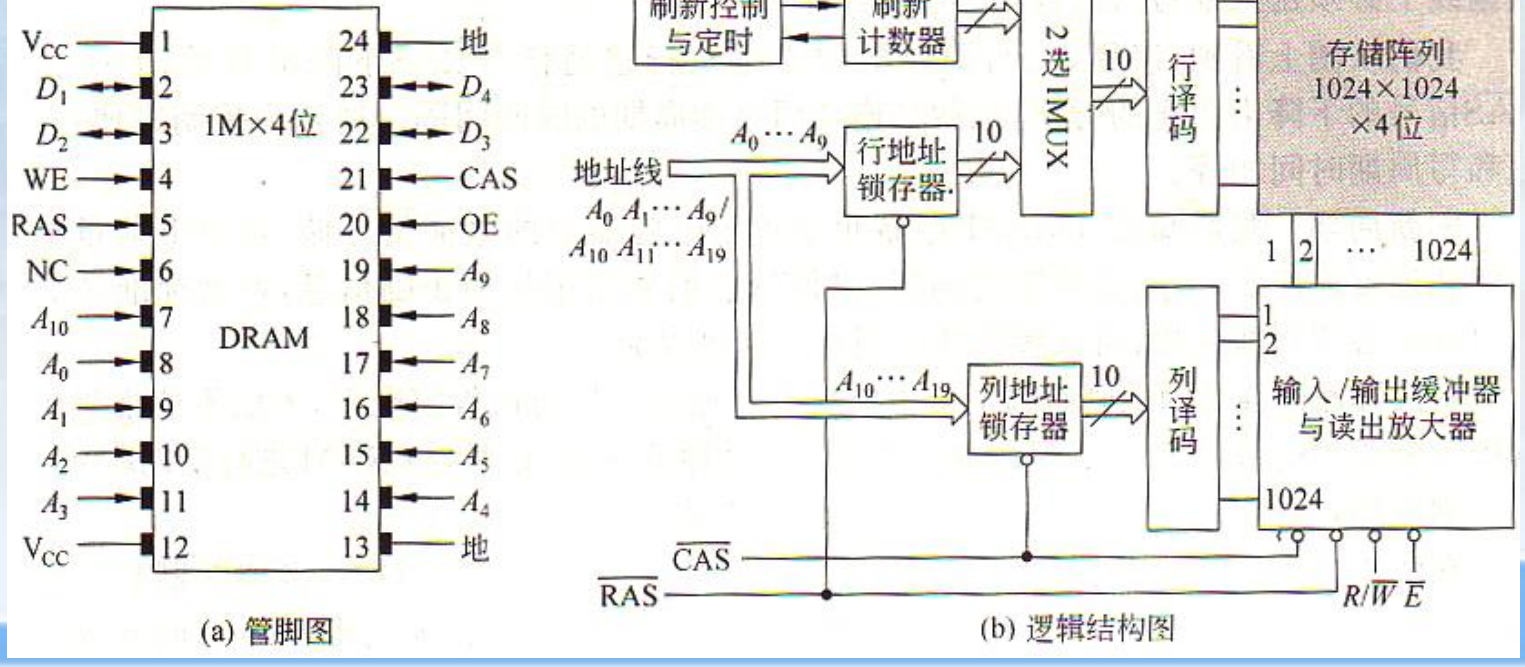
- 3.3.1 DRAM存储元的记忆原理
  - 怎么刷新
    - 读出D_{OUT}=1时刷新：D_{OUT}=1经过刷新缓冲器送到位线上，再经MOS管写到电容上



    - 定时刷新：按行刷新、规定时间范围内刷新全部存储元

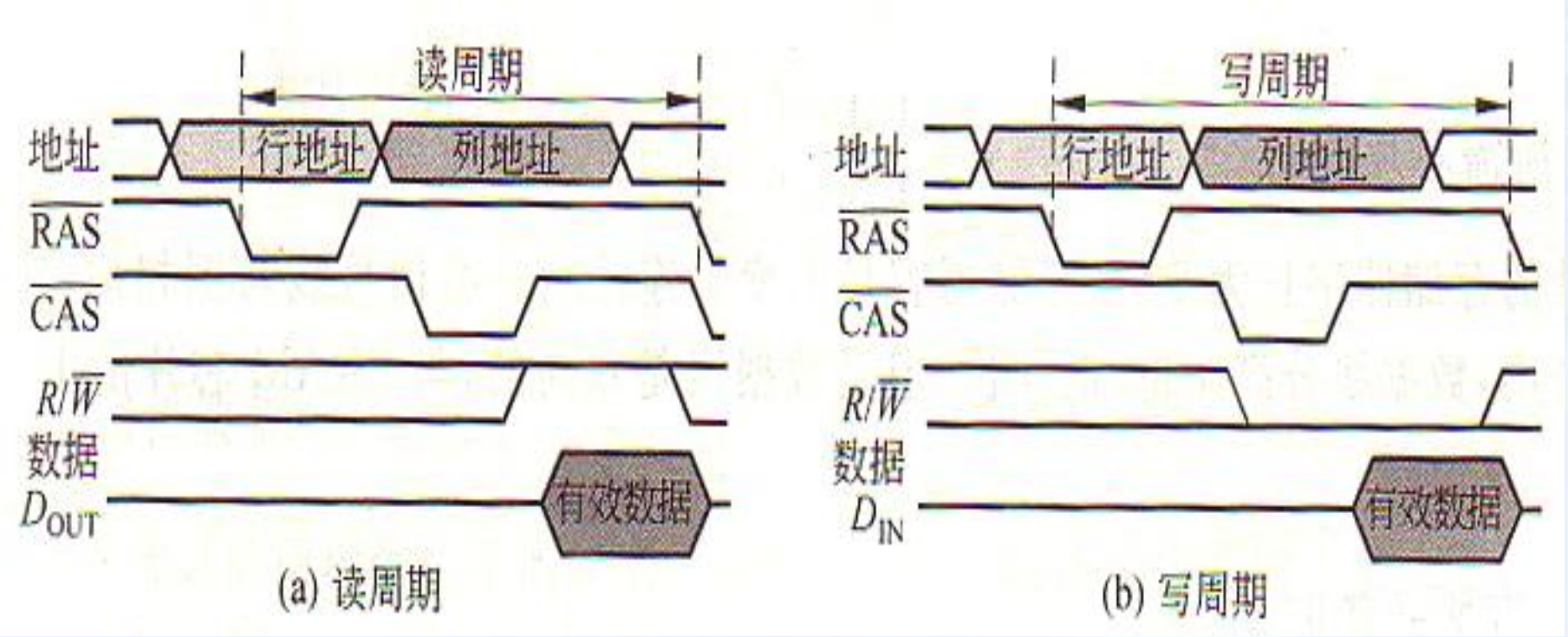
3.3 DRAM

- 3.3.2 DRAM芯片的逻辑结构
  - DRAM芯片示例：1M×4位，与SRAM相比增加了
    - (1)行/列地址锁存器：分时传送地址码
    - (2)刷新计数器



3.3 DRAM

- 3.3.3 读/写周期波形图
  - 注意行地址、列地址分2次输入的控制机制



3.3 DRAM

- 3.3.4 存储器容量的扩充
  - 字长位数扩展

例：利用1M×4位的SRAM芯片，设计一个存储容量为1M×8位的SRAM存储器。

解：

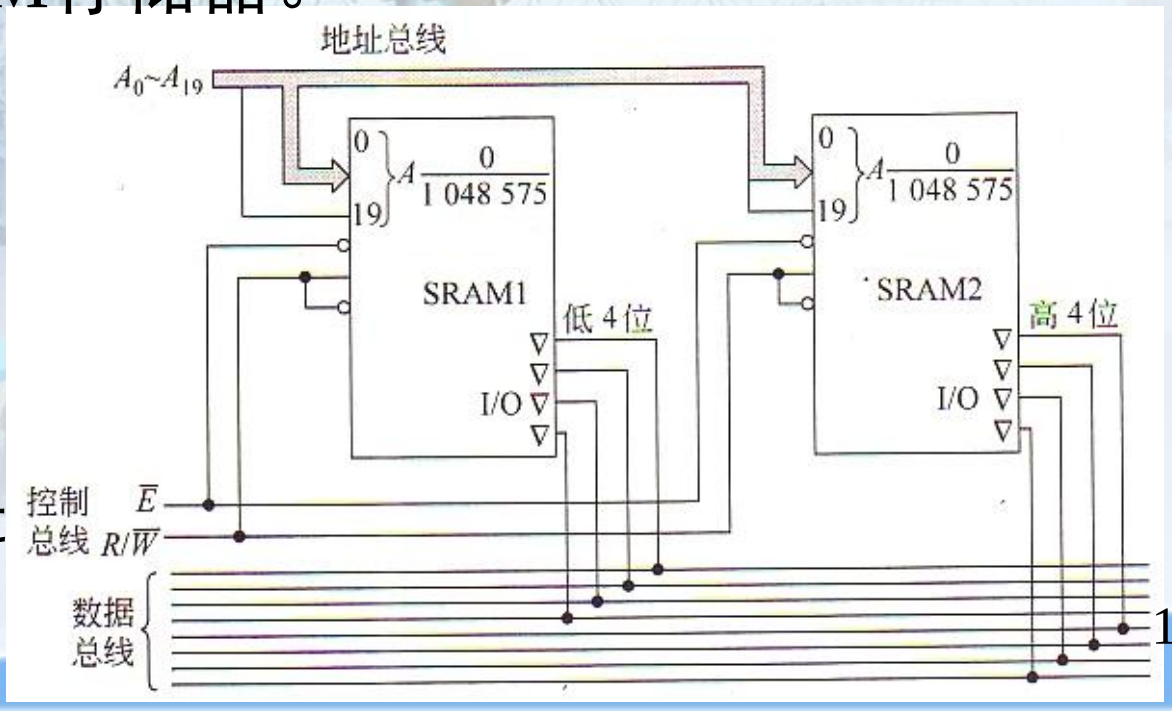
(1)计算芯片数

(2)“搭积木”

(3)写地址范围

(4)设计片选逻辑

(5)连线



3.3 DRAM

- 3.3.4 存储器容量的扩充
  - 字存储容量扩展

例：利用1M×8位的DRAM芯片，设计一个存储容量为2M×8位的DRAM存储器。

解：

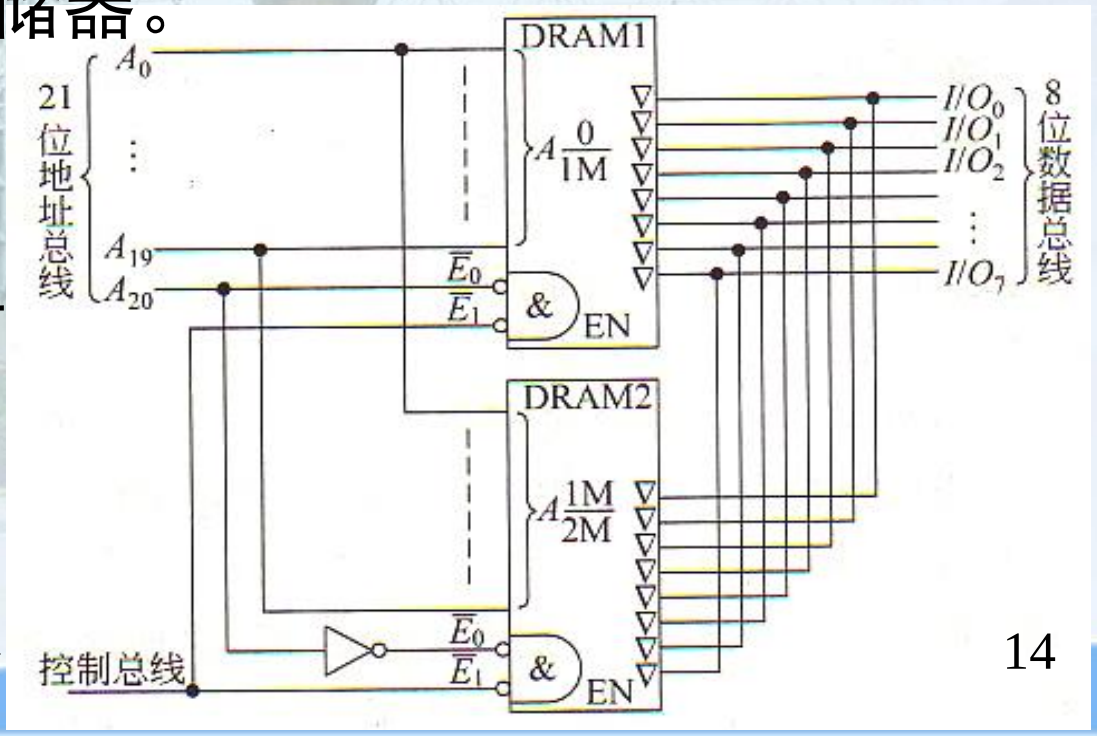
(1)计算芯片数

(2)构建地址空间分布

(3)写地址范围

(4)设计片选逻辑

(5)连接CPU与存储器



3.3 DRAM

- 3.3.4 存储器容量的扩充

例：设CPU的地址总线16根($A_{0\text{-}15}$，A_0为低位)，双向数据总线16根($D_{0\text{-}15}$)，控制总线中与主存有关的信号有$\overline{MREQ}$(允许访存，低电平有效)，R/$\overline{W}$(高电平为读命令，低电平为写命令)。

主存地址空间分配如下：0~8191为系统程序区（8K），由只读存储芯片组成；8192~32767为用户程序区（24K）；63487-65535为系统程序工作区（2K）。

现有如下存储器芯片：

EPROM：8K×8位(控制端仅有$\overline{CS}$)

SRAM：2K×8位，8K×8位（控制端有$\overline{CS}$、R/$\overline{W}$）

请从中选择适当芯片设计该计算机主存储器，画出主存储器逻辑框图，注意画出选片逻辑（可选用门电路及3:8译码器74LS138）与CPU的连接，说明选哪些芯片、选多少片。

3.3 DRAM

- 3.3.4 存储器容量的扩充

解：

(1)计算芯片数：

系统程序区8K，需要(8K×16位)/(8K×8位)=2片EPROM

用户程序区24K，需要(24K×16位)/(8K×8位)=6片SRAM

系统程序工作区2K，需要(2K×16位)/(2K×8位)=2片SRAM

(2)构建地址空间分布：

0 8191	8K（EPROM）	8K（EPROM）
8192	8K（SRAM）	8K（SRAM）
	8K（SRAM）	8K（SRAM）
32767	8K（SRAM）	8K（SRAM）
	30K（空）	30K（空）
63488	2K（SRAM）	2K（SRAM）
65535		

3.3 DRAM

- 3.3.4 存储器容量的扩充

解：(3)写地址空间范围

8K（EPROM）1#	8K（EPROM）
8K（SRAM）2#	8K（SRAM）
8K（SRAM）3#	8K（SRAM）
8K（SRAM）4#	8K（SRAM）
30K（空）	30K（空）
2K（SRAM）5#	2K（SRAM）

0000 0000 0000 0000	0000H
0001 1111 1111 1111	1FFFH
0010 0000 0000 0000	2000H
0011 1111 1111 1111	3FFFH
0100 0000 0000 0000	4000H
0101 1111 1111 1111	5FFFH
0110 0000 0000 0000	6000H
0111 1111 1111 1111	7FFFH
1000 0000 0000 0000	8000H
1111 0111 1111 1111	F7FFH
1111 1000 0000 0000	F800H
1111 1111 1111 1111	FFFFH

3.3 DRAM

- 3.3.4 存储器容量的扩充

解：(4)设计片选逻辑

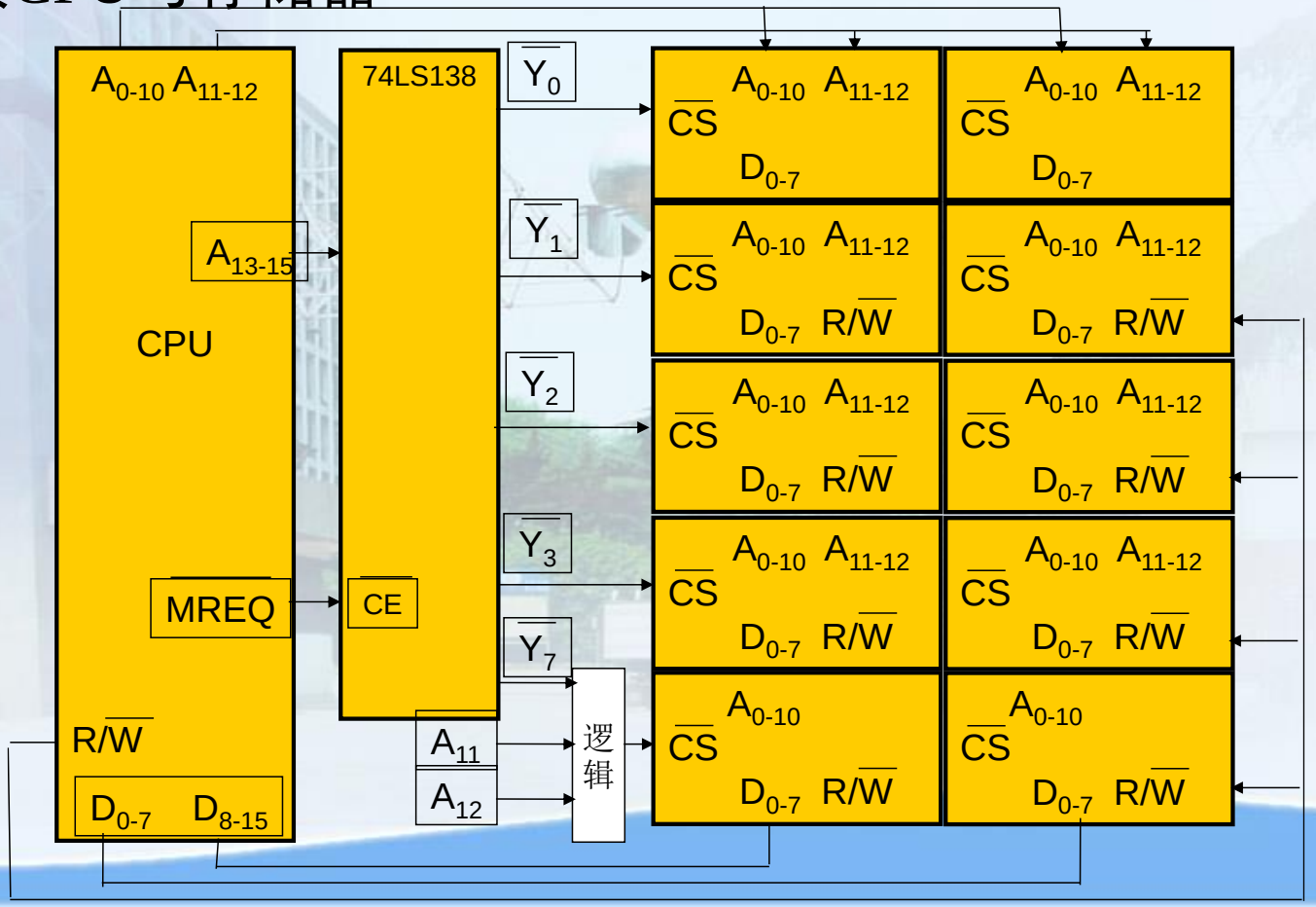
从二进制地址空间范围可以明显看出，采用A15、A14、A13三根地址线译码（74LS138），可以从地址上区分出各片组。

- 1#：$\overline{CS1}= \overline{A_{15}}\overline{A_{14}}\overline{A_{13}} =000 = \overline{Y0}$
- 2#：$\overline{CS2}= \overline{A_{15}}\overline{A_{14}}A_{13} =001 = \overline{Y1}$
- 3#：$\overline{CS3}= \overline{A_{15}}A_{14}\overline{A_{13}} =010 = \overline{Y2}$
- 4#：$\overline{CS4}= A_{15}\overline{A_{14}}\overline{A_{13}} =011 = \overline{Y3}$
- 5#：$\overline{CS5}= A_{15}A_{14}A_{13}A_{12}A_{11} =111\ 11= \overline{Y7}\ 11$

3.3 DRAM

- 3.3.4 存储器容量的扩充

解：(5)连接CPU与存储器



3.4 只读存储器和闪速存储器

只读存储器ROM

闪速存储器

3.4.1 只读存储器ROM

只读存储器特点：

正常工作情况下，只能读、不能写，读出的是事先存入的确定数据。

通过特定方式擦除，然后写入数据。

只读存储器由于工作可靠，保密性强，在计算机系统中得到广泛应用。

3.4.1 只读存储器ROM

1、一次性掩膜ROM

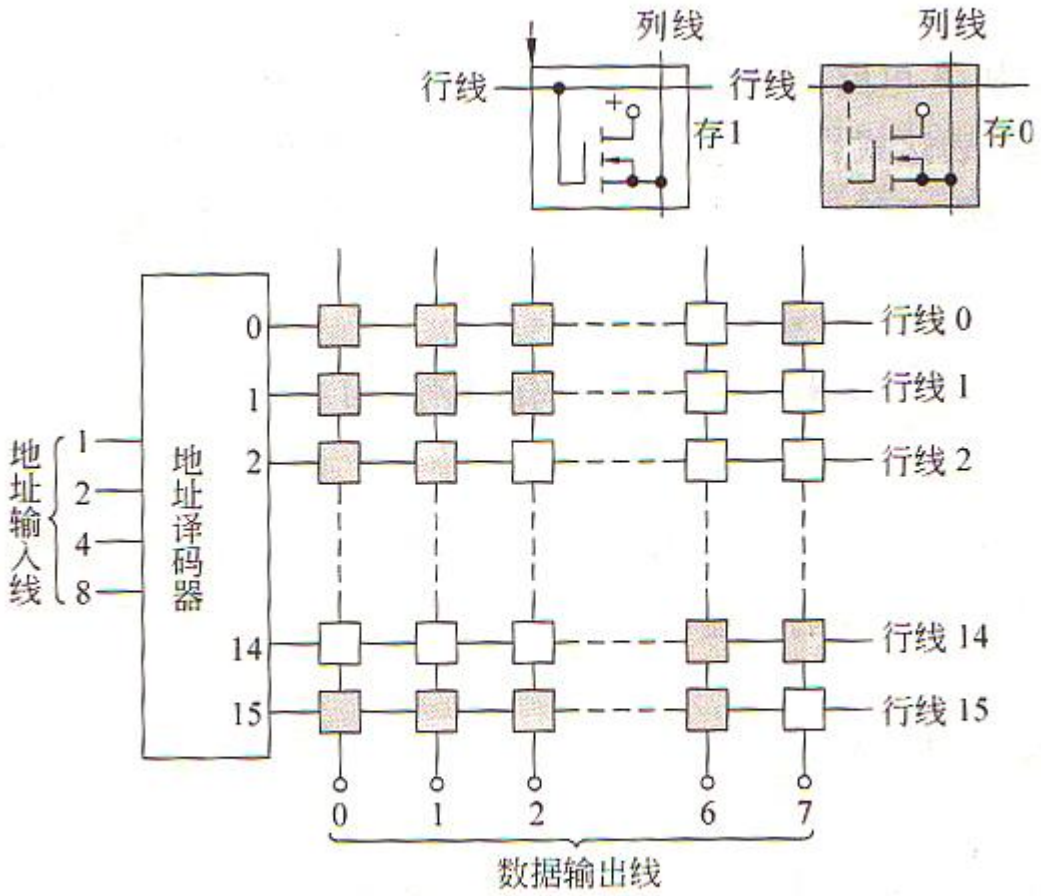


图 3.17　16×8 位 ROM 阵列结构示意图

（1）出厂前一次性烧成；

（2）MOS管的导通、截止；

（3）一旦烧成，不能改写；

000…10
000…11
001…11

111…00
000…01

3.4.1 只读存储器ROM

2、EPROM存储元

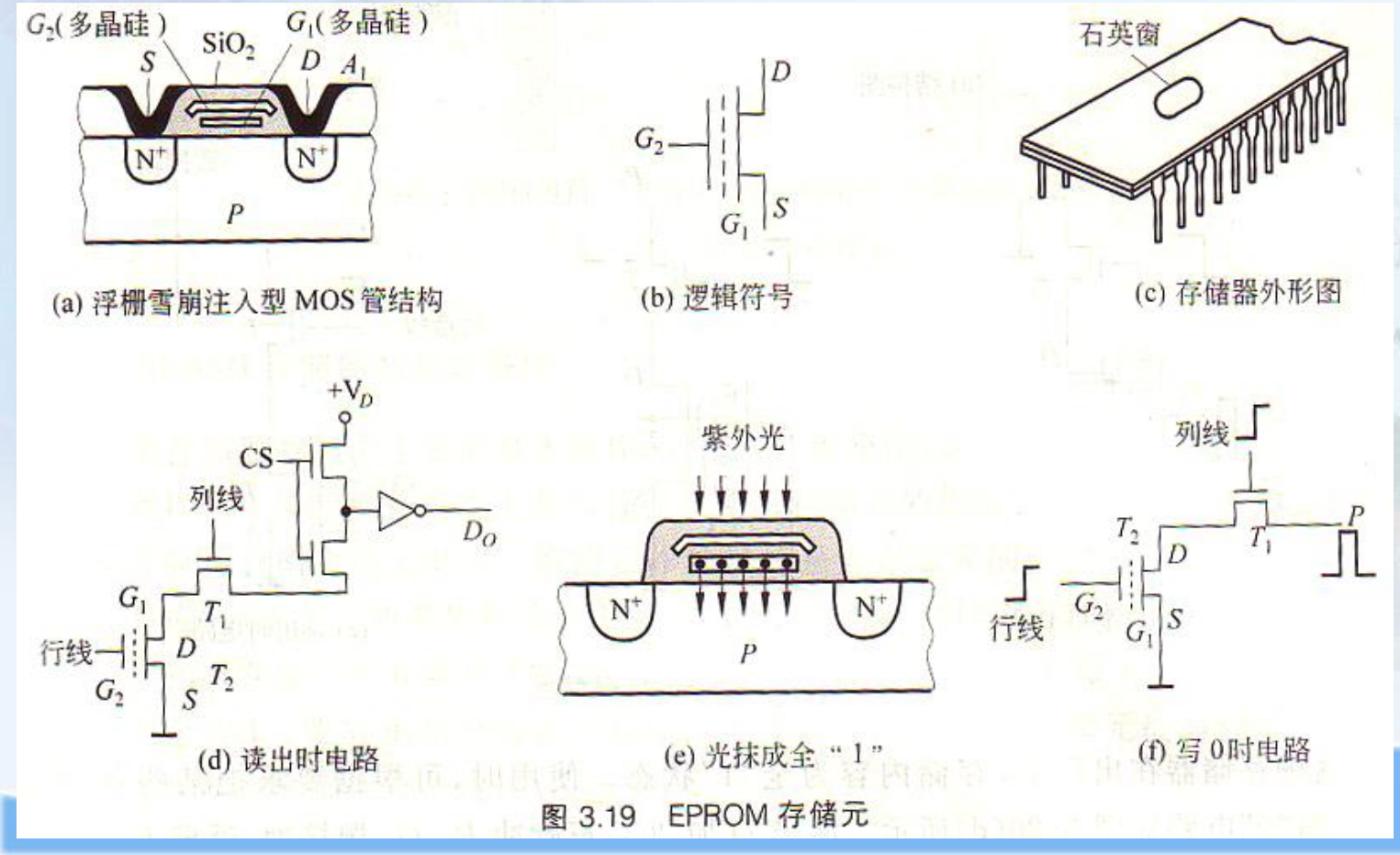
光擦除可编程可读存储器

（1）出厂时为全“1”

（2）几十伏高压写入“0”

（3）正常工作电压读出

（4）紫外线照射擦除、恢复为全“1”



3．4．1 只读存储器ROM

3、E²PROM存储元

电擦除可编程只读存储器。

（1）出厂时为全“1”

（2）几十伏高压写入“0”

（3）正常工作电压读出

（4）几十伏高压擦除、恢复为全“1”

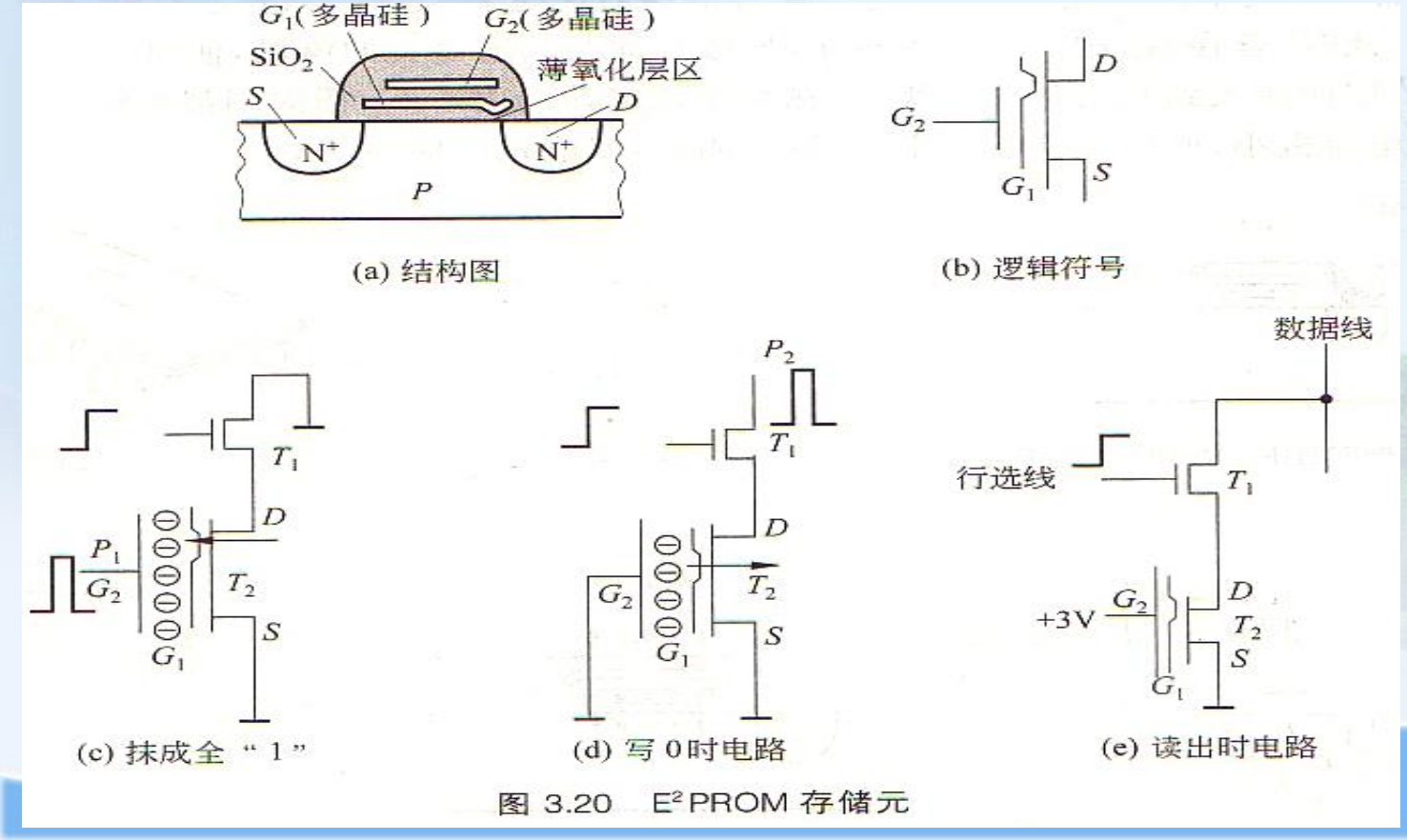
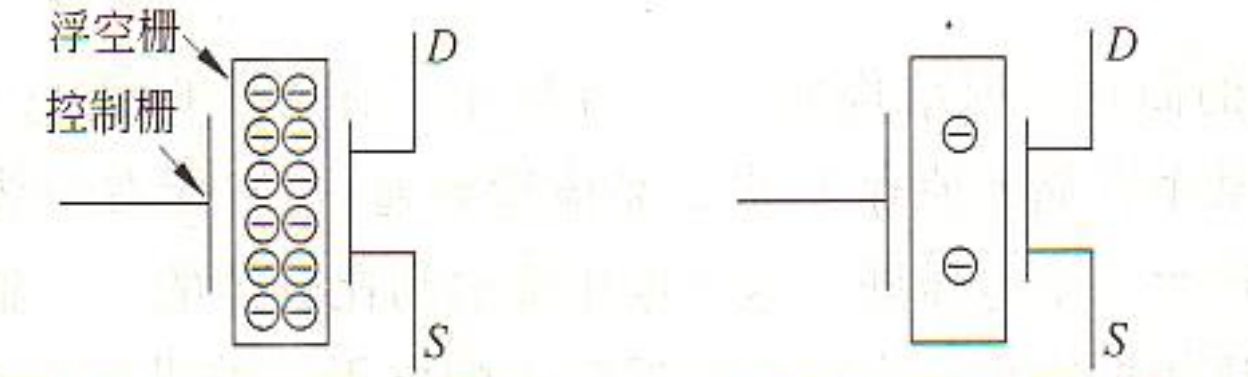


图 3.20 E²PROM 存储元

3.4.2 FLASH闪速存储器



许多电子(带负电荷),存储0　　少许电子(不带电荷),存储1

图 3.21　FLASH 存储元

高速高密度非易失性的读/写存储器

（1）高速：闪存，眨眼的功夫

（2）高密度：单个MOS管组成

（3）正常工作电压读/写/擦除，既有RAM的优点、又有ROM的优点

3.5 并行存储器

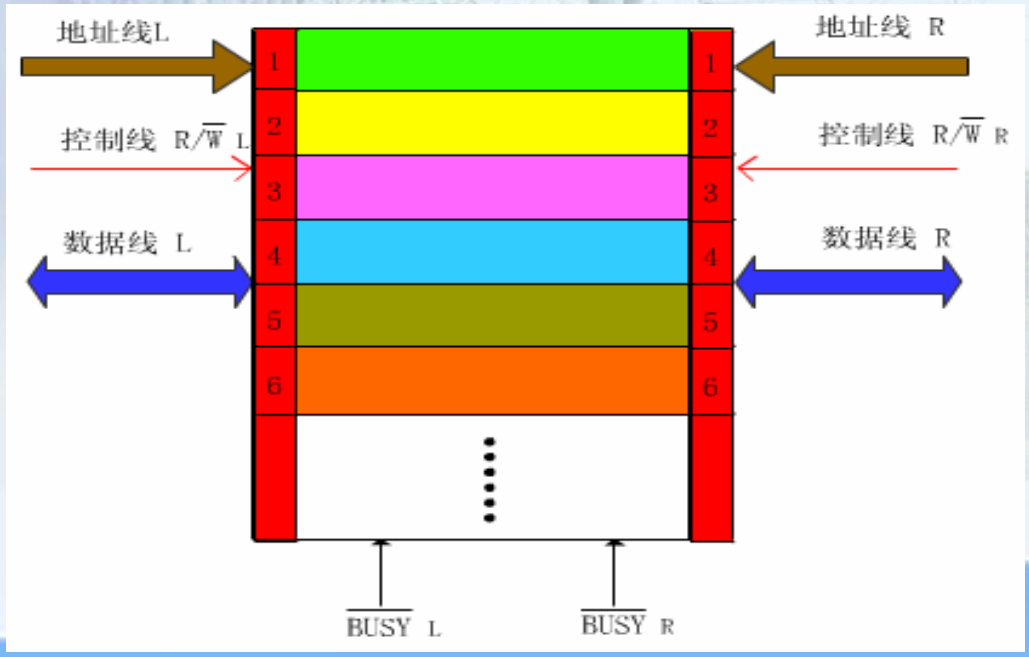
双端口存储器

多模块交叉存储器

3.5.1 双端口存储器

1、双端口存储器的逻辑结构

双端口存储器：同一个存储器具有两组相互独立的读写控制线路（即两个相互独立的访问端口），它们分别具有各自的地址线、数据线和控制线，可以进行并行的独立操作。



3.5.1 双端口存储器

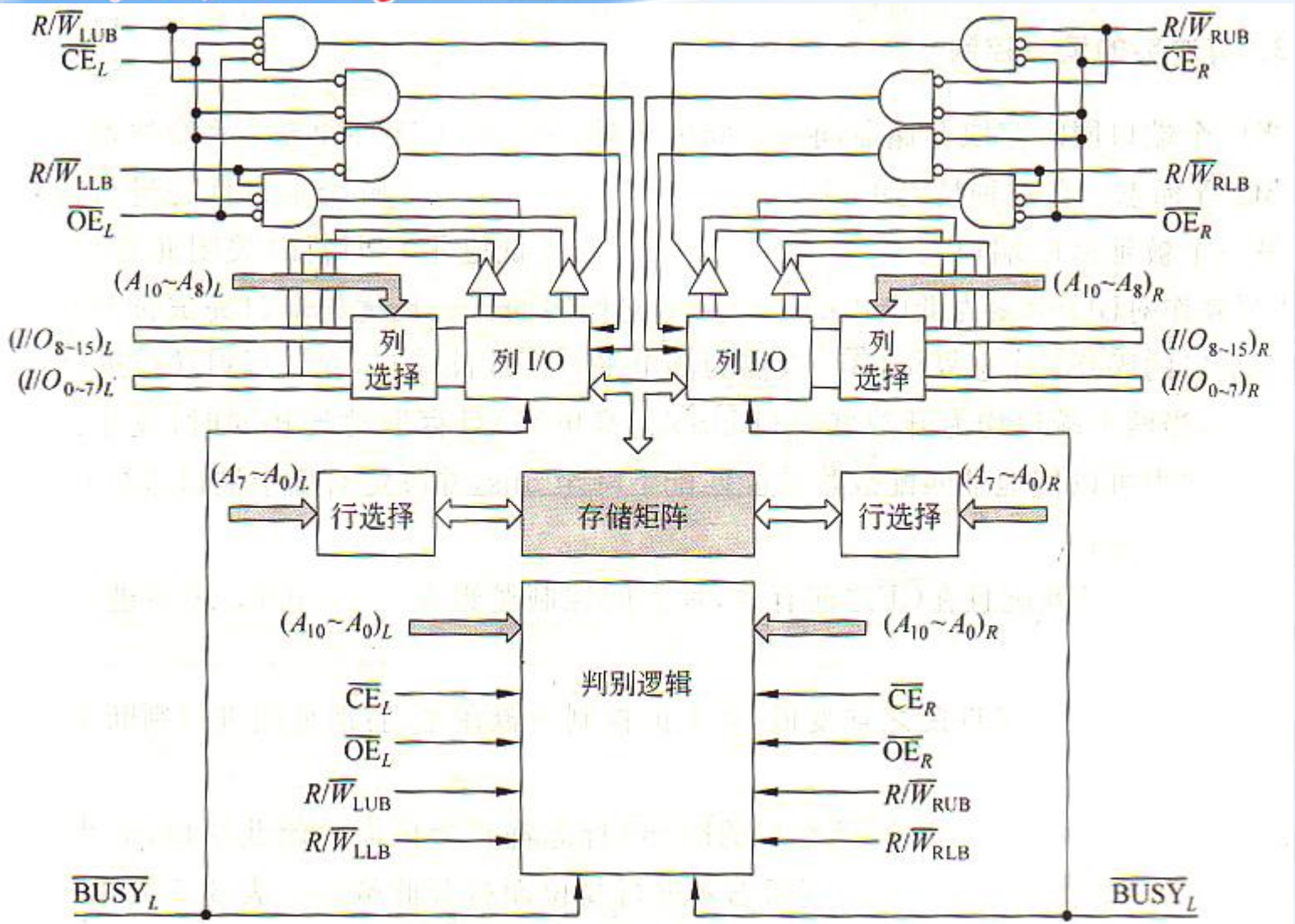


图 3.24　双端口存储器 IDT7133 逻辑框图

3.5.1 双端口存储器

2、无冲突读写控制 ■

当只有任意一个端口访问存储器时，当然它可以对整个存储器的任何单元进行存取，而且不存在冲突的问题。

当两个端口同时访问存储器，而访问地址不相同（不同存储单元）时，在两个端口上可以同时各自独立地并行进行读写操作，不会发生冲突。

3.5.1 双端口存储器

3、有冲突读写控制 ■

当两个端口同时存取（访问）存储器同一存储单元时，便发生读写冲突。

为解决此问题，特设置了BUSY标志。

在这种情况下，片上的判断逻辑可以决定哪个端口优先进行读写操作，而对另一个被延迟的端口置BUSY标志（低电平有效）。

3.5.2 多模块交叉存储器

1、存储器的模块化组织

顺序方式：各存储体依次顺序定义地址空间，一个存储体编完以后才到另一块，每个存储体中的地址是连续的。

地址范围：

M0：00 000-00 111　　M1：01 000-01 111

M2：10 000 10 111　　M3：11 000-11 111

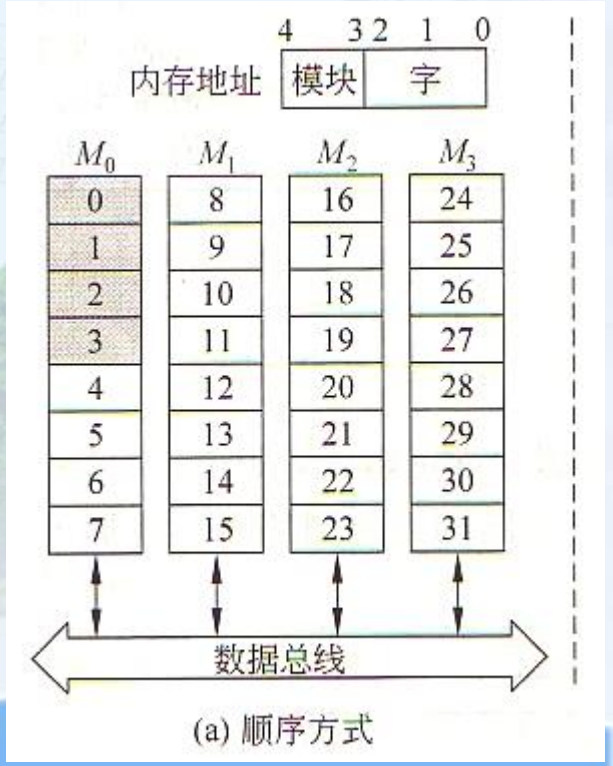
特点：

（1）某个模块进行存取时，其他模块不工作。

（2）某一模块出现故障时，其他模块可以照常工作。

（3）增添模块扩充容量比较方便。

（4）各模块串行工作，难以采用流水线提高带宽。



(a) 顺序方式

3.5.2 多模块交叉存储器

1、存储器的模块化组织

交叉方式：连续地址交叉分配在各个存储体中，每个存储体中的地址是不连续的。

地址范围：

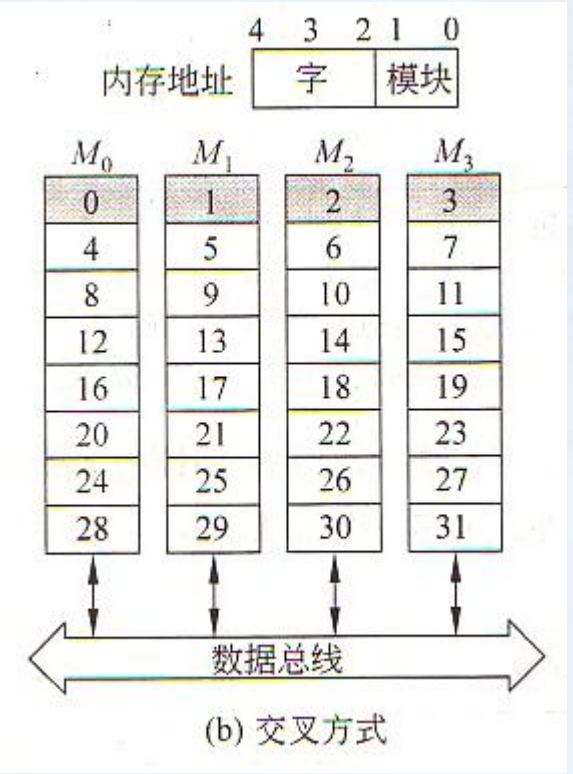
M0：000 00-111 00　　M1：000 01-111 01

M2：000 11-111 10　　M3：000 11-111 11

特点：

（1）同一个模块内的地址都是不连续的。

（2）对连续的成组访问可实现流水线并行存取，提高存储器的带宽。



(b) 交叉方式

3.5.2 多模块交叉存储器

2、多模块交叉存储器结构

（1）设模块存取一个存储单元的时间为T；

（2）总线传递一个数据到CPU，然后到运算器/控制器的时间为τ；

（3）流水线方式存取：M0取出一个数据D0，然后经过总线传递；当D0总线传递完成，马上就有M1中的一个数据D1，送给总线传递；依此类推……，充分利用总线，只要总线传递不重叠即可。

（4）显然，T = 4T，为了充分利用总线，必然有四个存储体的重叠操作，即交叉存储器的流水线方式存取。

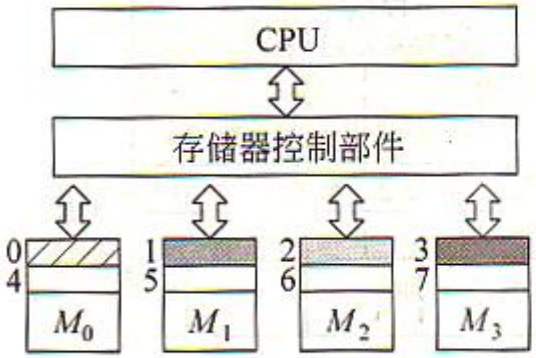


图 3.27　四模块交叉存储器结构框图

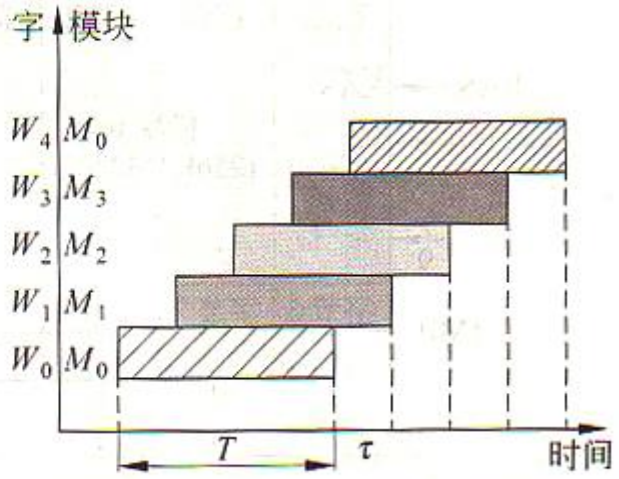


图 3.28　流水线方式存取示意图

3.5.2 多模块交叉存储器

2、多模块交叉存储器结构

（1）流水线的时钟周期为τ；

（2）采用交叉流水线完成n个任务所需要的时间为（T+τ）+(n-1)τ，为了和流水线时间公式一致，为T+(n-1)τ ；

（3）采用非流水完成n个任务所需要的时间为n（T+τ），可约为nT ；

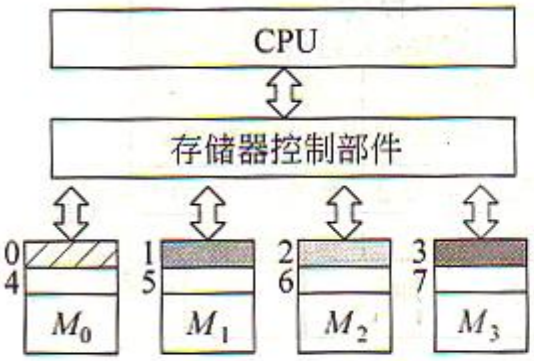


图 3.27　四模块交叉存储器结构框图

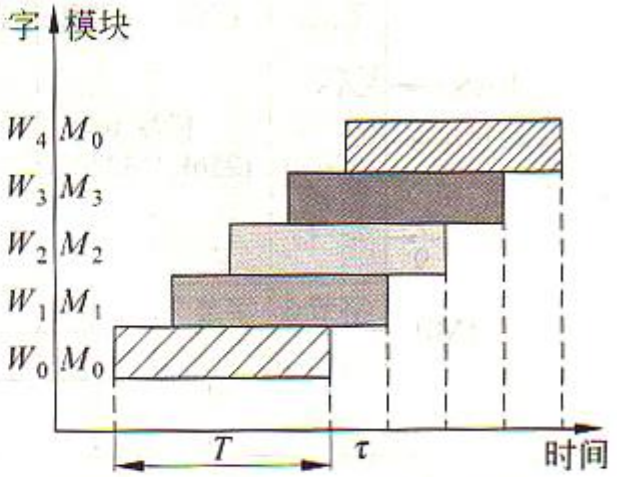


图 3.28　流水线方式存取示意图

3.5.2 多模块交叉存储器

【例5】设存储器容量为32字，字长64位，模块数m=4，分别用顺序方式和交叉方式进行组织。存储周期T=200ns，数据总线宽度为64位，总线传送周期 τ =50ns。问顺序存储器和交叉存储器的带宽各是多少?

【解】顺序存储器和交叉存储器连续读出m=4个字的信息总量都是： q=64位×4=256位。

顺序存储器和交叉存储器连续读出4个字所需的时间分别是：

$t2=mT=4\times200ns=800ns=8\times10^{-7}s$

$t1=T+(m-1)\tau=200ns+3\times50ns=350ns=35\times10^{-7}s$

顺序存储器和交叉存储器的带宽分别是：

$W2=q/t2=256\div(8\times10^{-7})=32\times10^{7}$ bps

$W1=q/t1=256\div(35\times10^{-7})=73\times10^{7}$ bps

3．6 cache存储器

Cache基本原理

主存与cache的地址映射

替换策略

Cache的写操作策略

Pentium 4的cache组织

3．6．1 cache基本原理

1、cache的功能

cache是一种高速缓冲存储器，为了解决CPU和主存之间速度不匹配而采用的一项重要技术。

（1）可高速存取的小容量的存储器，片内cache已经接近于CPU的处理速度；

（2）CPU可直接访问cache；

（3）可构造2级以上的cache系统；

（4）cache系统实现cache和主存、cache和cache之间的信息交换，对用户是透明的。

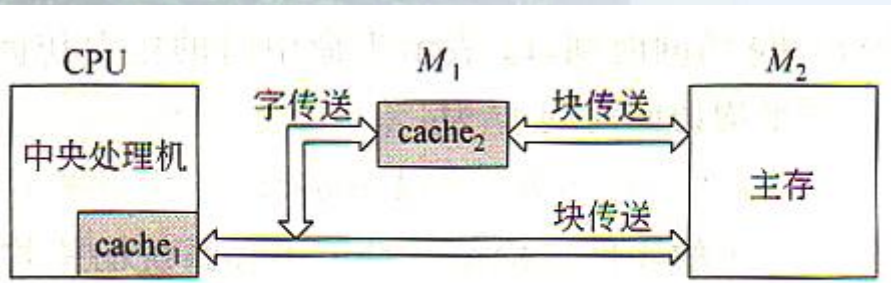


图 3.31　CPU 与存储器系统的关系

3．6．1 cache基本原理

2、cache的基本原理

主存和cache均按照约定长度划分为若干块；

主存中一个数据块调入到cache中，则将数据块地址（块编号）存放到相联存储器CAM中，将数据块内容存放在cache中；

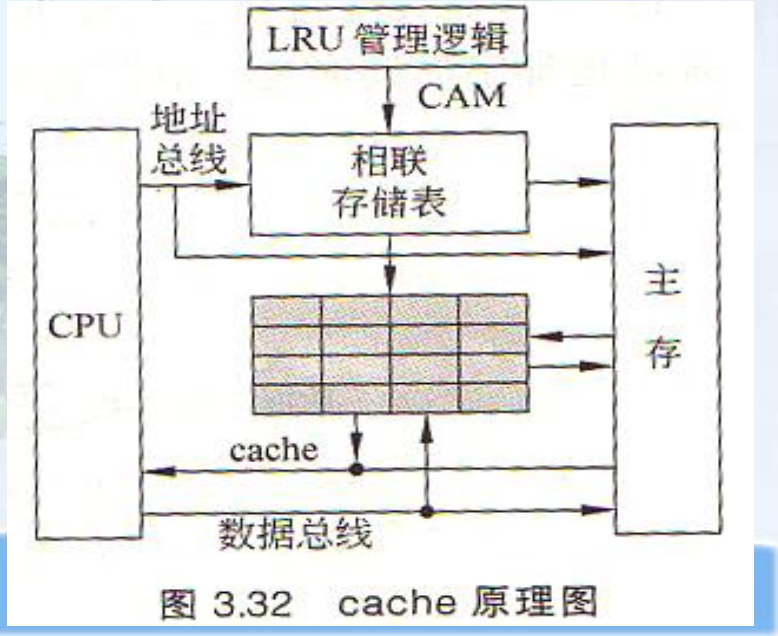


图 3.32　cache 原理图

3．6．1 cache基本原理

2、cache的基本原理

当CPU访问主存时，同时输出物理地址给主存、相联存储器CAM，控制逻辑判断所访问的块是否在cache中：

若在，则命中，CPU直接访问cache。

若不在，则未命中，CPU直接[illegible]元所在数据块交换到cache中。

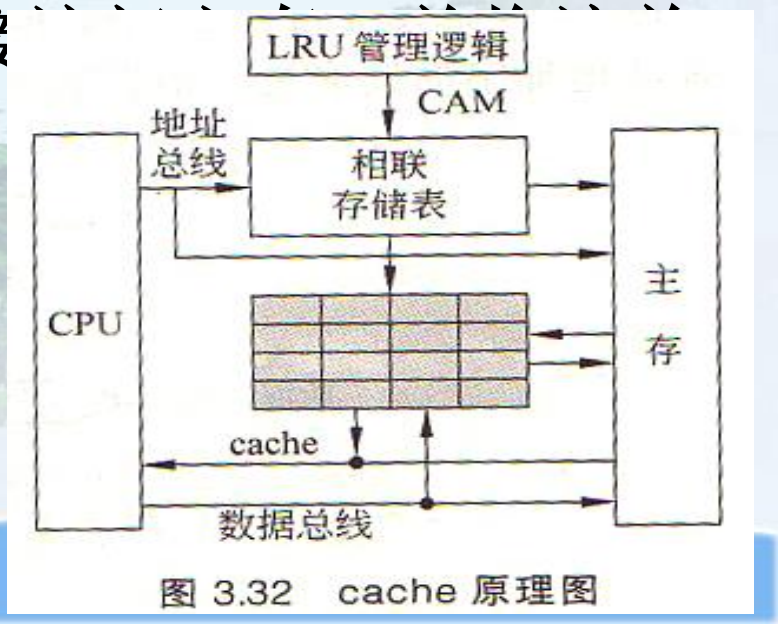


图 3.32　cache 原理图

3．6．1 cache基本原理

2、cache的基本原理

基于程序和数据的局部性访问原理；

通过cache和主存之间的动态数据块交换，尽量争取CPU访存操作在cache命中，从而总体提高访存速度；

cache实际上是主存的当前最活跃部分，即主存的一个子集。

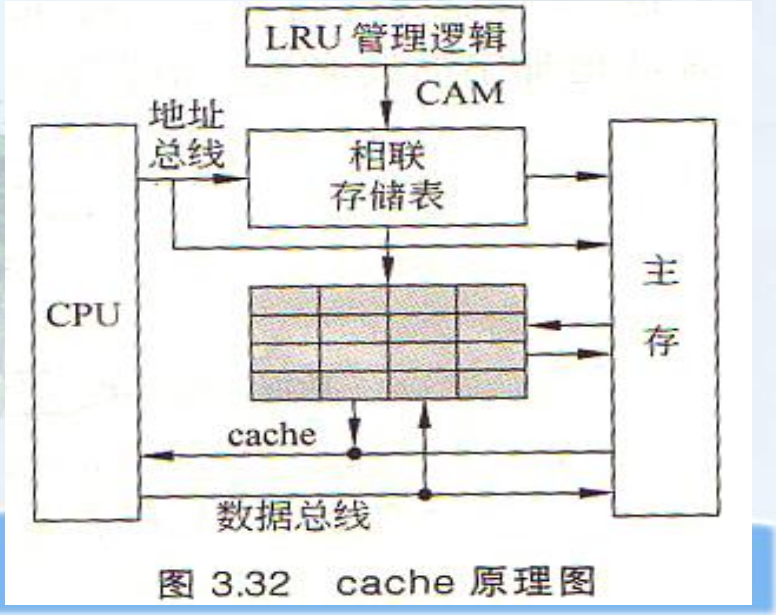


图 3.32　cache 原理图

3．6．1 cache基本原理

2、cache的基本原理

LRU管理逻辑：LRU是一种替换策略，当cache已满、且有新的数据块需要载入时，将最近最少使用的数据块替换出去。

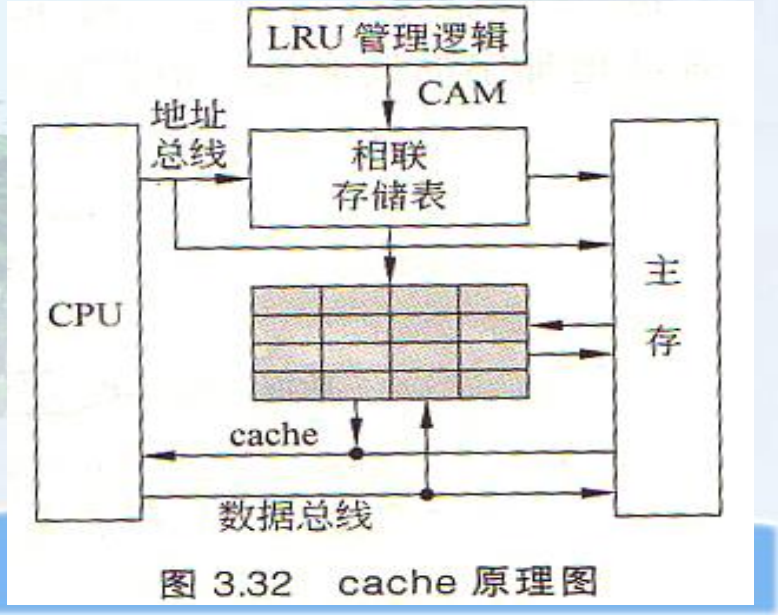


图 3.32　cache 原理图

3．6．1 cache基本原理

2、cache的基本原理

控制逻辑：

片外cache一般由主存/cache控制器提供控制逻辑，负责片外cache与主存、片外cache与片内cache、片外cache与CPU之间的数据交换及控制。

片内cache由CPU提供控制逻辑，负责片内cache与CPU、片内cache与片外cache、片内cache与主存之间的数据交换及控制。

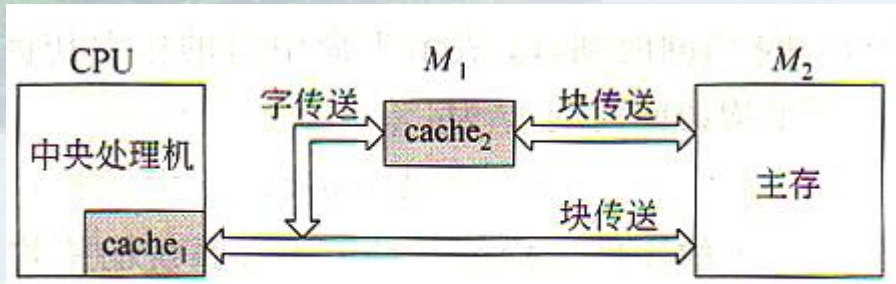


图 3.31　CPU 与存储器系统的关系

3．6．1 cache基本原理

2、cache的基本原理

数据交换：

CPU～cache：CPU以机器字为单位访问cache；

CPU～内存：CPU以机器字为单位访问内存；

cache～内存：以定长数据块为单位数据交换；

cache～cache：以定长数据块为单位数据交换。

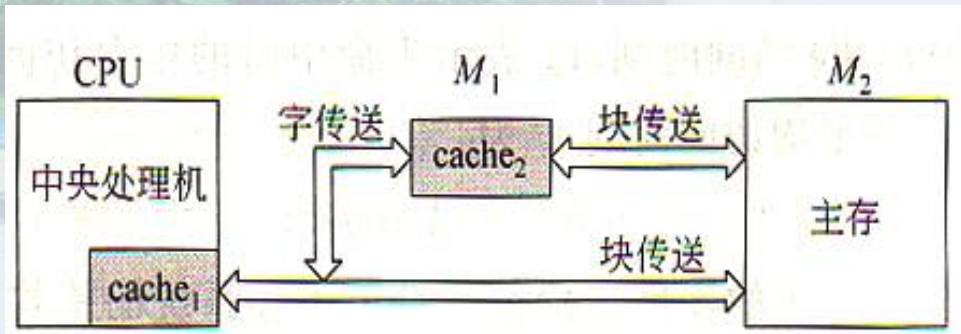


图 3.31　CPU 与存储器系统的关系

3、cache的命中率

命中率：

增加cache以后，就应该尽量争取在cache中命中越多越好。

在一个程序执行期间，设Nc表示cache命中完成存取的总次数，Nm表示未命中、主存完成存取的总次数，h定义为命中率，则有 $h=\frac{Nc}{Nc+Nm}$ 。

3．6．1 cache基本原理

3、cache的命中率

cache/主存系统的平均访问时间：

设tc表示命中时的cache访问时间，tm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ta为：$t_a = ht_c + (1-h)t_m$

3．6．1 cache基本原理

3、cache的命中率

访问效率：

设r=tm/tc表示主存慢于cache的倍率，e表示访问效率，则有：

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

3．6．1 cache基本原理

3、cache的命中率

命中率的影响因素：

程序行为，顺序程序比分支程序命中率高；

cache的容量；

组织方式；

数据块大小；

3.6.1 cache基本原理

【例6】CPU执行一段程序时，cache完成存取次数为1900次，主存完成存取次数为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=Nc/(Nc+Nm)=1900/(1900+100)= 0.95

r=tm/tc=250ns/50ns= 5

e=1/(r+(1-r)h)=1/(5+(1-5)×0.95)= 83.3%

ta=tc/e=50ns/0.833=60ns

3.6.2 主存与cache的地址映射

与主存容量相比，cache的容量很小，保存的内容只是主存内容的一个子集，且cache与主存的数据交换是以块为单位。

为了把主存块放到cache中合适的位置，必须应用某种方法把主存地址定位到cache中，称为地址映射。

“映射”一次的含义就是确定位置对应关系。

地址映射由硬件控制逻辑实现。

地址映射方式有全相联方式、直接方式和组相联方式三种。

3．6．2 主存与cache的地址映射

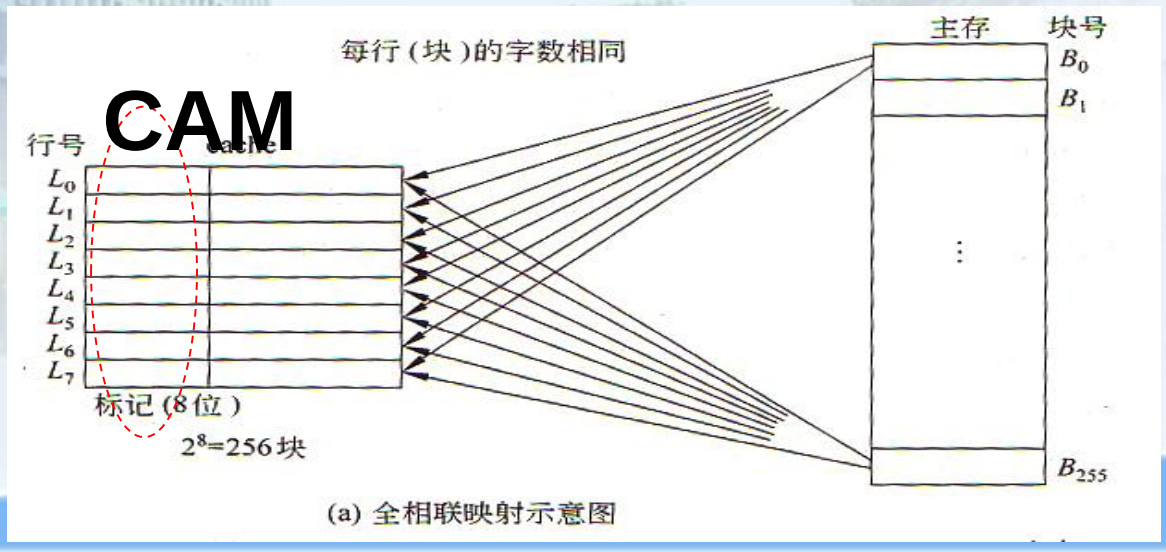
1、全相联映射方式

主存和cache都按照相同的约定的长度划分若干个块（行），当主存中某块（行）交换到cache的一个块（行）中时，将块地址集中存放在相联存储器CAM中。

（1）主存256块，cache8块；

（2）设块的长度为128B，则主存物理地址为：块地址8位 + 块内地址7位；

（3）主存中的任意一块（行）可以载入到cache中任意一块（行）位置上；



3.6.2 主存与cache的地址映射

1、全相联映射方式

根据物理地址中的块地址s（块号）和CAM中已存放的块地址（标记）进行比较：

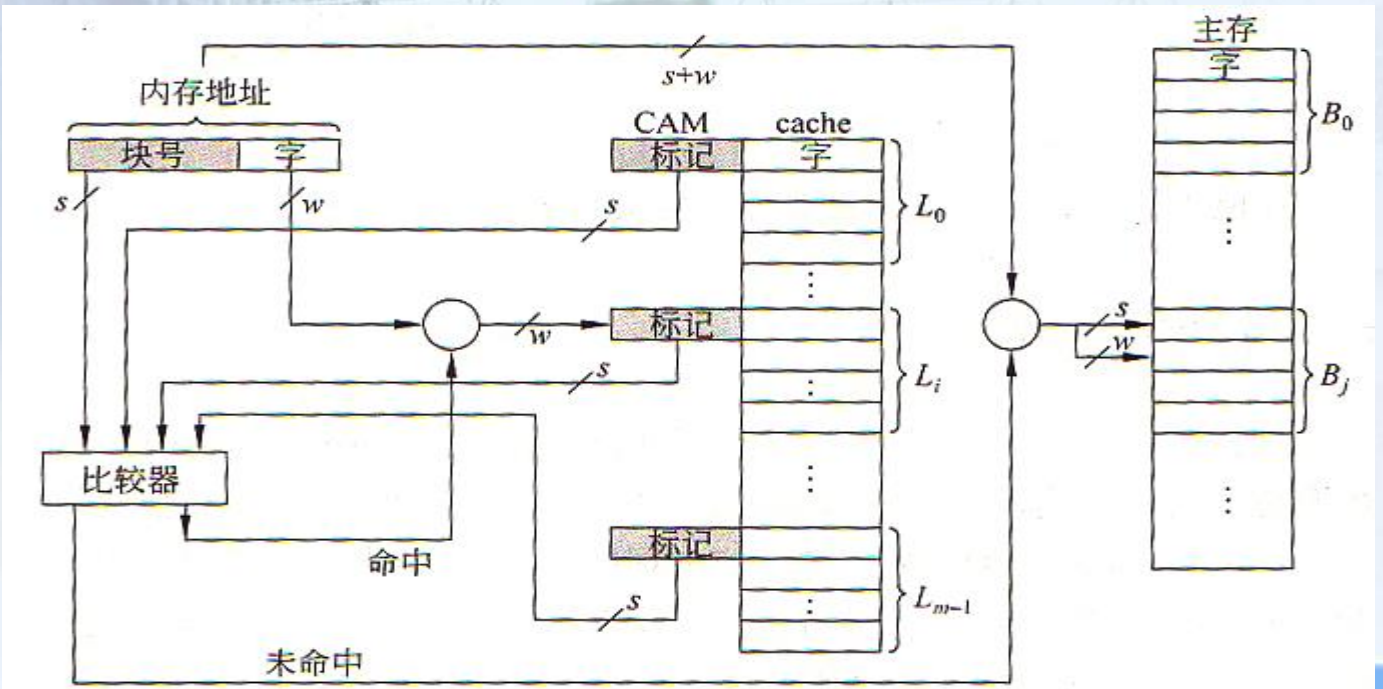
若命中，则根据块内地址w（字）访问cache指定单元；

若未命中，则根据s+w访问主存单元，并将该块载入cache中。

特点：

灵活；

比较器耗时；



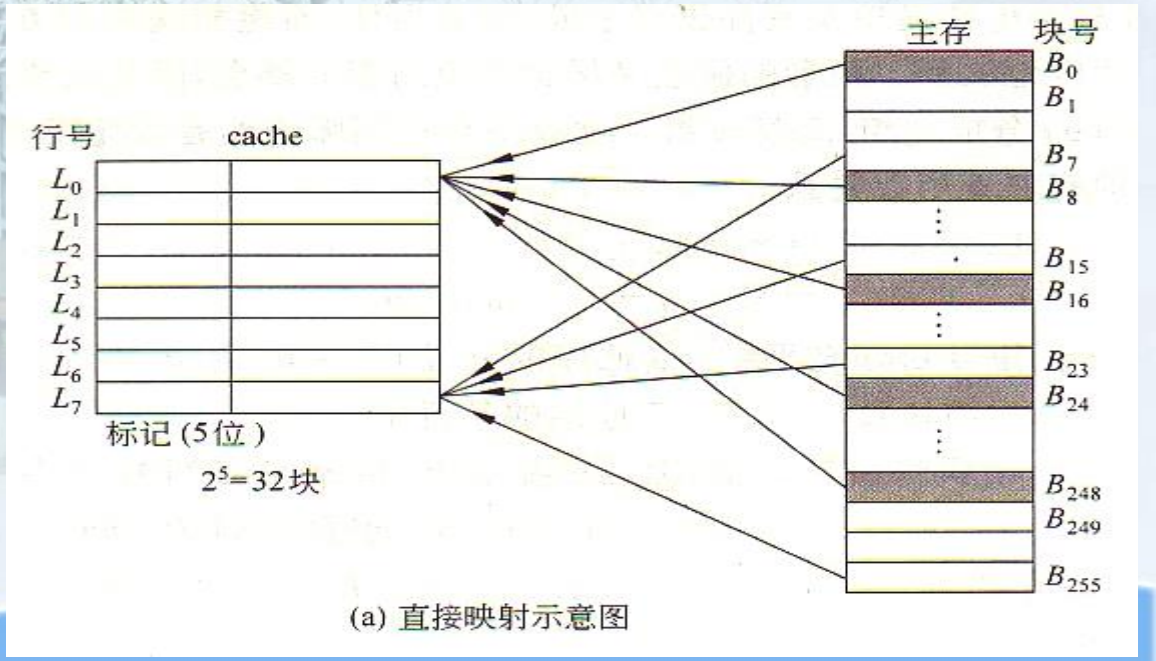
(b) 全相联 cache的检索过程

3．6．2 主存与cache的地址映射

2、直接映射方式

（1）主存按照cache的块数进行分组，如图主存中的256个数据块，按照8块一组，划分为32个组；

（2）主存物理地址=组号5位+组内块编号3位+块内地址7位；



(a) 直接映射示意图

3.6.2 主存与cache的地址映射

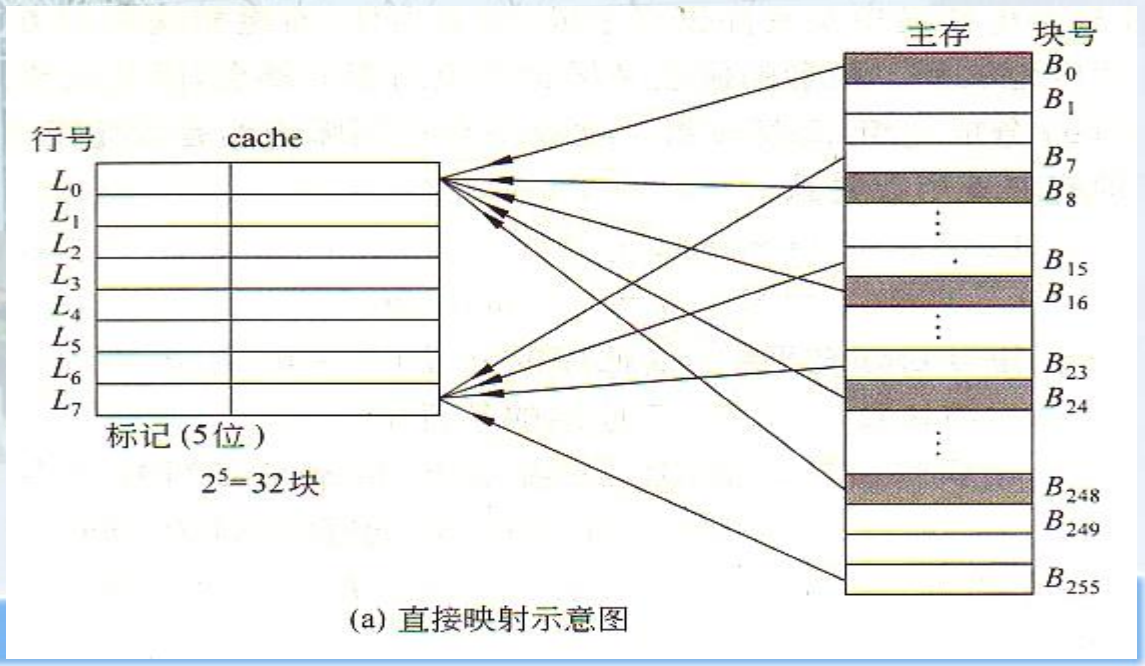
2、直接映射方式

主存中每组的第0块（B0、B8、B16…B248）只能载入到L0块（行）；

主存中每组的第1块（B1、B9、B17…B249）只能载入到L1块（行）；

……；

主存中每组的第7块（B7、B15、B23…B255）只能载入到L7块（行）。



(a) 直接映射示意图

3.6.2 主存与cache的地址映射

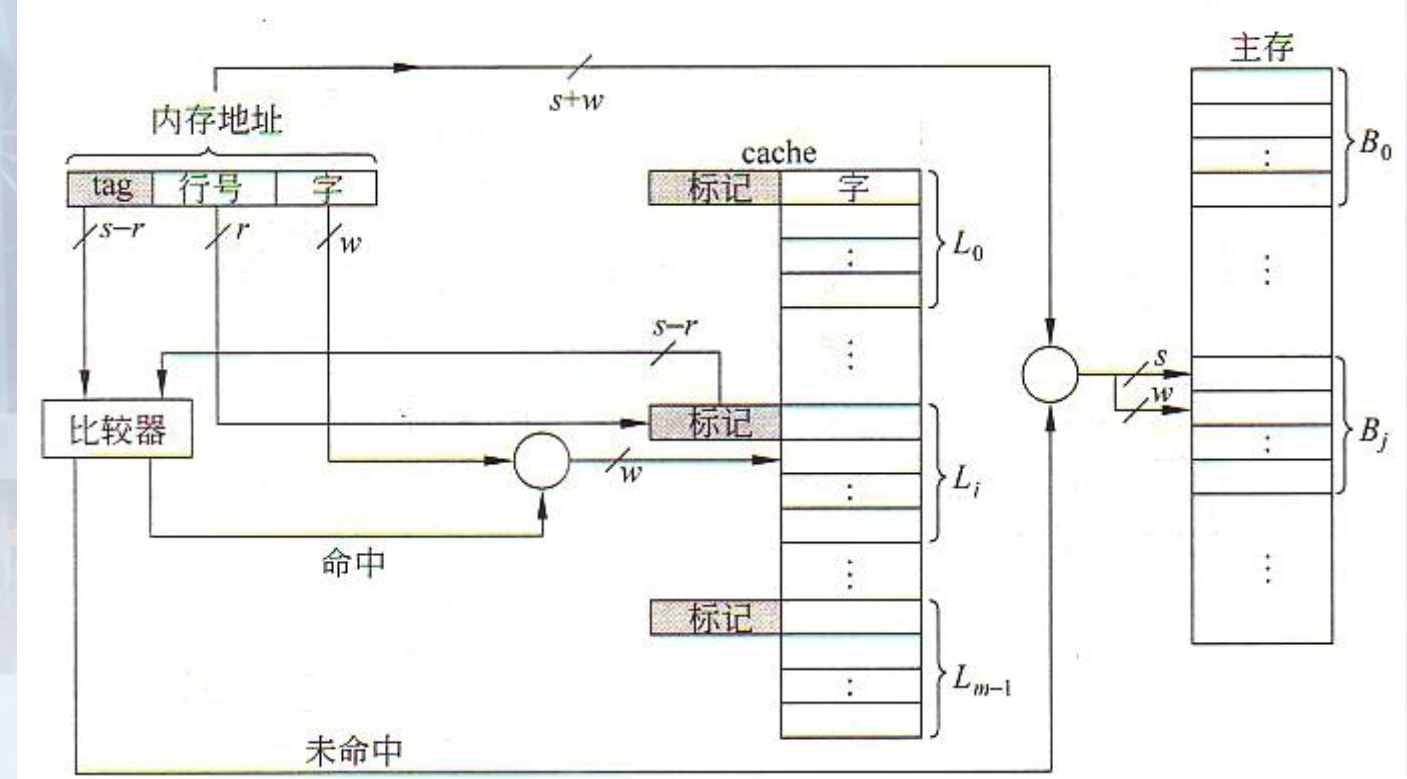
2、直接映射方式

（1）根据物理地址组内块编号r，直接对应cache中同编号块；

（2）比较组号s-r是否一致；若一致，则命中，否则未命中；

特点：

灵活性差；



(b) 直接映射 cache的检索过程

3.6.2 主存与cache的地址映射

3、组相联映射方式

全相联映射方式太灵活、比较工作量大；

直接映射方式太死；

组相联映射方式是前两种方式的折衷方案，即将cache、主存都分组，适当提高灵活性。

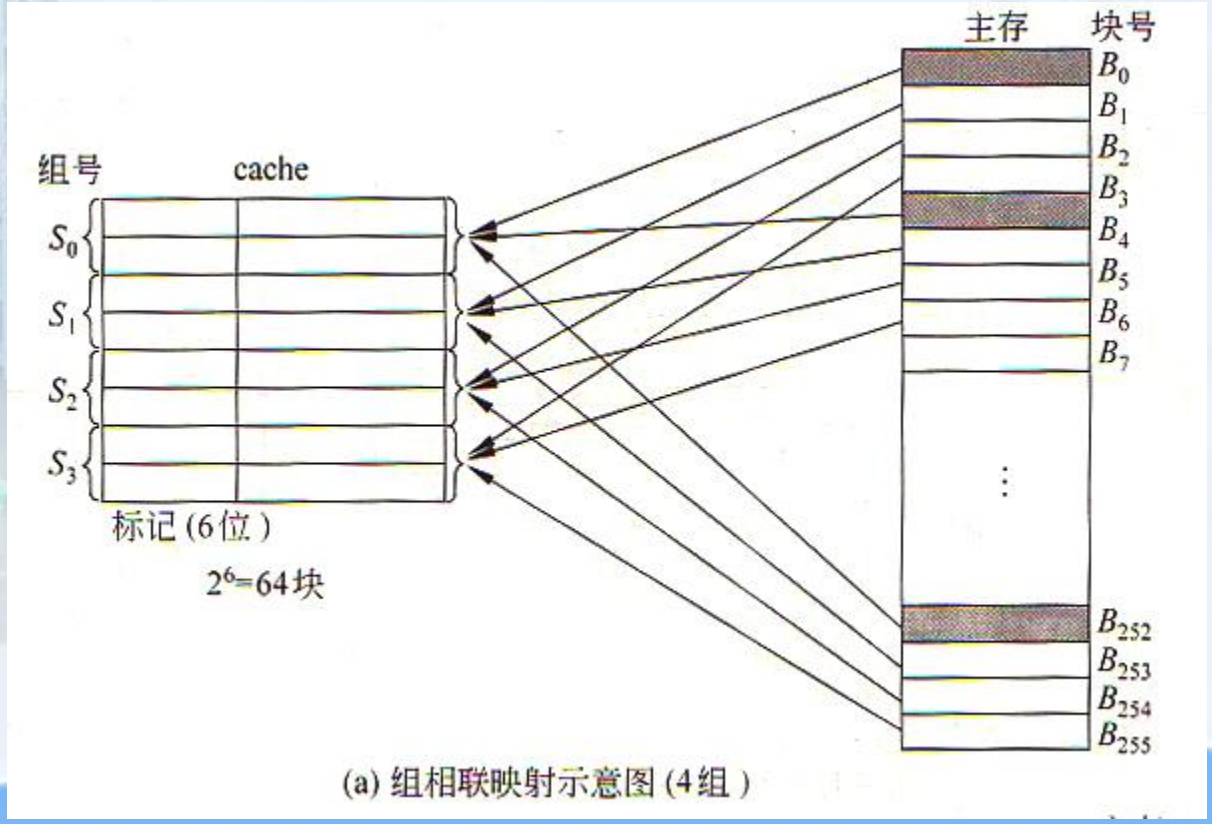
最常采用组相联映射方式。

3．6．2 主存与cache的地址映射

3、组相联映射方式

（1）cache中每两个块（行）为一组，共划分为4组；

（2）主存按照cache中的组数进行分组，共划分为64个组，每组4块；



(a) 组相联映射示意图 (4组)

3．6．2 主存与cache的地址映射

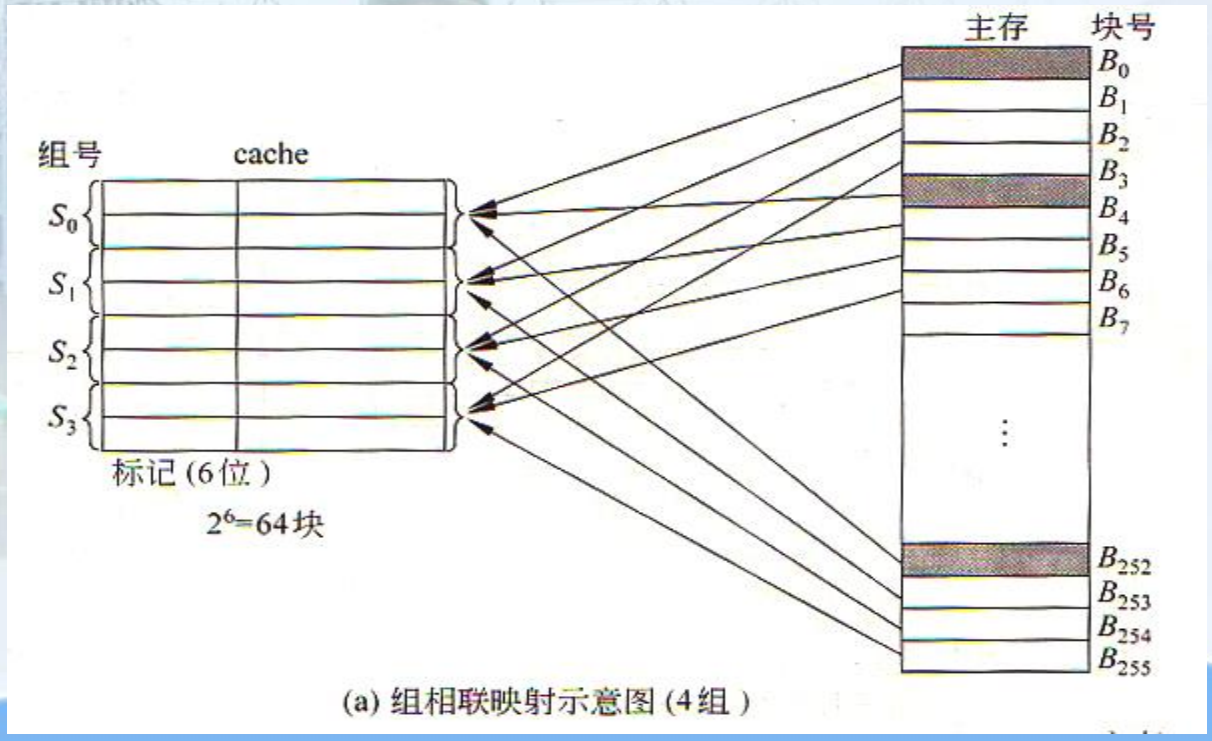
3、组相联映射方式

主存中每组的第0块（B0、B4、…B252）可以载入到S0组中任意一块；

主存中每组的第1块（B1、B5、…B253）可以载入到S1组中任意一块；

主存中每组的第2块（B2、B6、…B253）可以载入到S1组中任意一块;

主存中每组的第3块（B3、B7、…B255）可以载入到S3组中任意一块；



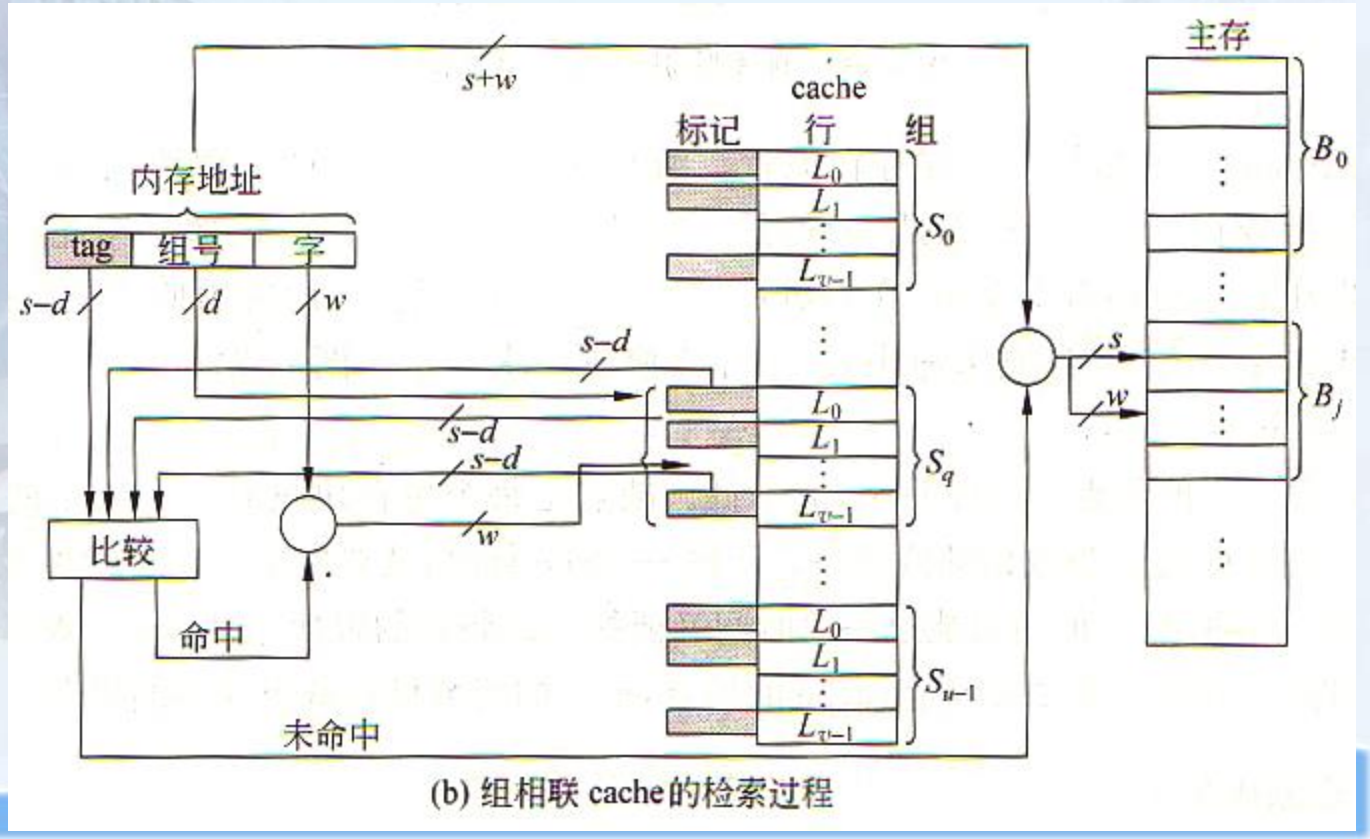
(a) 组相联映射示意图 (4组)

3．6．2 主存与cache的地址映射

3、组相联映射方式

（1）根据物理地址组内块编号d，直接对应cache中同编号组；

（2）比较cache组中所有块的s-d是否一致；若一致，则命中，否则未命中；



(b) 组相联 cache的检索过程

3.6.3 替换策略

替换策略：

当一个新的主存块需要载入到cache、而允许存放此块（行）的位置已满时，需要选择哪一块（行）被替换出cache，这需要一种策略机制。

直接映射方式：

内存中每组的第i块只能映射到cache中的第i块，是固定的一对一关系，不需要替换策略。

全相联映射方式：

内存中的每一块可以映射到cache中的任意块（行），存在替换策略的问题。

组相联映射方式：

内存中每组的第i块可映射到cache的第i组中的任意一块（行），存在替换策略的问题。

3.6.3 替换策略

1、最不经常使用(LFU)算法

上次替换到本次替换之间的一段时间内被访问次数最少的块替换出去，不能严格反映近期访问情况。

2、近期最少使用(LRU)算法

将近期内（未限定在两次替换之间）长久未被访问过的块换出。

3、随机替换策略

随机地选择一块替换出去。

3．6．4 cache的写操作策略

写操作策略：

由于cache的内容只是主存部分内容的复制，它应当与主存内容保持一致。而CPU对cache的写操作，更改了cache的内容，但并不意味着更改了主存的内容，必须有一种策略机制来保证cache和主存中的内容保持一致。

1、写回法

当CPU写cache命中时，只修改cache的内容，而不立即写入主存；只有当此行被换出时才写回主存；

这种方法减少了访问主存的次数,但是存在数据不一致性的隐患。

3．6．4 cache的写操作策略

2、全写法

当写cache命中时，cache与主存同时发生写修改；当写cache未命中时，直接向主存进行写入；

该方法维护了cache与主存的内容一致性，但是效率低。

3、写一次法

写一次法是指第一次写cache命中时要同时写入主存，以后的写操作只写cache，替换时再写主存。

第一次写cache命中时同时写入主存，目的是通知多cache系统中的其它cache和主存，这个数据块发生了写操作，应实施数据一致性策略。

3.6.5 Pentium 4的cache组织

片内三级cache：L1、L2、L3；

L1 cache：I-cache、D-cache；

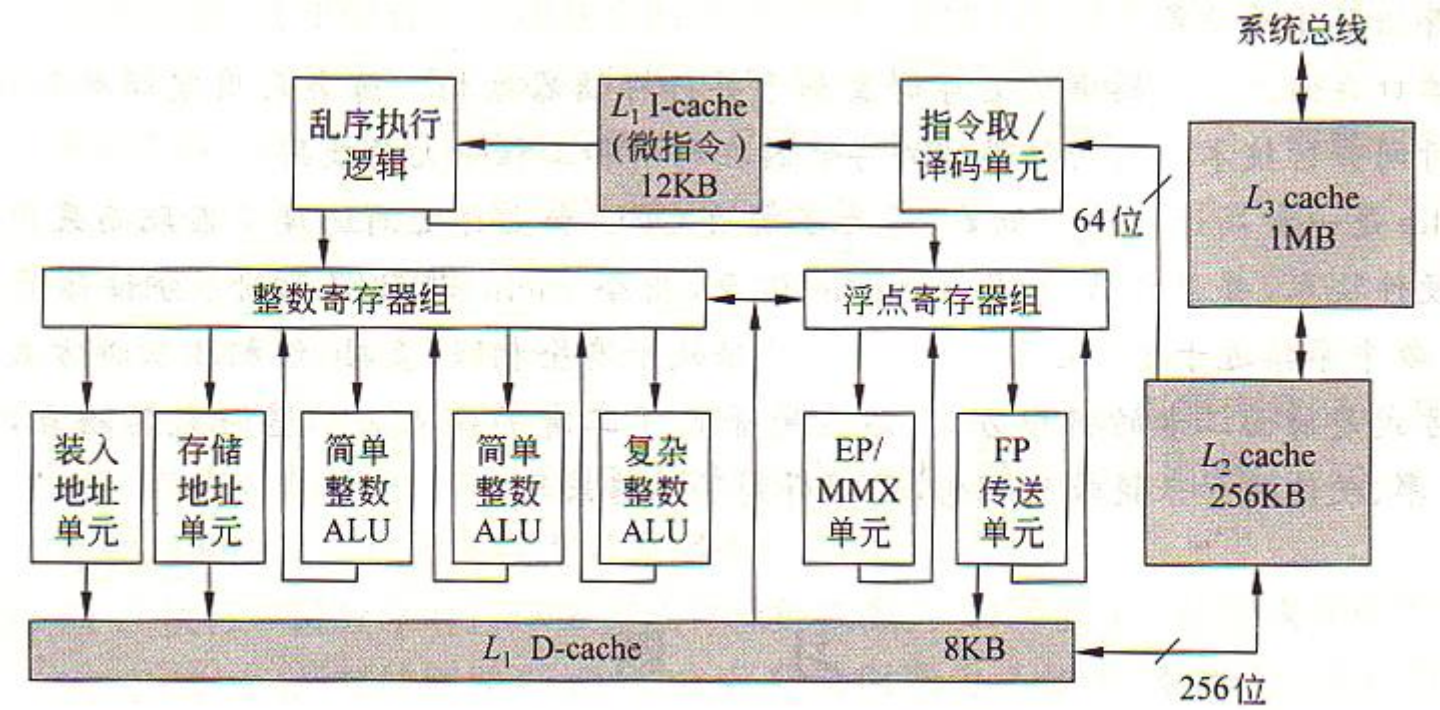


图 3.36　Pentium 4 的 cache 布局图

3.7 虚拟存储器

虚拟存储器的基本概念

页式虚拟存储器

段式虚拟存储器

段页式虚拟存储器

替换算法

虚拟存储器实例

3.7.1 虚拟存储器的基本概念

1、虚拟存储器的引入

在多用户多任务系统中，多个用户或多个任务共享全部主存，要求同时执行多道程序。这些同时运行的多道程序到底占用实际内存中的哪一部分，在编制程序时是无法确定的，必须等到程序运行时才动态分配。

为此，希望提供一个足够大的、独立编址的逻辑地址空间，不需考虑实际内存是否放得下、放什么位置。

3.7.1 虚拟存储器的基本概念

2、什么是虚拟存储器

虚拟存储器是通过硬件/操作系统，实现主存-外存之间的信息部分调入调出，为用户提供一个比实际物理内存容量大得多的存储器逻辑空间，使之为更大或更多的程序所使用。主存-外存之间的信息部分调入调出过程对用户透明。

3、虚拟存储器的实现原理

虚拟存储器技术基于程序的局部性原理实现：在一段时间范围内，执行的程序是一个大程序中相对集中的一部分程序（模块），也称为当前活跃部分。

4、虚拟存储器的技术意义

虚拟存储器的技术目的是侧重解决主存容量不足，属于主存和外存之间的问题。

以透明的方式为用户提供一个比实际主存空间大得多的逻辑空间，使之为更大或更多的程序所使用。用户在编写程序时不需要考虑所编程序在主存中是否放得下或放在什么位置等具体细节问题。

3.7.1 虚拟存储器的基本概念

5、物理地址空间、物理地址

物理存储器的实际地址空间称为物理地址空间；

物理地址空间受CPU外部地址总线控制，设地址总线为n根、则物理地址空间最大为2^n；

物理地址空间中的每一个存储单元都必须有一个唯一的地址编码，称为物理地址。

3.7.1 虚拟存储器的基本概念

6、虚拟地址空间、虚拟地址

通过虚拟存储器技术为用户透明提供的比实际物理地址空间大得多的逻辑空间，称为虚拟地址空间；

编译程序在逻辑空间基础上生成的逻辑地址，称为虚拟地址；

工作在虚拟地址模式下的CPU负责解释虚拟地址，并通过相应的策略机制转换为物理地址访问物理内存。

3.7.1 虚拟存储器的基本概念

7、cache与虚拟存储器的异同

P98：

（1）出发点相同

（2）原理相同

（3）侧重点不同

（4）数据通路不同

（5）透明性不同

（6）未命中时的损失不同

3．7．1　虚拟存储器的基本概念

8、虚拟存储器机制要解决的关键问题

P99：

（1）调度问题

（2）地址映射问题

（3）替换问题

（4）更新问题，即写策略

9、主存-外存之间的基本信息传送单位

主存-外存之间信息交换的基本传输单位可采用几种不同的方案：段、页、段页，相应地分别称为段式虚拟存储器、页式虚拟存储器、段页式虚拟存储器。

3.7.1 虚拟存储器的基本概念

10、段

段是利用程序的模块化性质，按照程序的逻辑结构划分成的多个相对独立部分。

段通常是指独立的功能模块、数据模块，如过程、子程序、数据文件等。

段作为独立的逻辑单位，可以被其它段调用，段间连接形成规模更大的程序。

3.7.1 虚拟存储器的基本概念

10、段

段的优点：段的分界与程序的自然分界相对应；段的逻辑独立性使它易于编译、管理、修改和保护；以段为单位在主存-外存之间调入调出不会改变程序的结构性质，保证程序的完整性、一致性。

段的缺点：由于段的长度各不相同，给主存空间的预留、分配带来麻烦。

3.7.1 虚拟存储器的基本概念

11、页

物理地址空间、虚拟地址空间均按定长划分，称为页。

优点：页的大小是固定的，新页调入主存容易合理地预留、分配存储空间。

缺点：因为页不是独立的逻辑单位，处理、保护和共享都不及段方便，可能破坏程序的逻辑结构。

3.7.1 虚拟存储器的基本概念

12、段页

采用分段和分页结合的方法。

程序按模块分段，段内再分页，进入主存以页为基本信息传送单位。

3.7.2 页式虚拟存储器

1、物理页、逻辑页

将物理地址空间、虚拟地址空间均按照约定长度划分为页面，主存和外存之间的调入调出以页为基本单位。

物理地址空间的页称为物理页（实页）；

物理地址 = 物理页号 + 页内偏移地址；

虚拟地址空间的页成为逻辑页（虚页）；

虚拟地址 = 虚拟页号 + 页内偏移地址；

3.7.2 页式虚拟存储器

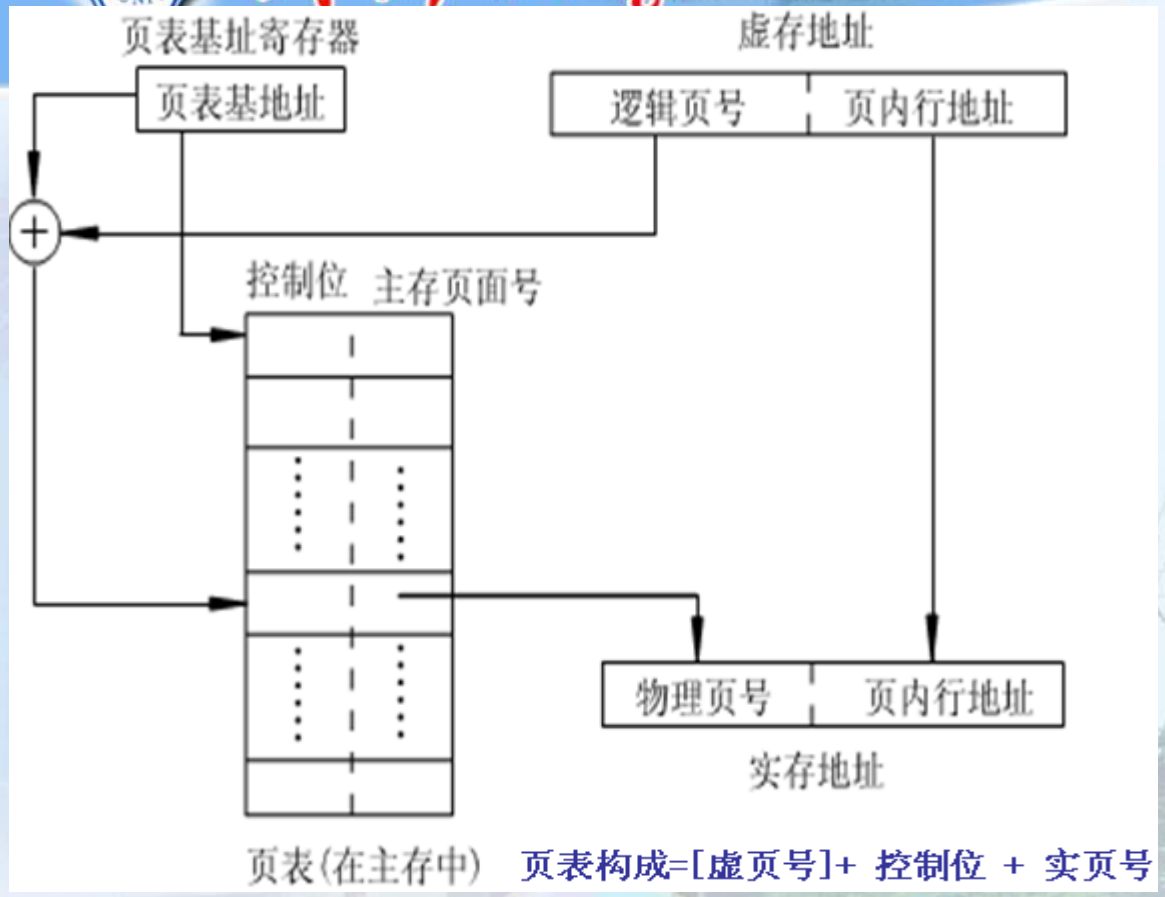
2、虚拟地址和物理地址之间的变换

变换的基础是系统建立的页表，页表由[虚页号]+控制位（装入标志位+修改标志位等）+实页号构成，每个虚页在页表中占一行。

当某个虚页调入内存，则在页表中对应的虚页行上记载实页号、装载标志、修改标志等信息。

当CPU访问主存时，根据虚拟地址中的虚页号查页表，获取实页号，和页内地址拼接形成物理地址。

3.7.2 页式虚拟存储器



根据逻辑页号查页表：

若装入标志为1，则取物理页号，形成物理地址；

若装入标志为0，则产生缺页中断，从外存调入页到内存；

缺页中断时，如内存已满，则需要替换策略；

页表基址寄存器：存放页表基地址（页表中首单元在内存中的物理地址），而逻辑页号相当于访问的页表行相对于页表表首单元的偏移量。

访问的页表行的地址=页表基地址+逻辑页号

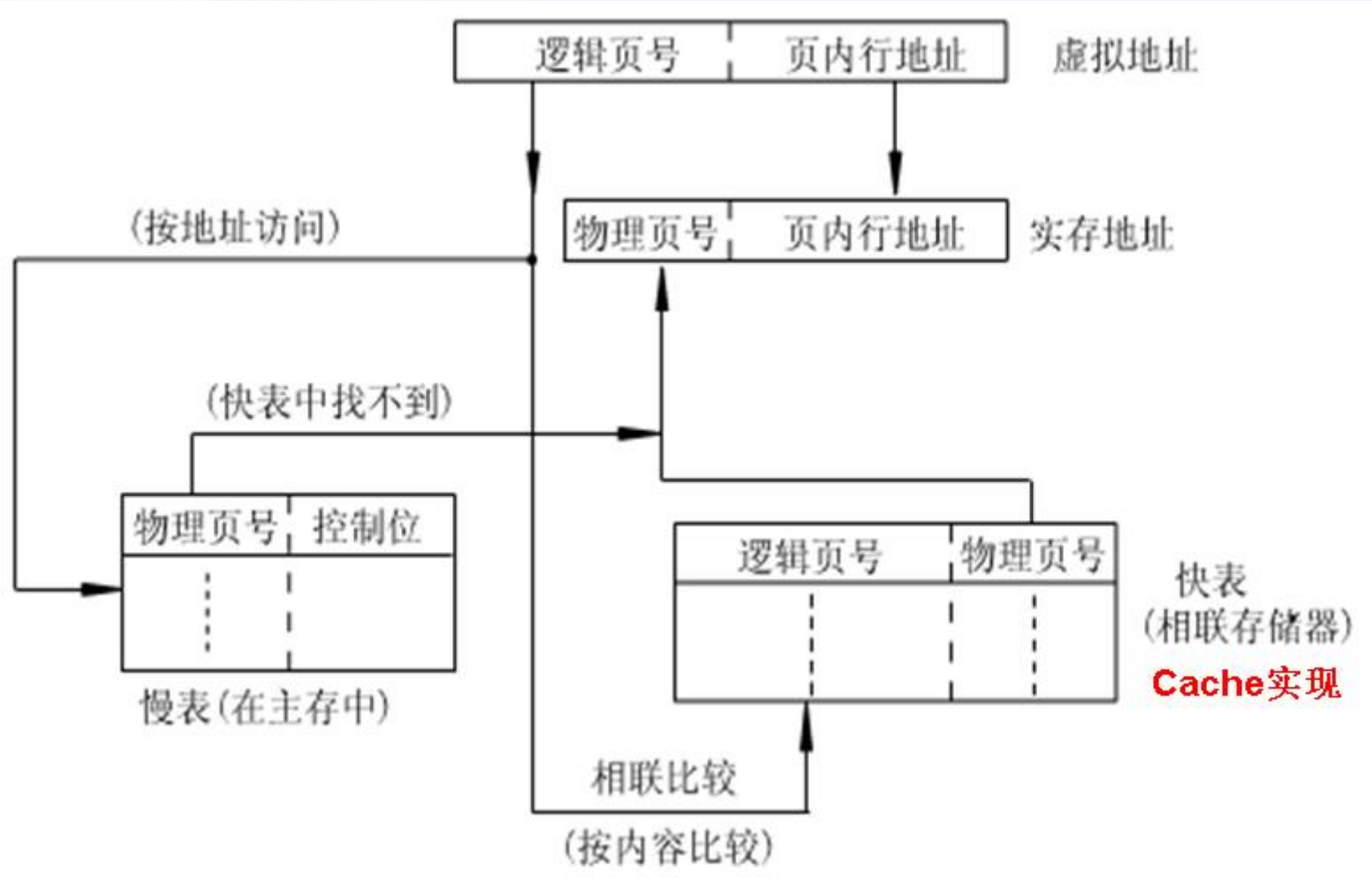
3.7.2 页式虚拟存储器

3、快表、慢表实现地址变换

在页式虚拟存储器中，虚拟地址通过查询页表转换为物理地址，而页表在主存中，因此进行一次访问主存储器操作必须要两次访问主存（一次根据虚拟地址查页表，获得物理地址；一次根据物理地址访问具体单元）。

显然效率不高，可以考虑把页表的当前最活跃部分存放在高速缓冲存储器cache中，即快表，从而提高页表的查询效率。

3.7.2 页式虚拟存储器


逻辑页号
页内行地址
虚拟地址
(按地址访问)
物理页号
页内行地址
实存地址
(快表中找不到)
物理页号
控制位
慢表(在主存中)
逻辑页号
物理页号
快表
(相联存储器)
Cache实现
相联比较
(按内容比较)

3.7.3 段式虚拟存储器

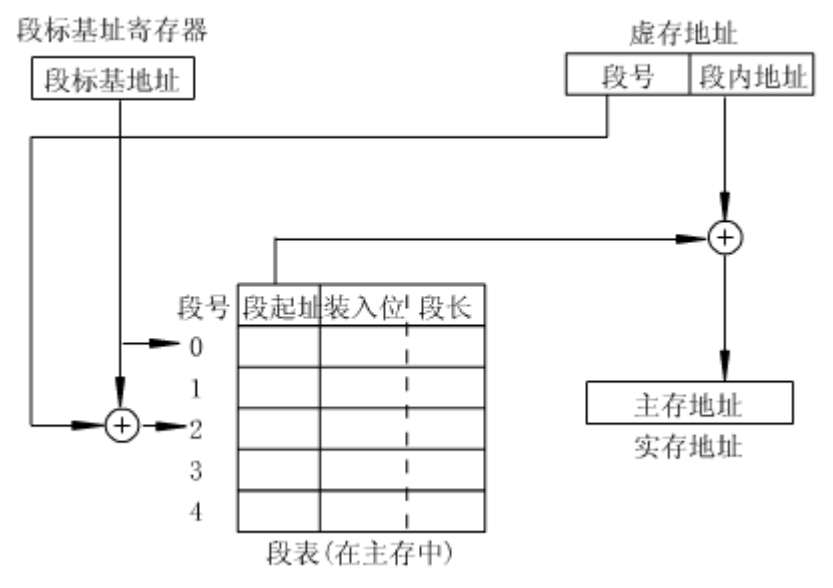
基于段表的管理和地址变换：

段式虚拟存储器基于段表管理和变换；

段表 = [段号]+段起始地址+装入标志位+段长度+修改标志位等；

根据虚拟地址中的逻辑段号、查询段表、取得段起始地址、与虚拟地址中的段内地址组合形成物理地址。

3.7.3 段式虚拟存储器



段表基址寄存器：

存放段表首单元的物理地址；

段号相当于访问行相对于段首单元的偏移量；

3.7.4 段页式虚拟存储器

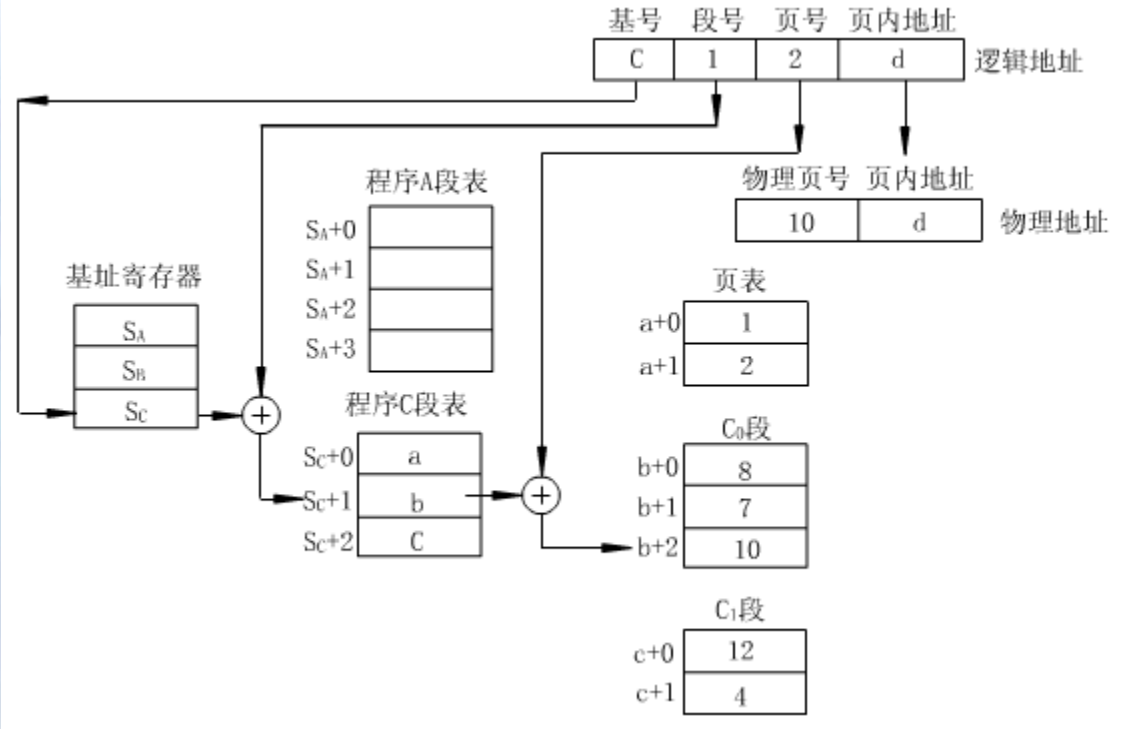
段表、页表：

先分段、然后分页，因此由虚拟地址变换为物理地址时需要先查段表、后查页表。

基号：

在多道程序运行的情况下、每道程序都有一个标识号，称为基号。

3.7.4 段页式虚拟存储器



基址寄存器：

每个基址寄存器存放每道程序所对应的段表的首单元的物理地址；

根据基号确定访问哪个基址寄存器；

3.7.5 替换算法

1、替换算法

访问的页（或段）不在主存中，同时主存已满，采用什么样的规则来将某一页（或段）调出主存，即替换算法。

LRU：近期最少使用算法

LFU：最不经常使用算法

FIFO：先进先出算法

2、写策略

对于将被替换出去的页（或段）面，假如该页（或段）调入主存后没有被修改，就不必进行处理，否则就把该页（或段）重新写入外存，以保证外存中数据的正确性。

为此，在页表/段表中的需要设置修改标志位。

3.7.5 替换算法

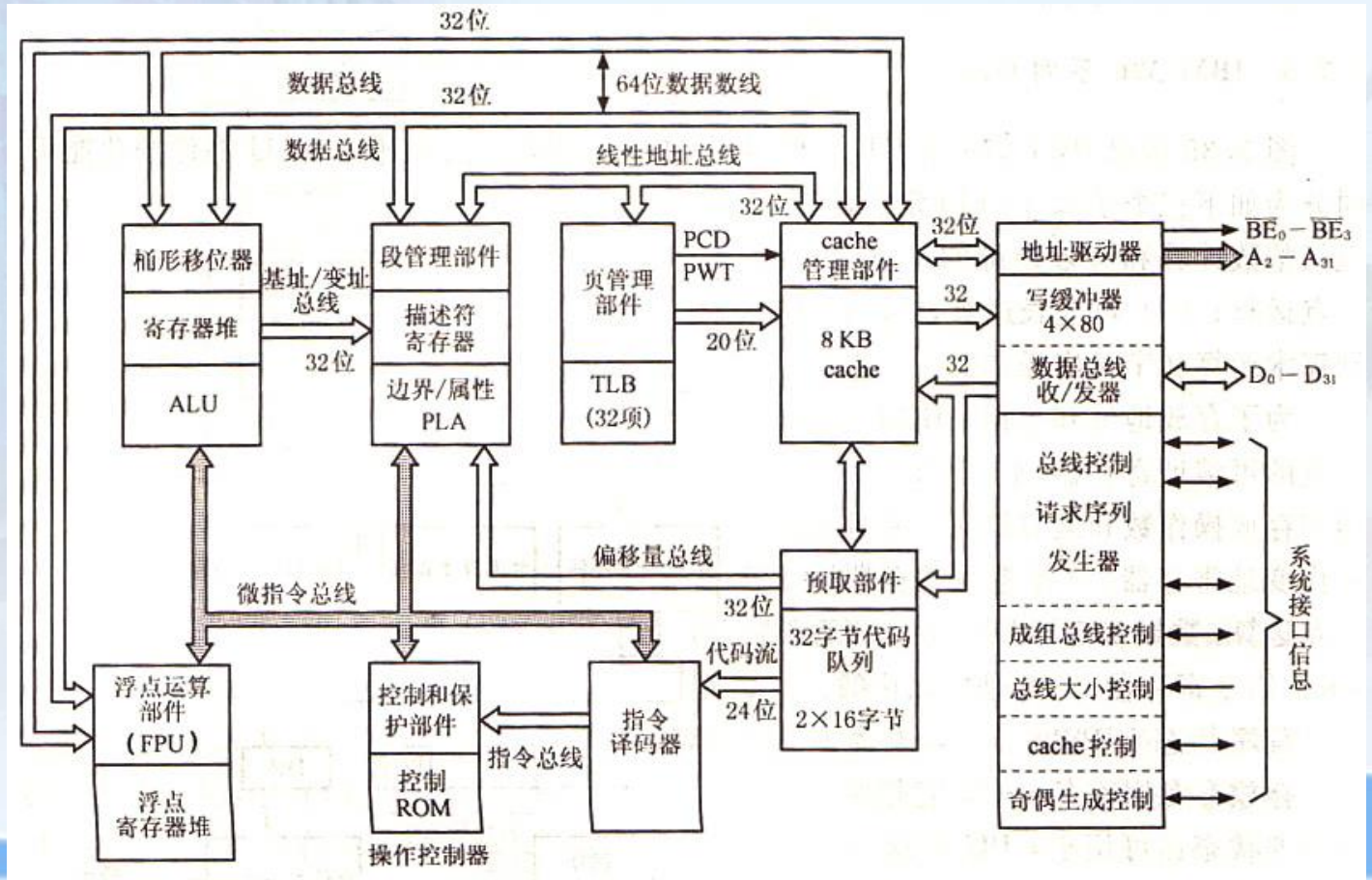
【例7】假设主存只有a,b,c三个页框，组成a进c出的FIFO队列，进程访问页面的序列是0，1，2，4，2，3，0，2，1，3，2号。若采用①FIFO算法，②FIFO算法+LRU算法，用列表法分别求两种替换策略情况下的命中率。

【解】

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=
	b		0	1	2	2	4	3	0	2	1	1	
算法	c			0	1	1	2	4	3	0	2	2	18.2%
						命中						命中	
FIFO+	a	0	1	2	4	2	3	0	2	1	3	2	
	b		0	1	2	4	2	3	0	2	1	3	3/11=
LRU	c			0	1	1	4	2	3	0	2	1	27.3%
算法						命中			命中			命中	

3.7.6 虚拟存储器实例

奔腾PC的虚地址模式:奔腾PC的存储管理部件MMU包括段管理部件SU和页管理部件PU两部份，可允许SU、PU单独工作或同时工作。



3.7.6 虚拟存储器实例

分段不分页模式：段式虚拟存储器

分段分页模式：段页式虚拟存储器

不分段分页模式：页式虚拟存储器