第三章　多层次的存储器

返回

3.1 存储器概述

3.1.1 存储器的分类

3.1.2 存储器的分级

3.1.3 主存储器的技术指标

返回

3.1.1 存储器的分类

- 按存储介质分类：磁表面/半导体存储器
- 按存取方式分类：随机/顺序存取（磁带）
- 按读写功能分类：ROM，RAM
  - RAM：双极型/MOS
  - ROM：MROM/PROM/EPROM/EEPROM
- 按信息的可保存性分类：永久性和非永久性的
- 按存储器系统中的作用分类：主/辅/缓/控

3.1.2 存储器的分级

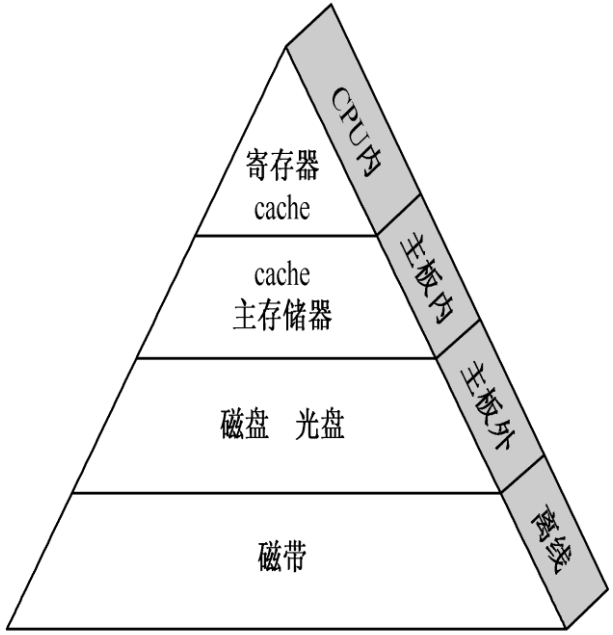
目前存储器的特点是：

- ❿ 速度快的存储器价格贵，容量小；
- ❿ 价格低的存储器速度慢，容量大。

在计算机存储器体系结构设计时，我们希望存储器系统的性能高、价格低，那么在存储器系统设计时，应当在存储器容量，速度和价格方面的因素作折中考虑，建立了分层次的存储器体系结构如下图所示。

3.1.2 存储器的分级

- 高速缓冲存储器简称cache，它是计算机系统中的一个高速小容量半导体存储器。
- 主存储器简称主存，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。
- 外存储器简称外存，它是大容量辅助存储器。



3.1.3主存储器的技术指标

- 字存储单元：存放一个机器字的存储单元，相应的单元地址叫字地址。
- 字节存储单元：存放一个字节的单元，相应的地址称为字节地址。
- 存储容量：指一个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。
- 存取时间又称存储器访问时间：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。通常取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。通常，存储周期略大于存取时间，其时间单位为ns。
- 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。

3.2 SRAM存储器

3.2.1 基本的静态存储元阵列

3.2.2 基本的SRAM逻辑结构

3.2.3 读/写周期波形图

3.2 SRAM存储器

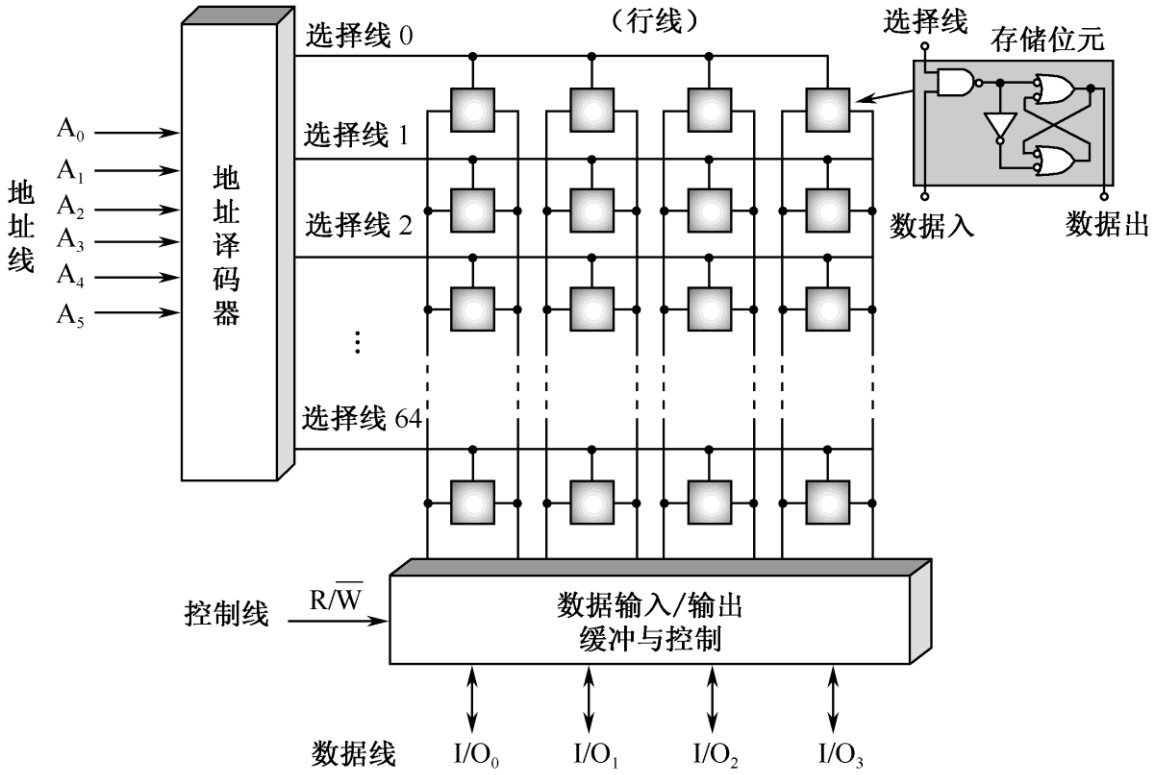
- 主存（内部存储器）是半导体存储器。根据信息存储的机理不同可以分为两类：
  - 静态读写存储器(SRAM)：存取速度快
  - 动态读写存储器(DRAM)：存储容量不如SRAM大。

3.2.1 基本的静态存储元阵列

1、存储位元

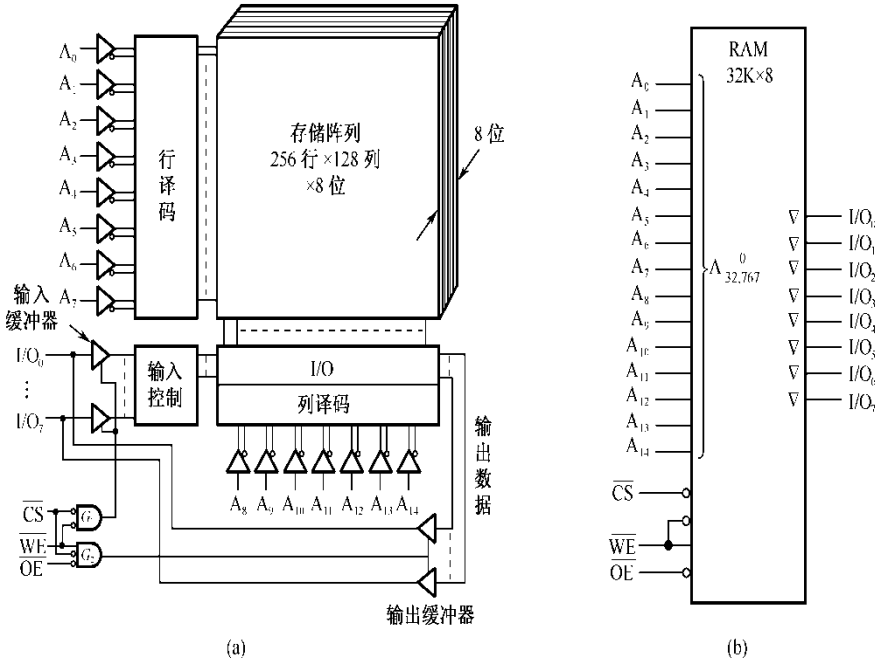
2、三组信号线

- 地址线
- 数据线
  - 行线
  - 列线
- 控制线



3.2.2 基本的SRAM逻辑结构

- SRAM芯大多采用双译码方式，以便组织更大的存储容量。采用了二级译码：将地址分成x向、y向两部分如图所示。



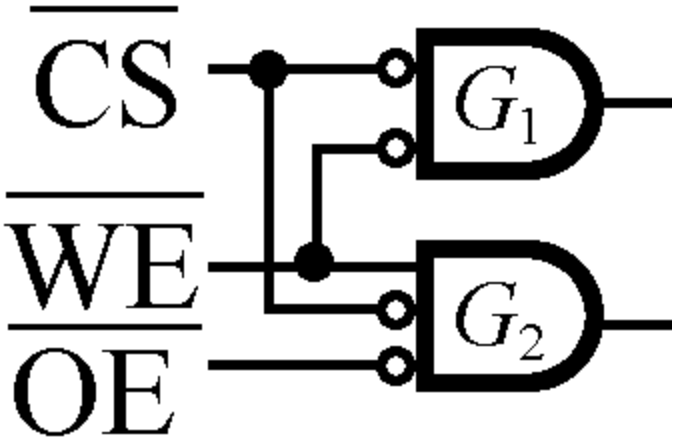
3.2.2 基本的SRAM逻辑结构

- 存储体（256×128×8）
  - 通常把各个字的同一个字的同一位集成在一个芯片（32K×1）中，32K位排成256×128的矩阵。8个片子就可以构成32KB。
- 地址译码器
  - 采用双译码的方式（减少选择线的数目）。
  - A0~A7为行地址译码线
  - A8~A14为列地址译码线

3.2.2 基本的SRAM逻辑结构

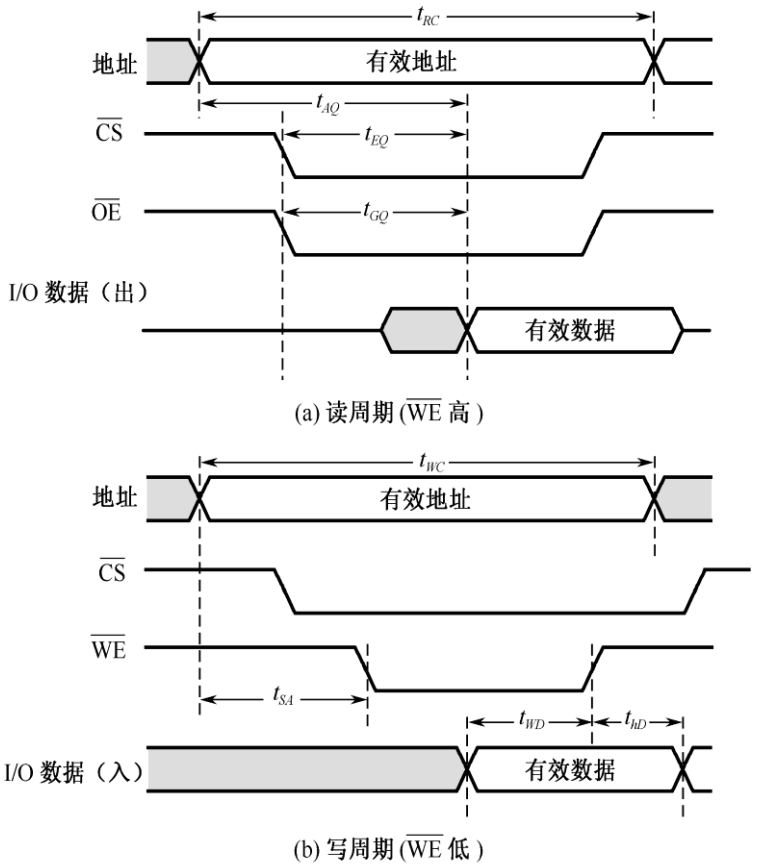
- 读与写的互锁逻辑

控制信号中CS是片选信号，CS有效时（低电平），门G1、G2均被打开。OE为读出使能信号，OE有效时（低电平），门G2开启，当写命令WE=1时（高电平），门G1关闭，存储器进行读操作。写操作时，WE=0，门G1开启，门G2关闭。注意，门G1和G2是互锁的，一个开启时另一个必定关闭，这样保证了读时不写，写时不读。

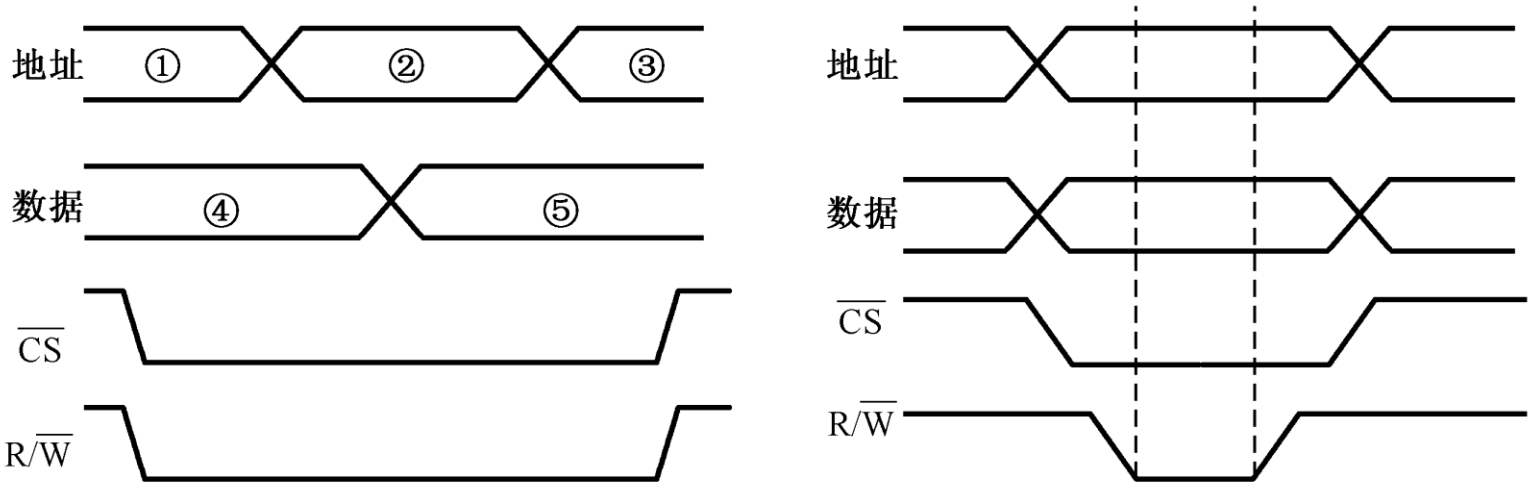


3.2.3 读/写周期波形图

- 读周期
  - 读出时间Taq
  - 读周期时间Trc
- 写周期
  - 写周期时间Twc
  - 写时间twd
- 存取周期
  - 读周期时间Trc=写时间twd



例1：图3.5(a)是SRA的写入时序图。其中R/W是读/写命令控制线，当R/W线为低电平时，存储器按给定地址把数据线上的数据写入存储器。请指出图3.5(a)写入时序中的错误，并画出正确的写入时序图。



(a) 错误时序　　(b) 正确时序

3.3 DRAM存储器

3.3.1 DRAM存储位元的记忆原理

3.3.2 DRAM芯片的逻辑结构

3.3.3 读/写周期、刷新周期

3.3.4 存储器容量的扩充

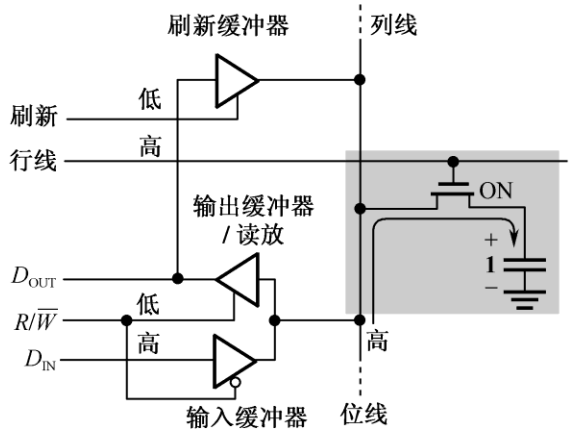
3.3.5 高级的DRAM结构

3.3.6 DRAM主存读/写的正确性校验

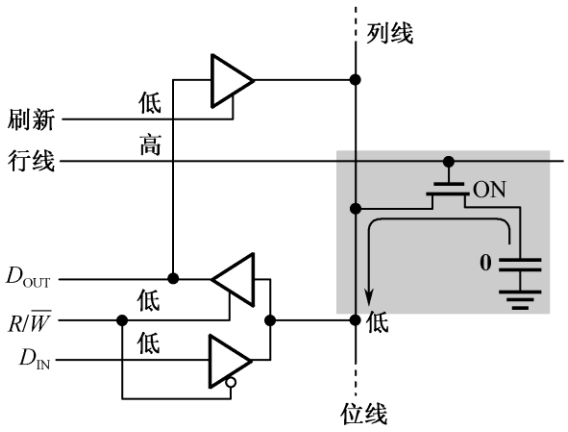
3.3.1 DRAM存储位元的记忆原理

SRAM存储器的存储位元是一个触发器，它具有两个稳定的状态。而DRAM存储器的存储位元是由一个MOS晶体管和电容器组成的记忆电路，如图3.6所示。

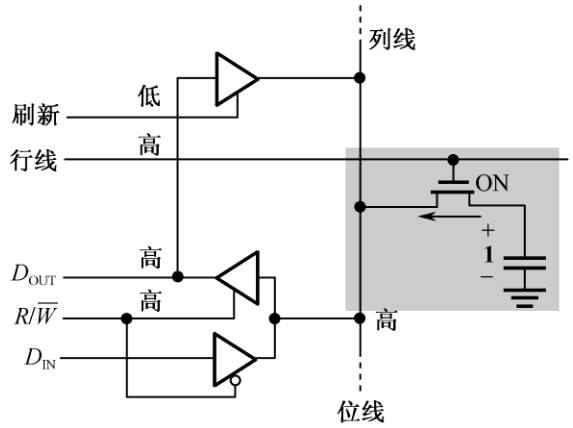
3.3.1 DRAM存储位元的记忆原理



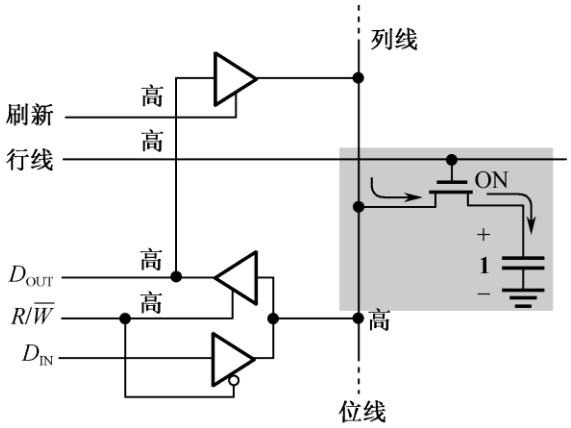
(a) 写 1 到存储位元



(b) 写 0 到存储位元



(c) 从存储位元读出 1



(d) 刷新存储位元的 1

3.3.2 DRAM芯片的逻辑结构

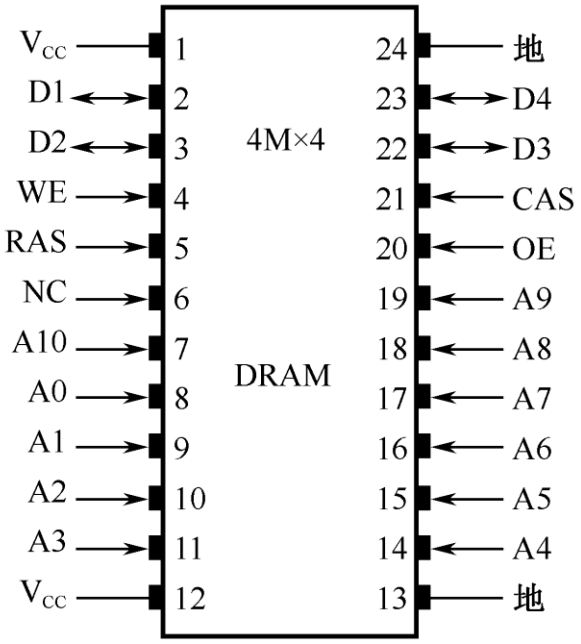
下面我们通过一个例子来看一下动态存储器的逻辑结构如图。

- 图3.7(a)示出1M×4位DRAM芯片的管脚图，其中有两个电源脚、两个地线脚，为了对称，还有一个空脚（NC）。
- 图3.7(b)是该芯片的逻辑结构图。与SRAM不同的是：

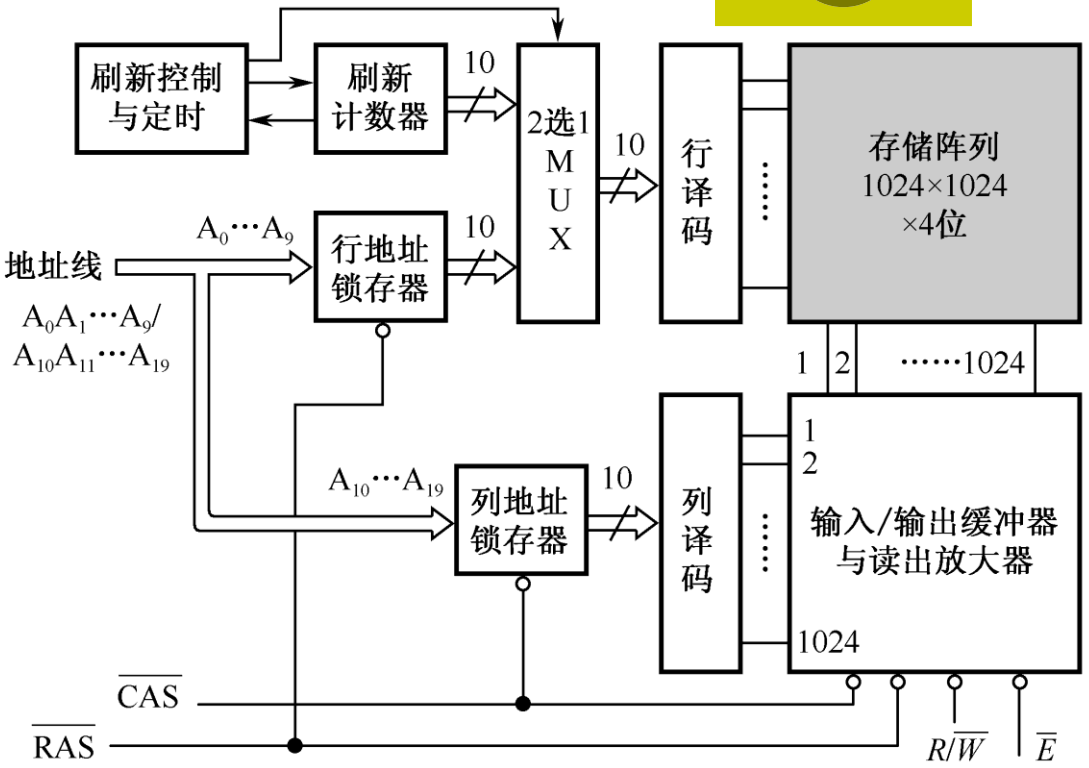
（1）增加了行地址锁存器和列地址锁存器。由于DRAM存储器容量很大，地址线宽度相应要增加，这势必增加芯片地址线的管脚数目。为避免这种情况，采取的办法是分时传送地址码。若地址总线宽度为10位，先传送地址码A0～A9，由行选通信号RAS打入到行地址锁存器；然后传送地址码A10～A19，由列选通信号CRS打入到列地址锁存器。芯片内部两部分合起来，地址线宽度达20位，存储容量为1M×4位。

（2）增加了刷新计数器和相应的控制电路。DRAM读出后必须刷新，而未读写的存储元也要定期刷新，而且要按行刷新，所以刷新计数器的长度等于行地址锁存器。刷新操作与读/写操作是交替进行的，所以通过2选1多路开关来提供刷新行地址或正常读/写的行地址。

3.3.2 DRAM芯片的逻辑结构



(a) 管脚图

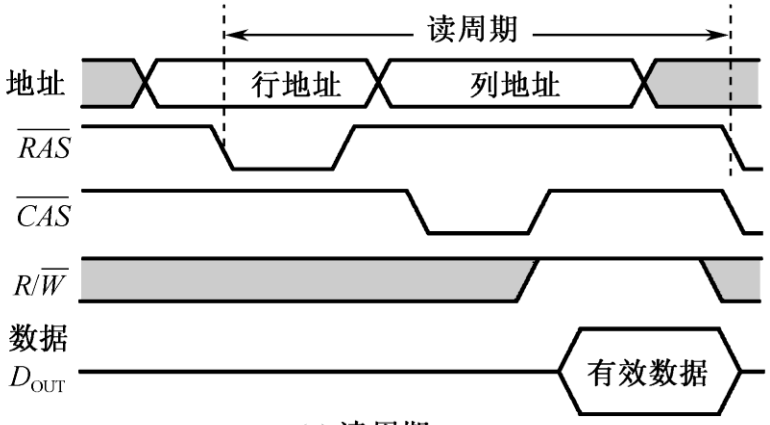


(b) 逻辑结构图

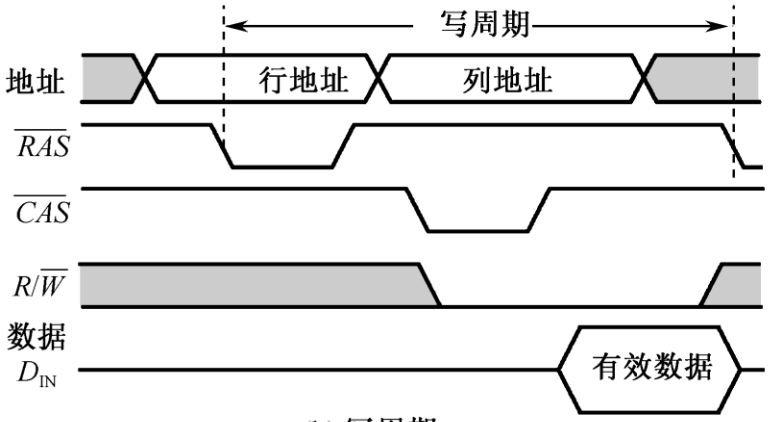
3.3.3 读/写周期、刷新周期

1、读/写周期

- 读周期、写周期的定义是从行选通信号RAS下降沿开始，到下一个RAS信号的下降沿为止的时间，也就是连续两个读周期的时间间隔。通常为控制方便，读周期和写周期时间相等。



(a) 读周期



(b) 写周期

3.3.3 读/写周期、刷新周期

2、 刷新周期

- 刷新周期：DRAM存储位元是基于电容器上的电荷量存储，这个电荷量随着时间和温度而减少，因此必须定期地刷新，以保持它们原来记忆的正确信息。
- 刷新操作有两种刷新方式：
  - 集中式刷新:DRAM的所有行在每一个刷新周期中都被刷新。
    - 例如刷新周期为8ms的内存来说，所有行的集中式刷新必须每隔8ms进行一次。为此将8ms时间分为两部分：前一段时间进行正常的读/写操作，后一段时间（8ms至正常读/写周期时间）做为集中刷新操作时间。
  - 分散式刷新:每一行的刷新插入到正常的读/写周期之中。
    - 例如p72图3.7所示的DRAM有1024行，如果刷新周期为8ms，则每一行必须每隔8ms÷1024=7.8us进行一次。

3.3.4 存储器容量的扩充

1、字长位数扩展

给定的芯片字长位数较短，不满足设计要求的存储器字长，此时需要用多片给定芯片扩展字长位数。三组信号线中，地址线和控制线公用而数据线单独分开连接。

d=设计要求的存储器容量/选择芯片存储器容量

[例2] 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：所需芯片数量=(1M×8)/(1M×4)=2片

设计的存储器字长为8位，存储器容量不变。连接的三组信号线与例相似，即地址线、控制线公用，数据线分高4位、低4位，但数据线是双向的，与SRAM芯片的I/O端相连接。见书上图3.9所示。

3.3.4 存储器容量的扩充

2、字存储容量扩展

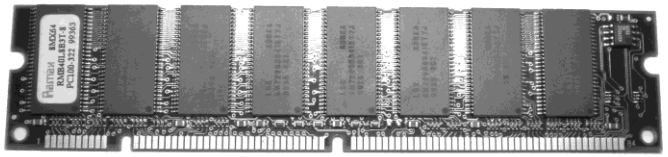
- 给定的芯片存储容量较小（字数少），不满足设计要求的总存储容量，此时需要用多片给定芯片来扩展字数。三组信号组中给定芯片的地址总线和数据总线公用，控制总线中R/W公用，使能端EN不能公用，它由地址总线的高位段译码来决定片选信号。所需芯片数仍由（d=设计要求的存储器容量/选择芯片存储器容量）决定。

[例3]利用1M×8位的DRAM芯片设计2M×8位的DRAM存储器

解：所需芯片数d=（2M×8）/（1M×8）=2(片)

设计的存储器见书上图3.10所示。字长位数不变，地址总线A_0~A_{19}同时连接到2片DRAM的地址输入端，地址总线最高位有A_{20}、$\overline{A}_{20}$，分别作为两片DRAM的片选信号，两个芯片不会同时工作。

3.3.4 存储器容量的扩充



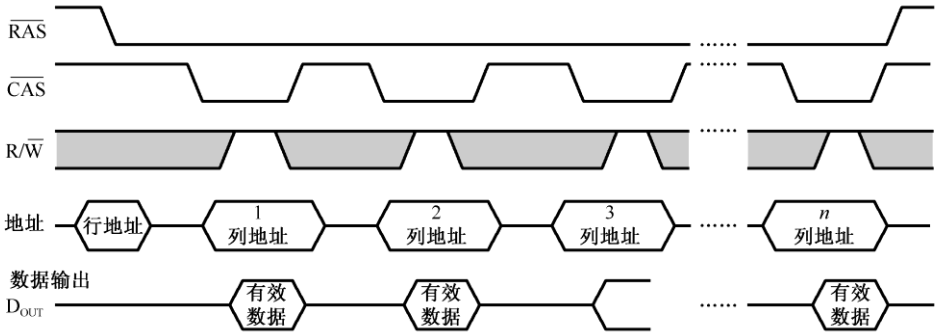
3、存储器模块条

- 存储器通常以插槽用模块条形式供应市场。这种模块条常称为内存条，它们是在一个条状形的小印制电路板上，用一定数量的存储器芯片，组成一个存储容量固定的存储模块。如图所示。
- 内存条有30脚、72脚、100脚、144脚、168脚等多种形式。
  - 30脚内存条设计成8位数据线，存储容量从256KB～32MB。
  - 72脚内存条设计成32位数据总线
  - 100脚以上内存条既用于32位数据总线又用于64位数据总线，存储容量从4MB～512MB。

3.3.5 高级的DRAM结构

1、FPM DRAM：

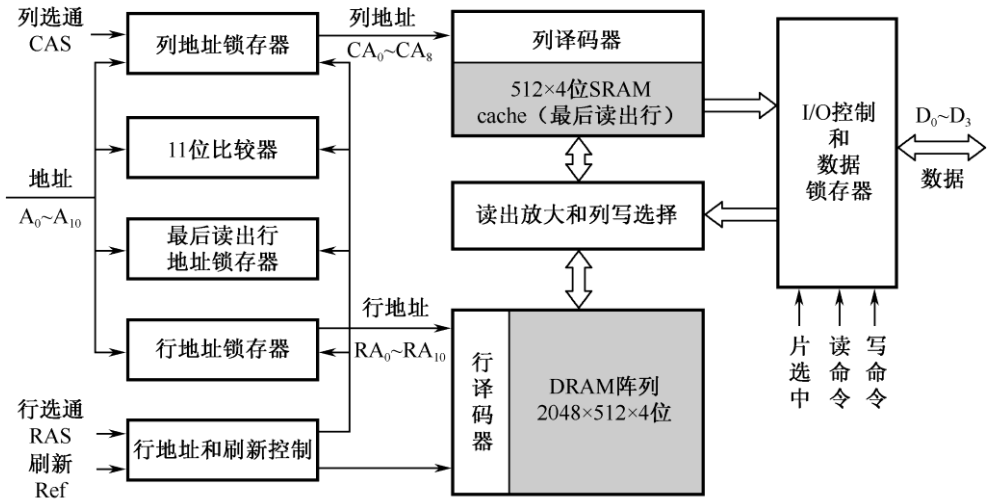
快速页模式动态存储器，它是根据程序的局部性原理来实现的。读周期和写周期中，为了寻找一个确定的存储单元地址，首先由低电平的行选通信号RAS确定行地址，然后由低电平的列选信号CAS确定列地址。下一次寻找操作，也是由RAS选定行地址，CAS选定列地址，依此类推，如下图所示。



3.3.5 高级的DRAM结构

2、CDRAM

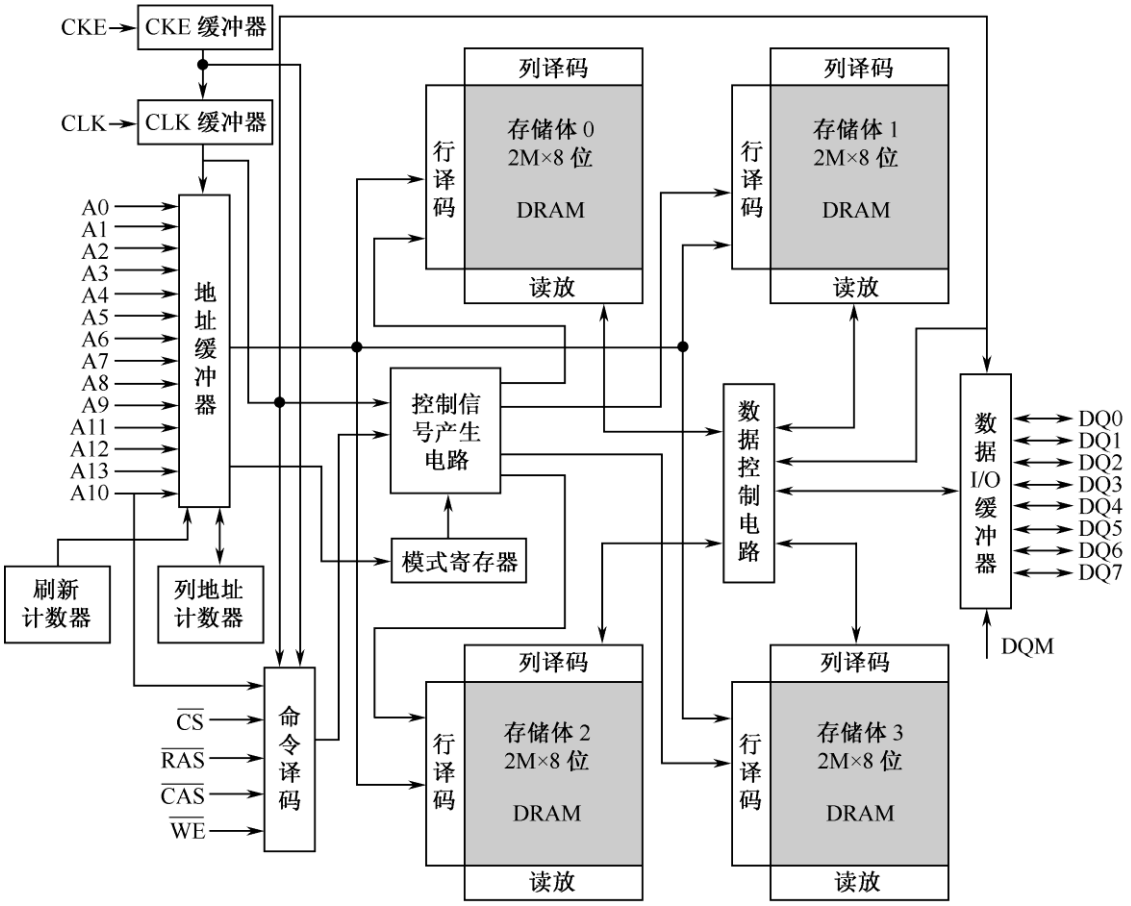
CDRAM称为带高速缓冲存储器（cache）的动态存储器，它是在通常的DRAM芯片内又集成了一个小容量的SRAM，从而使DRAM芯片的性能得到显著改进。如图所示出1M×4位CDRAM芯片的结构框图，其中SRAM为512×4位。



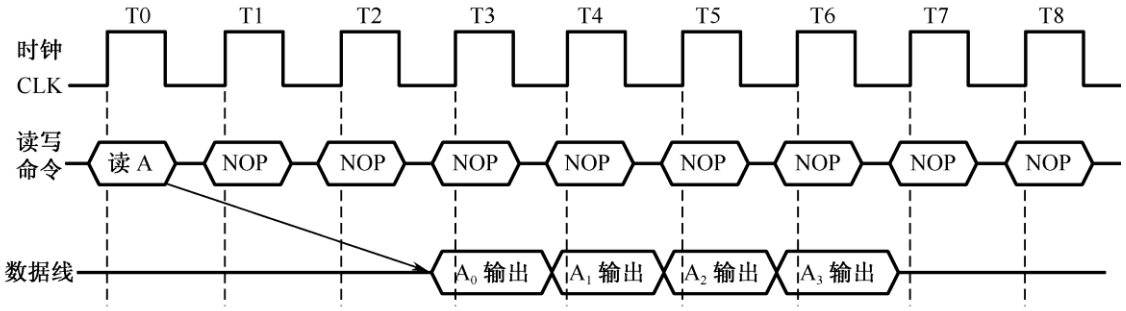
3.3.5 高级的DRAM结构

3、SDRAM

SDRAM称为同步型动态存储器。计算机系统中的CPU使用的是系统时钟，SDRAM的操作要求与系统时钟相同步，在系统时钟的控制下从CPU获得地址、数据和控制信息。换句话说，它与CPU的数据交换同步于外部的系统时钟信号，并且以CPU/存储器总线的最高速度运行，而不需要插入等待状态。其原理和时序关系见下一页图和动画。



(a) SDRAM 内部结构

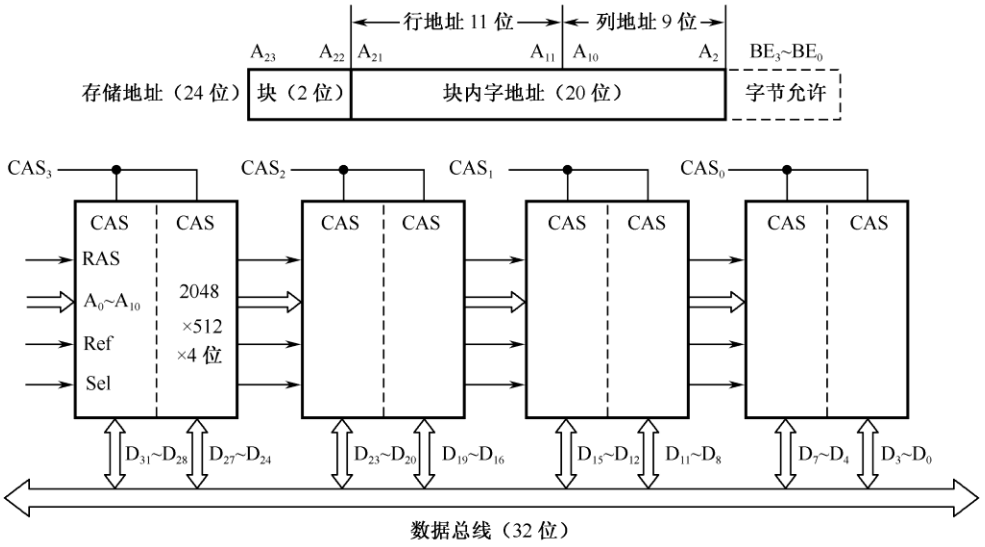


(b) SDRAM 读操作时序（猝发长度 =4　$\overline{\text{CAS}}$ 延时 =2）

3.3.5 高级的DRAM结构

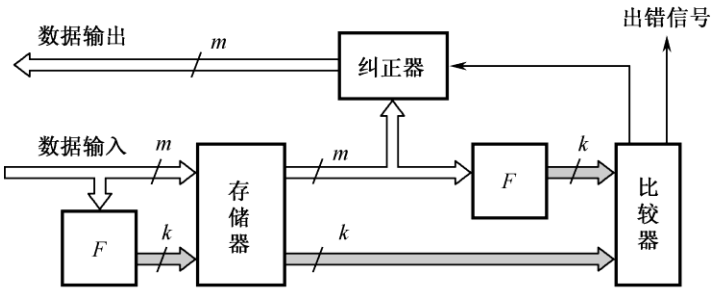
[例4] CDRAM内存条组成实例。

一片CDRAM的容量为1M×4位，8片这样的芯片可组成1M×32位4MB的存储模块，其组成如下图所示。

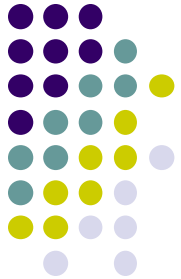


3.3.6 DRAM主存读/写的正确性校验

DRAM通常用做主存储器，其读写操作的正确性与可靠性至关重要。为此除了正常的数据位宽度，还增加了附加位，用于读/写操作正确性校验。增加的附加位也要同数据位一起写入DRAM中保存。其原理如图所示。



3.4 只读存储器和闪速存储器



3.4.1 只读存储器ROM

3.4.2 FLASH存储器

3.4.1 只读存储器ROM

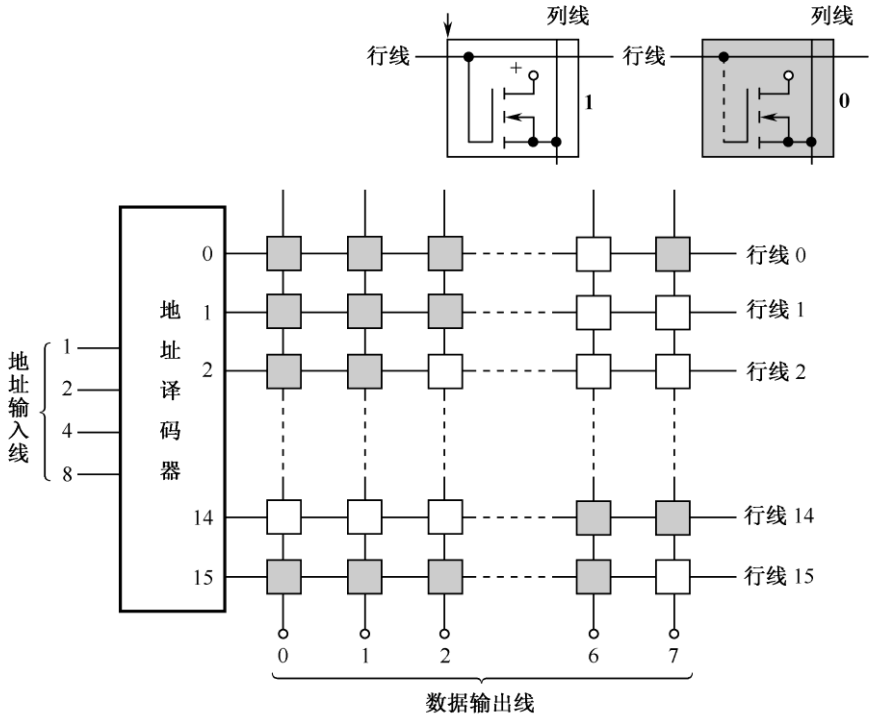
ROM叫做只读存储器。顾名思义，只读的意思是在它工作时只能读出，不能写入。然而其中存储的原始数据，必须在它工作以前写入。只读存储器由于工作可靠，保密性强，在计算机系统中得到广泛的应用。主要有两类：

- 掩模ROM：掩模ROM实际上是一个存储内容固定的ROM，由生产厂家提供产品。
- 可编程ROM：用户后写入内容，有些可以多次写入。
  - 一次性编程的PROM
  - 多次编程的EPROM和E^2PROM。

3.4.1 只读存储器ROM

1、掩模ROM

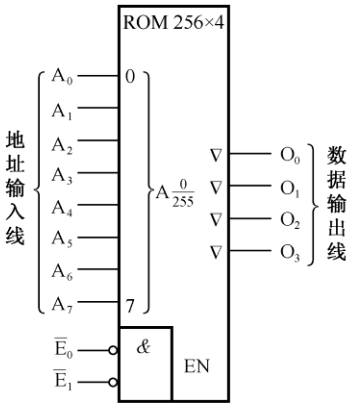
(1)掩模ROM的阵列结构和存储元



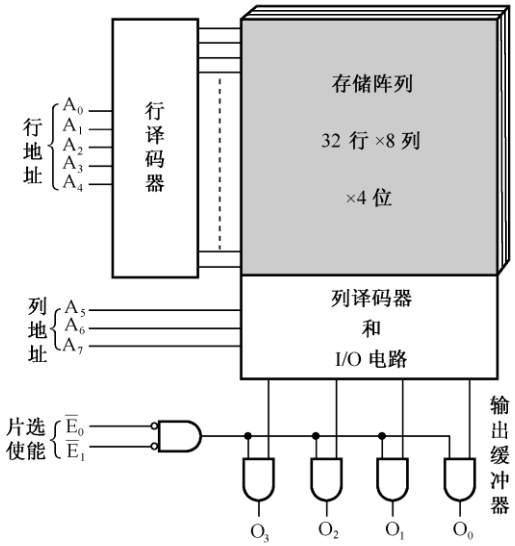
3.4.1 只读存储器ROM

1、掩模ROM

(2)掩模ROM的逻辑符号和内部逻辑框图



(a) 掩模 ROM 逻辑符号

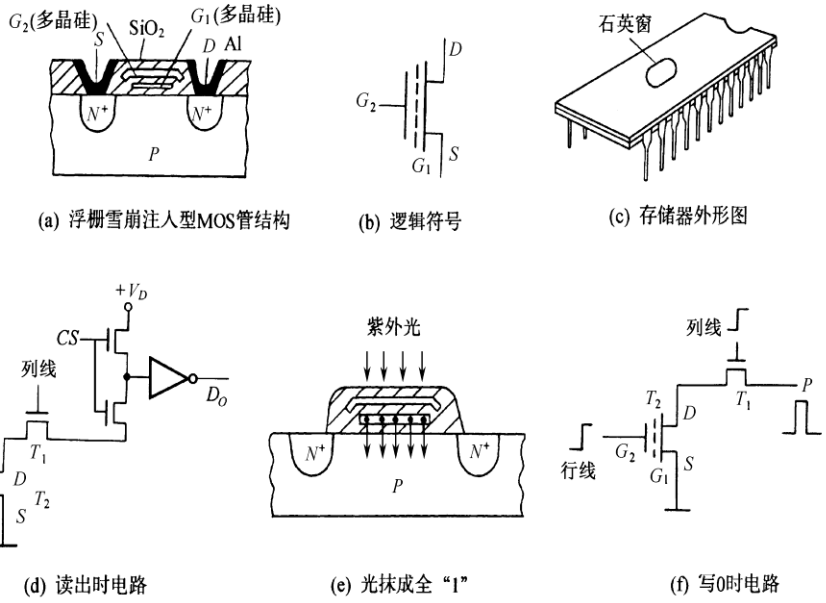


(b) 内部逻辑框图

3.4.1 只读存储器ROM

2、可编程ROM

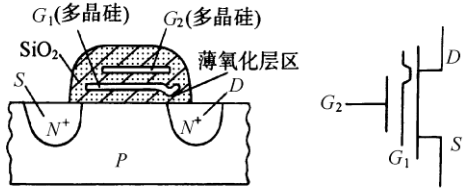
- EPROM叫做光擦除可编程可读存储器。它的存储内容可以根据需要写入，当需要更新时将原存储内容抹去，再写入新的内容。
- 现以浮栅雪崩注入型MOS管为存储元的EPROM为例进行说明，结构如右图所示。



3.4.1 只读存储器ROM

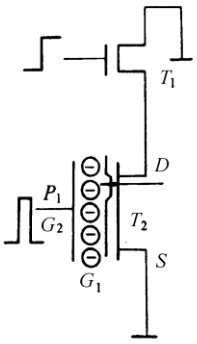
- **2、可编程ROM**
  E²PROM存储元

  EEPROM，叫做电擦除可编程只读存储器。其存储元是一个具有两个栅极的NMOS管，如图(a)和(b)所示，G1是控制栅，它是一个浮栅，无引出线；G2是抹去栅，它有引出线。在G1栅和漏极D之间有一小面积的氧化层，其厚度极薄，可产生隧道效应。如图(c)所示，当G2栅加20V正脉冲P1时，通过隧道效应，电子由衬底注入到G1浮栅，相当于存储了“1”。利用此方法可将存储器抹成全“1”状态。

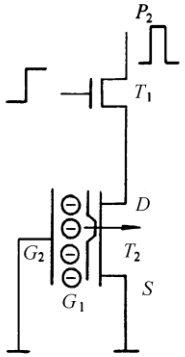


(a) 结构图

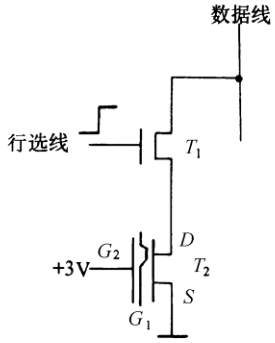
(b) 逻辑符号



(c) 抹成全“1”



(d) 写0时电路



(e) 读出时电路

3.4.2 FLASH存储器

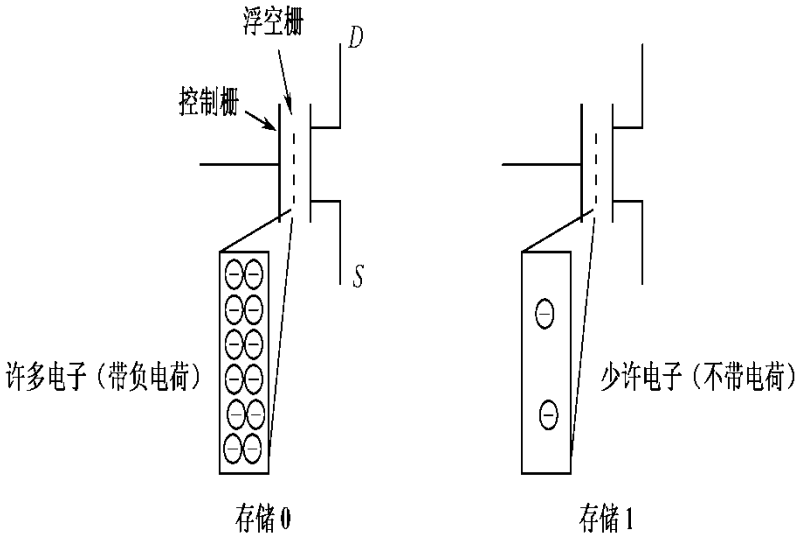
FLASH存储器也翻译成闪速存储器，它是高密度非失易失性的读/写存储器。高密度意味着它具有巨大比特数目的存储容量。非易失性意味着存放的数据在没有电源的情况下可以长期保存。总之，它既有RAM的优点，又有ROM的优点，称得上是存储技术划时代的进展。

3.4.2 FLASH存储器

- 1、FLASH存储元

  在EPROM存储元基础上发展起来的，由此可以看出创新与继承的关系。

- 如右图所示为闪速存储器中的存储元，由单个MOS晶体管组成，除漏极D和源极S外，还有一个控制栅和浮空栅。

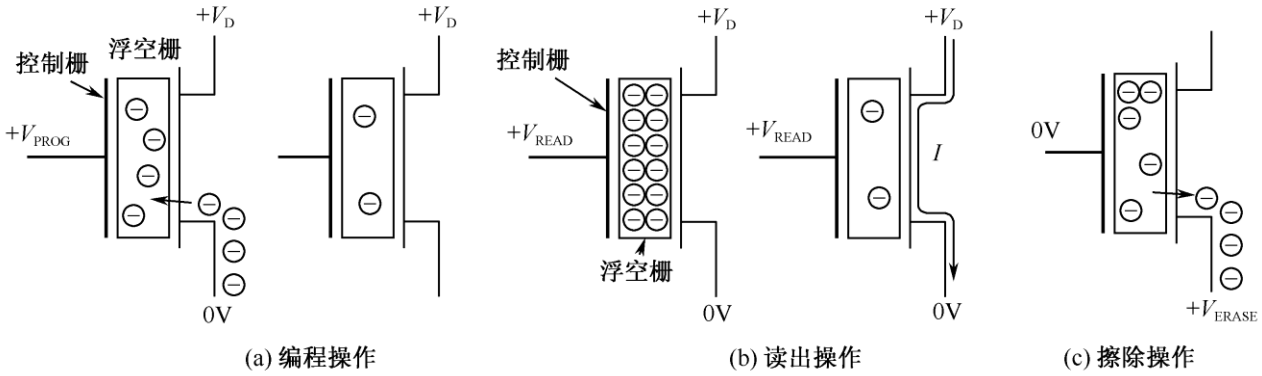


3.4.2 FLASH存储器

2、FLASH存储器的基本操作

编程操作、读取操作、擦除操作

- 如图(a)表示编程操作时存储元写0、写1的情况。实际上编程时只写0，不写1，因为存储元擦除后原始状态全为1。要写0，就是要在控制栅C上加正电压。一旦存储元被编程，存储的数据可保持100年之久而无需外电源。

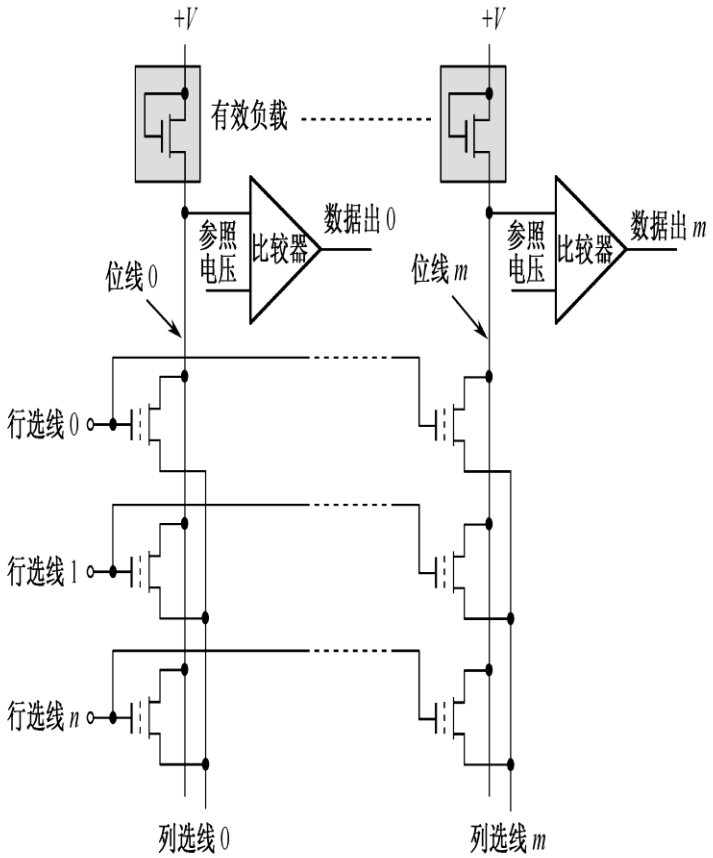


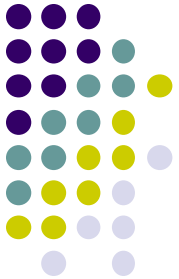
(a) 编程操作　　(b) 读出操作　　(c) 擦除操作

3.4.2 FLASH存储器

3、FLASH存储器的阵列结构

- FLASH存储器的简化阵列结构如右图所示。在某一时间只有一条行选择线被激活。读操作时，假定某个存储元原存1，那么晶体管导通，与它所在位线接通，有电流通过位线，所经过的负载上产生一个电压降。这个电压降送到比较器的一个输入端，与另一端输入的参照电压做比较，比较器输出一个标志为逻辑1的电平。如果某个存储元原先存0，那么晶体管不导通，位线上没有电流，比较器输出端则产生一个标志为逻辑0的电平。





3.5 并行存储器

3.5.1 双端口存储器

3.5.2多模块交叉存储器

3.5 并行存储器

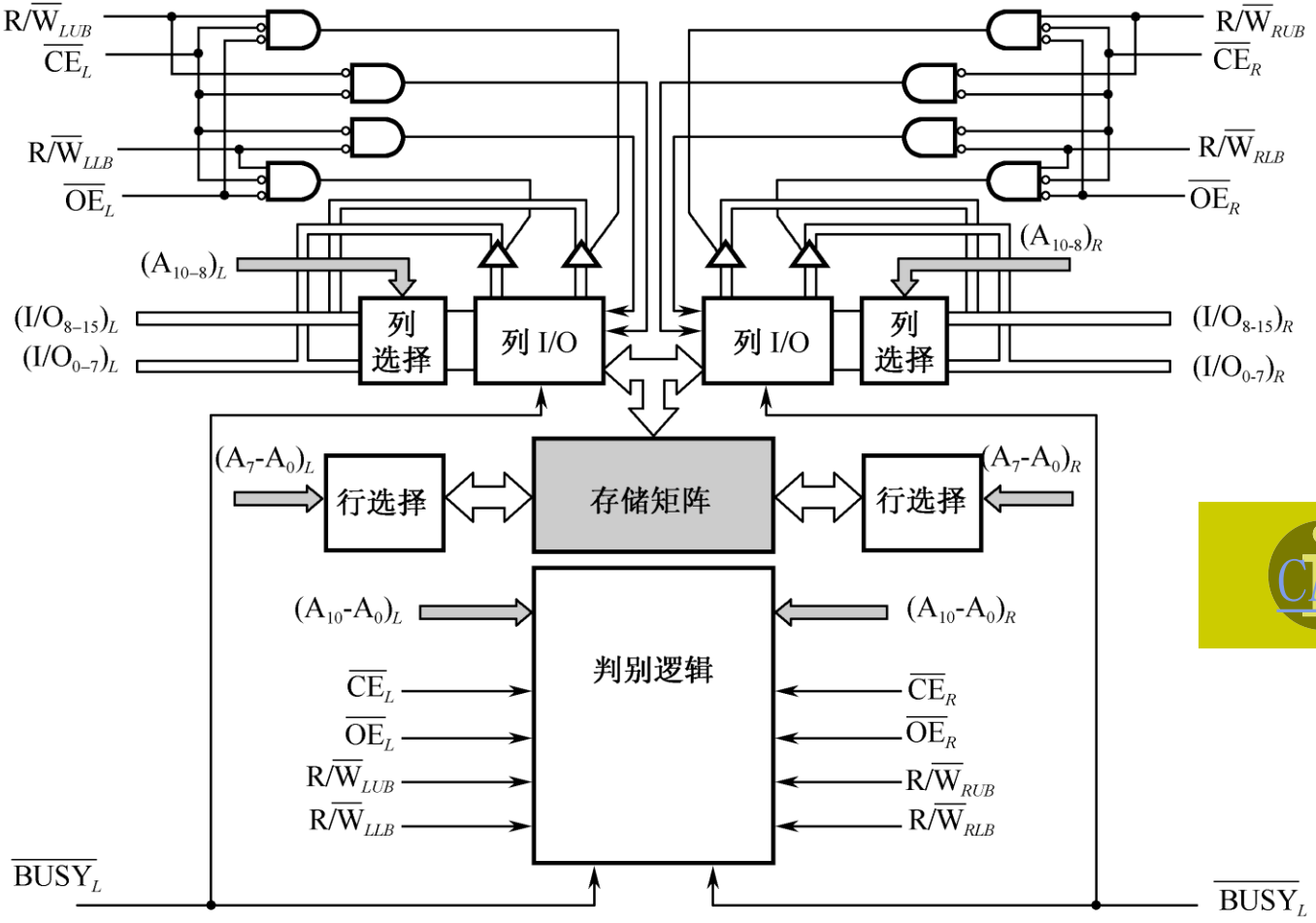
由于CPU和主存储器之间在速度上是不匹配的，这种情况便成为限制高速计算机设计的主要问题。为了提高CPU和主存之间的数据传输率，除了主存采用更高速的技术来缩短读出时间外，还可以采用并行技术的存储器。

3.5.1 双端口存储器

1、双端口存储器的逻辑结构

双端口存储器由于同一个存储器具有两组相互独立的读写控制电路而得名。由于进行并行的独立操作，因而是一种高速工作的存储器，在科研和工程中非常有用。 举例说明，双端口存储器IDT7133的逻辑框图 。如下页图。

3.5.1 双端口存储器


R/W_LUB
CE_L
R/W_LLB
OE_L
(A10–8)_L
(I/O8–15)_L
(I/O0–7)_L
列选择
列 I/O
列 I/O
列选择
R/W_RUB
CE_R
R/W_RLB
OE_R
(A10-8)_R
(I/O8-15)_R
(I/O0-7)_R
(A7-A0)_L
行选择
存储矩阵
行选择
(A7-A0)_R
(A10-A0)_L
判别逻辑
(A10-A0)_R
CE_L
OE_L
R/W_LUB
R/W_LLB
CE_R
OE_R
R/W_RUB
R/W_RLB
BUSY_L
BUSY_L

3.5.1 双端口存储器

2、无冲突读写控制

当两个端口的地址不相同时，在两个端口上进行读写操作，一定不会发生冲突。当任一端口被选中驱动时，就可对整个存储器进行存取，每一个端口都有自己的片选控制(CE)和输出驱动控制(OE)。读操作时，端口的OE(低电平有效)打开输出驱动器，由存储矩阵读出的数据就出现在I/O线上。

3、有冲突读写控制

当两个端口同时存取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对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(BUSY变为低电平)，即暂时关闭此端口。

3.5.1 双端口存储器

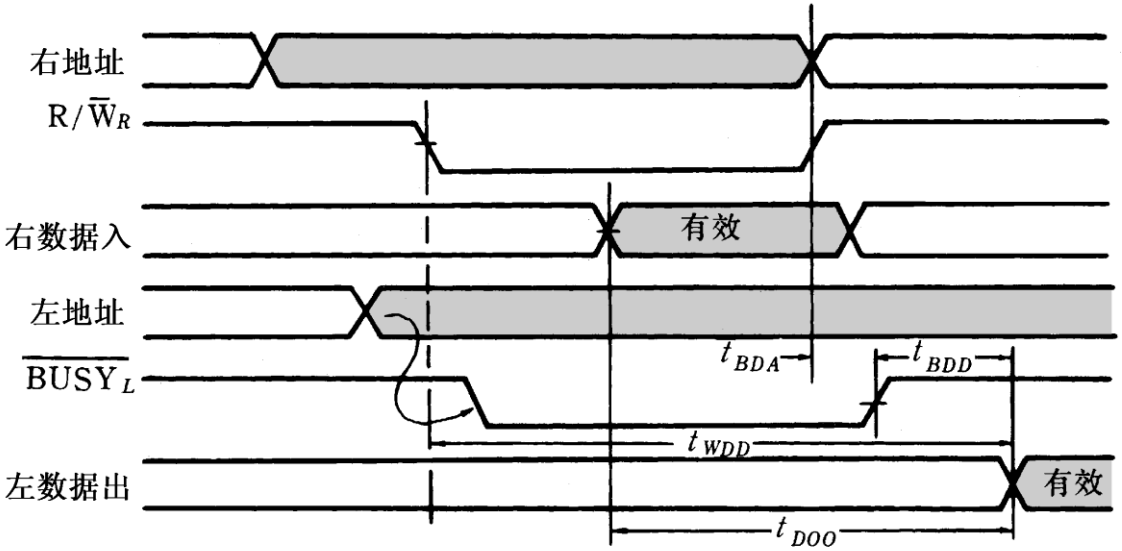
有冲突读写控制判断方法

(1)如果地址匹配且在CE之前有效，片上的控制逻辑在CEL和CER之间进行判断来选择端口(CE判断)。

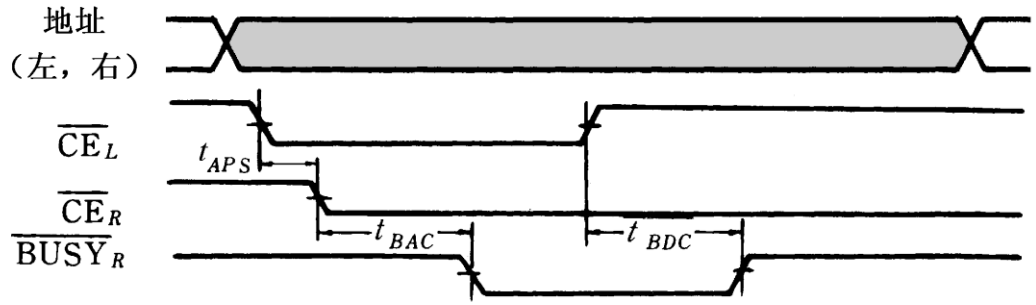
(2)如果CE在地址匹配之前变低，片上的控制逻辑在左、右地址间进行判断来选择端口(地址有效判断)。

无论采用哪种判断方式，延迟端口的BUSY标志都将置位而关闭此端口，而当允许存取的端口完成操作时，延迟端口BUSY标志才进行复位而打开此端口。

3.5.1双端口存储器



(a) 未发生冲突时读写时序波形（左读右写）

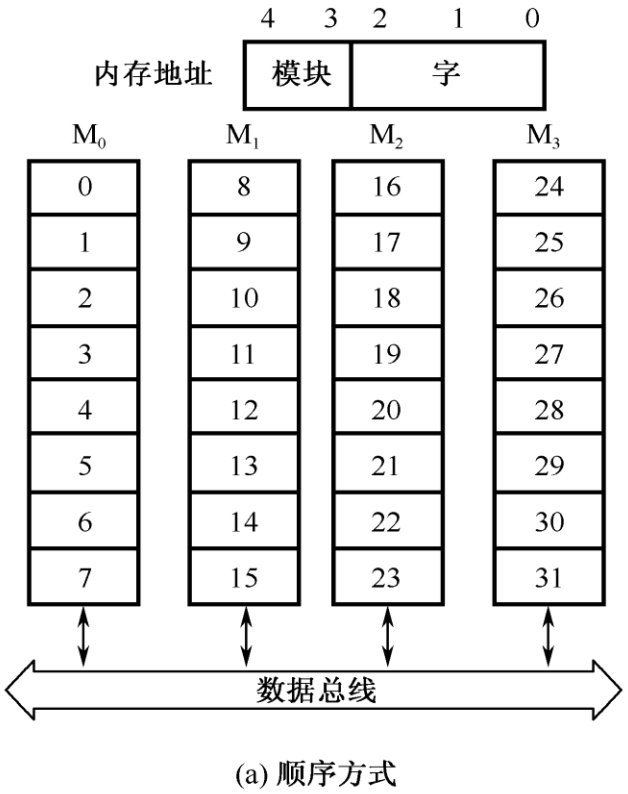


(b) 用$\overline{CE}$判断的冲突周期时序波形（$\overline{CE}_L$先有效）

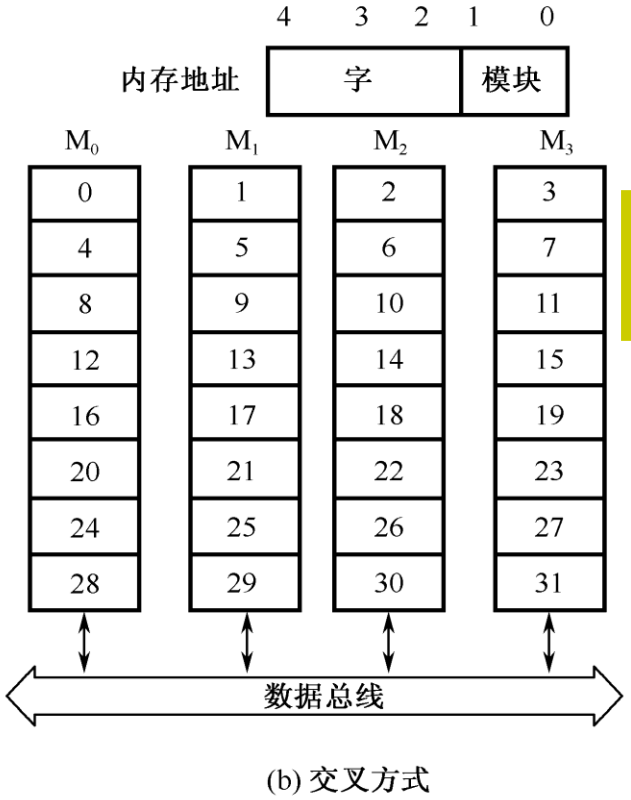
3.5 并行存储器

1、存储器的模块化组织

一个由若干个模块组成的主存储器是线性编址的。这些地址在各模块中如何安排，有两种方式：一种是顺序方式，一种是交叉方式



(a) 顺序方式



(b) 交叉方式

3.5.2 多模块交叉存储器

1、顺序方式

[例]M0—M3共四个模块，则每个模块8个字

顺序方式：　M0：0—7

M1：8—15

M2：16—23

M3：24—31

- 5位地址组织如下：　X X　X X X
- 高位选模块，低位选块内地址
- 特点：某个模块进行存取时，其他模块不工作，优点是某一模块出现故障时，其他模块可以照常工作，通过增添模块来扩充存储器容量比较方便。缺点是各模块串行工作，存储器的带宽受到了限制。

3.5.2 多模块交叉存储器

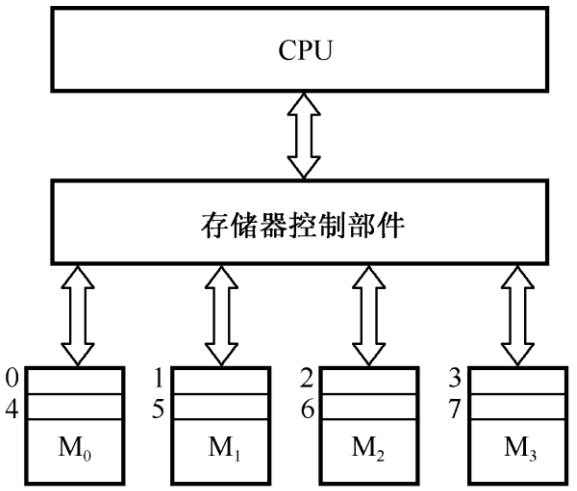
[例]M0—M3共四个模块，则每个模块8个字

- 交叉方式：
- M0：0，4,...除以4余数为0
- M1：1，5,...除以4余数为1
- M2：2，6,...除以4余数为2
- M3：3，7,...除以4余数为3
- 5位地址组织如下： X X X X X
- 高位选块内地址，低位选模块
- 特点：连续地址分布在相邻的不同模块内，同一个模块内的地址都是不连续的。优点是对连续字的成块传送可实现多模块流水式并行存取，大大提高存储器的带宽。使用场合为成批数据读取。

3.5.2 多模块交叉存储器

2、多模块交叉存储器的基本结构

右图为四模块交叉存储器结构框图。主存被分成4个相互独立、容量相同的模块M0，M1，M2，M3，每个模块都有自己的读写控制电路、地址寄存器和数据寄存器，各自以等同的方式与CPU传送信息。在理想情况下，如果程序段或数据块都是连续地在主存中存取，那么将大大提高主存的访问速度。

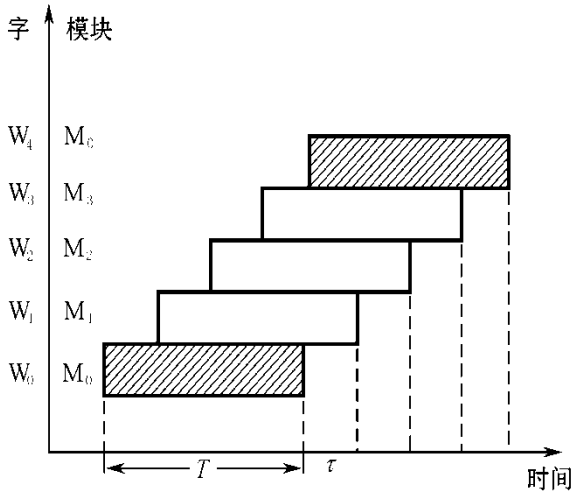


3.5.2 多模块交叉存储器

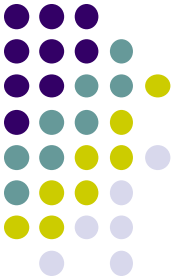
- 通常在一个存储器周期内，n个存储体必须分时启动，则各个存储体的启动间隔为 $t = T/n$ （n为交叉存取度）
- 整个存储器的存取速度有望提高n倍

$$t_{顺序} = xT$$

$$t_{交叉} = T + (x-1)t = T(\frac{x+n-1}{n})$$



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=50ns。若连续读出4个字，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：

$$q=64b\times4=256b$$

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：

$t_2=mT=4\times200ns=800ns=8\times10^{-7}s$

$t_1=T+(m-1)=200ns+350ns=350ns=35\times10^{-7}s$

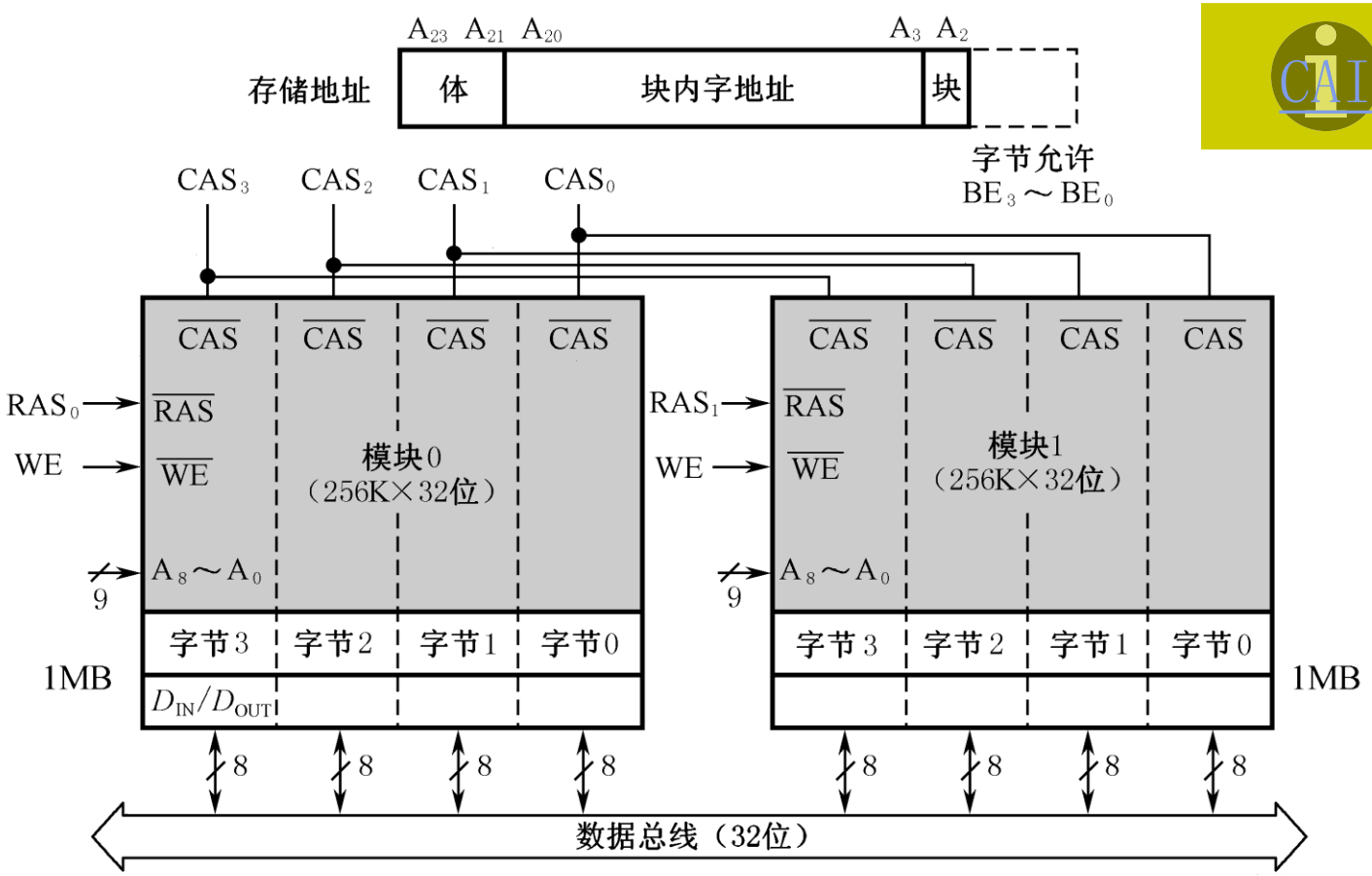
顺序存储器和交叉存储器的带宽分别是：

$W_2=q/t_2=256b\div(8\times10^{-7})s=320Mb/s$

$W_1=q/t_1=256b\div(35\times10^{-7})s=730Mb/s$

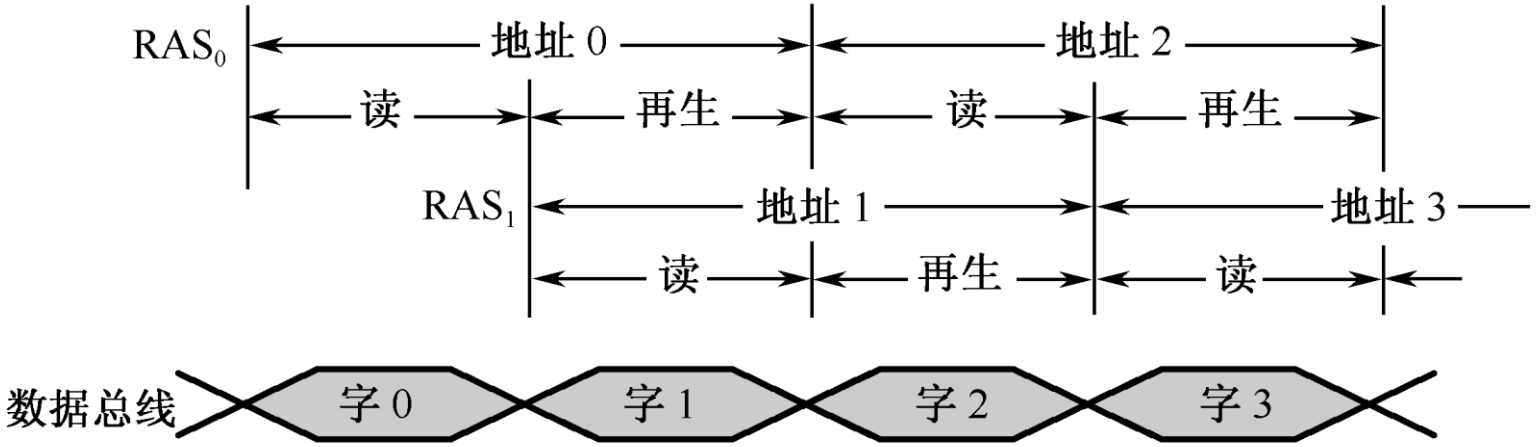
3.5.2 多模块交叉存储器

3、二模块交叉存储器举例



3.5.2 多模块交叉存储器

3、二模块交叉存储器举例



3.6 cache存储器

3.6.1 cache基本原理

3.6.2 主存与cache的地址映射

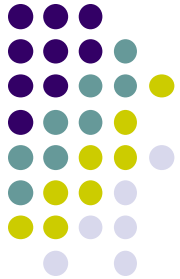
3.6.3替换策略

3.6.4 cache的写操作策略

3.6.5Pentium4的cache组织

3.6.6使用多级cache减少缺失损失

3.6.1 cache基本原理



1、cache的功能

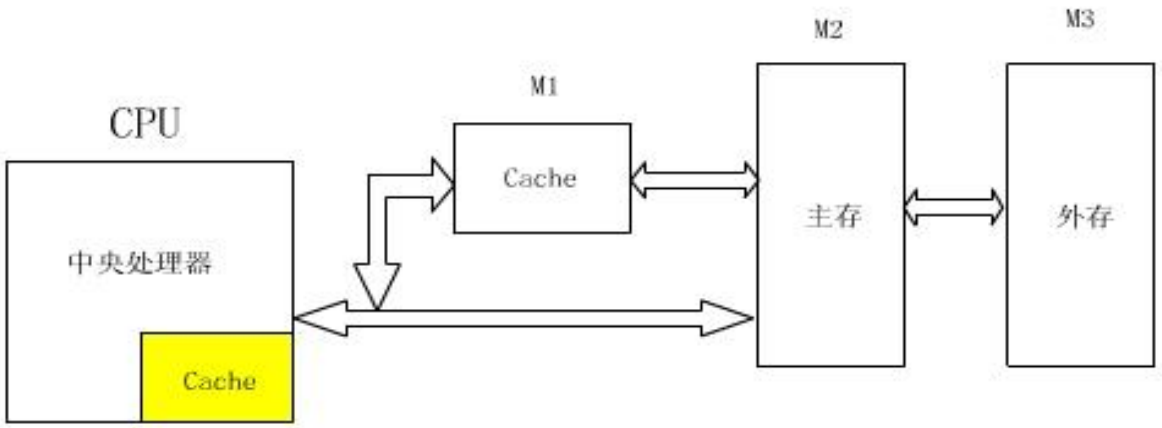
解决CPU和主存之间的速度不匹配问题

- 一般采用高速的SRAM构成。
- CPU和主存之间的速度差别很大采用两级或多级Cache系统
- 早期的一级Cache在CPU内，二级在主板上
- 现在的CPU内带L1 Cache和L2 Cache
- 全由硬件调度，对用户透明

3.6.1 cache基本原理

CPU与存储器系统的关系

图3.36 CPU与存储器系统的关系

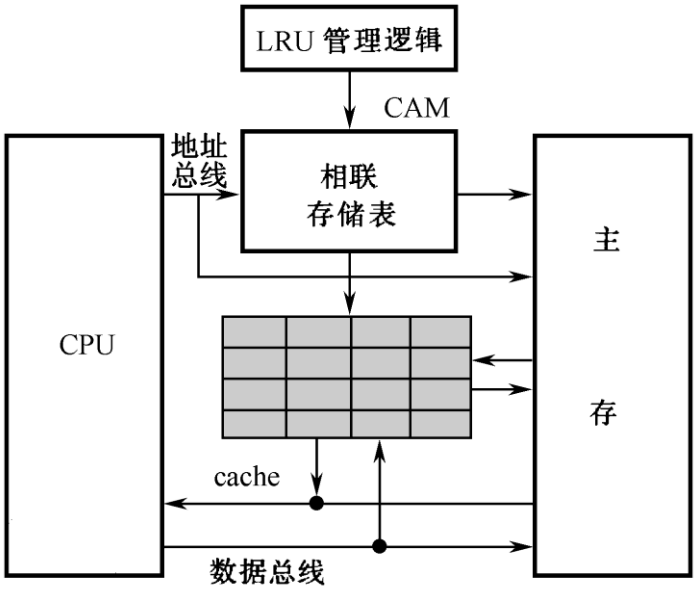


现在的Cache分片内Cache和片外Cache，片内Cache速度已接近CPU的速度.（完）

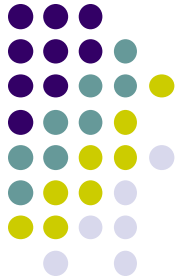
3.6.1 cache基本原理

2、cache基本原理

- 地址映射；
- 替换策略；
- 写一致性；
- 性能评价。



3.6.1 cache基本原理



3、 Cache的命中率

从CPU来看，增加一个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。为了达到这个目的，在所有的存储器访问中由cache满足CPU需要的部分应占很高的比例，即cache的命中率应接近于1。由于程序访问的局部性，实现这个目标是可能的。

3.6.1 cache基本原理

3、cache命中率公式

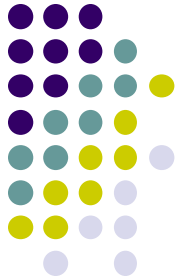
命中率 $$h = \frac{N_e}{N_e + N_m}$$

Cache/主存系统的平均访问时间 $$t_a = ht_c + (1-h)t_m$$

访问效率 $$e = \frac{t_a}{t_c} = \frac{1}{r + (1-r)h}$$

Cache与内存的速度比 $$r = t_m / t_c$$

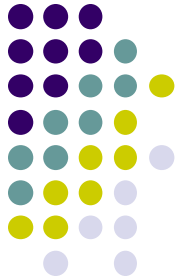
例6 CPU执行一段程序时，cache完成存取的次数为1900次，主存完成存取的次数为100次，已知cache存取周期为50ns，主存存取周期为250ns，求cache/主存系统的效率和平均访问时间。



解：

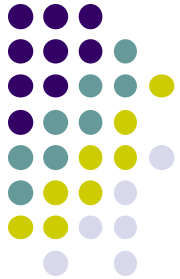
- h=Nc/（Nc+Nm）=1900/(1900+100)=0.95
- r=tm/tc=250ns/50ns=5
- e=1/(r+(1-r)h)=1/(5+(1-5)×0.95=83.3%
- ta=tc/e=50ns/0.833=60ns

3.6.2主存与Cache的地址映射



- 无论选择那种映射方式，都要把主存和cache划分为同样大小的“块”。
- 选择哪种映射方式，要考虑：
  - 硬件是否容易实现
  - 地址变换的速度是否快
  - 主存空间的利用率是否高
  - 主存装入一块时，发生冲突的概率
- 以下我们介绍三种映射方法

3.6.2 主存与cache的地址映射

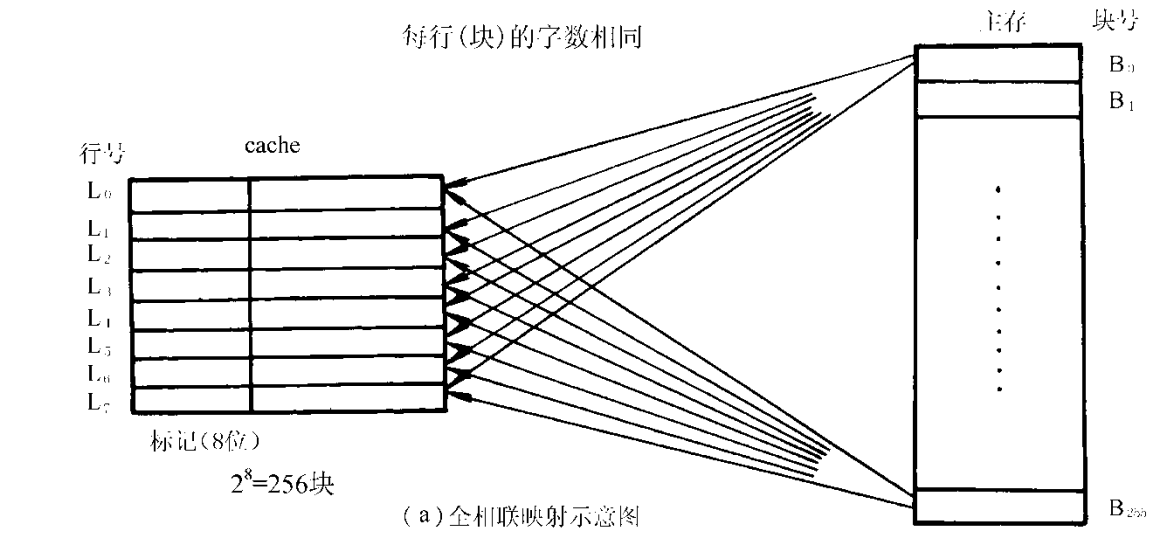


1、全相联的映射方式

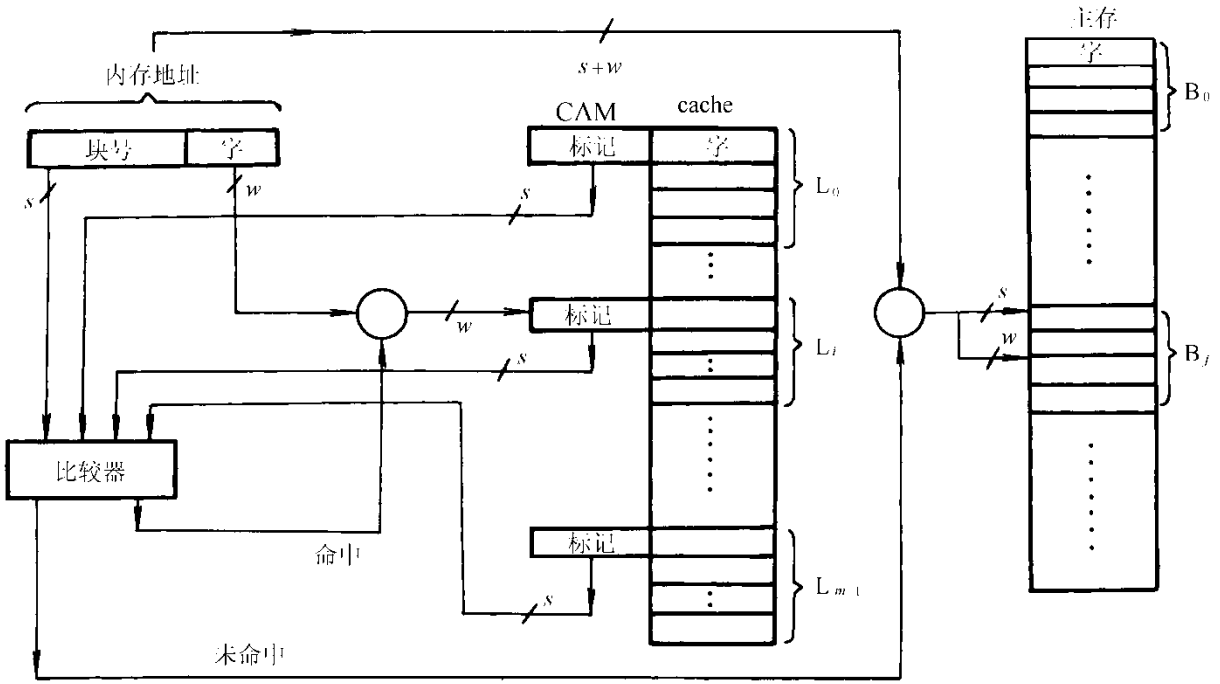
（1）将地址分为两部分（块号和字），在内存块写入Cache时，同时写入块号标记；

（2）CPU给出访问地址后，也将地址分为两部分（块号和字），比较电路块号与Cache 表中的标记进行比较，相同表示命中，访问相应单元；如果没有命中访问内存，CPU 直接访问内存，并将被访问内存的相对应块写入Cache。

1、全相联的映射方式



(a)全相联映射示意图



(b) 全相联cache的检索过程

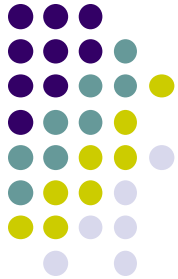
3.6.2 主存与cache的地址映射

1、全相联的映射方式

转换公式

主存地址长度＝(s+w)位
寻址单元数＝2^w个字或字节
块大小＝行大小＝2^w个字或字节
主存的块数＝2^s
标记大小＝s位
cache的行数＝不由地址格式确定

3.6.2 主存与cache的地址映射



1、全相联的映射方式

特点：

- 优点：冲突概率小，Cache的利用高。
- 缺点：比较器难实现，需要一个访问速度很快代价高的相联存储器

应用场合：

- 适用于小容量的Cache

3.6.2 主存与cache的地址映射

2、直接映射方式

映射方法（一对多）如：

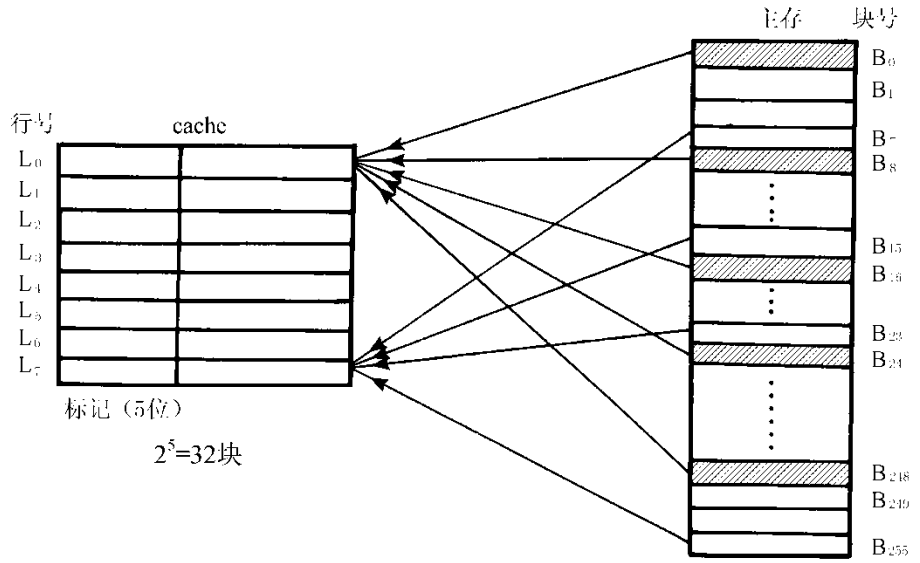
- i= j mod m
- 主存第j块内容拷贝到Cache的i行
- 一般I和m都是2^N级

[例]cache容量16字，主存容量256字，则地址2，18，34…..242等都存放在cache的地址2内，如果第一次2在cache中，下次访问34内容，则不管cache其他位置的内容访问情况，都会引起2块内容的替换

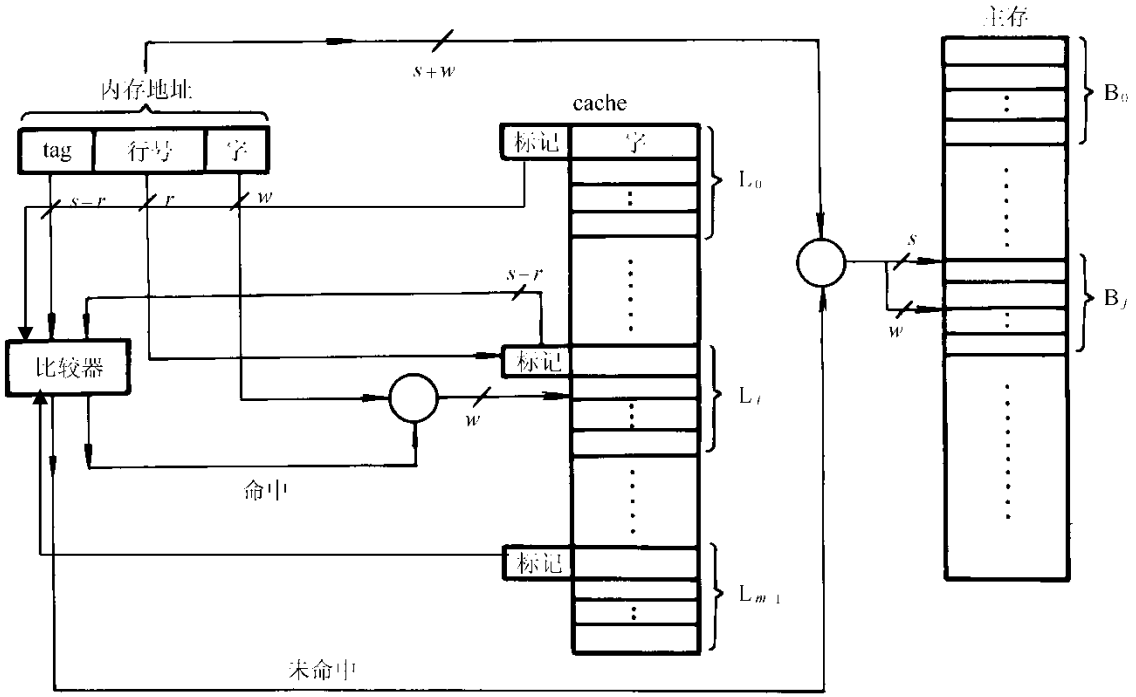
2、直接映射方式

2、基本原理

- 利用行号选择相应行；
- 把行标记与CPU访问地址进行比较，相同表示命中，访问Cache；
- 如果没有命中，访问内　存，并将相应块写入Cache



(a) 直接映射示意图



(b) 直接映射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

转换公式

主存地址长度＝(s+w)位
寻址单元数＝2^{s+w}个字或字节
块大小＝行大小＝2^{w}个字或字节
主存的块数＝2^{s}
cache的行数＝m＝2^{r}
标记大小＝(s-r)位

3.6.2 主存与cache的地址映射

2、直接映射方式

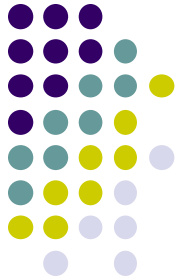
特点

- 优点：比较电路少m倍线路，所以硬件实现简单，Cache地址为主存地址的低几位，不需变换。
- 缺点：冲突概率高（抖动）

应用场合

- 适合大容量Cache

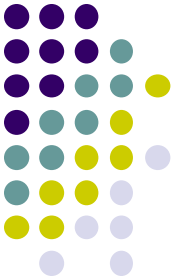
3.6.2 主存与cache的地址映射



3、组相联映射方式

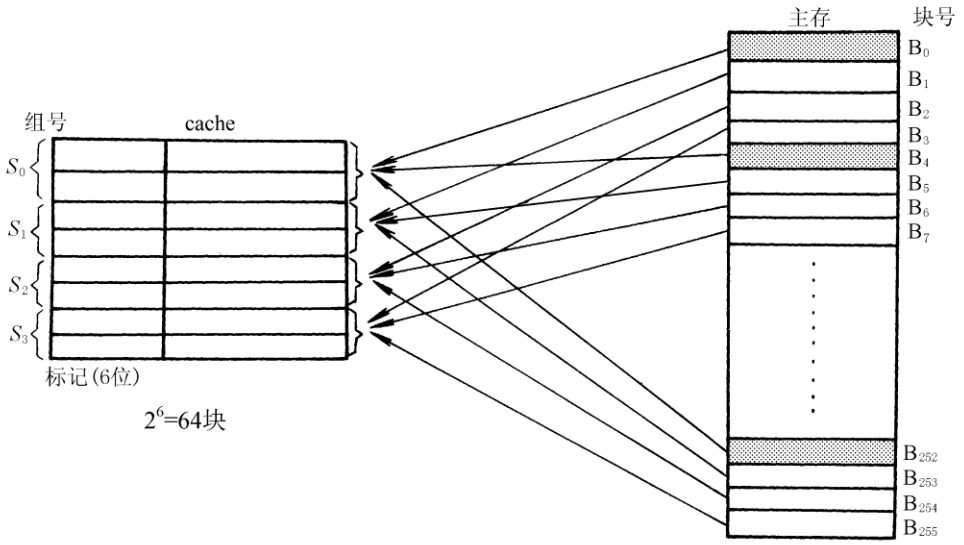
- 前两者的组合
  - Cache分组，组间采用直接映射方式，组内采用全相联的映射方式
  - Cache分组U，组内容量V
  - 映射方法（一对多）
    - q= j mod u
    - 主存第j块内容拷贝到Cache的q组中的某行
  - 地址变换
    - 设主存地址x，看是不是在cache中，先y= x mod u，则在y组中一次查找

3.6.2 主存与cache的地址映射

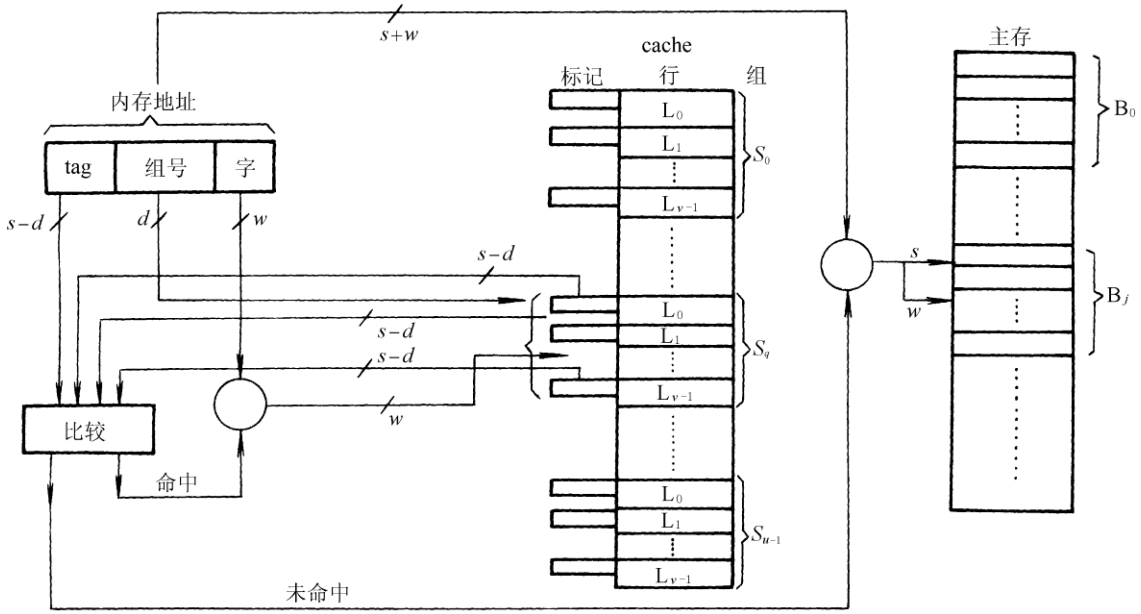


3、组相联映射方式

- 分析：比全相联容易实现，冲突低
- v=1，则为直接相联映射方式
- u=1，则为全相联映射方式
- v的取值一般比较小， 一般是2的幂，称之为v路组相联cache.



(a) 组相联映射示意图(4组)



(b) 组相联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

转换公式

主存地址长度＝(s+w)位
寻址单元数＝2^{s+w}个字或字节
块大小＝行大小＝2^{w}个字或字节
主存的块数＝2^{s}
每组的行数＝k
每组的v＝2^{d}
cache的行数＝kv
标记大小＝(s-d)位

3.6.2 主存与cache的地址映射

[例 7]直接映射方式的内存地址格式如下所示：

标记 s-r	行 r	字 w
8 位	14 位	2 位

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式 cache 的标记、行、字的值。

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

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

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

字　$w=(11)_2=(3)_{16}$

3.6.2 主存与cache的地址映射

例8：一个组相联cache由64个行组成，每组4行。主存包含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＝128＝2^7　　∴ w＝7

每组的行数k＝4

cache的行数＝kv＝K×2^d＝4×2^d＝64　　∴ d＝4

组数v＝2^d＝2^4＝16

主存的块数2^s＝4K＝2^2×2^{10}＝2^2　　∴ s＝12

标记大小(s-d)位＝12-4＝8位

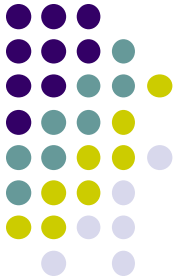
主存地址长度(s+w)位＝12+7＝19位

主存寻址单元数2^{s+w}＝2^{19}

故 k＝4各组相联的内存地址格式如下所示：

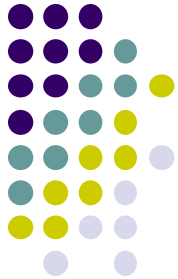
标记s-d	组号d	字号w
8位	4位	7位

3.6.3 替换策略



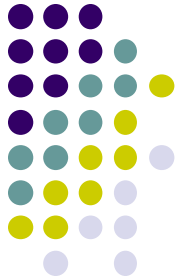
- LFU（最不经常使用）：被访问的行计数器增加1，换值小的行，不能反映近期cache的访问情况，
- LRU（近期最少使用）：被访问的行计数器置0，其他的计数器增加1，换值大的行，符合cache的工作原理
- 随机替换：随机替换策略实际上是不要什么算法，从特定的行位置中随机地选取一行换出即可。这种策略在硬件上容易实现，且速度也比前两种策略快。缺点是随意换出的数据很可能马上又要使用，从而降低命中率和cache工作效率。但这个不足随着cache容量增大而减小。随机替换策略的功效只是稍逊于前两种策略。

3.6.4 写操作策略



- 由于cache的内容只是主存部分内容的拷贝，它应当与主存内容保持一致。而CPU对cache的写入更改了cache的内容。如何与主存内容保持一致，可选用如下三种写操作策略。
  - 写回法：换出时，对行的修改位进行判断，决定是写回还是舍掉。
  - 全写法：写命中时，Cache与内存一起写
  - 写一次法：与写回法一致，但是第一次Cache命中时采用全写法。

3.6.5 Pentium 4的Cache组织



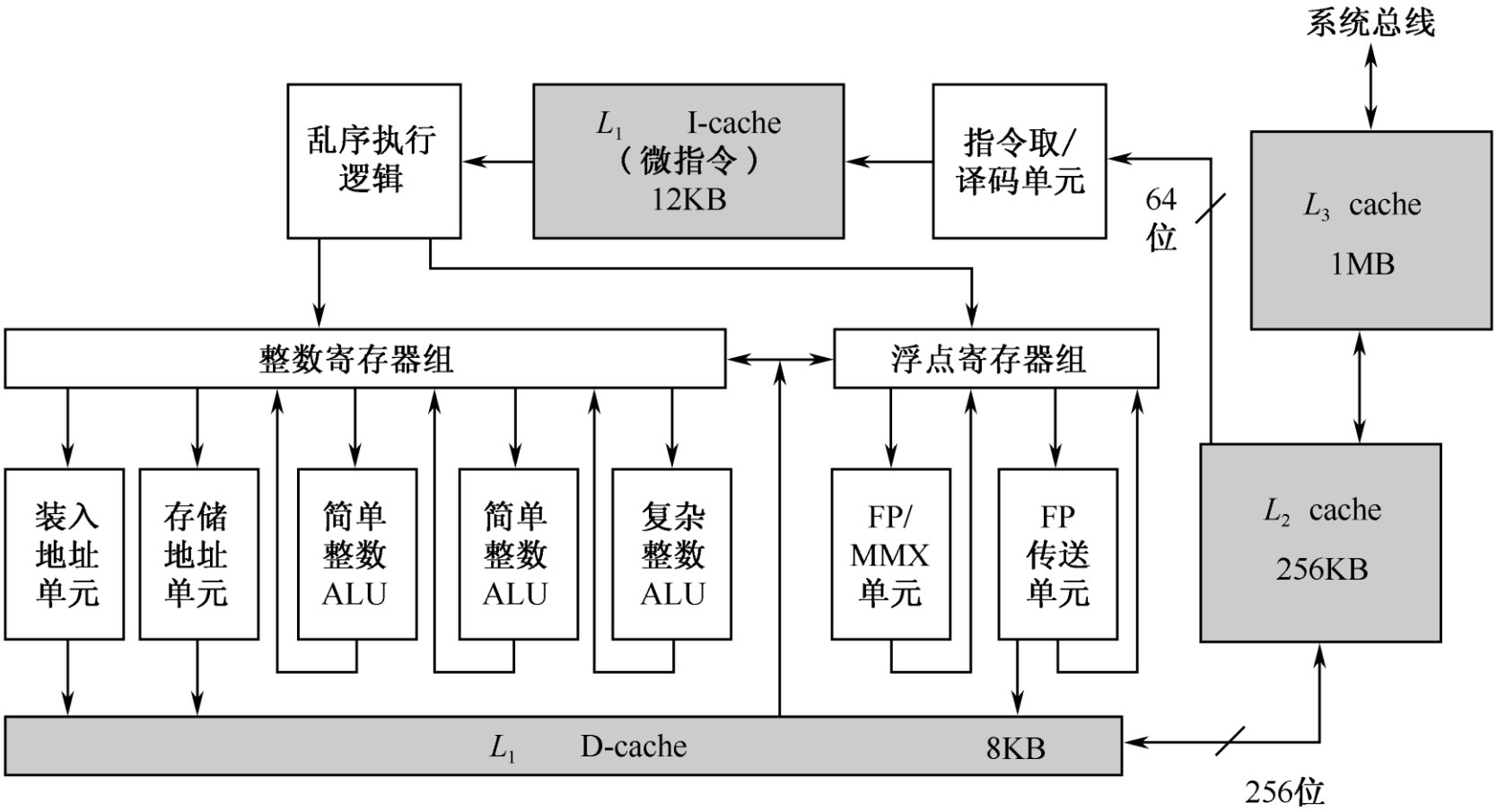
主要包括四个部分：

- 取指/译码单元：顺序从L2cache中取程序指令，将它们译成一系列的微指令，并存入L1指令cache中。
- 乱序执行逻辑：依据数据相关性和资源可用性，调度微指令的执行，因而微指令可按不同于所取机器指令流的顺序被调度执行。
- 执行单元：它执行微指令，从L1数据cache中取所需数据，并在寄存器组中暂存运算结果。
- 存储器子系统：这部分包括L2cache、L3cache和系统总线。当L1、L2cache未命中时，使用系统总线访问主存。系统总线还用于访问I/O资源。

不同于所有先前Pentium模式和大多数处理器所采用的结构，Pentium 4的指令cache位于指令译码逻辑和执行部件之间。其设计理念是：Pentium 4将机器指令译成由微指令组成的简单RISC类指令，而使用简单定长的微指令可允许采用超标量流水线和调度技术，从而增强机器的性能。

3.6.5 Pentium 的Cache组织

- 基本原理见下图



3.6.6 使用多级cache减少缺失损失

为进一步缩小现代CPU和DRAM访问速度的差距，CPU支持附加一级的cache。二级cache在访问主cache缺失时被访问，各级cache都不包含所访问数据时，需要访问主存储器。

[例10] 现有一处理器，基本CPI为1.0，所有访问在第一级cache中命中，时钟频率5GHz。假定访问一次主存储器的时间为100ns，其中包括所有缺失处理。设平均每条指令在第一级cache中产生的缺失率为2%。若增加一个二级cache，命中或缺失的访问时间都为5ns，且容量大到可使必须访问主存的缺失率降为0.5%，问处理器速度提高多少。

解得

只有一级cache的CPU：总的CPI＝11.0

有二级cache的CPU：总的CPI＝4.0

后者是前者CPU性能的：11.0÷4.0＝2.8倍

3.7 虚拟存储器

3.7.1 虚拟存储器的基本概念

3.7.2 页式虚拟存储器

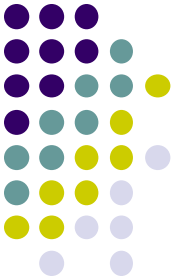
3.7.3段式虚拟存储器和段页式虚拟存储器

3.7.4 虚存的替换算法

3.7.1 虚拟存储器的基本概念

1、实地址与虚地址:用户编制程序时使用的地址称为虚地址或逻辑地址，其对应的存储空间称为虚存空间或逻辑地址空间；而计算机物理内存的访问地址则称为实地地或物理地址，其对应的存储空间称为物理存储空间或主存空间。程序进行虚地址到实地址转换的过程称为程序的再定位。

3.7.1 虚拟存储器的基本概念



2、虚存的访问过程

虚存空间的用户程序按照虚地址编程并存放在辅存中。程序运行时，由地址变换机构依据当时分配给该程序的实地址空间把程序的一部分调入实存。每次访存时，首先判断该虚地址所对应的部分是否在实存中：如果是，则进行地址转换并用实地址访问主存；否则，按照某种算法将辅存中的部分程序调度进内存，再按同样的方法访问主存。由此可见，每个程序的虚地址空间可以远大于实地址空间，也可以远小于实地址空间。前一种情况以提高存储容量为目的，后一种情况则以地址变换为目的。后者通常出现在多用户或多任务系统中：实存空间较大，而单个任务并不需要很大的地址空间，较小的虚存空间则可以缩短指令中地址字段的长度。

3.7.1 虚拟存储器的基本概念

3、cache与虚存的异同

- 从虚存的概念可以看出，主存辅存的访问机制与cache　主存的访问机制是类似的。这是由cache存储器、主存和辅存构成的三级存储体系中的两个层次。
- cache和主存之间以及主存和辅存之间分别有辅助硬件和辅助软硬件负责地址变换与管理，以便各级存储器能够组成有机的三级存储体系。cache和主存构成了系统的内存，而主存和辅存依靠辅助软硬件的支持构成了虚拟存储器。

3.7.1 虚拟存储器的基本概念

在三级存储体系中，cache　主存和主存辅存这两个存储层次有许多相同点：

(1)出发点相同 二者都是为了提高存储系统的性能价格比而构造的分层存储体系，都力图使存储系统的性能接近高速存储器，而价格和容量接近低速存储器。

(2)原理相同 都是利用了程序运行时的局部性原理把最近常用的信息块从相对慢速而大容量的存储器调入相对高速而小容量的存储器。

但cache　主存和主存辅存这两个存储层次也有许多不同之处：

(3)侧重点不同 cache主要解决主存与CPU的速度差异问题；而就性能价格比的提高而言，虚存主要是解决存储容量问题，另外还包括存储管理、主存分配和存储保护等方面。

(4)数据通路不同 CPU与cache和主存之间均有直接访问通路，cache不命中时可直接访问主存；而虚存所依赖的辅存与CPU之间不存在直接的数据通路，当主存不命中时只能通过调页解决，CPU最终还是要访问主存。

(5)透明性不同 cache的管理完全由硬件完成，对系统程序员和应用程序员均透明；而虚存管理由软件（操作系统）和硬件共同完成，由于软件的介入，虚存对实现存储管理的系统程序员不透明，而只对应用程序员透明（段式和段页式管理对应用程序员“半透明”）。

(6)未命中时的损失不同 由于主存的存取时间是cache的存取时间的5～10倍，而主存的存取速度通常比辅存的存取速度快上千倍，故主存未命中时系统的性能损失要远大于cache未命中时的损失。

3.7.1 虚拟存储器的基本概念

4、虚存机制要解决的关键问题

(1)调度问题决定哪些程序和数据应被调入主存。

(2)地址映射问题在访问主存时把虚地址变为主存物理地址（这一过程称为内地址变换）；在访问辅存时把虚地址变成辅存的物理地址（这一过程称为外地址变换），以便换页。此外还要解决主存分配、存储保护与程序再定位等问题。

(3)替换问题决定哪些程序和数据应被调出主存。

(4)更新问题确保主存与辅存的一致性。

- 在操作系统的控制下，硬件和系统软件为用户解决了上述问题，从而使应用程序的编程大大简化。

3.7.2 页式虚拟存储器

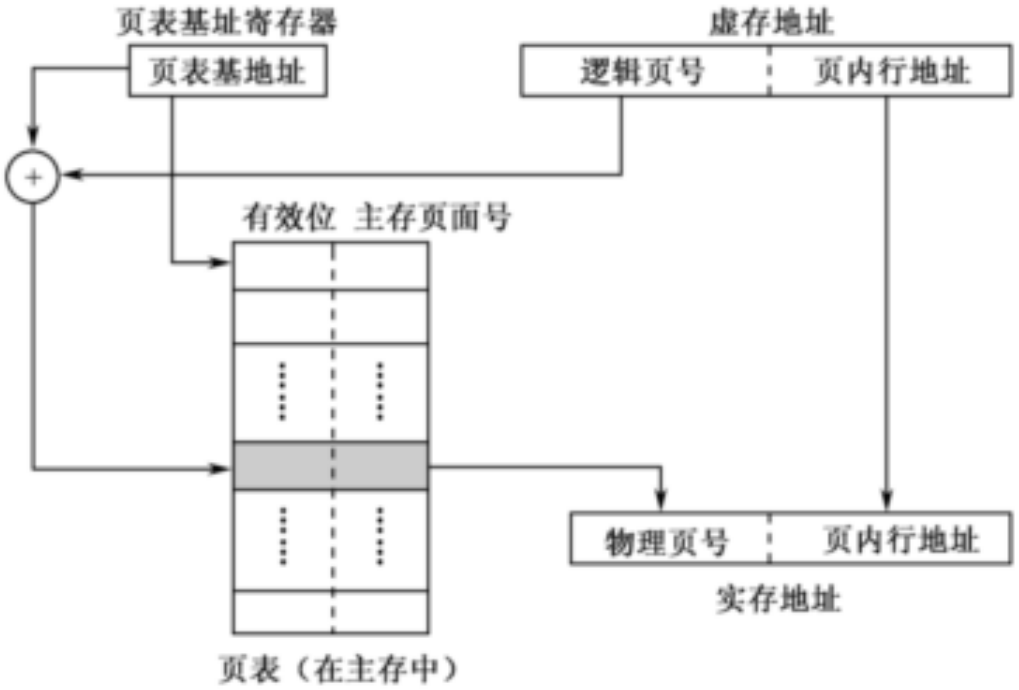
1、页式虚存地址映射

页式虚拟存储系统中，虚地址空间被分成等长大小的页，称为逻辑页；主存空间也被分成同样大小的页，称为物理页。相应地，虚地址分为两个字段：高字段为逻辑页号，低字段为页内地址（偏移量）；实存地址也分两个字段：高字段为物理页号，低字段为页内地址。通过页表可以把虚地址（逻辑地址）转换成物理地址。

页式虚拟存储器的地址映射过程见下图。

3.7.2 页式虚拟存储器

1、页式虚存地址映射

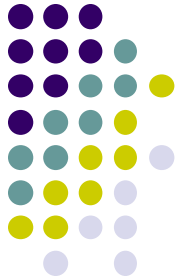


3.7.2 页式虚拟存储器

1、页式虚存地址映射

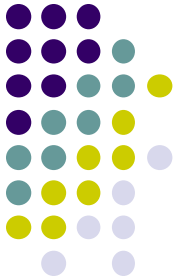
- 在大多数系统中，每个进程对应一个页表。页表中对应每一个虚存页面有一个表项，表项的内容包含该虚存页面所在的主存页面的地址（物理页号），以及指示该逻辑页是否已调入主存的有效位。地址变换时，用逻辑页号作为页表内的偏移地址索引页表（将虚页号看作页表数组下标）并找到相应物理页号，用物理页号作为实存地址的高字段，再与虚地址的页内偏移量拼接，就构成完整的物理地址。现代的中央处理机通常有专门的硬件支持地址变换。
- 每个进程所需的页数并不固定，所以页表的长度是可变的，因此通常的实现方法是把页表的基地址保存在寄存器中，而页表本身则放在主存中。由于虚存地址空间可以很大，因而每个进程的页表有可能非常长。例如，如果一个进程的虚地址空间为2G字节，每页的大小为512字节，则总的虚页数为$2^{31}/2^{9}=2^{22}$。

3.7.2 页式虚拟存储器



- 为了节省页表本身占用的主存空间，一些系统把页表存储在虚存中，因而页表本身也要进行分页。当一个进程运行时，其页表中一部分在主存中，另一部分则在辅存中保存。
- 另一些系统采用二级页表结构。每个进程有一个页目录表，其中的每个表项指向一个页表。因此，若页目录表的长度（表项数）是m,每个页表的最大长度（表项数）为n,则一个进程最多可以有m×n个页。
- 在页表长度较大的系统中，还可以采用反向页表实现物理页号到逻辑页号的反向映射。页表中对应每一个物理页号有一个表项，表项的内容包含该物理页所对应的逻辑页号。访存时，通过逻辑页号在反向页表中逐一查找。如果找到匹配的页，则用表项中的物理页号取代逻辑页号；如果没有匹配表项，则说明该页不在主存中。这种方式的优点是页表所占空间大大缩小，但代价是需要对反向页表进行检索，查表的时间很长。有些系统通过散列（哈希）表加以改进。

3.7.2 页式虚拟存储器

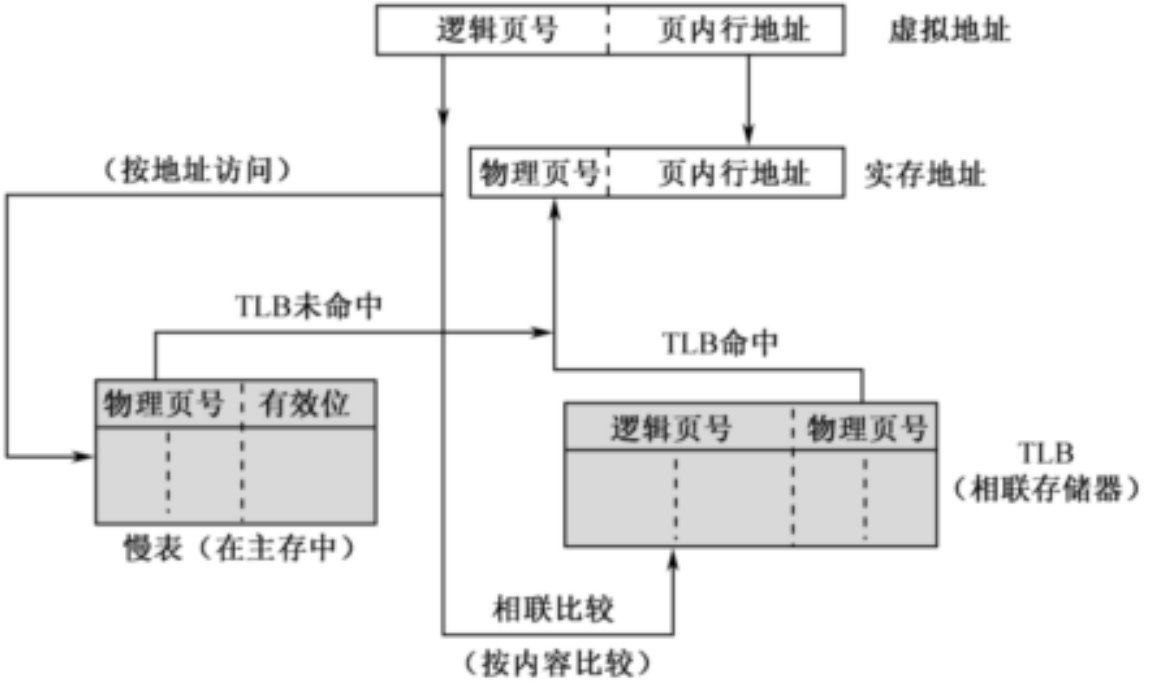


2、转换后援缓冲器

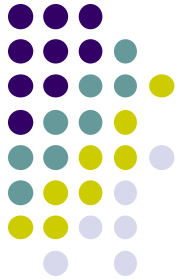
- 由于页表通常在主存中，因而即使逻辑页已经在主存中，也至少要访问两次物理存储器才能实现一次访存，这将使虚拟存储器的存取时间加倍。为了避免对主存访问次数的增多，可以对页表本身实行二级缓存，把页表中的最活跃的部分存放在高速存储器中，组成快表。这个专用于页表缓存的高速存储部件通常称为转换后援缓冲器(TLB)。保存在主存中的完整页表则称为慢表。

3.7.2 页式虚拟存储器

TLB的地址映射过程见图



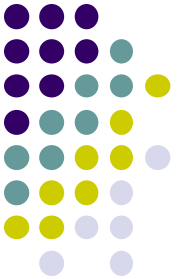
3.7.2 页式虚拟存储器



- 内页表和外页表

  页表是虚地址到主存物理地址的变换表，通常称为内页表。与内页表对应的还有外页表，用于虚地址与辅存地址之间的变换。当主存缺页时，调页操作首先要定位辅存，而外页表的结构与辅存的寻址机制密切相关。例如对磁盘而言，辅存地址包括磁盘机号、磁头号、磁道号和扇区号等。

3.7.3 段式虚拟存储器和段页式虚拟存储器



1、段式虚拟存储器：段是按照程序的自然分界划分的长度可以动态改变的区域。通常，程序员把子程序、操作数和常数等不同类型的数据划分到不同的段中，并且每个程序可以有多个相同类型的段。在段式虚拟存储系统中，虚地址由段号和段内地址（偏移量）组成。虚地址到实主存地址的变换通过段表实现。每个程序设置一个段表，段表的每一个表项对应一个段。每个表项至少包含下面三个字段：

(1)有效位：指明该段是否已经调入实存。

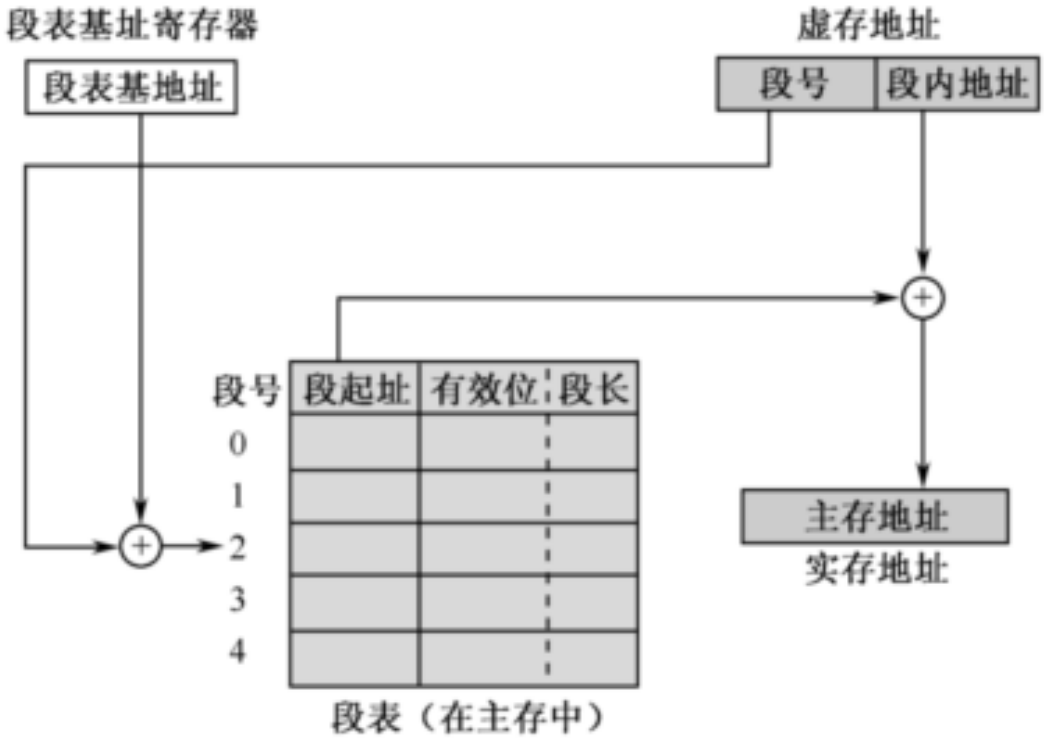
(2)段起址：指明在该段已经调入实存的情况下，该段在实存中的首地址。

(3)段长：记录该段的实际长度。设置段长字段的目的是为了保证访问某段的地址空间时，段内地址不会超出该段长度导致地址越界而破坏其他段。

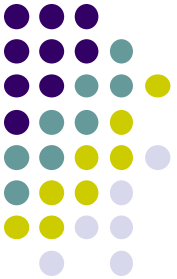
段表本身也是一个段，可以存在辅存中，但一般是驻留在主存中。

3.7.3 段式虚拟存储器和段页式虚拟存储器

段式虚地址向实存地址的变换过程



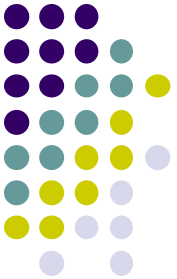
3.7.3 段式虚拟存储器和段页式虚拟存储器



段式虚拟存储器的特点

- 段式虚拟存储器有许多优点：①段的逻辑独立性使其易于编译、管理、修改和保护，也便于多道程序共享。②段长可以根据需要动态改变，允许自由调度，以便有效利用主存空间。
- 段式虚拟存储器也有一些缺点：①因为段的长度不固定，主存空间分配比较麻烦。②容易在段间留下许多外碎片，造成存储空间利用率降低。③由于段长不一定是2的整数次幂，因而不能简单地像分页方式那样用虚地址和实地址的最低若干二进制位作为段内偏移量，并与段号进行直接拼接，必须用加法操作通过段起址与段内偏移量的求和运算求得物理地址。因此，段式存储管理比页式存储管理方式需要更多的硬件支持。

3.7.3 段式虚拟存储器和段页式虚拟存储器



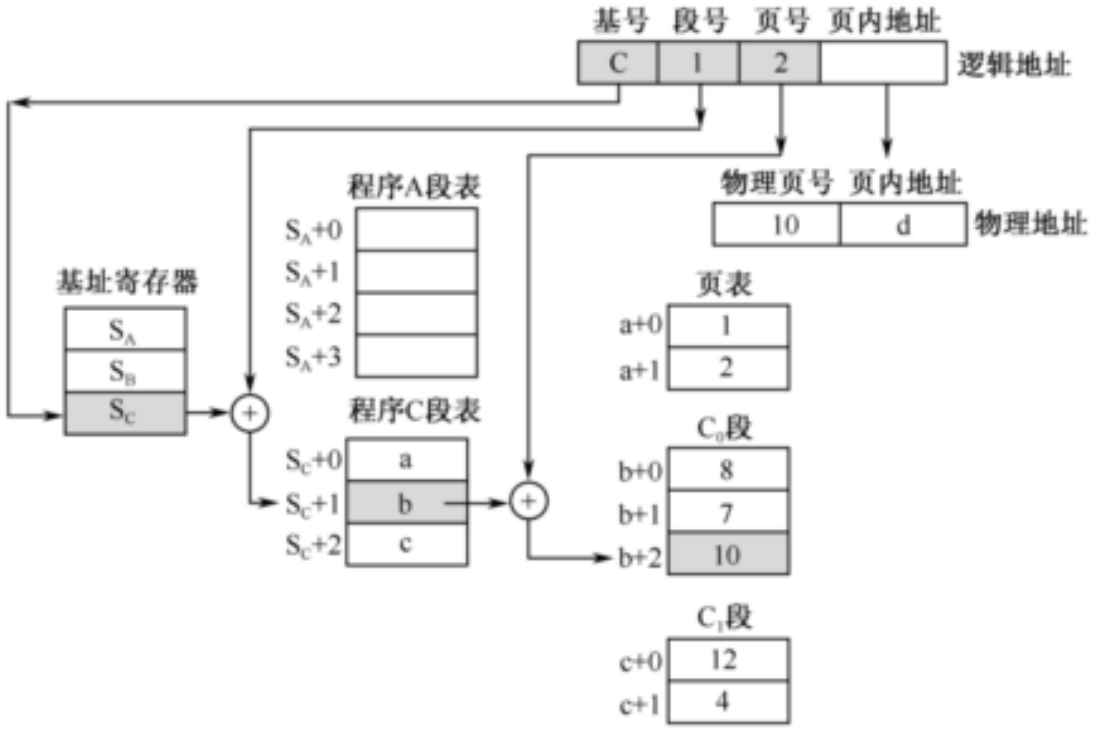
2、段页式虚拟存储器

- 段页式虚拟存储器是段式虚拟存储器和页式虚拟存储器的结合。
- 实存被等分成页。每个程序则先按逻辑结构分段，每段再按照实存的页大小分页，程序按页进行调入和调出操作，但可按段进行编程、保护和共享。

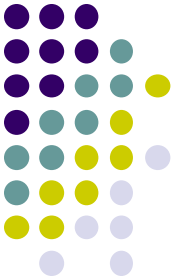
3.7.3 段式虚拟存储器和段页式虚拟存储器

[例1] 假设有三道程序，基号用A、B和C表示，其基址寄存器的内容分别为S_A、S_B和S_C。程序A由4个段构成，程序C由3个段构成。段页式虚拟存储系统的逻辑地址到物理地址的变换过程如图所示。在主存中，每道程序都有一张段表，A程序有4段，C程序有3段，每段应有一张页表，段表的每行就表示相应页表的起始位置，而页表内的每行即为相应的物理页号。请说明虚实地址变换过程。

3.7.3 段式虚拟存储器和段页式虚拟存储器



3.7.3 段式虚拟存储器和段页式虚拟存储器



[例1] 解：地址变换过程如下：

（1）由存储管理部件根据基号C找到段表基址寄存器表第c个表项，获得程序C的段表基址SC。再根据段号S(=1)找到程序C段表的第S个表项，得到段S的页表起始地址b。

（2）根据段内逻辑页号P(=2)检索页表，得到物理页号（图中为10）。

（3）物理页号与页内地址偏移量拼接即得物理地址。

假如计算机系统中只有一个基址寄存器，则基号可不要。多道程序切换时，由操作系统修改基址寄存器内容。

实际上，上述每个段表和页表的表项中都应设置一个有效位。只有在有效位为1时才按照上述流程操作，否则需中断当前操作先进行建表或调页。

可以看出，段页式虚拟存储器的缺点是在由虚地址向主存地址的映射过程中需要多次查表，因而实现复杂度较高.

3.7.4 虚存的替换算法

当从辅存调页至主存而主存已满时，也需要进行主存页面的替换。虚拟存储器的替换算法与cache的替换算法类似，有FIFO算法、LRU算法、LFU算法等。

3.7.4 虚存的替换算法

虚拟存储器的替换算法与cache的替换算法不同的是：

（1）cache的替换全部靠硬件实现，而虚拟存储器的替换有操作系统的支持。

（2）虚存缺页对系统性能的影响比cache未命中要大得多，因为调页需要访问辅存，并且要进行任务切换。

（3）虚存页面替换的选择余地很大，属于一个进程的页面都可替换。

3.7.4 虚存的替换算法

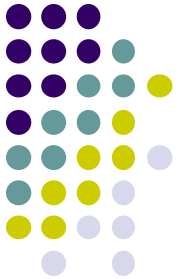
[例2] 假设主存只允许存放a、b、c三个页面，逻辑上构成a进c出的FIFO队列。某次操作中进程访存的序列是0,1,2,4,2,3,0,2,1,3,2（虚页号）。若分别采用FIFO算法、FIFO+LRU算法，请用列表法分别求两种替换策略情况下主存的命中率。

解：求得

FIFO算法命中率：2/11=18.2%

FIFO+LRU算法命中率：3/11=27.3%

（见书上289页表）

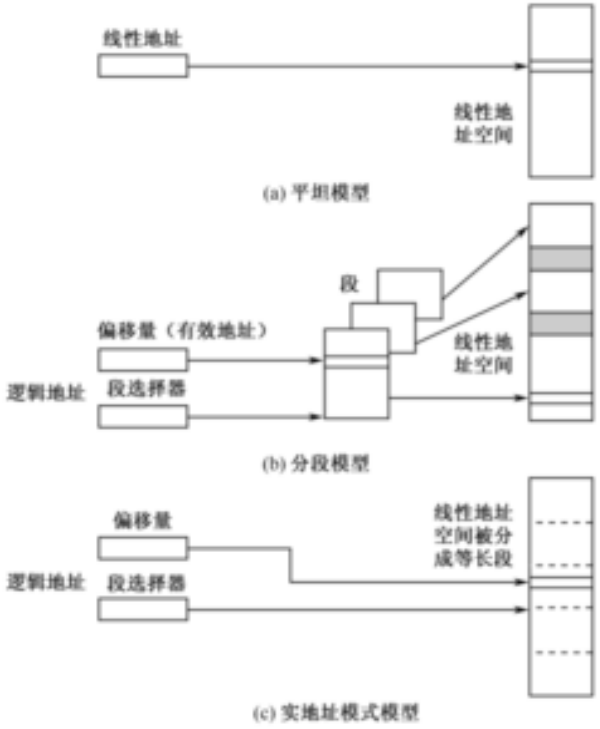


3.8 奔腾系列机的虚存组织

3.8.1 存储器模型
3.8.2 虚地址模式
3.8.3 分页模式下的地址转换

3.8.1 存储器模型

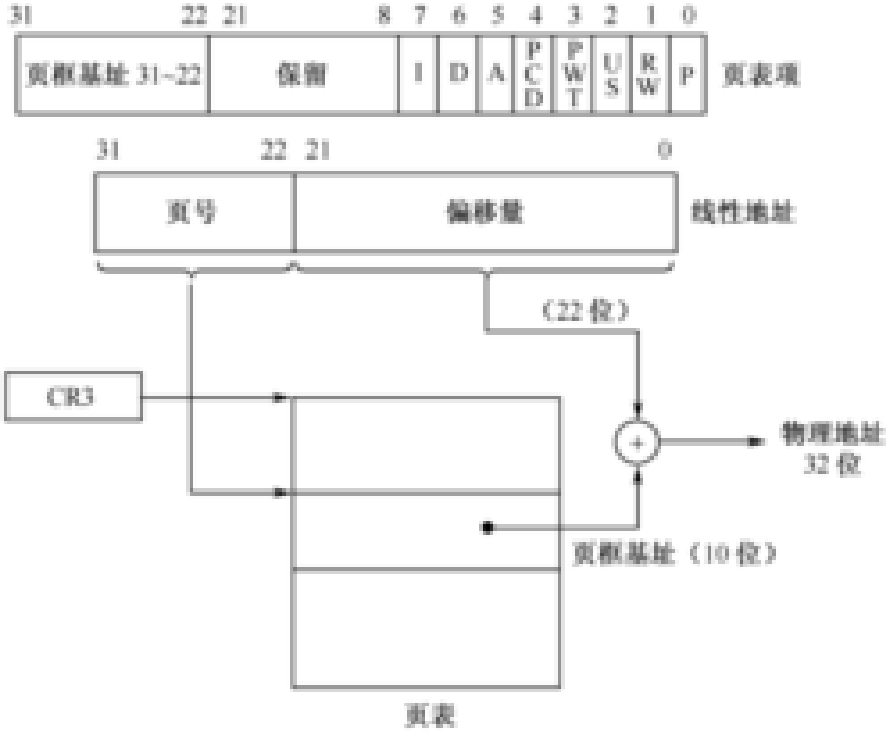
平坦存储器模型
分段存储器模型
实地址模式存储器模型



3.8.2 虚地址模式

IA32体系结构微处理机的虚拟存储器可以通过两种方式实现：分段和分页。存储管理部件包括分段部件SU和分页部件PU两部分。分段部件将程序中使用的虚地址转换成线性地址。而分页部件则将线性地址转换为物理地址。

3.8.3 分页模式下的地址转换



本章小结

- 对存储器的要求是容量大、速度快、成本低。为了解决了这三方面的矛盾，计算机采用多级存储体系结构，即cache、主存和外存。CPU能直接方问内存(cache、主存），但不能直接访问外存。存储器的技术指标有存储容量、存取时间、存储周期、存储器带宽。
- 广泛使用的SRAM和DRAM都是半导体随机读写存储器，前者速度比后者快，但集成度不如后者高。二者的优点是体积小，可靠性高，价格低廉，缺点是断电后不能保存信息。

本章小结

- 只读存储器和闪速存储器正好弥补了SRAM和DRAM的缺点，即使断电也仍然保存原先写入的数据。特别是闪速存储器能提供高性能、低功耗、高可靠性以及移动性，是一种全新的存储器体系结构。
- 双端口存储器和多模块交叉存储器属于并行存储器结构。前者采用空间并行技术，后者采用时间并行技术。这两种类型的存储器在科研和工程中大量使用。

本章小结

- cache是一种高速缓冲存储器，是为了解决CPU和主存之间速度不匹配而采用的一项重要的硬件技术，并且发展为多级cache体系，指令cache与数据cache分设体系。要求cache的命中率接近于1。主存与cache的地址映射有全相联、直接、组相联三种方式。其中组相联方式是前二者的折衷方案，适度地兼顾了二者的优点又尽量避免其缺点，从灵活性、命中率、硬件投资来说较为理想，因而得到了普遍采用。

本章小结

- 操作系统是计算机硬件资源的管理器，其管理功能主要包括处理机管理、存储管理和设备管理等。
- 操作系统的管理功能只有在专门的硬件支持下才能充分保证系统工作的高效与安全。硬件系统在设计层面上为操作系统提供了支持，包括处理机状态控制、特权指令、寄存器访问权限控制、程序状态字和程序执行现场保护与切换、中断机制、存储管理硬件等。操作系统设计者则应根据硬件特性和用户的使用需要，采用各种有效的管理策略。
- 处理机调度是操作系统的核心功能之一。按照调度的层次，处理机调度可以分成作业调度、交换调度和进程调度。其中，进程调度的运行频率最高。作业调度的周期较长，往往发生在一个作业运行完毕并将退出系统，需要重新调入一个作业进入内存时。交换调度的运行频率介于作业调度和进程调度之间。

本章小结

- 存储管理是操作系统的另外一个核心功能。存储管理主要解决存储器的分配与回收、存储器地址变换、存储器扩充、存储器共享与保护等问题。
- 为了支持多个程序并发执行，现代操作系统逐渐引入了分区式存储管理，以及交换技术和分页技术。在存储管理部件MMU的支持下，虚拟存储器技术可以彻底解决存储器的调度与管理问题。
- 用户程序按照虚地址（逻辑地址）编程并存放在辅存中。程序运行时，由地址变换机构依据当时分配给该程序的实地址空间把程序的一部分调入实存（物理存储空间或主存空间）。由操作系统在硬件的支持下对程序进行虚地址到实地址的变换，这一过程称为程序的再定位。每次访存时，首先判断该虚地址所对应的部分是否在实存中：如果是，则进行地址转换并用实地址访问主存；否则，按照某种算法将辅存中的部分程序调度进内存，再按同样的方法访问主存。对应用程序而言，如果主存的命中率很高，虚存的访问时间就接近于主存访问时间，而虚存的大小仅仅依赖于辅存的大小。

本章小结

- 虚存机制也要解决一些关键问题，包括调度问题、地址映射问题和替换问题等。在操作系统的控制下，硬件和系统软件为用户解决了上述问题，从而使应用程序的编程大大简化。
- 页式虚拟存储系统中，虚地址空间和主存空间都被分成大小相等的页，通过页表可以把虚地址转换成物理地址。为了避免对主存访问次数的增多，可以对页表本身实行二级缓存，把页表中的最活跃部分存放在转换后援缓冲器（TLB）中。

本章小结

- 分页方式的缺点是页长与程序的逻辑大小不相关，而分段方式则可按照程序的自然分界将内存空间划分为长度可以动态改变的存储区域。在段式虚拟存储系统中，虚地址由段号和段内地址（偏移量）组成。虚地址到实主存地址的变换通过段表实现。
- 段页式虚拟存储器是段式虚拟存储器和页式虚拟存储器的结合，程序按页进行调入和调出操作，但可按段进行编程、保护和共享。
- 虚拟存储器还解决存储保护等问题。在虚拟存储系统中，通常采用页表保护、段表保护和键式保护方法实现存储区域保护。还可以结合对主存信息的使用方式实现访问方式保护。