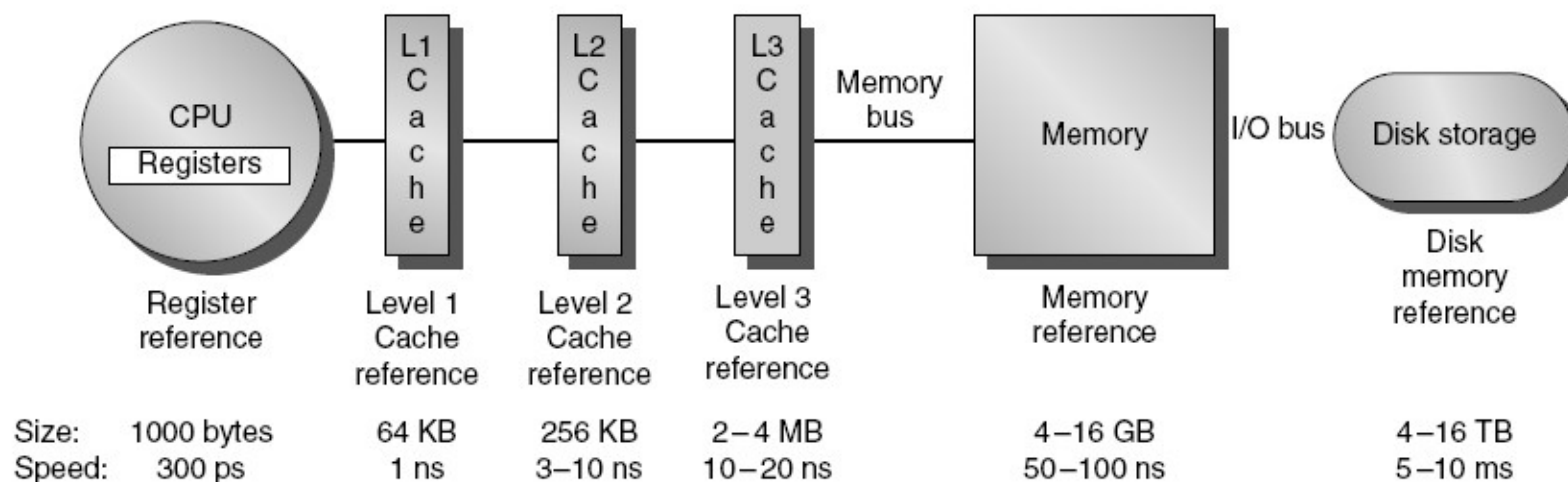


第五章 主存储器与存储系统

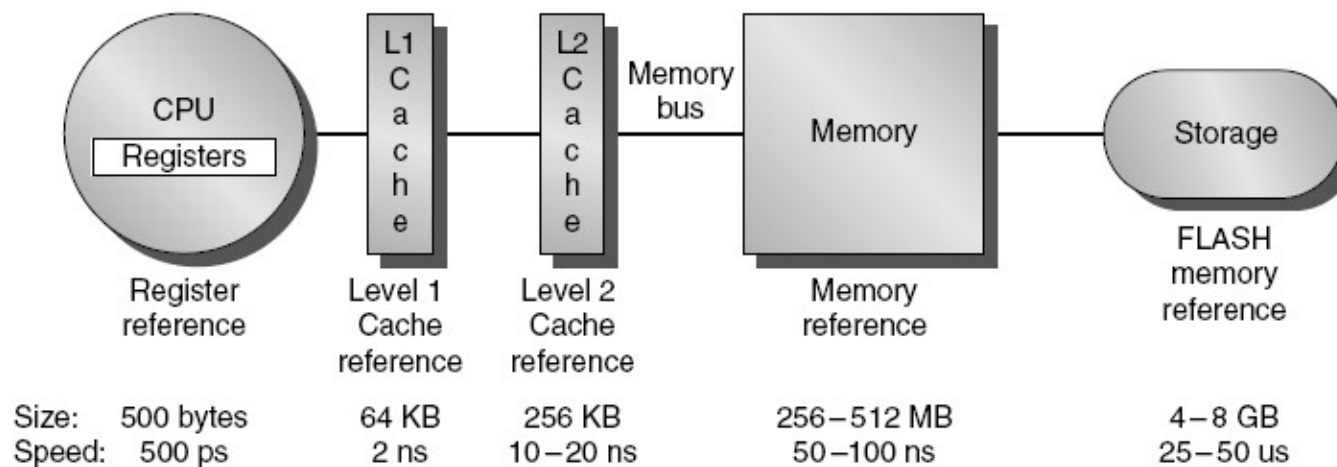
课前思考

- 1) 日常使用的PC/平板电脑/手机等内存容量多大, 单位是什么?
- 2) 内存与CPU之间如何传输数据?
- 3) 内存如何构成?
- 4) 存储系统具体由哪些部分组成?

存储系统示例



(a) Memory hierarchy for server



(b) Memory hierarchy for a personal mobile device

CPU与存储器

CPU和存储器的特性对比

		1980	1990	2000	2010	2010:1980
CPU	Name	8080	386	Pentium II	Core i7	/
	Clock rate(MHz)	1	20	600	2,500	2,500
	Cycle time(ns)	1,000	50	1.6	0.4	2,500
	Cores	1	1	1	4	4
	Effective Cycle time(ns)	1,000	50	1.6	0.1	10,000
SRAM	\$/MB	19,200	320	100	60	320
	access time(ns)	300	35	3	1.5	200
DRAM	\$/MB	8,000	100	1	0.06	130,000
	access time(ns)	375	100	60	40	9
	typical size(MB)	0.064	4	64	8,000	125,000
Disk	\$/MB	500	8	0.01	0.0003	1,600,000
	access time(ms)	87	28	8	3	29
	typical size(MB)	1	160	20,000	1,500,000	1,500,000

本章主要内容

本章讲述存储器的分类、主存储器的构成和并行主存储器的思想及设计方法；存储系统的概念及存储系统的构成。

重点掌握主存储器的构成，Cache的组成原理与地址映射方式、替换算法；掌握段式、页式和段页式虚拟存储器的构成原理、地址变换方式等。

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

存储器分类

从不同的角度出发，对存储系统可以做不同的分类。

- 1) 按存储器**所采用的元件**（介质）分，有磁芯存储器、半导体存储器、磁表面存储器（包括磁带、磁鼓、硬磁盘等）和激光存储器。
- 2) 按**存储器和CPU的关系**来分，有高速缓冲器(Cache)、内存储器和外存储器。直接与CPU相联系，作为计算机的组成部分，用于存放正在使用的部分程序 and 数据的存储器是主存储器（也称为内存储器）。外存储器是不直接和CPU相联系的存储器，属于计算机系统的外设部分。

存储器分类

3) 主存储器可分为

- 随机存储器（Random Access Memory, RAM）。
- 只读存储器（Read Only Memory, ROM）。
- 可编程只读存储器（Programmable ROM, PROM）
- 可擦除可编程只读存储器（Erasable PROM, EPROM）
- 电可擦除可编程只读存储器（Electrically EPROM, E²PROM）
- 非易失性存储器（FLASH Memory, STT-RAM, ReRAM）

主存储器的主要技术指标

主存储器的主要性能指标为主存容量、存储器存取时间和存储器存储周期。

主存容量（Memory Size）

存储器存取时间（Memory Access Time）

存储周期（Memory Cycle Time）

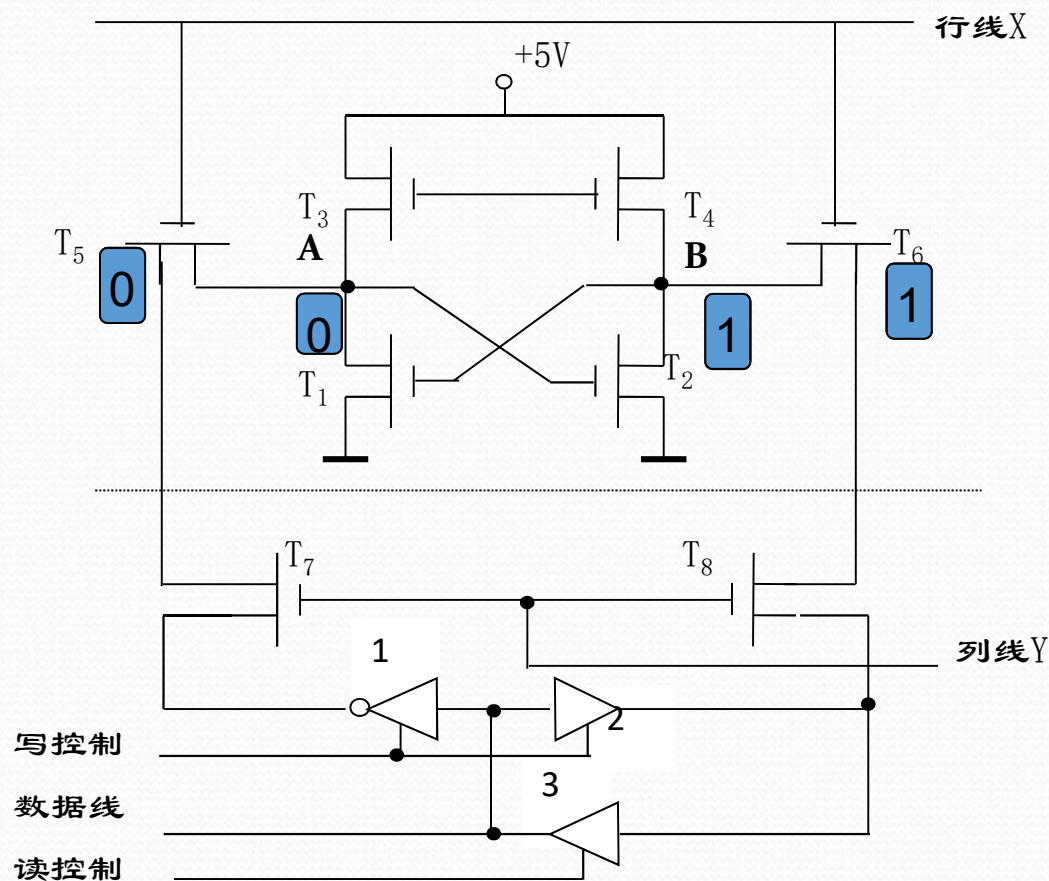
主存储器的速度和容量两项技术指标。

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

静态RAM的基本电路

静态MOS 6管
基本存储电路



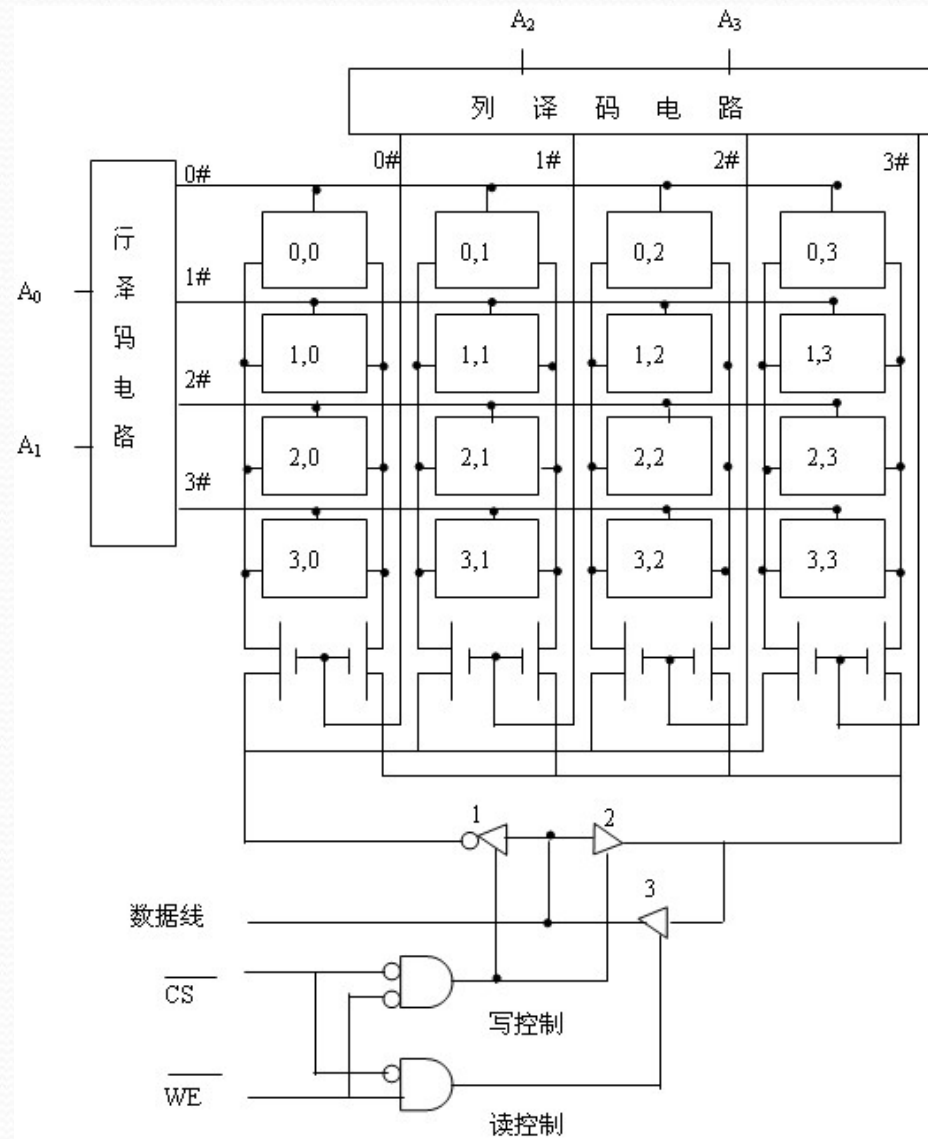
静态MOS六管基本存储电路

静态RAM的结构

将静态RAM的**基本存储电路**排成阵列，加上**地址译码电路**和**读写控制电路**，就可以构成读写存储器的内部结构。下面以4行4列共16个基本存储电路构成 16×1 静态RAM为例来说明RAM的基本原理。图中所示是一个 16×1 的静态RAM(一共16个字，每个字1位)

静态RAM的结构

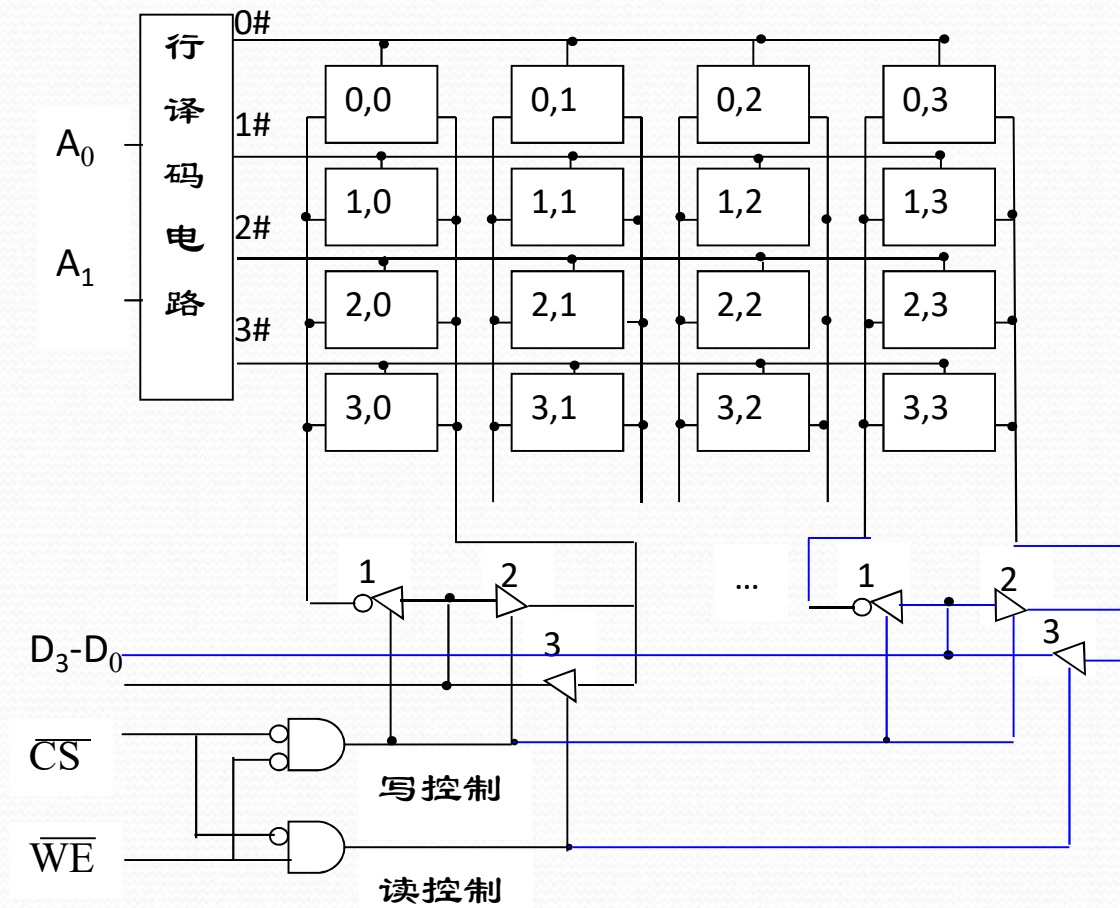
容量: 16×1



静态RAM的结构

容量=?

4×4



静态RAM的结构

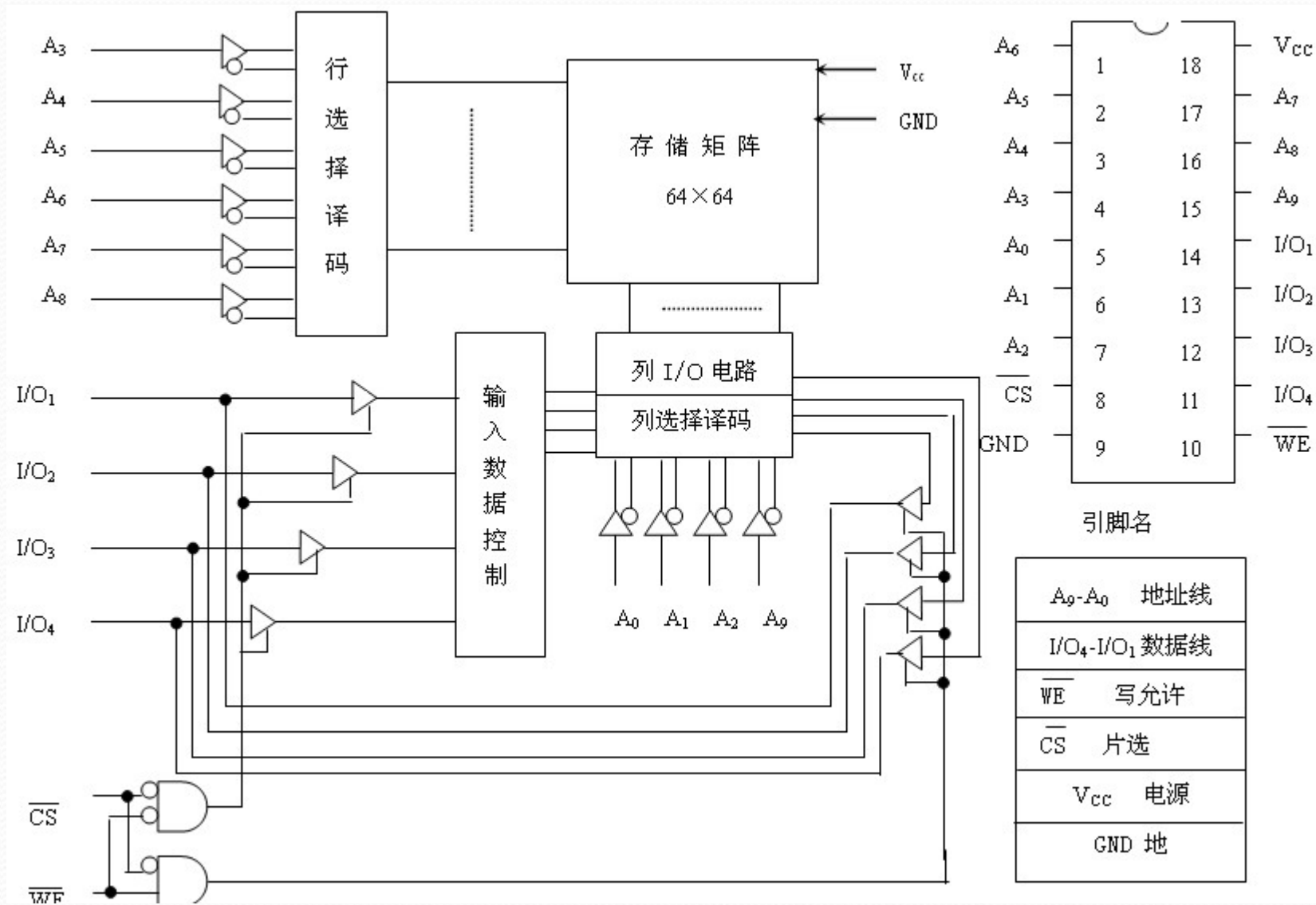
此静态RAM它由以下四部分构成：

- (1) 16个基本存储电路组成的 4×4 存储矩阵。
- (2) 两套(行与列)地址译码电路。
- (3) 四套列开关。
- (4) 一套读写控制电路(片选、读写等)。

静态RAM芯片实例

静态RAM芯片实例——Intel 2114

Intel 2114是 $1K \times 4$ 的静态RAM芯片，单一的+5V电源，所有的输入端、输出端均与TTL电路兼容。采用NMOS工艺。其结构框图、引脚排列和逻辑符号如图所示。



静态RAM的结构

Intel 2114 SRAM芯片的地址输入端10个($A_9 \sim A_0$)。在片内可寻址1K个存储单元。4位共用的数据输入/输出端($I/O_4 \sim I/O_1$)采用三态控制，每个存储单元可存储4位二进制信息。故2114芯片的存储容量为？

芯片中共有4096个6管NMOS静态基本存储电路，它们排成 64×64 的矩阵。10条地址线中的 $A_8 \sim A_3$ 通过地址译码电路产生64条行选择线，对存储矩阵的行进行控制；另外4条地址线 A_0, A_1, A_2, A_9 通过列地址译码电路对存储阵列的列线进行控制（共16条列线，每条列线可同时接至4位，所以实际为64列）

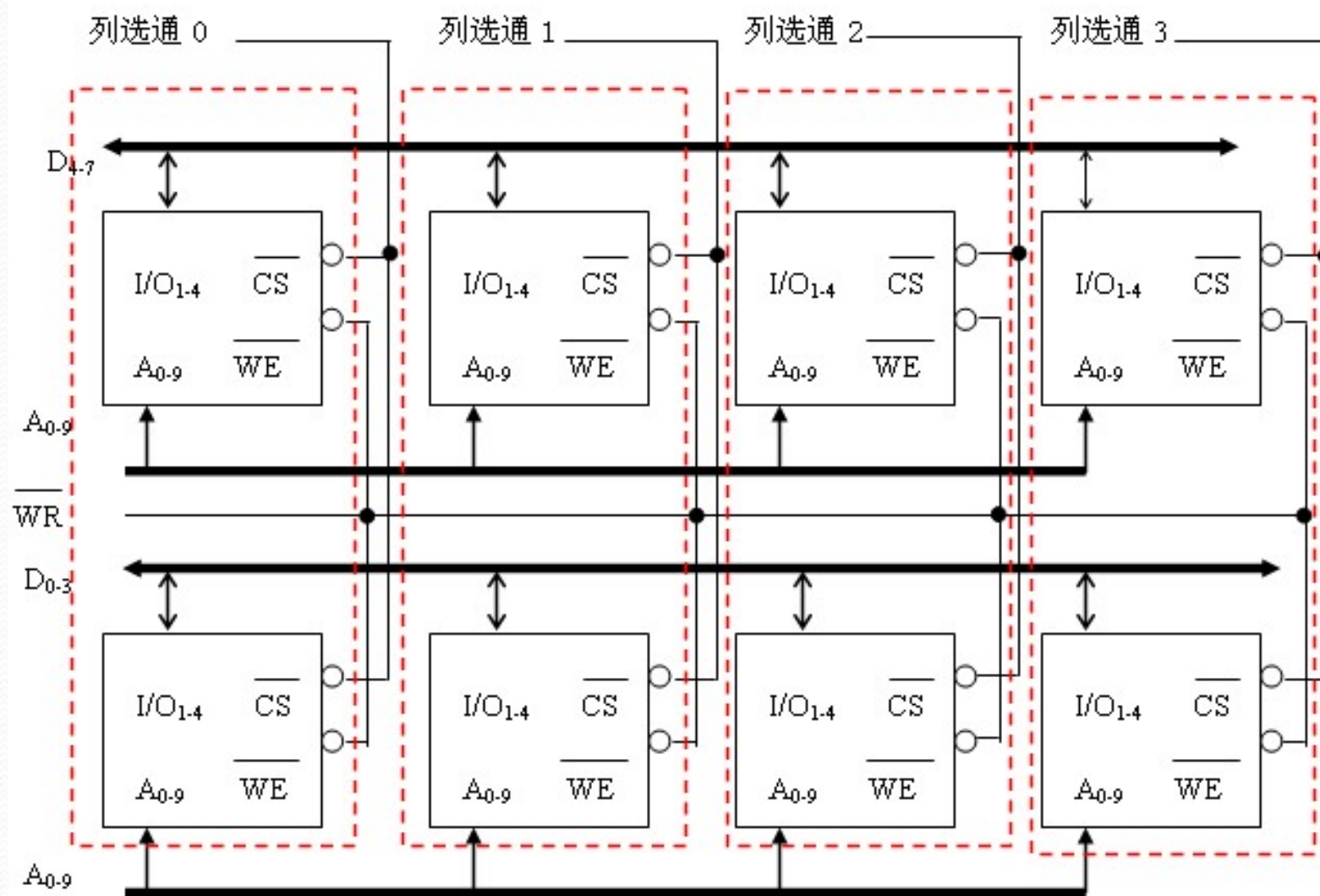
构成规定容量的存储器

例 用若干片2114构成一容量为 $4K \times 8$ 的存储器。

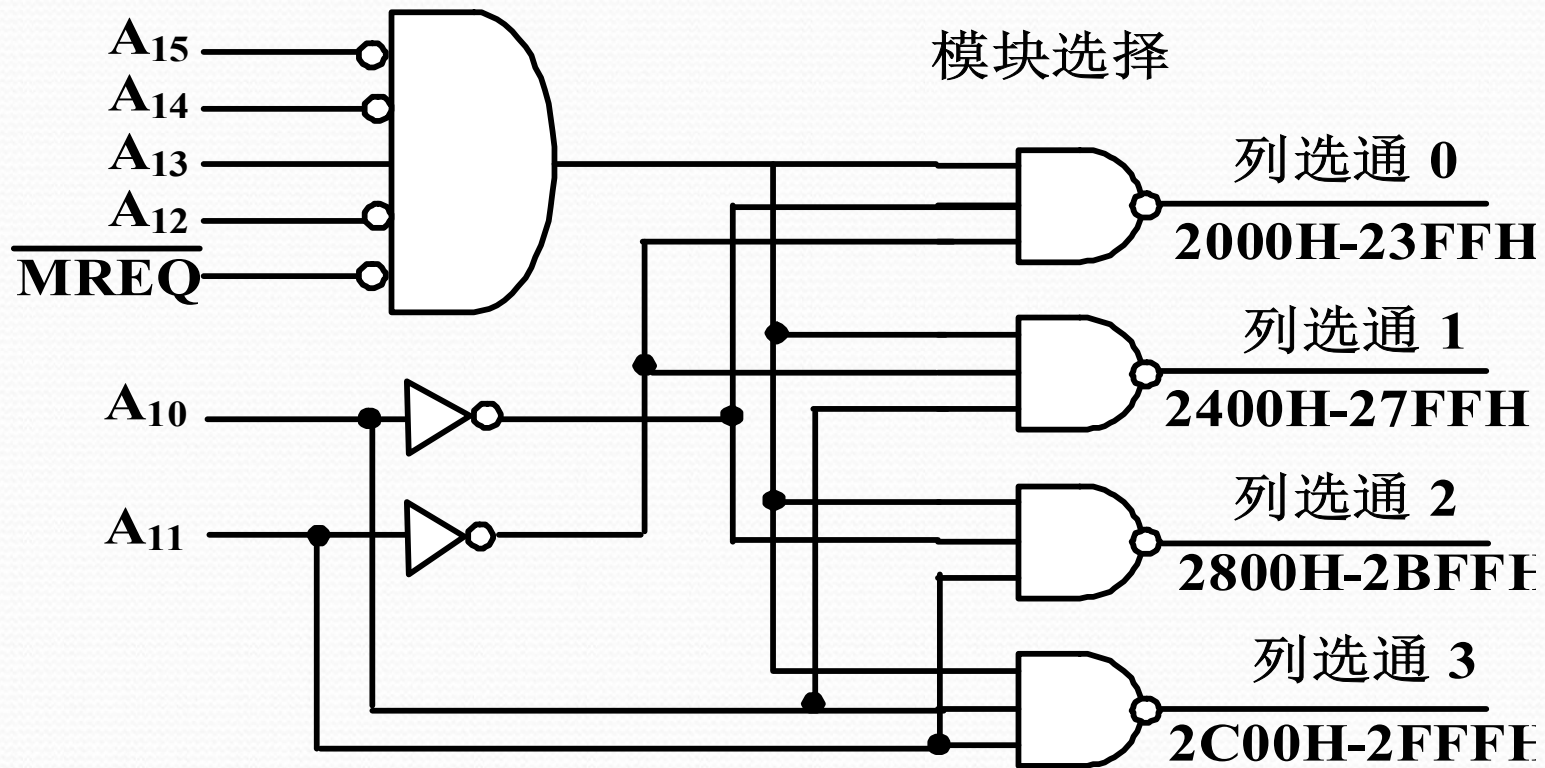
思考：

- (1) 2114容量多大？一片能否满足要求？
- (2) 需要多片2114作扩展，需要多少片？
- (3) 扩展连接时，地址线、数据线、控制线如何处理？

构成规定容量的存储器



构成规定容量的存储器

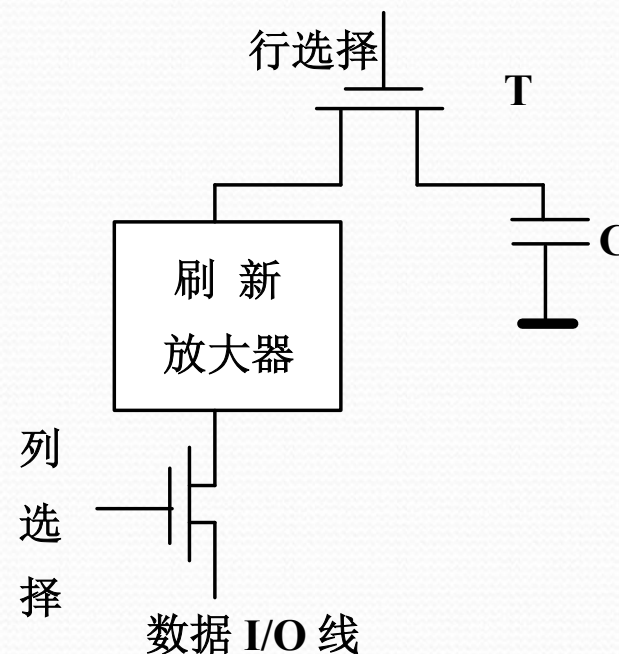


动态RAM

与静态RAM相似，动态RAM存储器器件的基本存储电路也是按行和列组成存储矩阵的，主要区别是基本存储电路不同。与静态RAM中信息存储的方式不同，动态RAM是利用MOS管栅源间的极间电容来存储信息的。

动态RAM的基本存储电路

动态RAM基本存储电路如图所示。当电容充有电荷时，称存储的信息为1，电容上没有电荷时，表示存储的信息为0。由于电容上存储的电荷不能长时间保存，总会泄漏，因此必须定时给电容补充电荷，这个过程称为“刷新”或“再生”。



动态RAM的基本存储电路

上图所示为一个NMOS单管动态基本存储电路。它由一个管子T和一个电容C构成，这个基本存储电路所存储的信息是0还是1由电容上是否充有电荷决定。图中的刷新放大器为同一列所有基本存储电路共用。

在进行读操作时，译码器对行地址（低位地址）译码，使对应行选择线变为高电平。处于该行选择线控制下的该行上所有基本存储电路的T管均导通。这样，各列的刷新放大器便可读取相应电容上的电压电平，形成1或0信号。列地址（高位地址）允许选中的一行中的一个基本存储电路输出。在这个过程中，整个一行上所有的电容都会受到干扰，这样的读出称为破坏性读出。为保持存储的信息不变，由刷新放大器对该行中的各基本存储电路按读取的状态进行重写。在进行写操作时，与此类似，只是输入的数据被存入选中的那个基本存储电路中。

动态RAM的特点

由上述可以看出，由于动态RAM的基本存储电路为电容，而电容上所存储的电荷会漏掉，也就是说，其中存储的信息在一定时间后会丢失。所以，动态RAM需定时“刷新”，这是其缺点所在。动态RAM具有以下主要特点：

- (1) 基本存储电路简单，集成度高。
- (2) 需定时刷新，且刷新是在对其读出信息的过程中完成的。
- (3) 外围电路复杂。

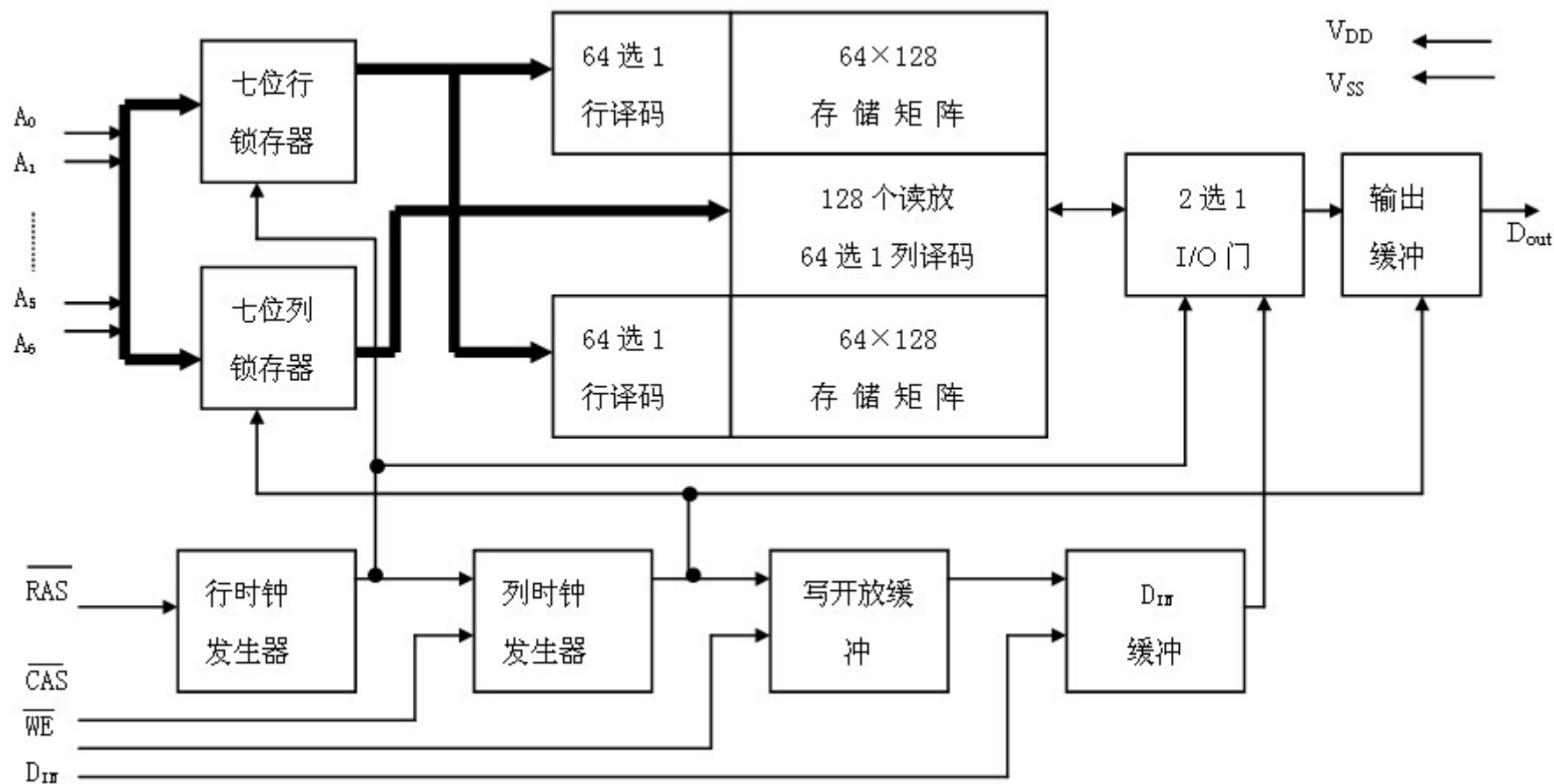
由于动态RAM有上述特点，所以在微电子技术相对不太发达的过去，大多内存主要采用静态RAM。随着技术的进步及工艺的提高，如今的微机内存中已大量使用动态RAM了。

动态RAM实例

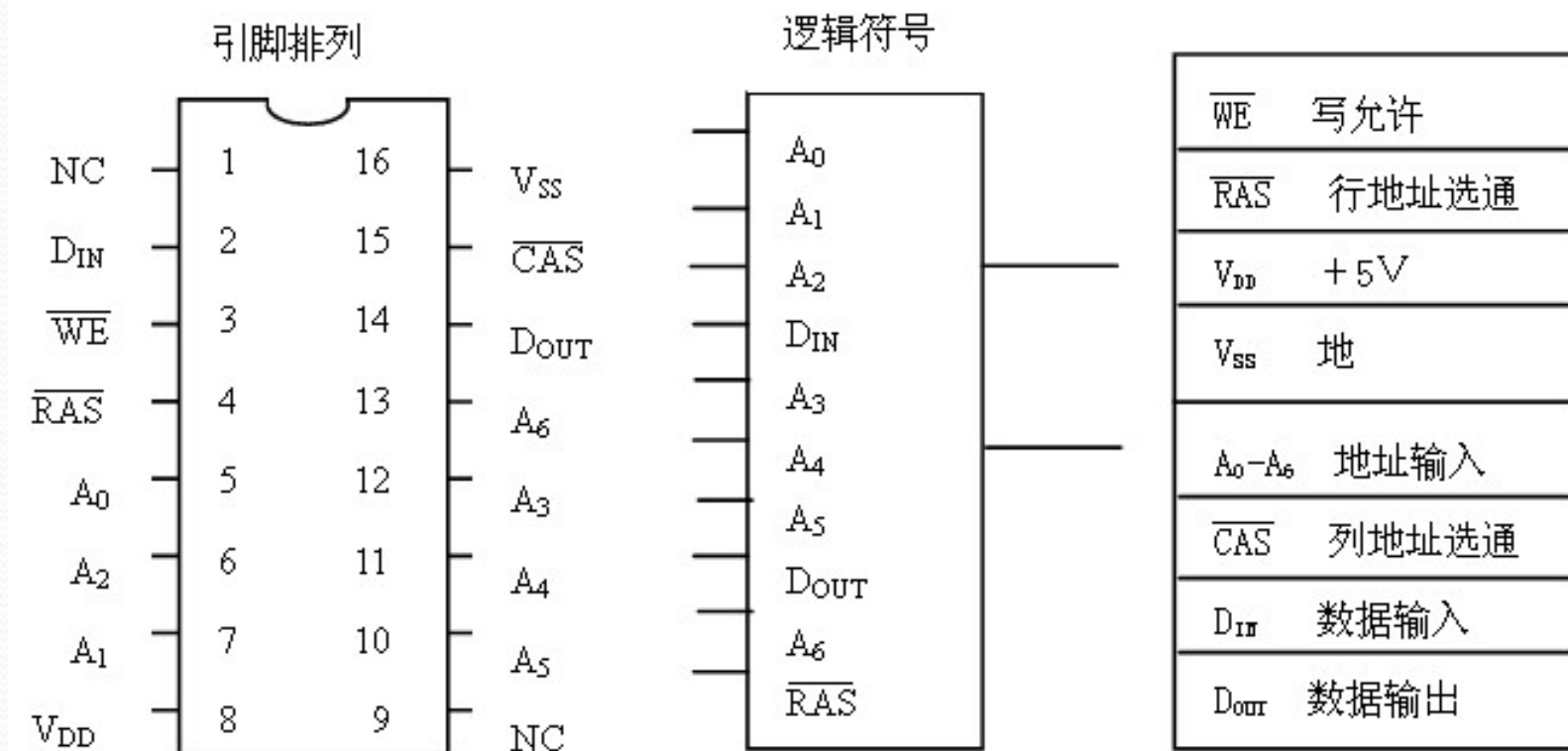
动态RAM典型的例子是Intel 2116，该芯片是 $16\text{K} \times 1$ 动态RAM，采用HMOS工艺。单管动态基本存储电路，单一+5V电源，所有的输入/输出引脚都与TTL电路兼容。2116共有16个引脚，其结构框图、引脚排列及逻辑符号如图所示。地址码的输入和控制方式不同于静态RAM。

2116是 $16\text{K} \times 1$ 的芯片，需要有14位地址码对其控制，所以芯片本应由14个引脚作为地址线，但实际上只有7个引脚用作地址引线。为了实现14位地址控制，采用分时技术将14位地址码分两次从7条地址引线上送入芯片内部，而在片内设置两个7位地址锁存器，分别称为行锁存器和列锁存器。14位地址码分成行地址（低7位地址）和列地址（高7位地址），在两次输入后分别寄存在行锁存器内和列锁存器内。基本存储电路按行和列排列成两个 128×64 的存储矩阵。

动态RAM实例



动态RAM实例



动态RAM实例

Intel 2116地址选择器操作这样进行：用行地址选通信号 \overline{RAS} 把先出现的7位地址送到行地址锁存器，由随后出现的列地址选通信号 \overline{CAS} 把后出现的7位地址送到列地址锁存器。行译码器和列译码器把存于行锁存器和列锁存器的地址分别译码。形成128条行选择线和128条列选择线对 128×128 存储阵列进行选址。

动态RAM实例

读写操作：当全部地址码输入后，128行中必有一行被选中，这一行中的128个基本存储电路的信息都选通到各自的读出放大器，在那里，每个基本存储电路存储的逻辑电平都被鉴别、放大和刷新。列译码的作用是通过选通128个读出放大器中的一个，从而唯一地确定欲读写的基本存储电路。并将被选中的基本存储电路通过读出放大器，I/O控制门输入数据锁存器或输出数据锁存器及缓冲器相连，以便完成对该基本存储电路的读写操作。读出与写入操作是由写允许信号控制的，当 \overline{WE} 为高电平时，进行读操作，数据从引脚 D_{out} 输出；当 \overline{WE} 为低电平时，进行写操作，数据从 D_{in} 引脚输入并锁存于输入锁存器中，再写入选定的基本存储电路。三态数据输出端受信号 \overline{CAS} 控制而与 \overline{RAS} 信号无关。

动态RAM实例

由于2116是由动态基本存储电路构成，因此需要定时对其存储电路进行刷新。对2116 DRAM的刷新方法是对128行逐行进行选择的，同时行选通信号 \overline{RAS} 加低电平，但列选通信号 \overline{CAS} 为高电平。这样，虽然成行对基本存储电路进行读操作，把一行中128个基本存储电路存储的信息被选通到各自的读出放大器，进行放大锁存，但不进行列选择。没有真正的输出，而是把锁存的信息再写回原来的基本存储电路，实现刷新。

用多块2116芯片形成一个存储器模块的基本原理同用2114 SRAM芯片构成的存储器模块相同，只是外加刷新逻辑，这里不再重复了。

动态RAM的刷新

从动态RAM基本存储电路中可以看出，行选择线为低电平时，T管截止，电容C上的电荷因无放电回路而保存下来。然而虽然MOS管入端阻抗很高，但总有一定的泄漏电流，这样会引起电容放电。为此必须定时重复地对动态RAM的基本存储电路存储的信息进行读出和恢复，这个过程叫[存储器刷新](#)。器件工作温度增高会使放电速度变快。刷新时间间隔一般要求在1~100ms之间，工作温度为70°时，典型的刷新时间间隔为2ms。一般C=0.2pF，若允许C两端电压变化差为 $\Delta V=1V$ ，泄漏电流 $I = 10^{-10}A$ 。则：

$$\Delta T = \frac{C \cdot \Delta V}{I} = \frac{0.2 \times 10^{-12} \times 1}{10^{-10}} = 2ms$$

因此，2ms之内必须对存储信息进行刷新。

动态RAM的刷新

在存储器刷新周期中，将一个行地址发送给存储器件，然后执行一次读操作，便可完成对选中的行中各个基本存储电路的刷新。刷新周期和正常的存储器读周期的不同之处主要有以下几点：

(1) 在刷新周期中输入至存储器件的地址一般并不来自地址总线，而是由一个以计数方式工作的寄存器提供。每经过一次（即一行）存储器刷新，该计数器加1，所以它可以顺序提供所有的行地址，每一行中各个基本存储电路的刷新是同时进行的，所以不需要列地址。而在正常的读周期中，地址来自地址总线，既有行地址，又有列地址。

(2) 在存储器的刷新周期中，存储器模块中每块芯片的刷新是同时进行的，这样可以减少刷新周期数。而在正常的读周期中，只能选中一行存储器芯片。

(3) 在存储器刷新周期中，存储器模块中各芯片的数据输出是高阻状态，即片内数据线与外部数据线完全隔离。

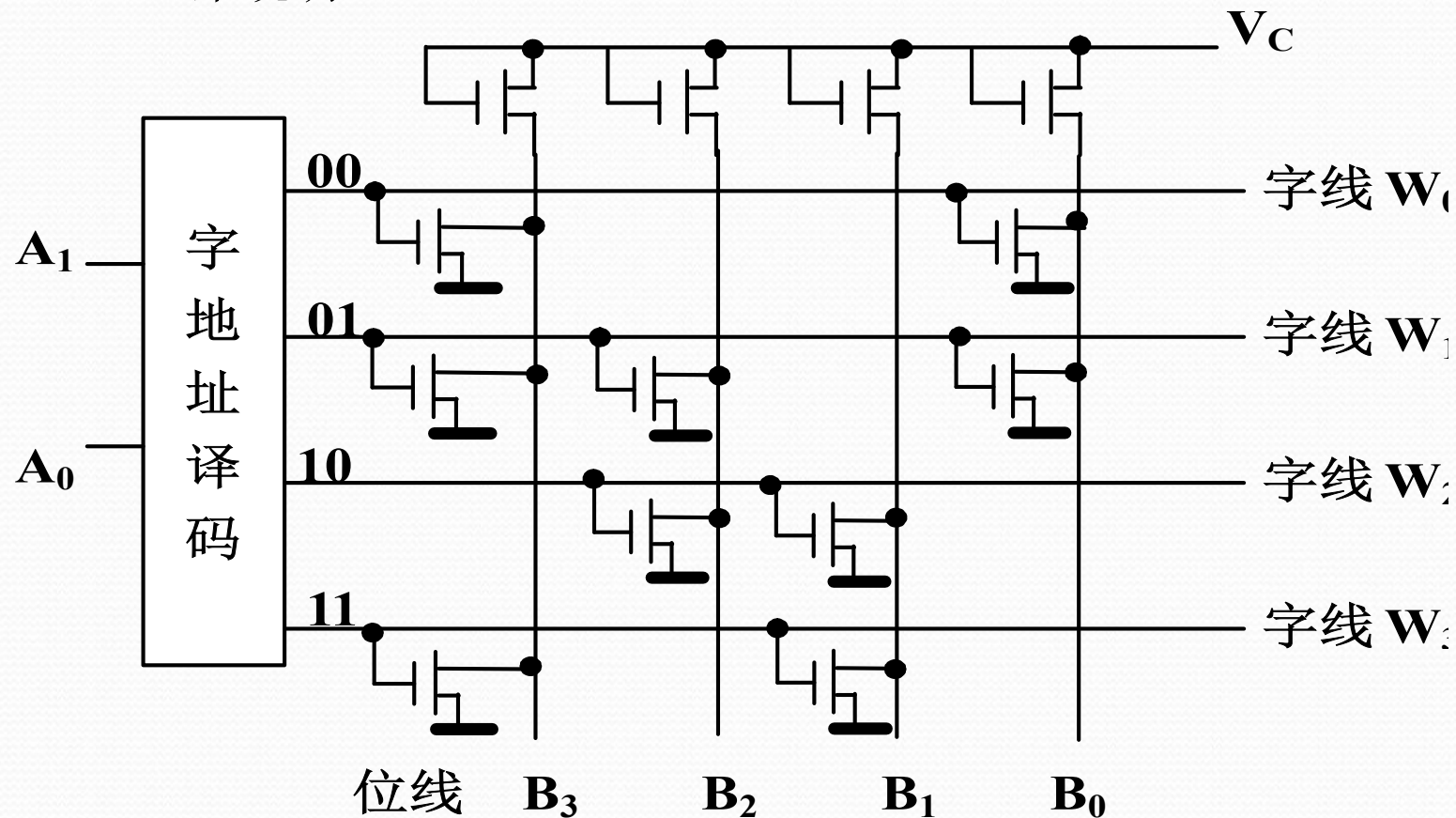
从用于刷新的时间来说，刷新可采用“集中”或“分散”两种方式中的任何一种。集中刷新方式是在信息保存允许的时间范围内(2ms)，集中一段时间对所有基本存储电路一行一行地顺序进行刷新，刷新结束后再开始工作周期。分散刷新工作方式是把各行的刷新分散在2ms的时间内完成。

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

只读存储器(ROM)

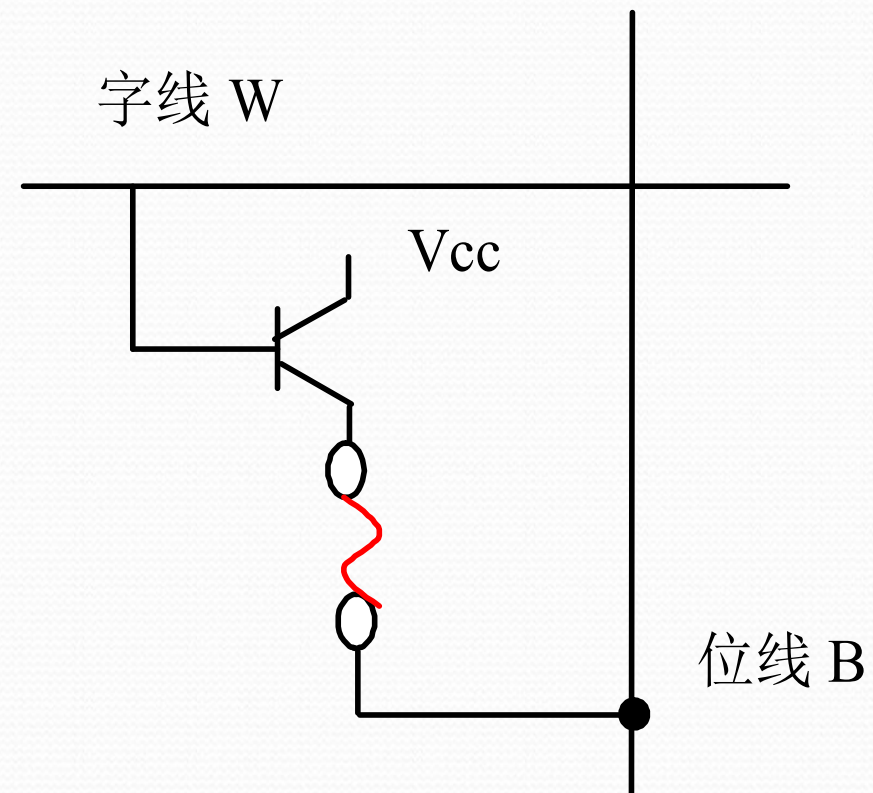
只读存储器ROM在生产过程中的掩模工艺中将程序写入了芯片中，因此又称为掩模ROM，掩模ROM的基本原理可用如图中给出的 4×4 MOS ROM来说明。



可编程只读存储器(PROM)

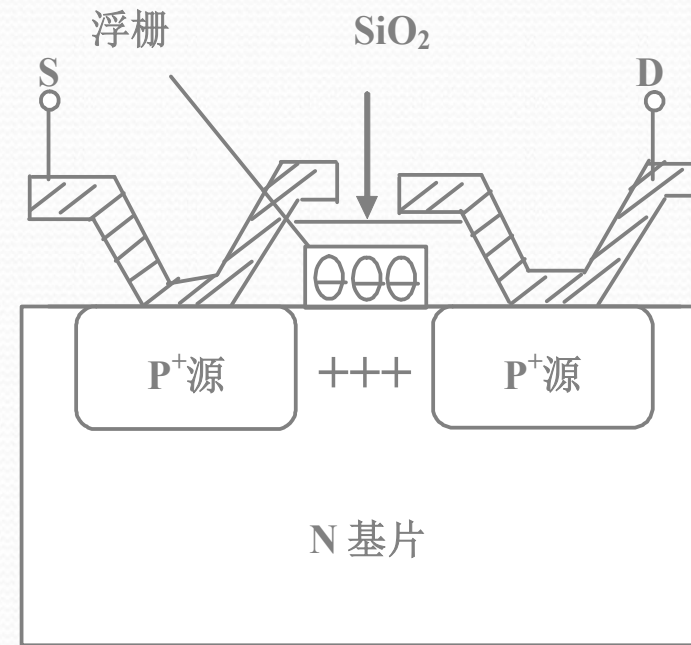
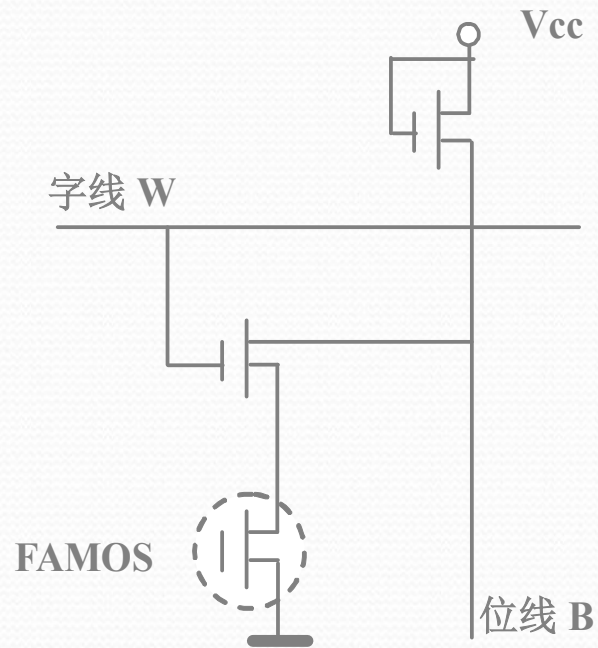
可编程只读存储器

(Programmable ROM)的基本存储电路为一个晶体管，以字位结构进行说明。晶体管的集电极接 V_{cc} ，它的基极连接字线，发射极通过一个熔丝与位线相连，如图所示。制造时每条字线与所有位线之间都跨接一个带熔丝的双极型晶体管，就构成了可编程只读存储器。



可擦除可编程只读存储器

EPR0M的存储电路



P沟道FAMOS管基本存储电路及P沟道FAMOS管结构

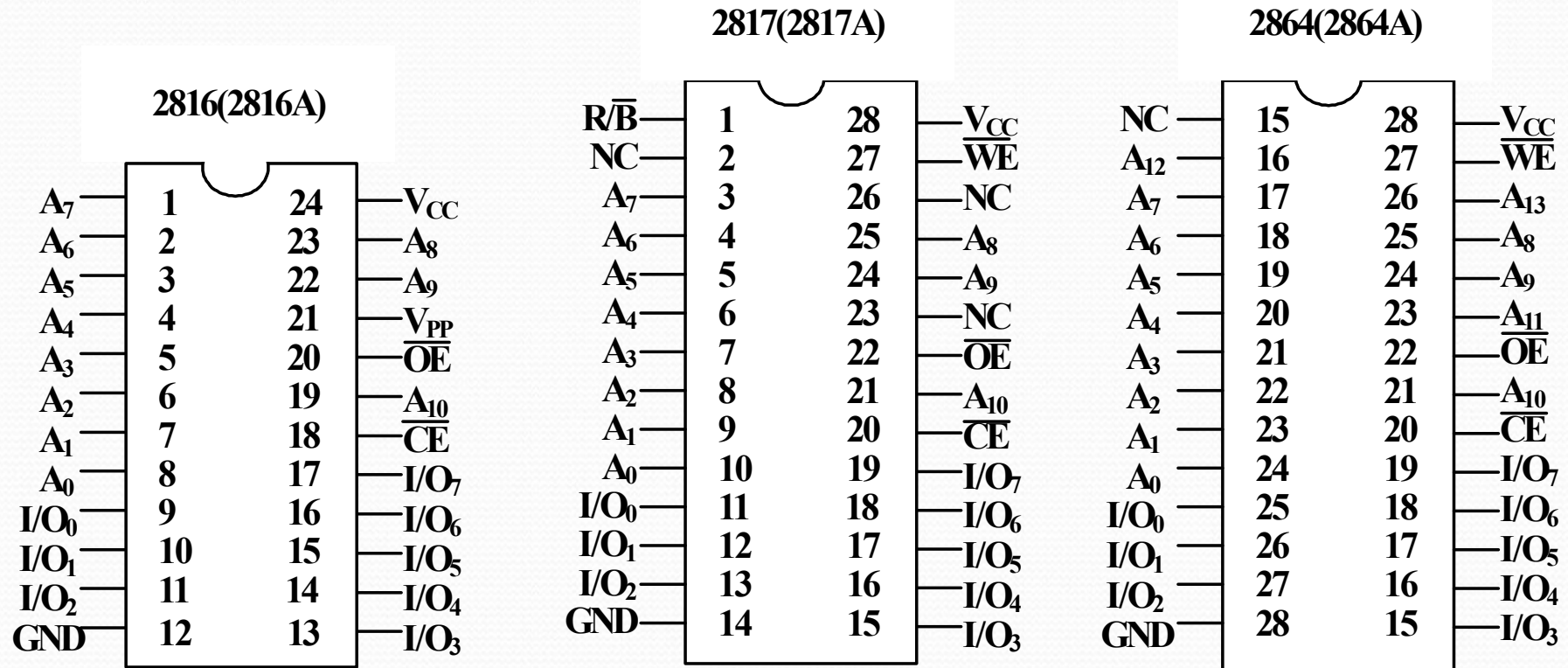
电可擦除可编程只读存储器

E²PROM的特点

E²PROM（Electric Erasable PROM）突出的优点是可以在线擦除和改写。

E²PROM既具有ROM的非易失性的优点，又能像RAM一样随机地进行读写，每个单元可重复进行一万次改写，保留信息的时间长达20年，不存在EPROM在日光下信息缓慢丢失的问题。

E²PROM芯片介绍



常用E²PROM管脚图

快速擦写存储器(Flash)

快速擦写存储器（Flash Memory）是Intel公司于20世纪80年代后期推出的新型存储器，由于众多的特点而深受用户的青睐。

Flash的主要性能特点：

- 1) 高速芯片整体电可擦除。芯片整体擦除只用约1s，而一般的EPROM擦除约花15~20 min。
- 2) 高速编程。采用快速脉冲编程方法，对于28F256A、28F512、28F010、28F020，每个字节的编程仅花10 μ s。对于以上4种芯片整个芯片编程时间分别为0.5s、1s、2s、4s。
- 3) 通常可进行100000次擦除/编程，最少可有10000次。
- 4) 改进后的Flash，内部集成了一个DC/DC变换器，可采用单一5V电压供电。

Flash的主要性能特点

- 5) 高速的存储器访问——最大读取时间不超过200ns，高速Flash可低至60ns。
- 6) 低功耗——最大工作电流为30mA，备用状态下的最大电流为100 μ A。
- 7) 内部的命令寄存器结构可用于微处理器/微控制器兼容的写入接口。
- 8) 抗噪声特性一次允许有 $\pm 10\%$ 的VCC误差。
- 9) 与E²PROM相比，Flash具有密度大、价格低、可靠性高的明显优势。

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

主存储器的组成

主存储器的组成是在给定存储器芯片的前提下，扩展成一定容量的存储器。存储器扩展有以下三种方法：

- (1) 位扩展
- (2) 字扩展
- (3) 字位扩展

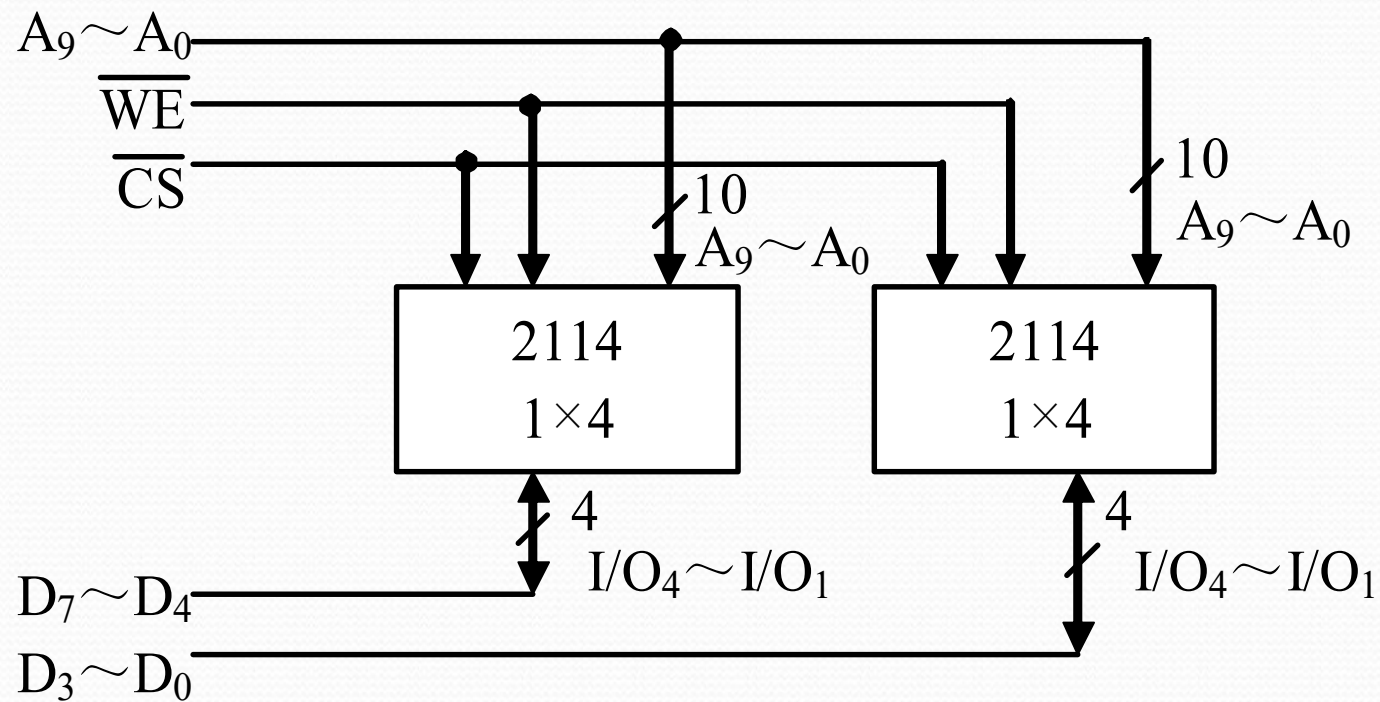
位扩展

指用多个存储器器件对字长进行扩充。位扩展的一般方法是：

- (1) 在给定的芯片中选择合适的芯片，并确定使用数量；
- (2) 将选中芯片的地址线、读写线、片选线对应连接；
- (3) 将数据线单独连接，拼接成要求的数据宽度。

位扩展

例如，使用Intel 2114（1K×4）芯片扩展成为1K×8容量的存储器。根据要求可以选用2片2114，按如图所示连接。



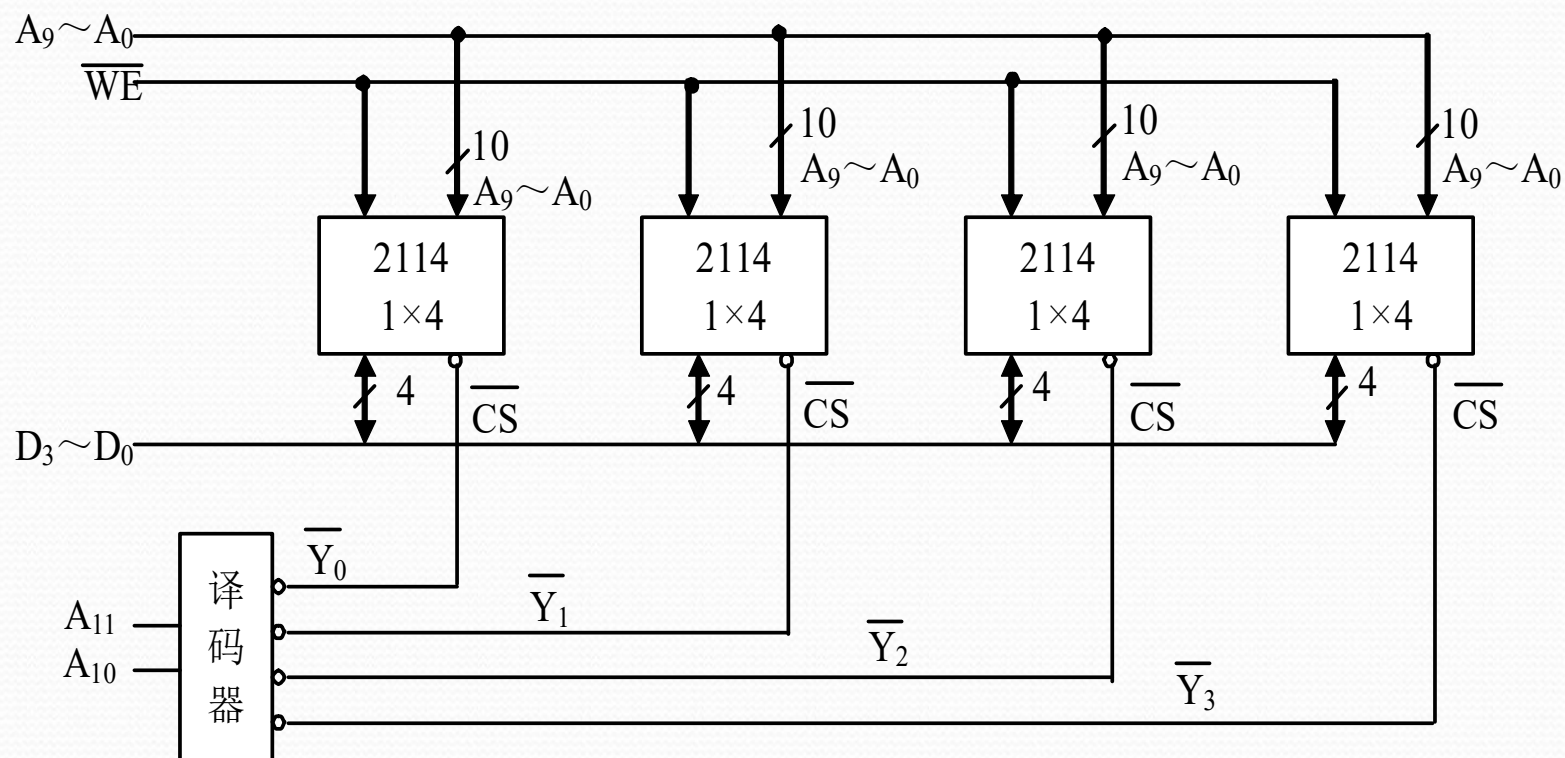
字扩展

指用多个存储器器件对字数进行扩充。字扩展的一般方法是：

- (1) 在给定的芯片中选择合适的芯片，并确定使用数量；
- (2) 将选中芯片的低位地址线、读写线、数据线对应连接；
- (3) 用高位地址线译码，将输出接至各芯片的片选端。

字扩展

例如，使用Intel 2114（1K×4）芯片扩展成为4K×4容量的存储器。根据要求可以选用4片2114，按照如图所示的方式连接。



字位扩展

如果已有芯片 $m \times n$ 若干块，现在要扩展为 $M \times N$ （设 $M > m$, $N > n$ ）容量的存储器，则字位扩展共需要 $m \times n$ 的芯片数量为：

$$C = \left\lceil \frac{M}{m} \times \frac{N}{n} \right\rceil$$

字位扩展的一般方法：

（1）选择芯片先进行位扩展，扩展成“组”，使得“组”的字长达到要求的字长；

（2）再用“组”进行字扩展，按照字扩展的方法将字数增加到目标字数。

主存储器与CPU的连接

- (1)、根据CPU芯片提供的地址线数目，确定CPU访存的地址范围，并写出相应的二进制地址码；
- (2)、根据地址范围的容量，确定各种类型存储器芯片的数目和扩展方法；
- (3)、分配CPU地址线。**CPU地址线的低位**（数量＝存储芯片的地址线数量）直接连接存储芯片的地址线；**CPU高位地址线**皆参与形成存储芯片的片选信号；
- (4)、连接数据线、R/W#等其他信号线，MREQ#信号一般可用作地址译码器的使能信号。

需要说明的是，主存的扩展及与CPU连接在做法上并不唯一，应该具体问题具体分析！

例题

设 CPU 有 16 根地址线，8 根数据线，
MREQ 访存控制信号（低电平有效），
WR 读/写控制信号（高电平为读，低电平为写）
RAM：1K×4位；4K×8位；8K×8 位
ROM：2K×8位；4K×8位；8K×8 位
74LS138 译码器和各种门电路

画出 CPU 与存储器的连接图，要求：

- ① 主存地址空间分配：
 - 6000H-67FFH 为系统程序区；
 - 6800H-6BFFH 为用户程序区。
- ② 合理选用上述存储芯片，说明各选几片？
- ③ 详细画出存储芯片的片选逻辑图。

例题

(1) 写出对应的二进制地址码

A_{15}	A_{14}	A_{13}		A_{11}	A_{10}	...	A_7	...	A_4	A_3	...	A_0
0	1	1	0	0	0	0	0	0	0	0	0	0
⋮												
0	1	1	0	0	1	1	1	1	1	1	1	1
0	1	1	0	1	0	0	0	0	0	0	0	0
⋮												
0	1	1	0	1	0	1	1	1	1	1	1	1

1片2K×8位

ROM

RAM

2片1K×4

(2) 确定芯片的数量及类型

(3) 分配地址线

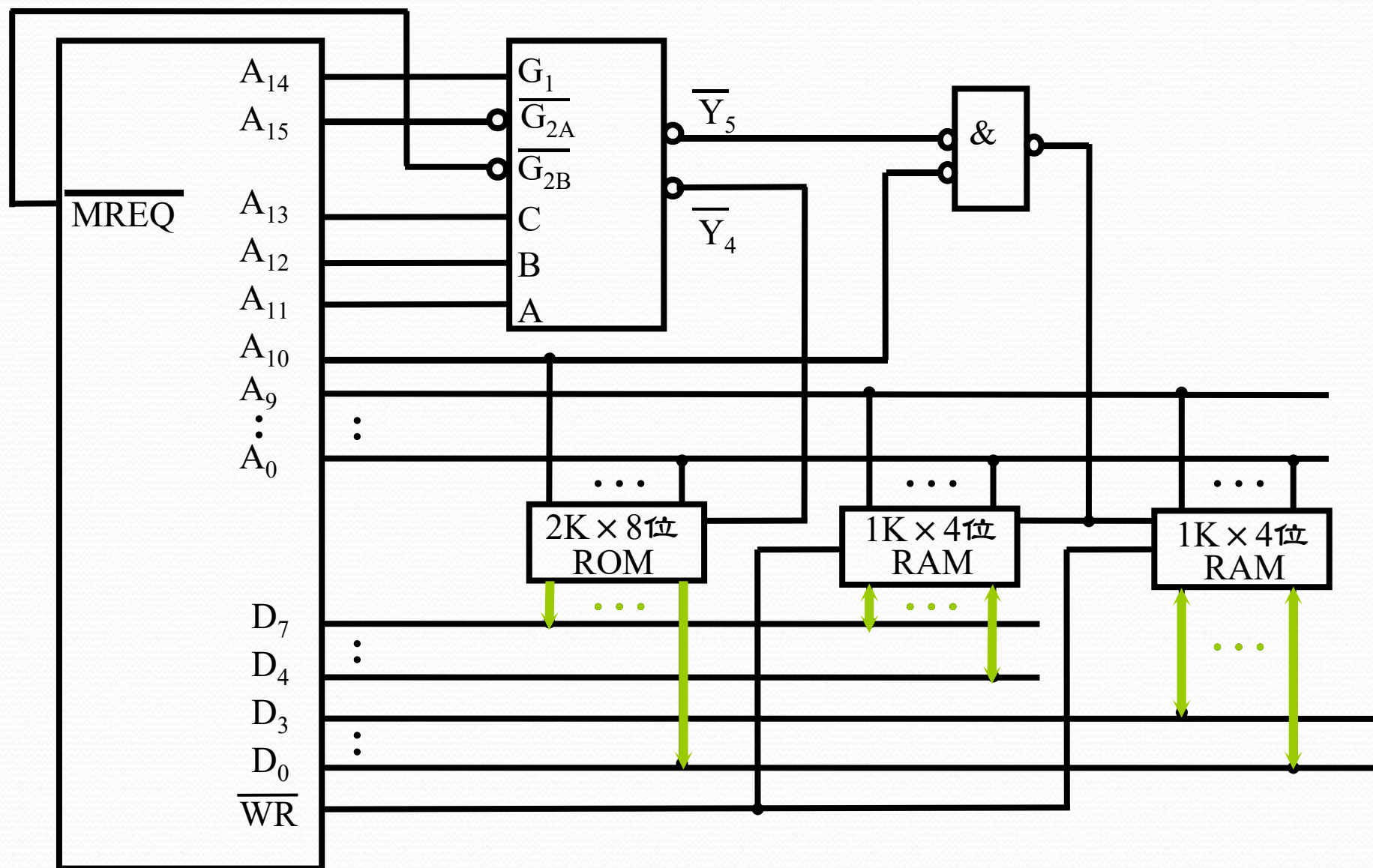


$A_{10} \sim A_0$ 接 $2K \times 8$ 位 ROM 的地址线

$A_9 \sim A_0$ 接 $1K \times 4$ 位 RAM 的地址线

(4) 确定片选信号

(5) CPU 与存储器的连接图



例题

设CPU有16根地址线，8根数据线，并用MREQ#作访存控制信号（低电平有效），用R/W#作读/写控制信号（高电平为读，低电平为写）。现有下列存储芯片：1K×4位SRAM；4K×8位SRAM；8K×8位SRAM；2K×8位ROM；4K×8位ROM；8K×8位ROM；及3:8译码器和各种门电路。

要求：主存的地址空间满足下述条件：最小8K地址为系统程序区，与其相邻的16K地址为用户程序区，最大4K地址空间为系统程序工作区。

- (1)请画出存储芯片的片选逻辑，存储芯片的种类、片数。
- (2)画出CPU与存储器的连接图。

例题

第一步：写出二地址地址码

[illegible]

例题

第二步：选择芯片

- 最小8K系统程序区 $\leftarrow 8K \times 8$ 位ROM，1片；
- 16K用户程序区 $\leftarrow 8K \times 8$ 位SRAM，2片；
- 4K系统程序工作区 $\leftarrow 4K \times 8$ 位SRAM，1片。

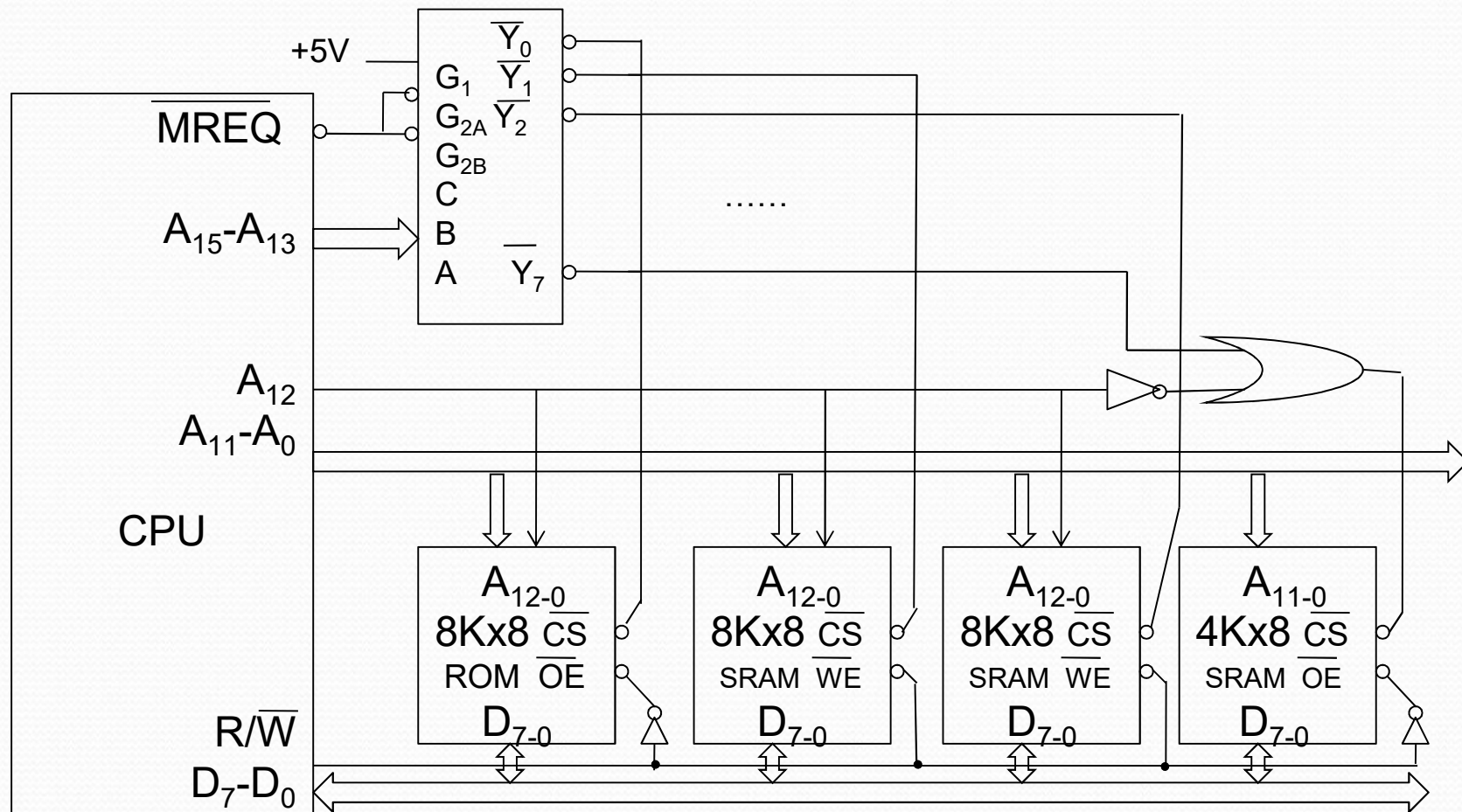
第三步，分配CPU地址线

- CPU的低13位地址线 $A_{12}-A_0$ 与1片 $8K \times 8$ 位ROM和两片 $8K \times 8$ 位SRAM芯片提供的地址线相连；将CPU的低12位地址线 $A_{11}-A_0$ 与1片 $4K \times 8$ 位SRAM芯片提供的地址线相连。

第四步，译码产生片选信号

例题

第五步：画出电路图



第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

存储系统的概念

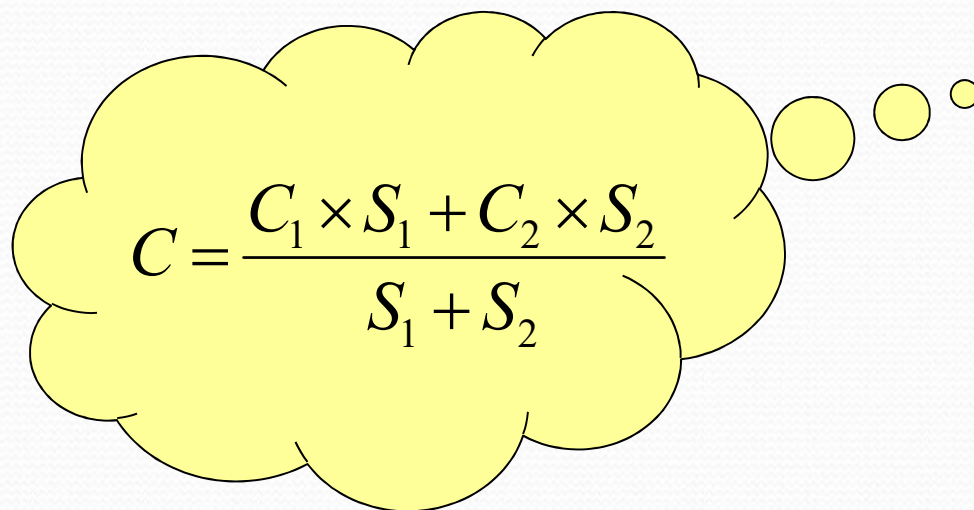
存储系统是指两个或两个以上速度、容量和价格不相同的存储器用硬件、软件、或软件与硬件相结合的方法连接起来成为一个系统。

存储系统的性能指标

以由两个存储器构成的存储系统为例：

1. 存储容量 $S = S_1 + S_2$
2. 位价格 C

整个存储系统的平均位价格可以这样来计算：


$$C = \frac{C_1 \times S_1 + C_2 \times S_2}{S_1 + S_2}$$

存储系统的性能指标

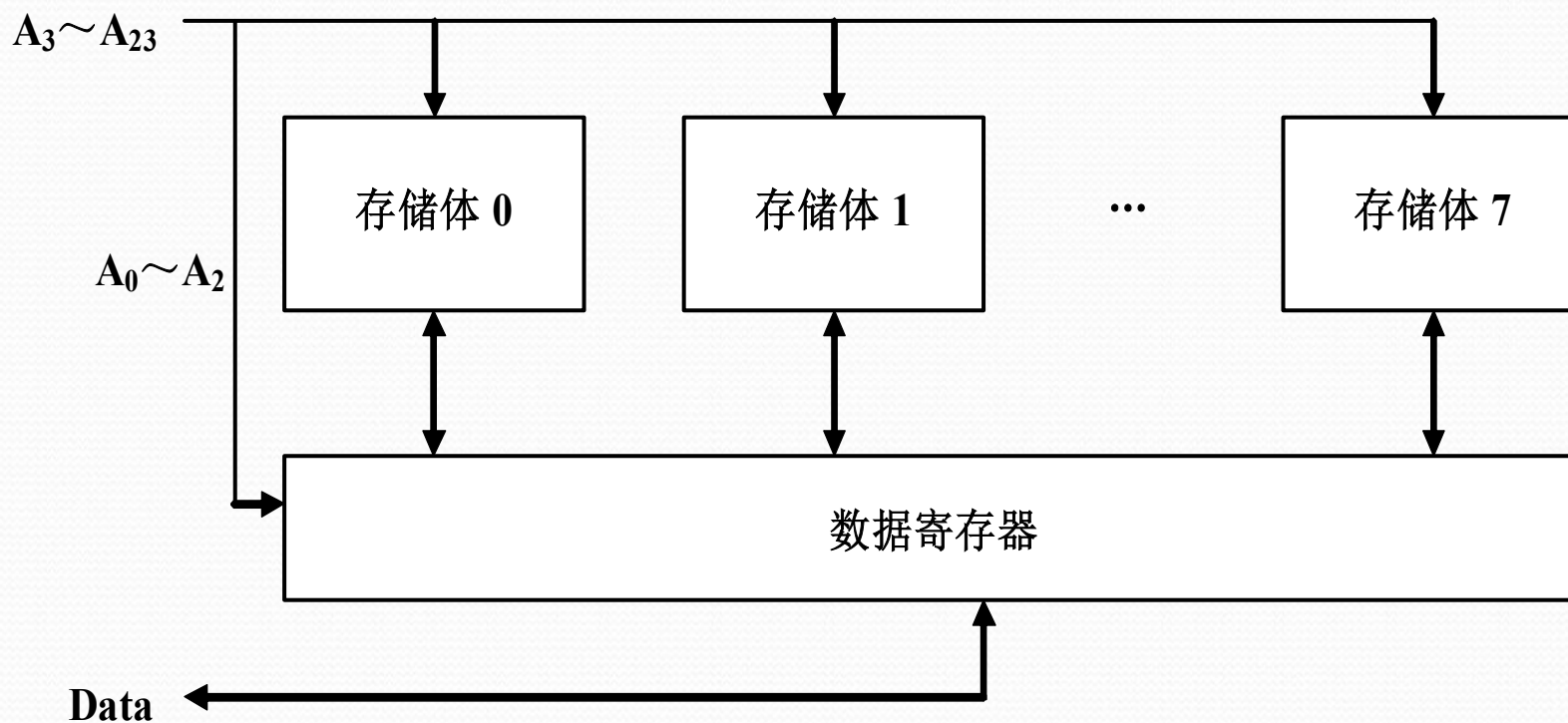
3. 访问周期 T

$$H = \frac{N_1}{N_1 + N_2}$$

$$T = H \times T_1 + (1 - H) \times T_2$$

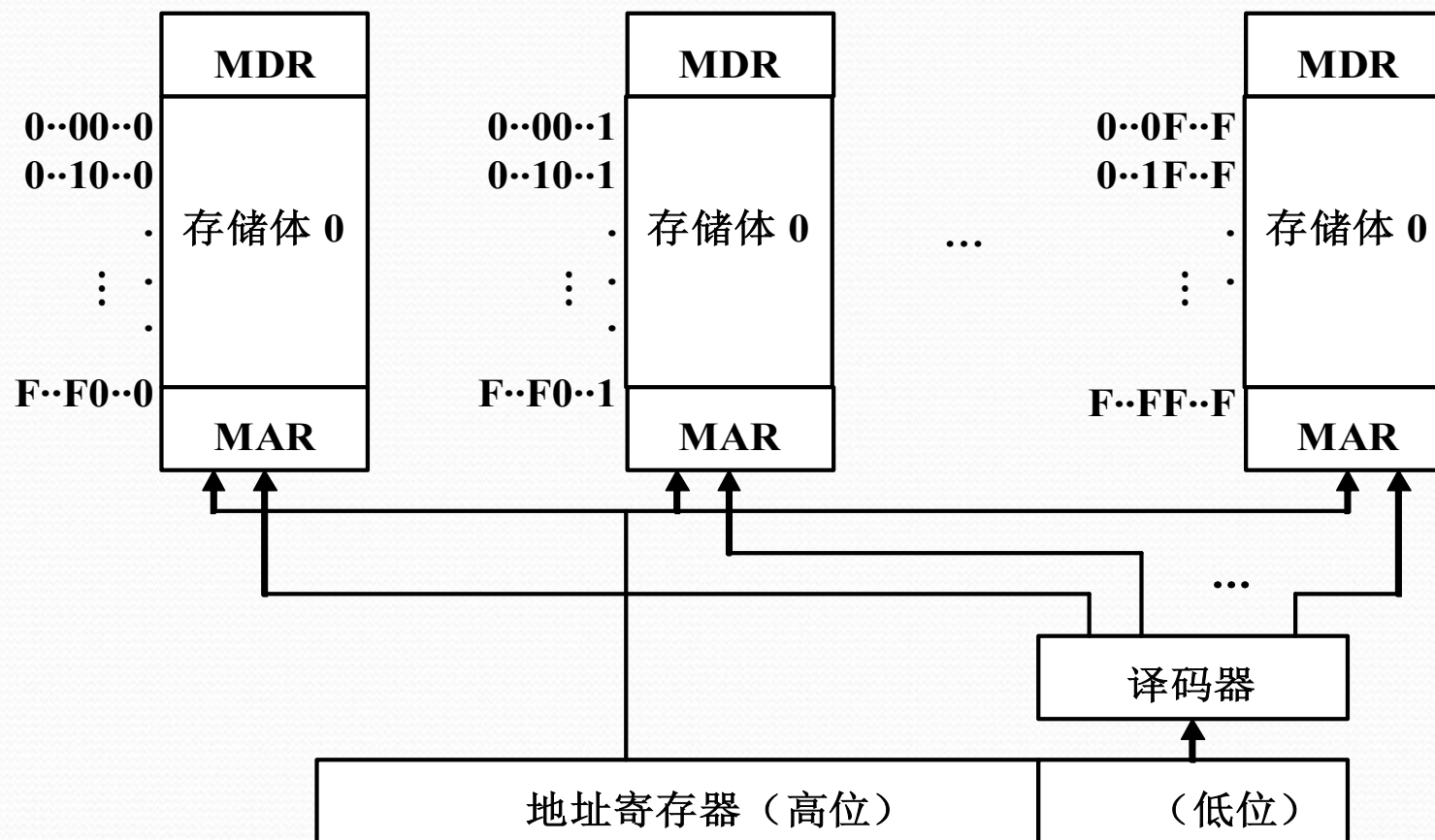
其中， H 表示命中率，在程序执行过程中对 $M1$ 存储器的访问次数为 N_1 （ N_2 同理）， T 表示整个系统的访问周期。

增加存储器的数据宽度



宽字存储器

多体交叉存储技术



低位交叉访问存储器的结构

一种无冲突访问的存储器

采用低位交叉访问技术时，一个由 n 个存储体构成的主存储器，其速度并不能提高到单个存储体的 n 倍，其根本原因在于存在访问冲突。产生访问冲突的根本原因主要有两个，一个是程序中的转移指令，二是数据的随机性。事实上，数据的随机性往往更为重要。

如：一维数组的例子。如果采用低位交叉访问方式的并行存储器由4个存储体构成，交叉存放一维数组 a_0, a_1, a_2, \dots 。如果每次都按连续地址访问，就不存在冲突问题。一个存储周期可以读出四个操作数，如果一个存储周期需要读出更多的操作数，只需要增加存储体的个数就可以了。如果要求按位移量2的变址方式访问并行存储器（读下标为奇数或偶数的操作数），则存储器的频带宽度就要降低一半，即可能有一半的地址是冲突的。

一种无冲突访问的存储器

	0号存储体	1号存储体	2号存储体	3号存储体
体内地址 0	a_0	a_1	a_2	a_3
1	a_4	a_5	a_6	a_7
2	a_8	a_9	a_{10}	a_{11}
3

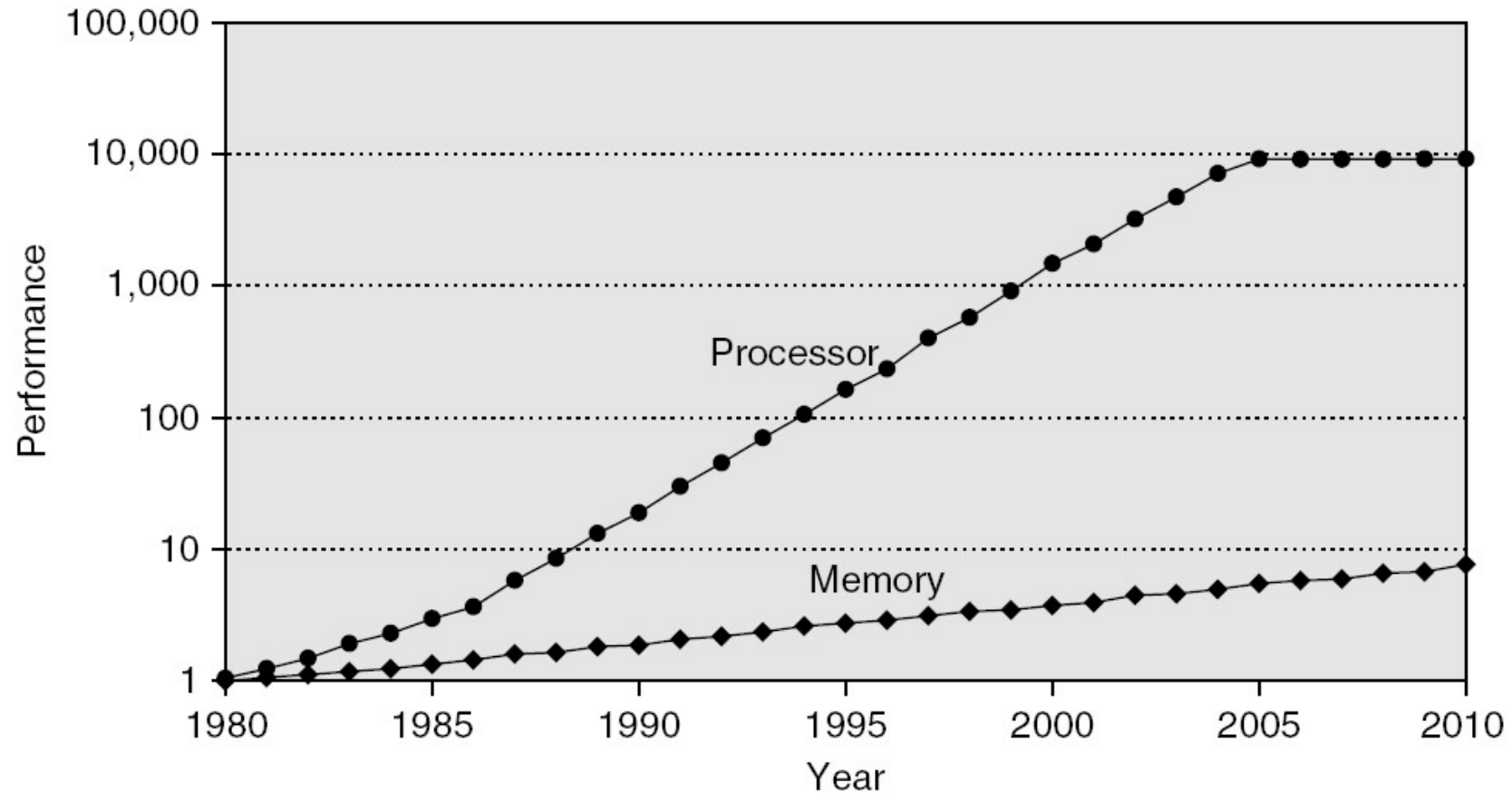
一维数组的存储方案

一种无冲突访问的存储器

事实上，造成上述访问冲突的原因是由于构成存储器的存储体个数 n 为2的整数幂，因此，变址位移量正好是 n 的约数。解决这一问题的方法很简单，只要把存储体的个数 n 选为质数，变址位移量就必然与 n 互质，一维数组的访问冲突问题自然就不存在了。

思考：现在要求对这个二维数组实现按行、按列、按对角线访问，据此思路，设一个 $n \times n$ 的二维数组存储在一个并行存储器中。并且在不同的变址位移量的情况下，都能实现无冲突访问。

处理器和主存储器的性能



处理器和主存储器的性能

- Memory hierarchy design becomes more crucial with recent multi-core processors:
 - Aggregate peak bandwidth grows with # cores:
 - Intel Core i7 can generate two references per core per clock
 - Four cores and 3.2 GHz clock
 - 25.6 billion 64-bit data references/second +
 - 12.8 billion 128-bit instruction references
 - = 409.6 GB/s!
 - DRAM bandwidth is only 6% of this (25 GB/s)
 - Requires:
 - Multi-port, pipelined caches
 - Two levels of cache per core
 - Shared third-level cache on chip

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

Cache概述

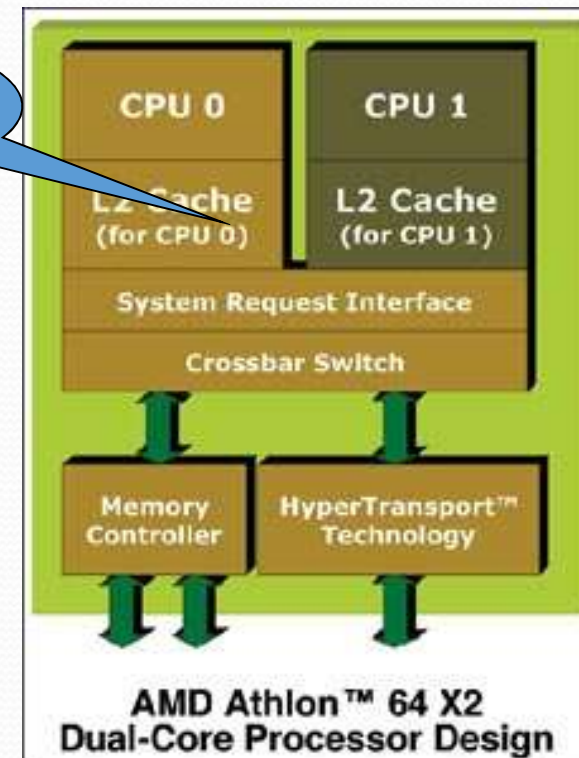
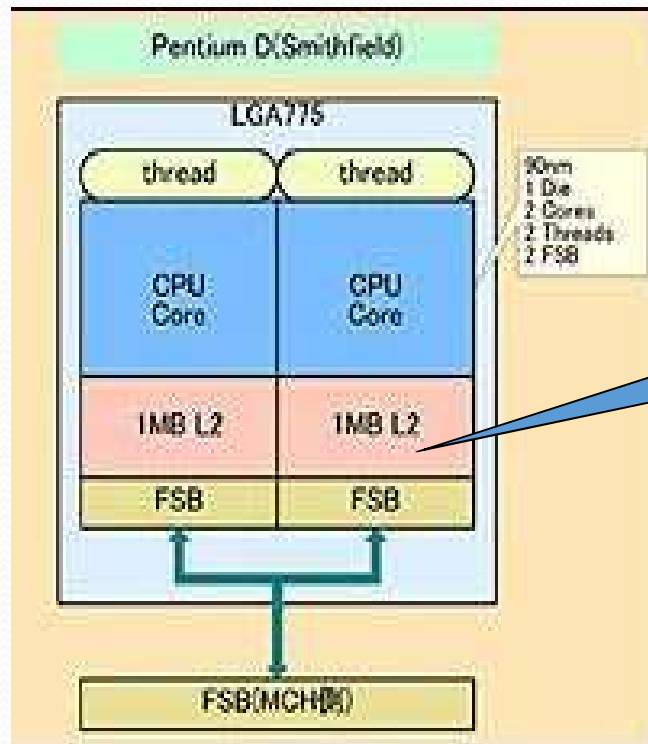
思考：

- 1) 什么是Cache? 为什么要用Cache?
- 2) 使用Cache有实现的可能和必要吗?
- 3) 原理是什么样的? 怎么实现?

Cache是为了解决存储器的速度问题而设计的!

Cache概述

现代计算机均采用**cache**结构！



Cache概述

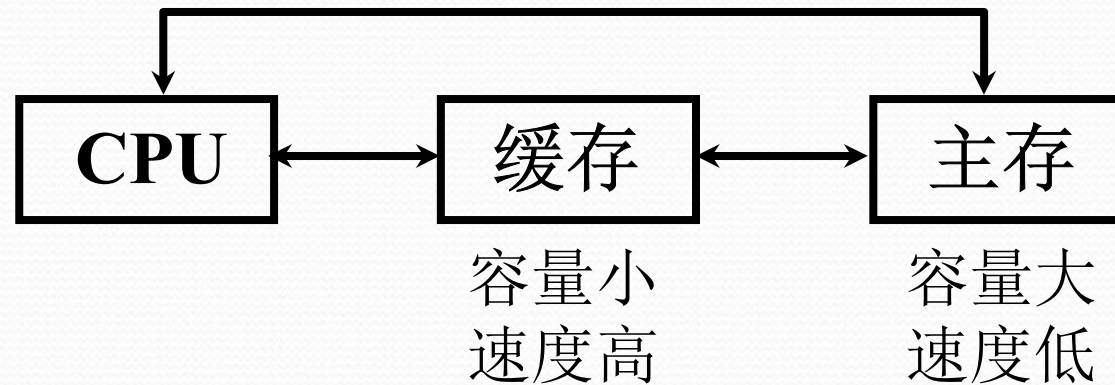
1980年代x86 CPU的Cache设计



CPU	典型主频	访存周期	DRAM延迟	Cache设计
8088	4.77MHz (210ns)	4 (840ns)	250ns	无需Cache
80286	10MHz (100ns)	2 (200ns)	220ns	无需Cache
80386	25MHz (40ns)	2 (80ns)	190ns	片外Cache
80486	33MHz (30ns)	2 (60ns)	165ns	8KB片内 Cache

Cache的工作原理

CPU 和主存（DRAM）的**速度**存在差异



程序访问的**局部性**原理

程序的局部性

经过对大量典型程序的运行分析表明：当CPU从内存中取出指令和数据时，在一个较短的时间间隔内，由程序产生的地址往往局限在一个很小的区域内，在一个较短的时间间隔内，对局部范围的存储器地址的频繁访问，而对此范围以外的地址访问甚少的现象，称为程序的局部性。

Cache技术正是基于这种程序的局部性原理。把程序中正在使用的部分存放在一个高速的Cache中，使CPU的访存操作大多针对Cache进行。

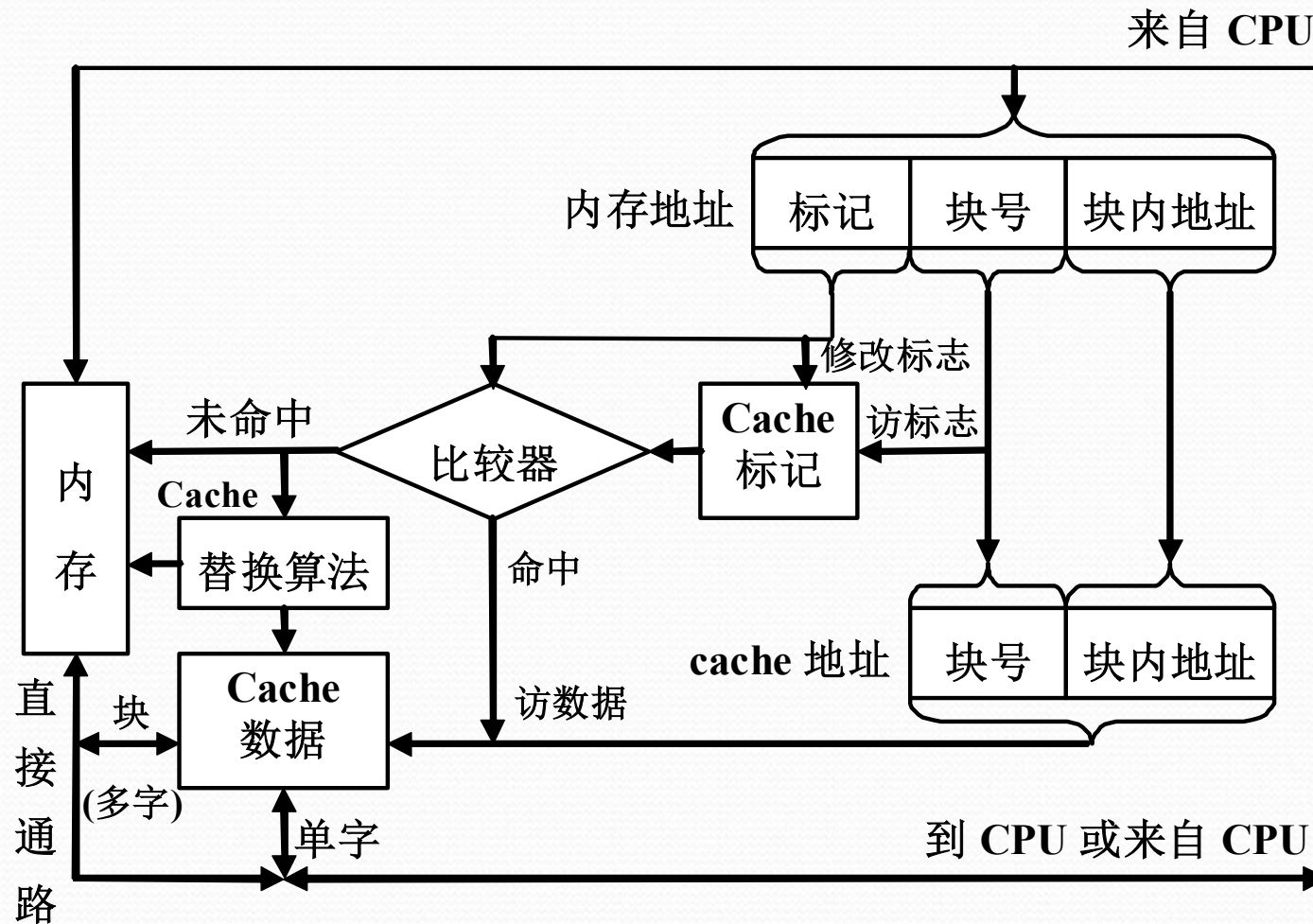
Cache的基本结构

Cache的物理位置**介于CPU和内存之间**，与内存有机地结合起来，借助辅助硬件构成Cache-内存层次。Cache的存取速度与CPU相匹配，但是容量较小，Cache中的信息是内存中信息的一部分。

在Cache系统中，把Cache和主存都划分成大小相等的块，每个块由若干个字节组成。由于Cache的容量远小于内存的容量，所以Cache中的块数要远少于内存中块的数量，Cache中保存的信息只是内存中最活跃的若干块的副本。

用内存地址的块号字段访问Cache标记，并将取出的标记和内存地址的标记字段相比较。若相等，说明访问的数据在Cache中，称为**命中**；若不相等，说明访问的数据不在Cache中，称为**不命中或失效**。

Cache的基本结构



Cache的读写过程

读操作:

当CPU发出读请求时，有两种情况：

- 1) **Cache命中**，直接对Cache进行读操作，不对内存进行操作；
- 2) **Cache未命中**，称为读缺失(read-miss)，则访问内存，CPU从内存中读取需要的信息，同时将对应块的信息调入Cache，若此时Cache已满，则须根据替换算法，用这个块替换掉Cache中原来的某块信息。

Cache的读写过程

写操作:

当CPU发出写操作请求时，如果Cache**未命中**，出现写缺失(write-miss)，先将对应的块调入Cache；如果Cache命中，就会出现如何保持Cache中的内容与内存中内容的一致性问题，一般的处理方法有：

- (1) **写直达法** (write-through)
- (2) **写回法** (write-back)

将CPU要写的信息暂时只写入Cache，并用标志将该块加以注明，直到该块从Cache中替换出去时才一次写回内存。

Cache的地址映像与地址变换

1. 直接映像及地址变换
2. 全相联映像及地址变换
3. 组相联映像及地址变换

直接映像及地址变换

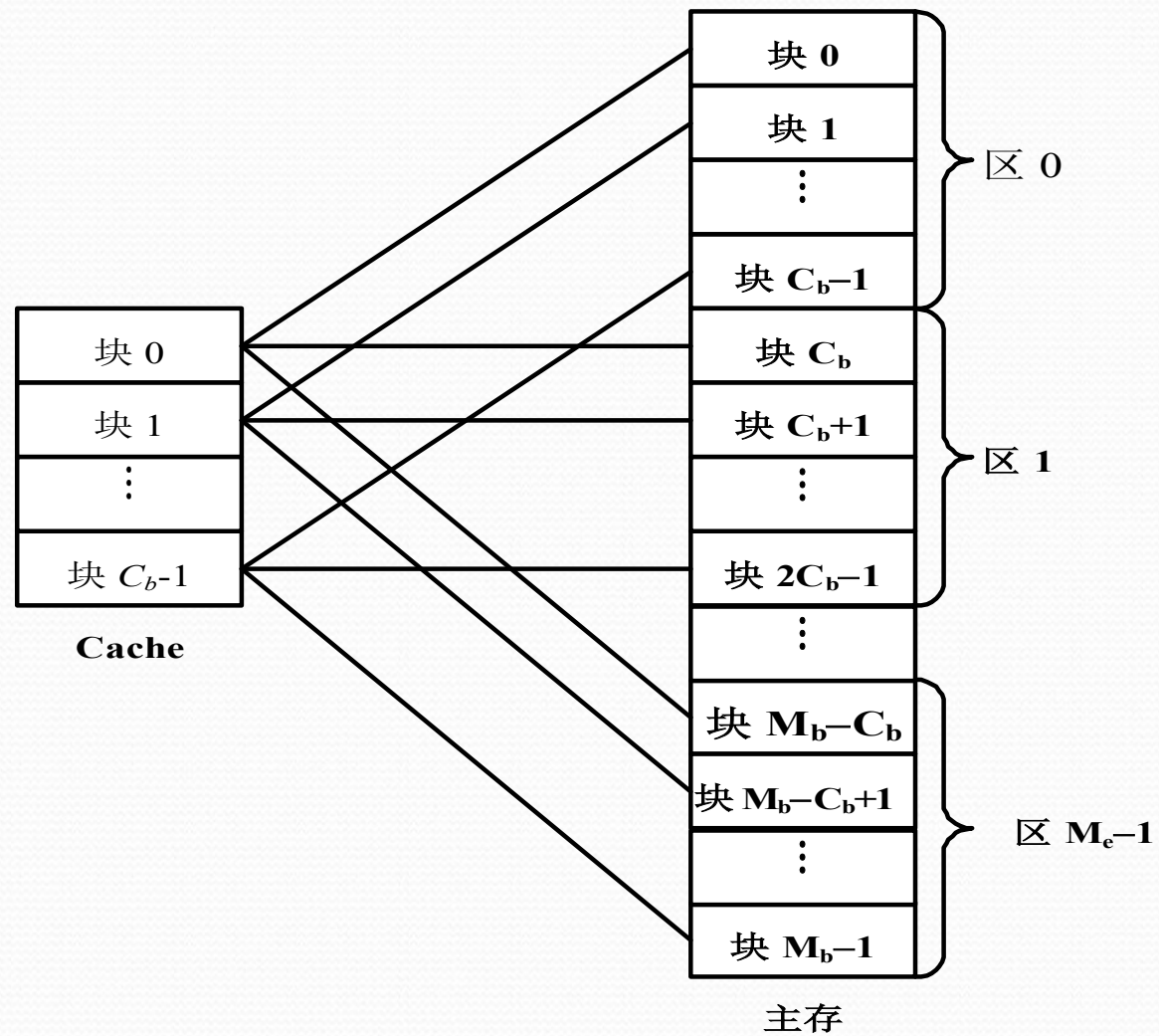
直接映像是将主存中的一块唯一映像到Cache中的一块，而Cache中的一块要对应主存中的若干块。

设主存中的块号为 B 、Cache块的块号为 b ，若主存的块数为 M_b ，Cache的块数为 C_b ，则映像关系可以表示为：

$$b = B \bmod C_b$$

可见：**cache**中块号为**b**对应主存中的若干块 (M_b/C_b)

直接映像方式

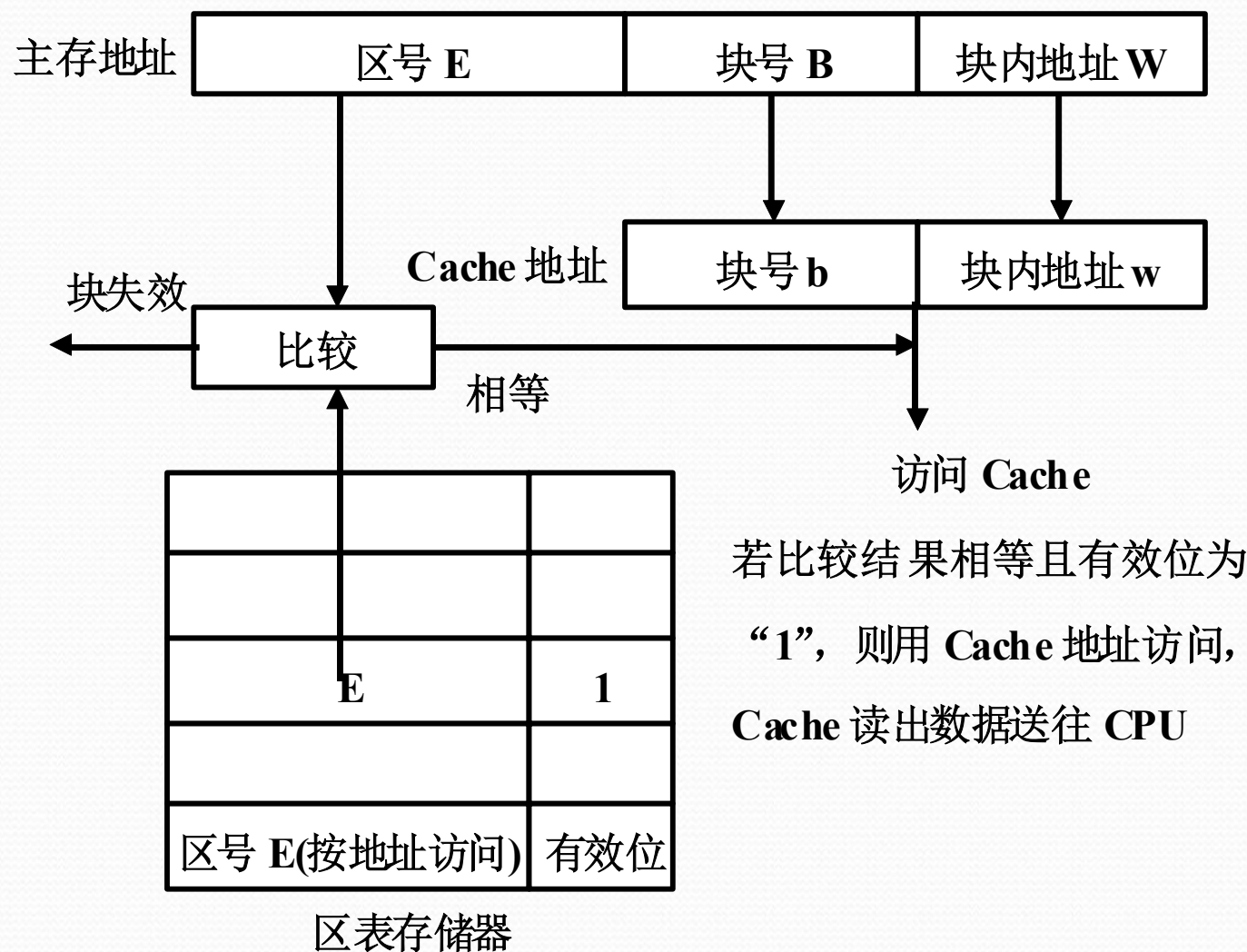


直接映像方式

C_b 是Cache的块容量，将块容量为 M_b 的主存按Cache的大小划分成了 $M_e = M_b / C_b$ 个区。在直接映像方式中，主存各个区中区内块号 B 相同的那些块都映像到Cache中同一个块号的位置上。因此，Cache的地址与主存的地址除区号外的低位部分完全相同。

在直接映像方式中，用于保存地址变换信息的表称为区表，区表存储的行数与Cache的块数 C_b 相同，字长为主存地址中区号的长度，另加一个有效位。

地址变换



地址变换

用主存地址中的块号 B 去访问区表存储器，把读出的区号与区表中的 E 相比较。比较结果有四种情况：

(1) 区号相同，有效位也为“1”，则命中且相应Cache块为有效块，以主存地址中的块号 B 和块内地址 W 为Cache地址，访问Cache。

(2) 区号相同，有效位为“0”，则表示Cache中虽然已有要访问的块，但该块与已经被修改过的主存副本块内容不一致，已经被作废，需要再从主存中调入一次来重写该块，并且把有效位置成“1”。

(3) 区号比较结果不相等，有效位为“1”，则表示没有命中，但该Cache块为有效块，需要把该Cache块调出，写入主存，修改主存中相应的副本块，然后从主存中调入所需的新块。

(4) 区号比较结果不相等，有效位为“0”，则表示没有命中而且该Cache块已作废，可以直接调入所需要的新块。

例题

例：假设在某个计算机系统中Cache容量为64K字节，数据块大小是16个字节，主存容量是4M，地址映象为直接相联方式。

- (1) 主存地址多少位？如何分配？
- (2) Cache地址多少位？如何分配？
- (3) 目录表（区表）的格式和容量？

解：

(1) 主存地址共22位（因为 $4M=2^{22}$ ）。主存可分为 $4M/64K=64$ 个区，每个区的块数为 $64K/16=4K$ 。因此主存的地址结构为：区号（6位）、区内块号（12位）、块内字地址（4位）。

(2) Cache地址为16位（因为 $64K=2^{16}$ ）。Cache地址分为块号和块内地址，Cache可分为 $64K/16=2^{12}$ 块，因此块地址为12位，块内字地址为4位。

(3) 区表的格式中包括6位区号和1位有效位，共7位构成。区表的容量与缓冲块容量相同，共 $2^{12}=4K$ 。（ $4K*7$ ）

直接映像的优缺点

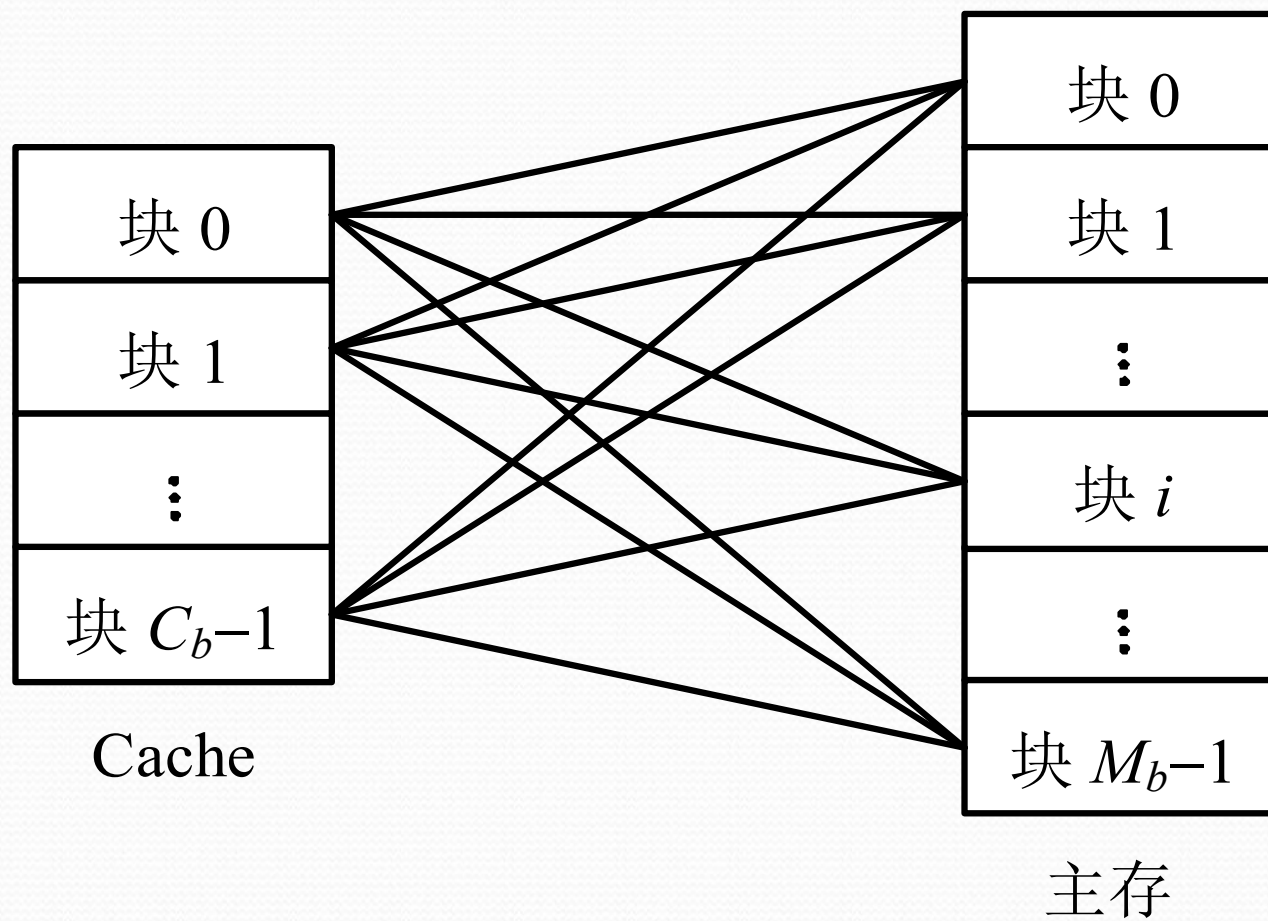
优点：映像函数简单，实现硬件简单，不需要相联存储器，地址变换的速度也较快。

缺点：块冲突率比较高，当两个或两个以上的块映射到相同的Cache块位置而发生冲突时，即使其他Cache块位置空闲也不能被使用。

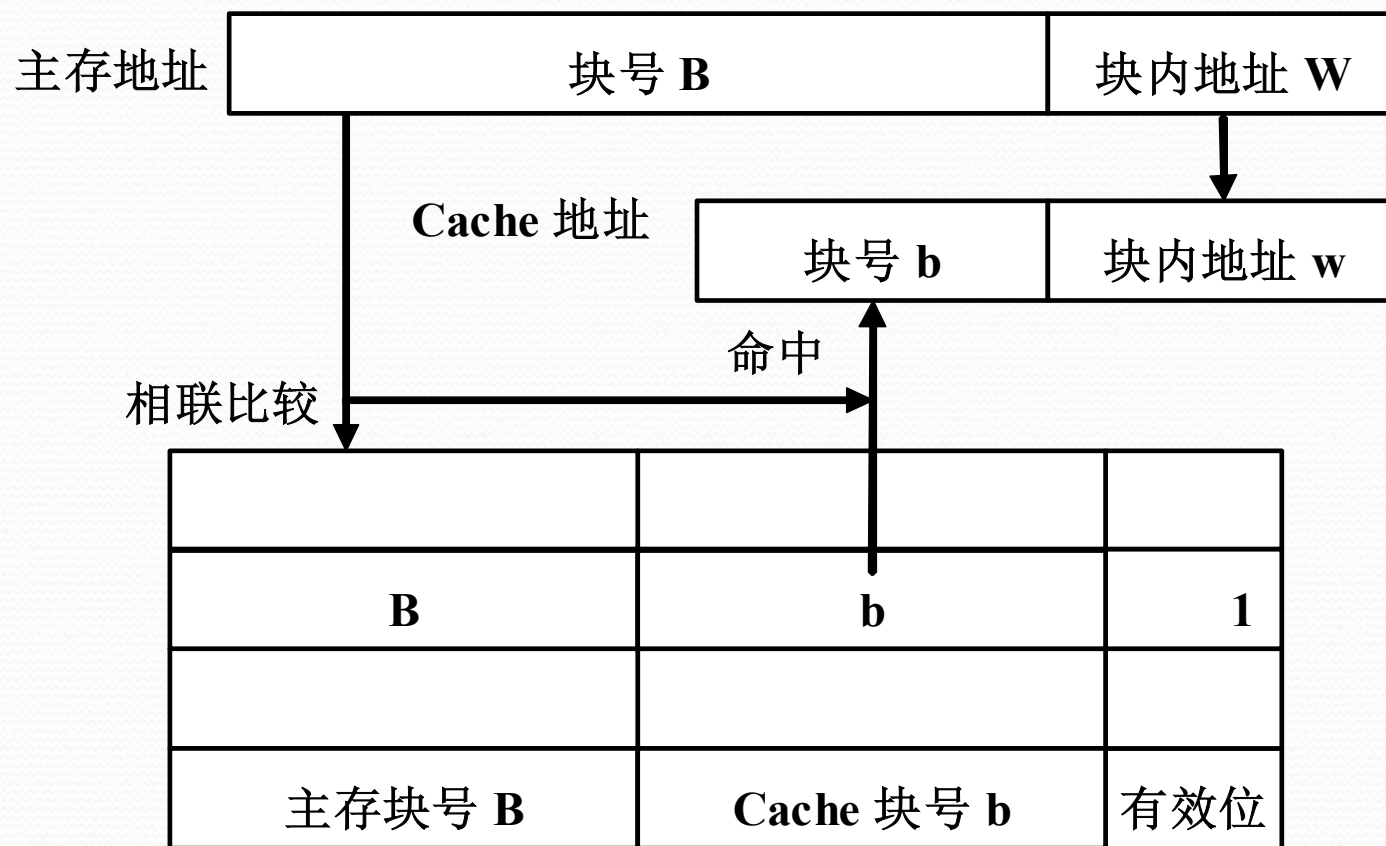
全相联映像及地址变换

全相联映像方式是主存中的任意一块可以映像到Cache中任意的块位置上。如果Cache的块数为 C_b ，主存的块数为 M_b ，则主存和Cache块之间的映像关系共有 $C_b \times M_b$ 种。

全相联映像方式



全相联映像地址变换方式



目录表（由相联存储器构成，共 C_b 个字）

例题

例: 假设在某个计算机系统中Cache容量为32K字节, 数据块大小是16个字节, 主存容量是1MB, 地址映象为全相联方式。

(1) 主存地址多少位? 如何分配?

(2) Cache地址多少位? 如何分配?

(3) 目录表的格式和容量?

解:

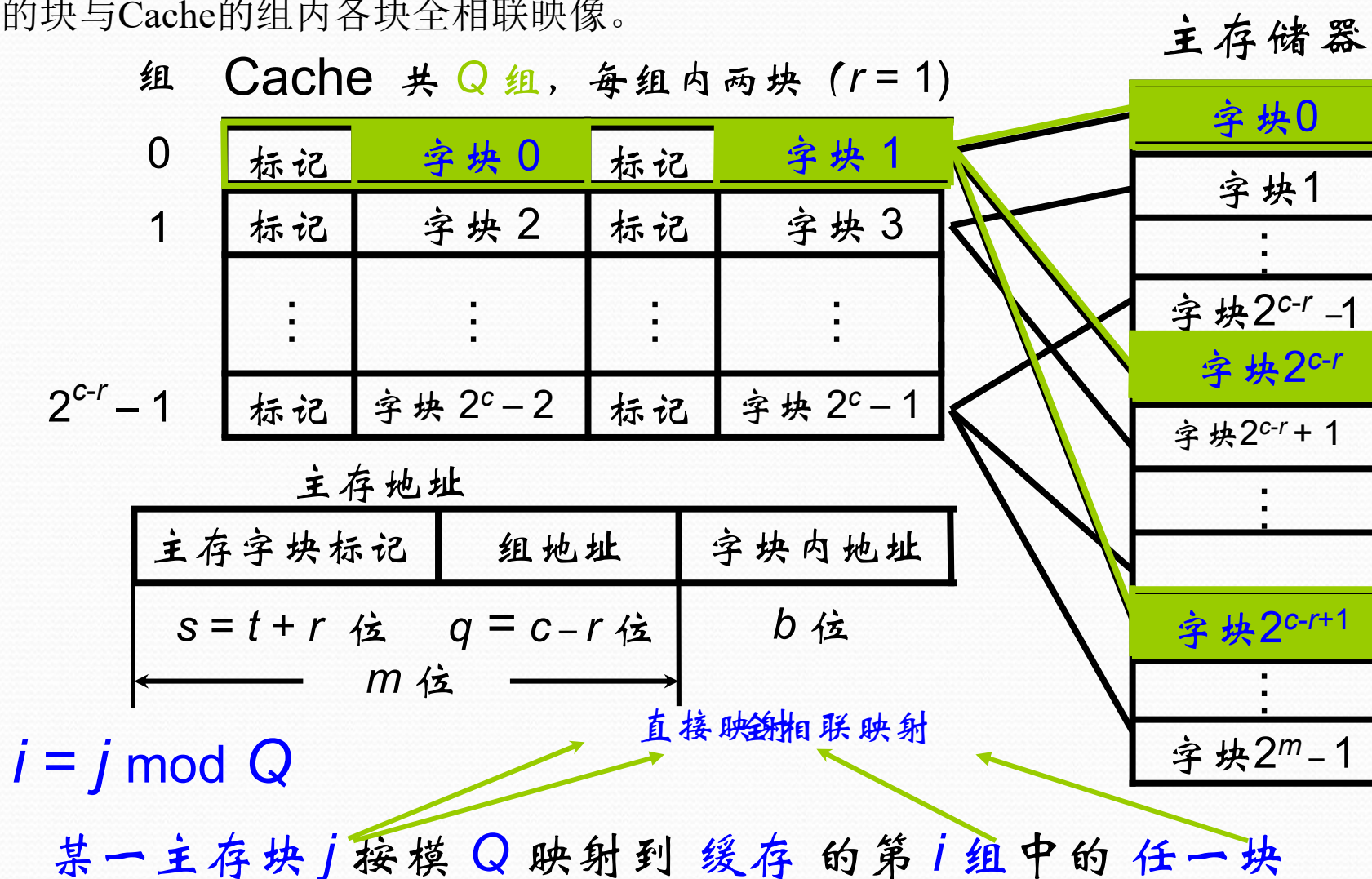
(1) 主存地址20位, 分为块号和块内地址, 主存共分为 $1\text{M}/16=2^{16}$ 块, 因此块号占16位, 块内地址占4位。

(2) Cache地址共15位, 块地址11位, 块内地址4位。

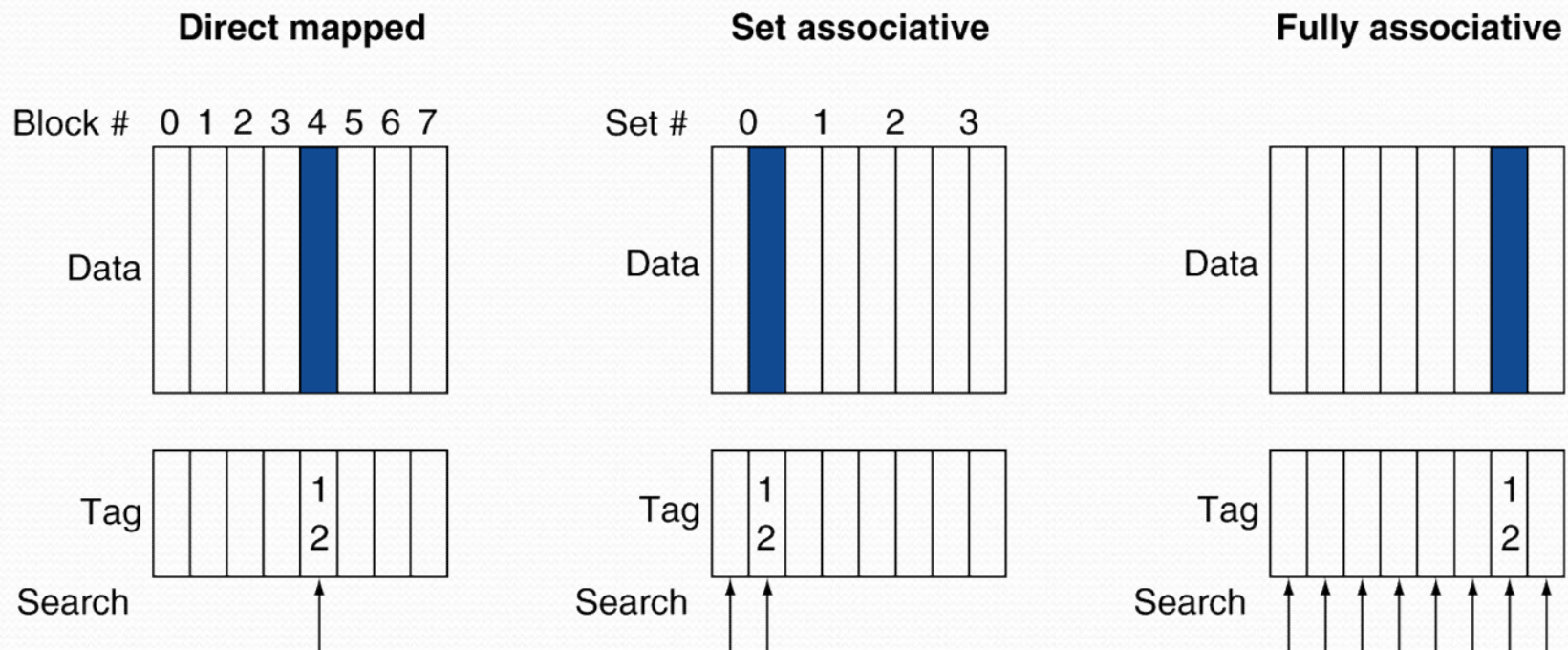
(3) 目录表由主存块号16位、Cache块号11位和1位有效位共28位构成, 容量为Cache的块数, 即2K。(2K*28)

组相联映像及地址变换

组相联映像方式把Cache分为若干组，Cache的组与主存中的块直接映像，主存中的块与Cache的组内各块全相联映像。



组相联映像示例



各种映像方式的关系

For a cache with 8 entries

One-way set associative (direct mapped)

Block	Tag	Data
0		
1		
2		
3		
4		
5		
6		
7		

Two-way set associative

Set	Tag	Data	Tag	Data
0				
1				
2				
3				

Four-way set associative

Set	Tag	Data	Tag	Data	Tag	Data	Tag	Data
0								
1								

Eight-way set associative (fully associative)

[illegible]

分组数量的影响

➤ 2-way set associative

Block address	Cache index	Hit/miss	Cache content after access			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0	miss	Mem[0]	Mem[8]		
0	0	hit	Mem[0]	Mem[8]		
6	0	miss	Mem[0]	Mem[6]		
8	0	miss	Mem[8]	Mem[6]		

➤ Fully associative

Block address		Hit/miss	Cache content after access			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	Mem[0]	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	Mem[8]	Mem[6]	

替换算法

在采用全相连映像方式和组相连映像方式从内存向Cache传送一个新块，而Cache中的可用位置已被占满时，就需要使用替换算法，将Cache中的块替换出去而调入新块。常用的替换算法有以下几种：

- 1) 随机算法
- 2) 先进先出算法
- 3) 近期使用最少算法
- 4) 最优化算法

替换算法

(1).随机 (RAND) 算法

该算法完全不管Cache块的过去、现在及将来的使用情况，而是根据一个随机数，选择一块替换掉。

(2).先进先出 (FIFO) 算法

该算法的思想是按调入Cache的先后决定替换的顺序，即在需要替换时，将最先进入Cache的块作为被替换的块。

(3).近期使用最少 (LRU) 算法

该算法是根据块的使用状况将CPU近期最少使用的块作为被替换的块。

(4).最优化 (OPT) 算法

OPT算法是一种以将来使用最少作为替换的目标的一种算法。

Cache的加速比与效率

Cache系统的加速比 S_p 定义为:

$$S_p = \frac{T_m}{T} = \frac{T_m}{H \times T_c + (1-H)T_m} = \frac{1}{(1-H) + H \times \frac{T_c}{T_m}}$$

其中, T_c 为Cache的访存周期, T_m 为主存的访问周期,
 T 为Cache系统的等效访问周期, Cache的命中率为 H 。

Cache系统的效率定义为 $e=T_c/T$ 。

例：假设在某个计算机系统中Cache容量为8K字节，数据块大小是8个字，每个字32位，主存容量是16MB，地址映象为四路组相联方式。

(1) 主存地址多少位？如何分配？

(2) 设cache初始为空，CPU依次访问主存的第0, 1, 2, ..., 99号单元读出100个字（主存每次读出1个字），重复读10次，命中率是多少

(3) 若Cache的速度是主存速度的5倍，有cache和无cache相比，速度提高了多少倍？

(4) cache系统的效率是多少？

解：

(1) 主存地址共24位（因为 $16M=2^{24}$ ）。主存的地址结构为：主存字块标记（13位）、组地址（6位）、字块内地址（5位）。

(2) 命中率 $(1000-13)/1000=0.987$ 。

(3) 设cache的存取周期为 t ，则主存为 $5t$ ，提高到 $5t/(Ht+(1-H)5t)$ 倍,带入计算可得4.75倍，提高了3.75倍；

(4) 效率为 $t/(Ht+(1-H)5t)$,带入计算可得95%

Cache的改进

1) 增加 Cache 的级数

片载（片内）Cache

片外 Cache

2) 统一缓存和分立缓存

指令 Cache

数据 Cache

与主存结构有关

与指令执行的控制方式有关

是否流水

Pentium 8K 指令 Cache 8K 数据 Cache

PowerPC620 32K 指令 Cache 32K 数据 Cache

Cache小结

1) 为了提高存储系统的速度, 引入cache

2) Cache的基本原理

- 命中、未命中、读、写

3) Cache的映像方式

- 直接映像
- 全相联映像
- 组相联映像

4) Cache的替换算法

- 随机算法
- FIFO算法
- LRU算法
- OPT算法

第五章 主存储器与存储系统

- 5.1 存储器分类与技术指标
- 5.2 读写存储器
- 5.3 非易失性半导体存储器
- 5.4 主存储器组成
- 5.5 存储系统与并行存储器*
- 5.6 高速缓冲器Cache
- 5.7 虚拟存储器原理

虚拟存储器的工作原理

虚拟存储器主要由软件管理。

根据采用的存储映像算法，可以将虚拟存储器的管理方式分成段式、页式和段页式三种。

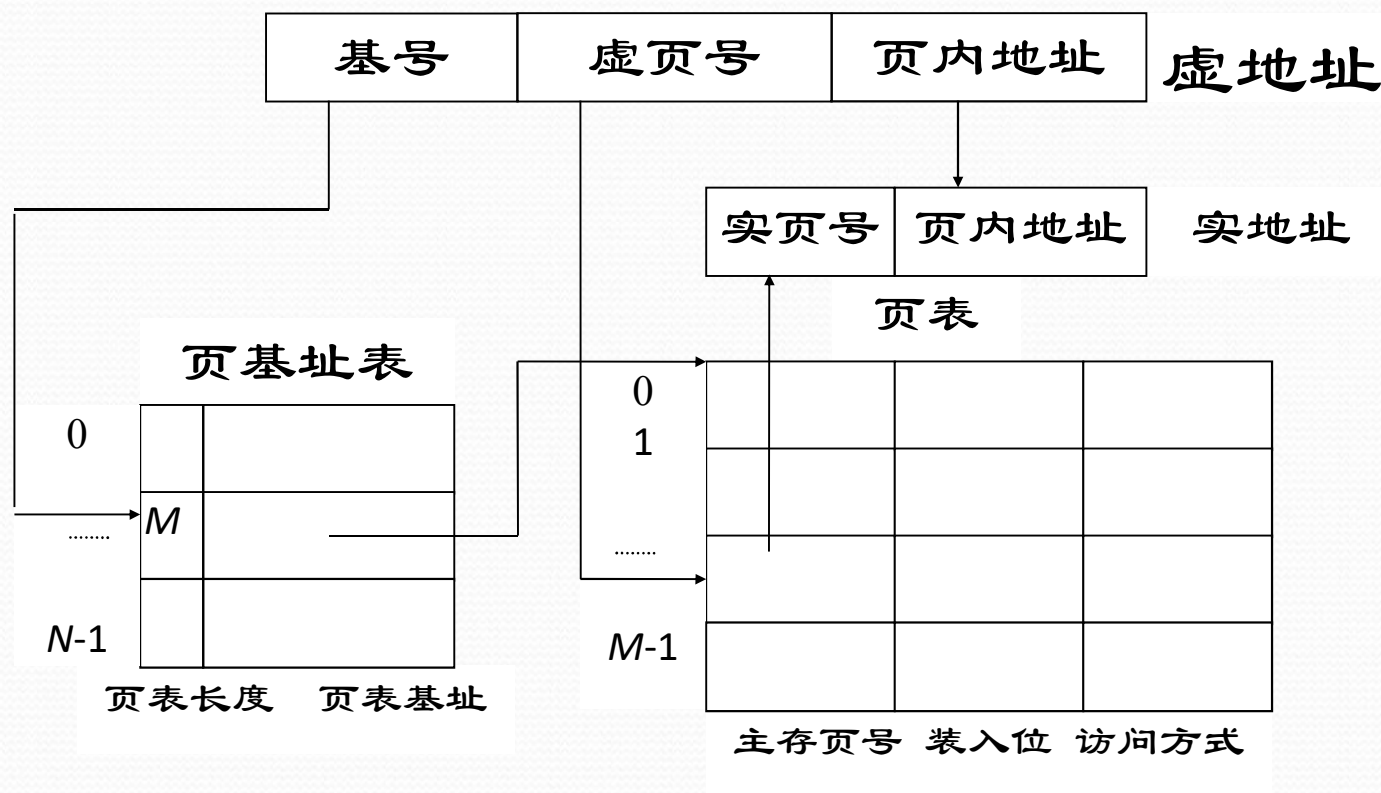
在虚拟存储器中，有三种地址空间，第一种是虚拟地址空间，也称为虚空间或虚拟存储器空间，它是应用程序员用来编写程序的地址空间；第二种空间是主存储器的地址空间，也称主存地址空间、主存物理空间或实存地址空间；第三种是辅存地址空间。

地址映像就是把虚拟地址空间映像到主存地址空间。

目前主要有页式虚拟存储器、段式虚拟存储器和段页式虚拟存储器三种

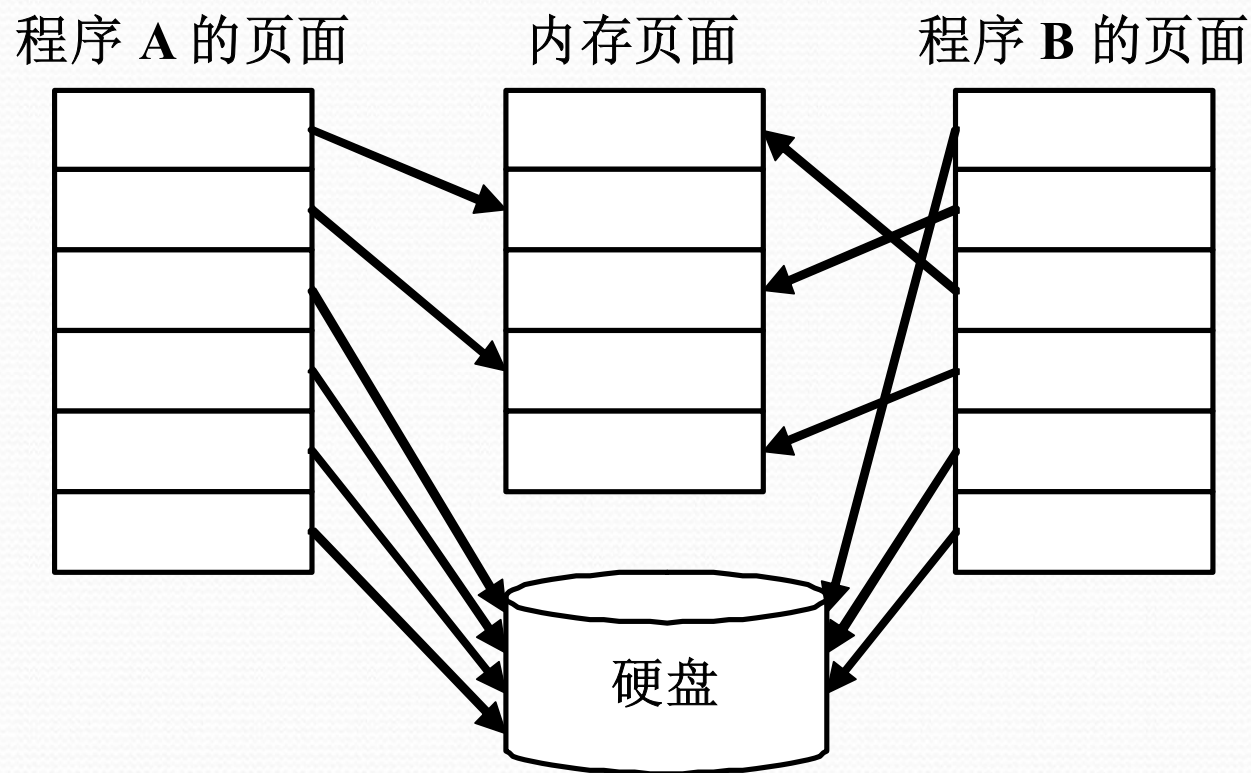
页式虚拟存储器

页式虚拟存储器是把虚拟存储空间和主存实空间划分成固定容量的页（Page），各虚拟页可装入主存中不同的实际页面位置。



页式虚拟存储器的地址映像方式

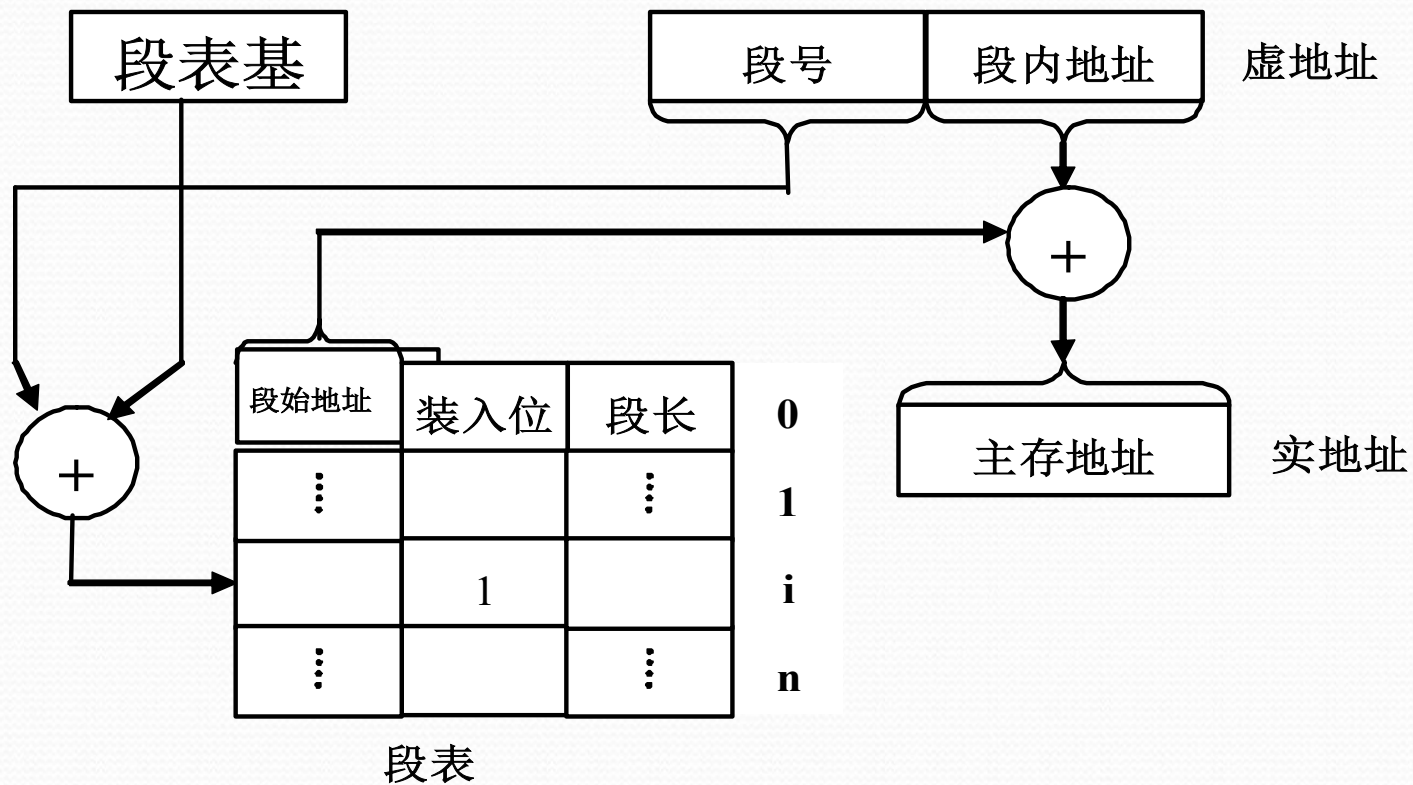
页式虚拟存储器



页式虚拟存储器页面映像的一个实例

段式虚拟存储器

段式虚拟存储器以段为单位与主存进行数据交换，操作系统通过段表对段进行管理。



段式虚拟存储器的地址映像方式

段页式虚拟存储器

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。

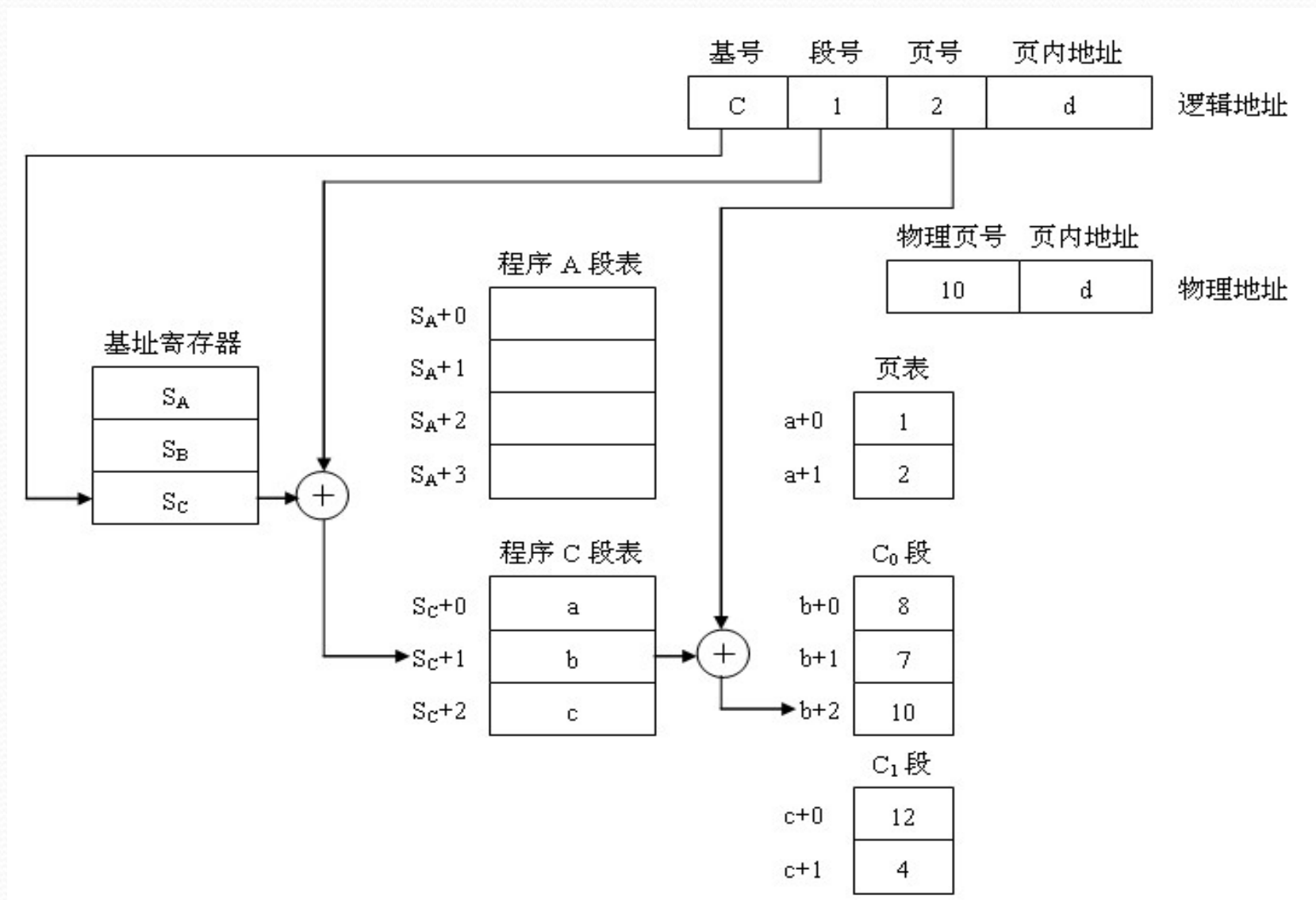
虚拟地址格式为：

基号	段号	页号	页内地址
----	----	----	------

例题

例: 假设有三道程序(用户标志号为A, B, C), 其基址寄存器内容分别为 S_A , S_B , S_C , 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。

例题



例题

解：由段页式虚拟存储器的构成原理可知，段页式虚拟存储器的逻辑地址由基号、段号、页号和页内地址构成。每道程序由若干段组成，而每段又由若干页组成。如程序A由4段组成，程序C由3段组成。每段应有一张页表。

程序C的地址变换过程为：

- (1) 根据基号C执行 S_c+1 操作，得到段表相应行地址，其内容为页表的起始地址b；
- (2) 执行 $b+2$ ，得到物理页号的地址，其内容即为物理页10；
- (3) 物理页号与页内地址拼装，即得物理地址。

可以看出，段页式虚拟存储系统由虚拟地址向主存地址的变换至少需要查两次表。

页面替换算法及其实现

页面替换算法与Cache中的行替换算法的不同点：

- (1) 缺页至少要涉及前一次磁盘存取，读取所缺的页。
- (2) 页面替换是由操作系统软件实现的。
- (3) 页面替换的选择余地很大，属于一个进程的页面都可替换。

虚拟存储器中的替换策略一般采用LRU算法、LFU算法和FIFO算法，或将两种算法结合起来使用。

例题

例：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，②LRU算法，用列表法分别求两种替换策略情况下的命中率。

解：求解过程如下所示：

例题

页面访问序列		0	1	2	4	2	3	0	2	1	3	2	命中率
FIFO 算法	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*	
	注释	调入	调入	调入	替换	命中	替换	替换	替换	替换	替换	命中	
LRU 算法	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*	
	注释	调入	调入	调入	替换	命中	替换	替换	命中	替换	替换	命中	
OPT 算法	a	0	1	2	2	2	2	2	2	2	2	2	5/11= 45.5%
	b		0	1*	4*	4*	3*	3	3	3	3	3	
	c			0	0	0	0	0*	0*	1*	1*	1	
	注释	调入	调入	调入	替换	命中	替换	命中	命中	替换	命中	命中	

小结

1. 存储器的分类
2. 半导体存储器（RAM、ROM）
3. 存储器构成（扩展）
4. 高速缓冲器（原理、映射关系、替换算法）
5. 虚拟存储器（原理、管理方式）

作业：

P.150-152:4.1, 4.11, 4.15, 4.17, 4.18, 4.25, 4.30, 4.32

谢谢！