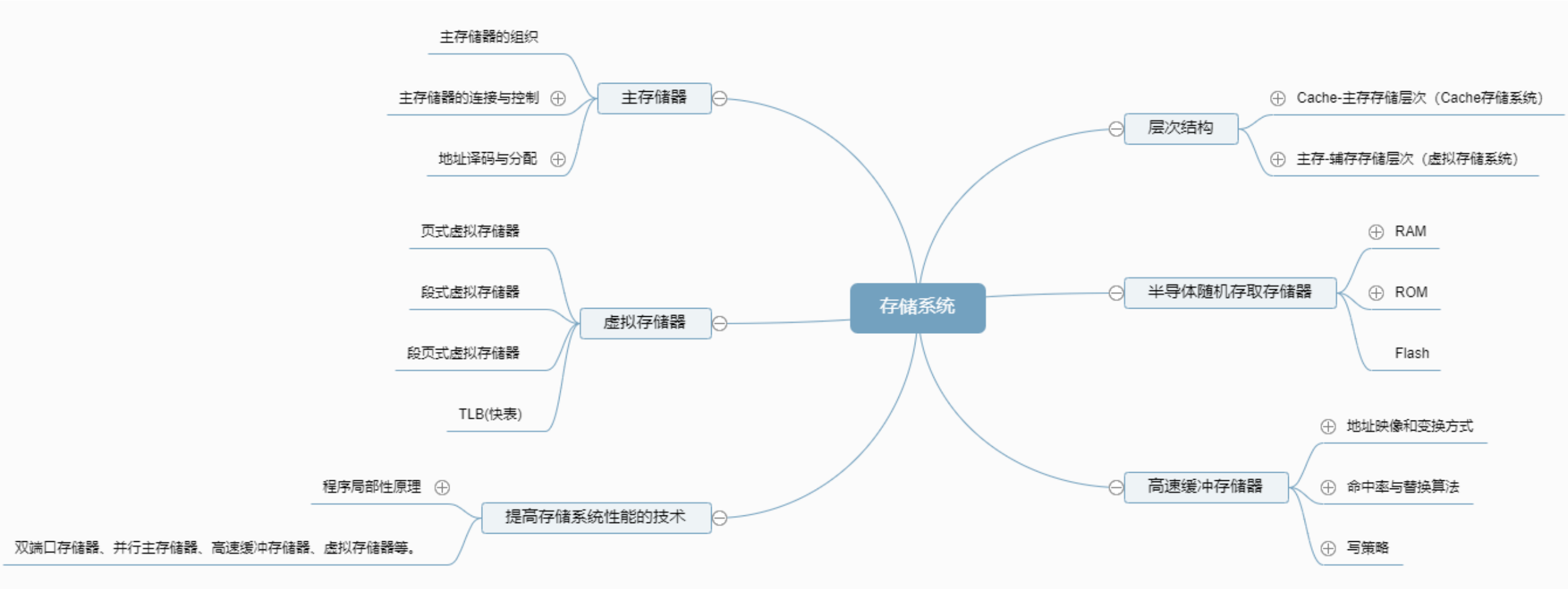
计算机组成与结构-存储系统

人工智能专业

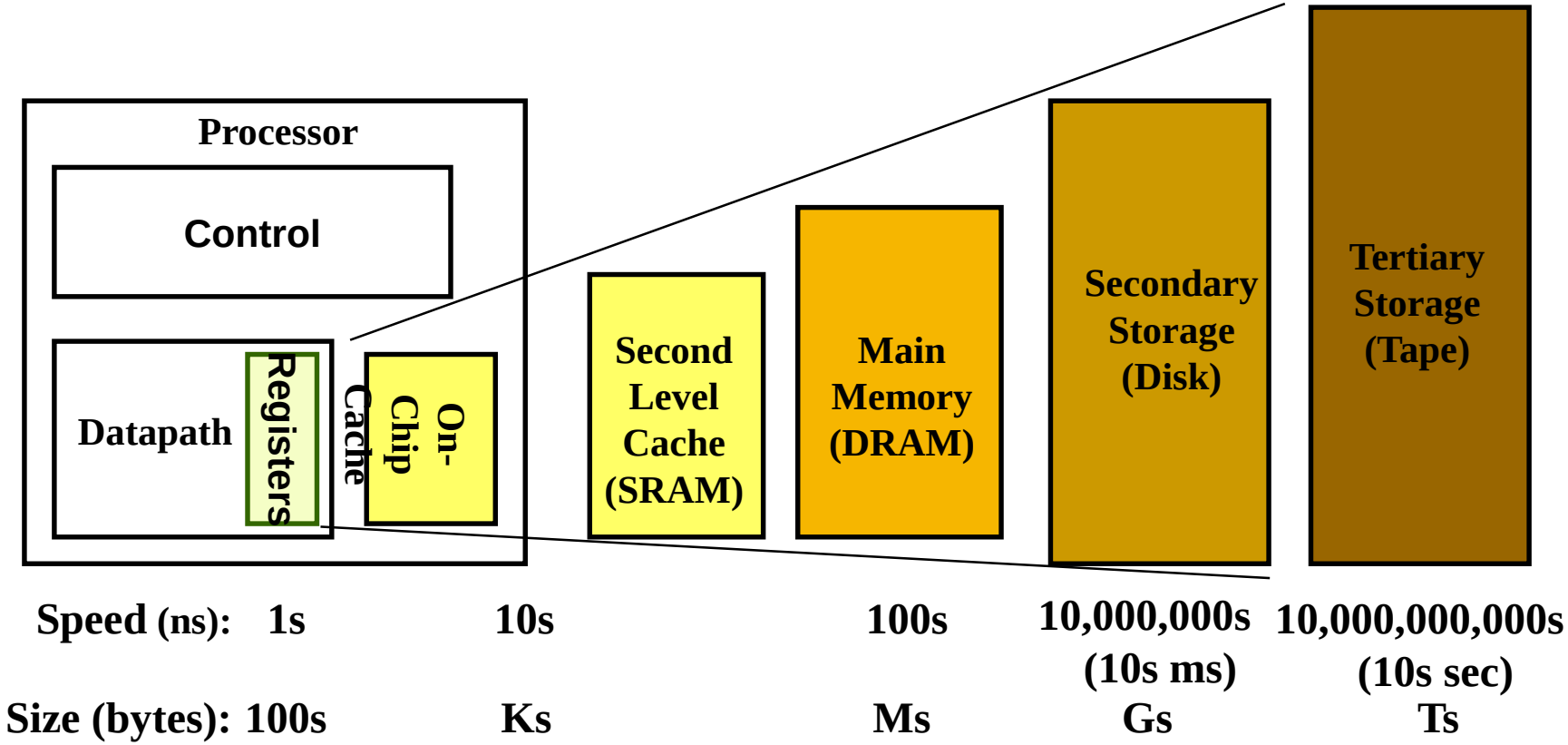
主讲教师：王 娟

存储系统思维导图


存储系统
主存储器
主存储器的组织
主存储器的连接与控制
地址译码与分配
虚拟存储器
页式虚拟存储器
段式虚拟存储器
段页式虚拟存储器
TLB(快表)
提高存储系统性能的技术
程序局部性原理
双端口存储器、并行主存储器、高速缓冲存储器、虚拟存储器等。
层次结构
Cache-主存存储层次（Cache存储系统）
主存-辅存存储层次（虚拟存储系统）
半导体随机存取存储器
RAM
ROM
Flash
高速缓冲存储器
地址映像和变换方式
命中率与替换算法
写策略

存储系统概念与层次化结构

为了解决存储容量、存取速度和价格之间的矛盾，通常把各种不同存储容量、不同存取速度的存储器，按一定的体系结构组织起来，形成一个统一整体的存储系统。



存储访问的局部性原理

指在访问存储器时，无论是存取指令或存取数据所访问的存储单元都趋于聚集在一个较小的连续单元区域中。

时间局部性：最近的访问的存储单元在不久的将来仍将被访问。（顺序存储）

空间局部性：下次访问的存储单元很可能就在刚刚访问的存储单元附近。（程序循环、数组等）

微机中常用三级两层存储系统：

高速缓存和主存间称为Cache-主存存储层次（Cache存储系统）；

主存与辅助存储器称为主存-辅存存储层次（虚拟存储系统）。

Cache 存储系统与虚拟存储系统的比较

存储系统	Cache 存储系统	虚拟存储系统
设计目标	提高速度	扩大容量
实现方法	全部硬件	软件为主，硬件为辅
等效存储容量	主存储器	虚拟存储器
透明性	对系统和应用程序员	仅对应用程序员

存储器的其他分类方式

按存取方式分类

(1)随机存取存储器RAM

CPU可以对RAM单元的内容随机地读写访问。CPU对任何一个存储单元的读写时间是一样的，即**存取时间是相同的**。

(2)只读存储器ROM

ROM可以看作RAM的一种特殊方式，存储器的内容只能随机读出而不能写入。

(3)顺序存取存储器SAM

SAM的内容只能按某种顺序存取，**存取时间与信息在存储体上的物理位置有关**。

(4)直接存取存储器DAM

当要存取所需的信息时，第一步直接指向整个存储器中的某个小区域（如磁盘上的磁道），第二步在小区域内顺序检索或等待，直至找到目的地后再进行读写操作。

按存储介质分类

(1)磁芯存储器

利用两种不同的剩磁状态表示“1”或“0”。磁芯存储器的特点是信息可以长期存储，不会因断电而丢失；但磁芯存储器的读出是破坏性读出，读出后需要重写（再生）。

(2)半导体存储器

采用半导体器件制造的存储器，主要有双极型（TTL电路或ECL电路）存储器和MOS型存储器两大类。

(3)磁表面存储器

在金属或塑料基体上，涂复一层磁性材料，用磁层存储信息，常见的有磁盘、磁带等。

(4)光存储器

采用激光技术控制访问的存储器，如CD-ROM（只读光盘）、WORM（CD-R，写一次多次读光盘）、CD-RW（可读可写光盘）。

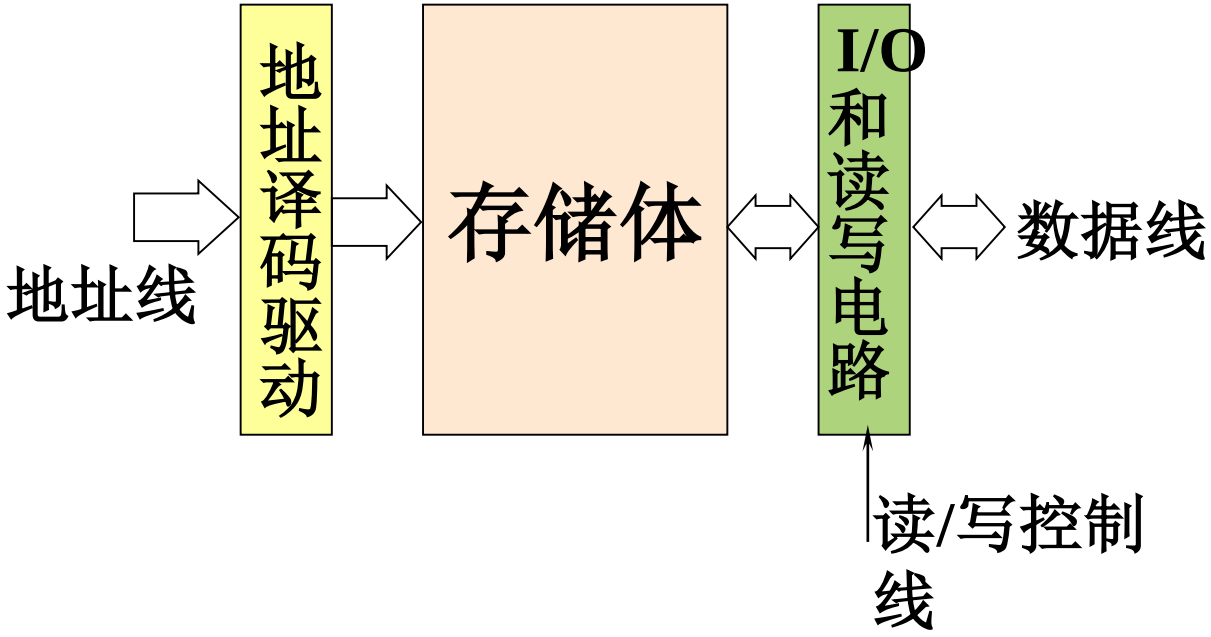
主存储器的组织

主存储器是整个存储系统的核心，它用来存放计算机运行期间所需要的程序和数据，CPU可直接随机地对它进行访问。

存储体：主存储器的核心，程序和数据都存放在存储体中。

地址译码驱动电路：包含译码器和驱动器两部分。

I/O和读写电路：包括读出放大器、写入电路和读/写控制电路。



主存储器的存储单元

位是二进制数的最基本单位，也是存储器存储信息的最小单位。

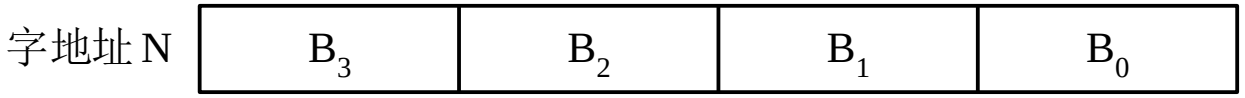
一个二进制数由若干位组成，当这个二进制数作为一个整体存入或取出时，这个数称为**存储字**。

存放存储字或存储字节的主存空间称为**存储单元或主存单元**。**存储单元是CPU对主存可访问操作的最小存储单位。**一个存储单元可能存放一个字，也可能存放一个字节，这是由计算机的结构确定的。对于字节编址的计算机，最小寻址单位是一个字节，相邻的存储单元地址指向相邻的存储字节；对于字编址的计算机，最小寻址单位是一个字，相邻的存储单元地址指向相邻的存储字。

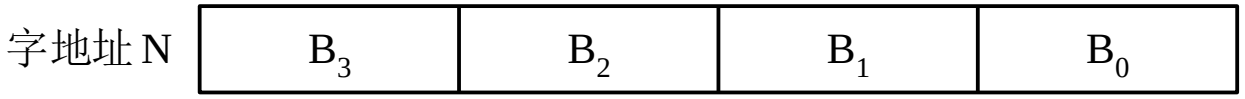
大量存储单元的集合构成一个**存储体**，程序和数据都存放在存储体中，它是存储器的核心。

大端方案和小端方案

假设一个字由四个字节组成，我们使用B_3、B_2、B_1、B_0来分别表示这四个字节，其中B_3是字的最高有效字节，B_0是最低有效字节。字节编址计算机的主存地址安排有两种方案，但字地址总是等于4的整数倍。



(a)



(b)

图 (a)称为小端方案。假设字地址为N，则字节B_3、B_2、B_1、B_0依次存放在地址为N+3、N+2、N+1、N+0的存储单元，即字地址等于最低有效字节地址。采用小端方案的计算机有Intel 80X86、DEC VAX等。

图 (b)称为大端方案。假设字地址为N，则字节B_3、B_2、B_1、B_0依次存放在地址为N+0、N+1、N+2、N+3的存储单元，即字地址等于最高有效字节地址。采用大端方案的计算机有IBM360/370、Motorola 68000等。

数据0x11223344，存在起始地址为00的内存中，如何存?

地址	小端	大端	地址	小端	大端
00	44	11	10	22	33
01	33	22	11	11	44

主存储器的主要技术指标

1.存储容量

存储容量是指主存所能容纳的二进制信息总量。对于字节编址的计算机，以字节数来表示容量；对于字编址的计算机，以字数与其字长的乘积来表示容量。

如某计算机的容量为64K×16，表示它有64K个字，每个字的字长为16位，若用字节数表示，则可记为128K字节（128KB）。

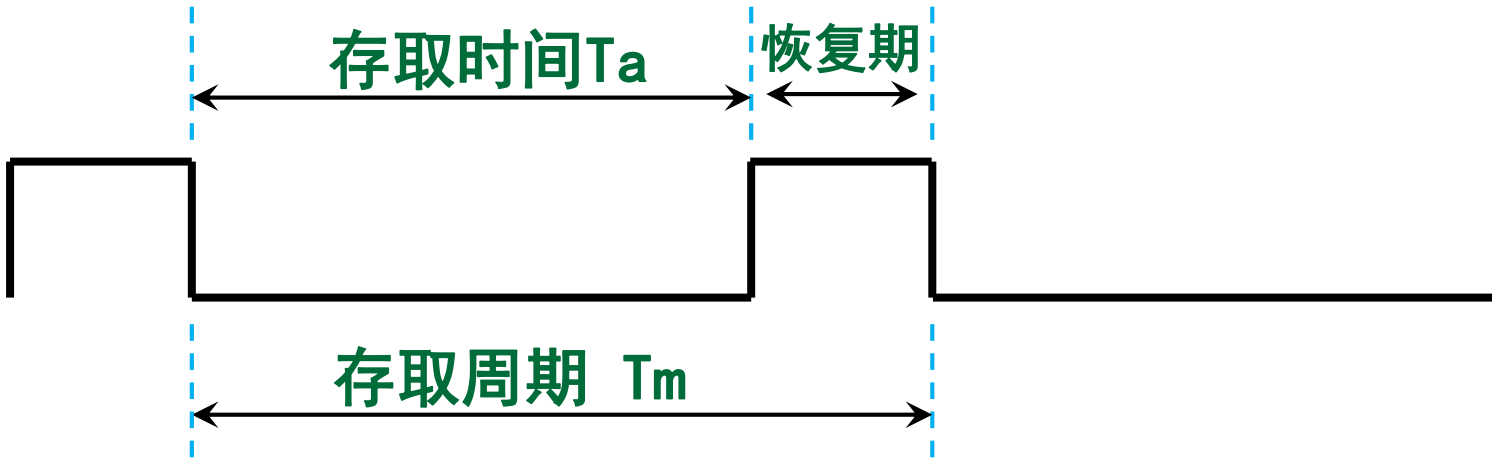
主存储器的主要技术指标

2.存取速度

(1)存取时间T_a：存取时间又称为访问时间或读/写时间，它是指从启动一次存储器操作到完成该操作所经历的时间。

(2)存取周期T_m：存取周期又可称作读写周期、访存周期，它是指存储器进行一次完整的读写操作所需的全部时间，即连续两次访问存储器操作之间所需要的最短时间。

一般情况下，$T_m > T_a$。



主存储器的主要技术指标

(3)主存带宽B_m

与存取周期密切相关的指标是主存的带宽，它又称为数据传输率，表示每秒从主存进出信息的最大数量，单位为字/秒或字节/秒或位/秒。

B_m=主存等效工作频率×主存位宽÷8

=内存时钟频率×倍增系数×主存位数÷ 8。

如何改进?

可以采取的措施有：

·缩短存取周期；

·增加存储字长；

·增加存储体。

以DDR400内存为例，它的运行频率为200MHz，数据总线位数为64bit，由于上升沿和下降沿都传输数据，因此倍增系数为2，此时带宽为：200×2×64/8＝3.2GB/s。

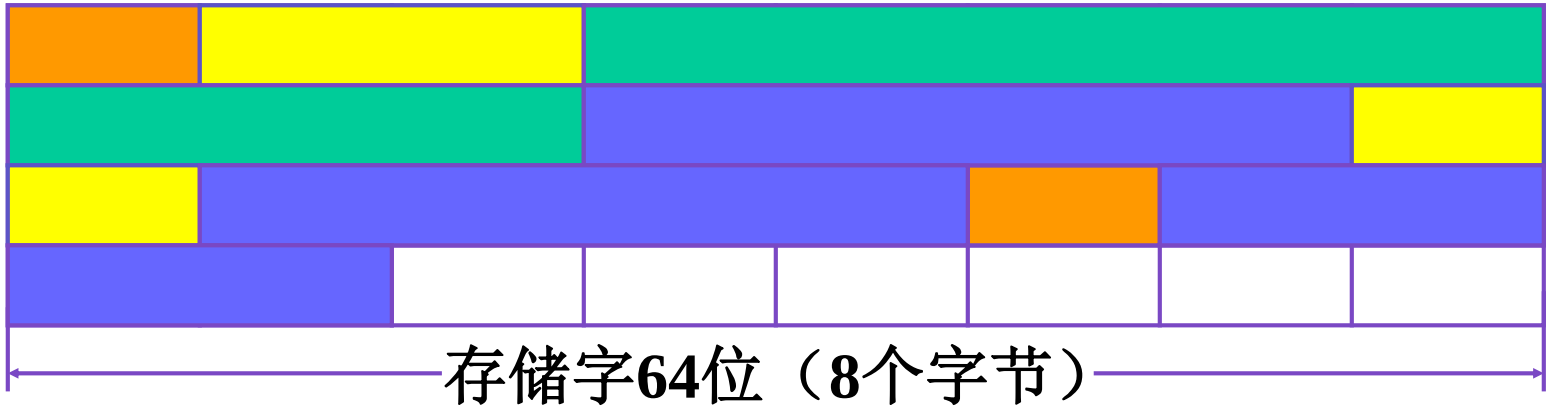
数据在主存中的存放

在采用字节编址的情况下，数据在主存储器中的三种不同存放方法。假设，存储字为64位（8个字节），读/写的数据有四种不同长度，它们分别是字节（8位）、半字（16位）、单字（32位）和双字（64位）。请注意：此例中数据字长（32位）不等于存储字长（64位）。

现有一批数据，它们依次为：字节、半字、双字、单字、半字、单字、字节、单字。

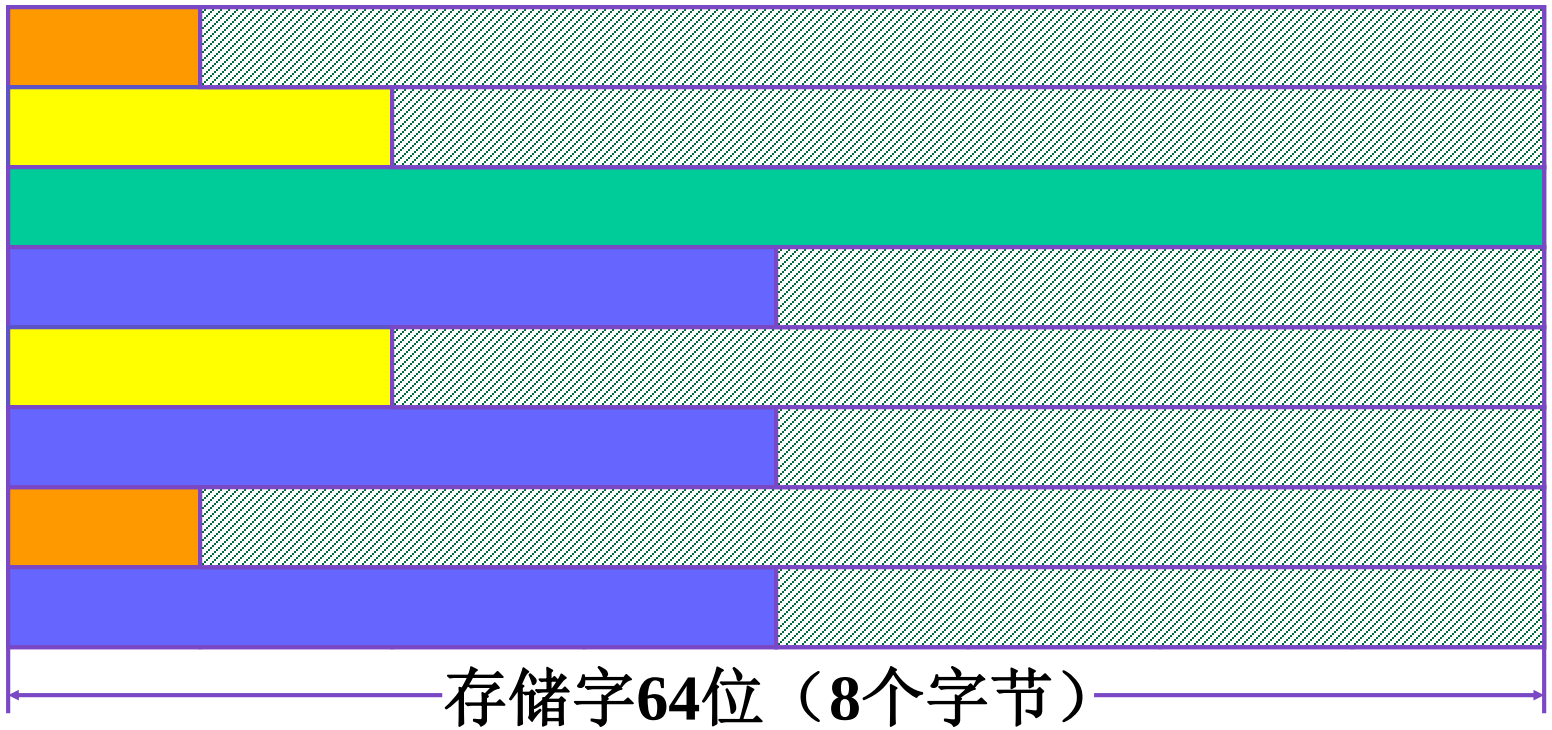
(1)不浪费存储器资源的存放方法

四种不同长度的数据一个紧接着一个存放。优点是不浪费宝贵的主存资源，但存在的问题是：当访问的一个双字、单字或半字跨越两个存储字时，存储器的工作速度降低了一倍，而且读写控制比较复杂。



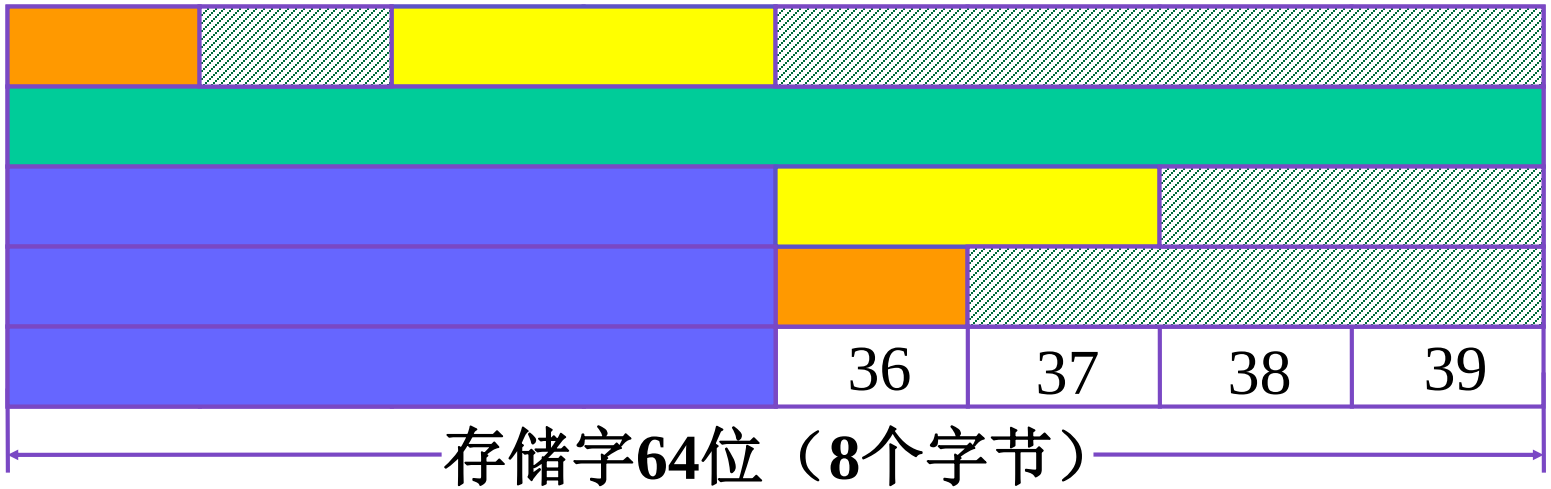
(2)从存储字的起始位置开始存放方法

无论要存放的是字节、半字、单字或双字，都必须从存储字的起始位置开始存放，而空余部分浪费不用。优点是：无论访问一个字节、半字、单字或双字都可以在一个存取周期内完成，读写数据的控制比较简单。缺点是：浪费了宝贵的存储器资源。


存储字64位（8个字节）

(3)边界对齐的数据存放方法

双字地址的最末三个二进制位必须为000，单字地址的最末两位必须为00，半字地址的最末一位必须为0。它能够保证无论访问双字、单字、半字或字节，都在一个存取周期内完成，尽管存储器资源仍然有浪费，但是浪费比第(2)种存放方法要少得多。



半导体随机存储器和只读存储器

主存储器通常分为RAM和ROM两大部分。RAM可读可写，ROM只能读不能写。MOS型存储器根据记忆单元的结构又可分为静态RAM和动态RAM两种。

静态RAM:

SRAM（Static RAM），其存储电路以双稳态触发器为基础；

功耗较大，速度快，常用作Cache。

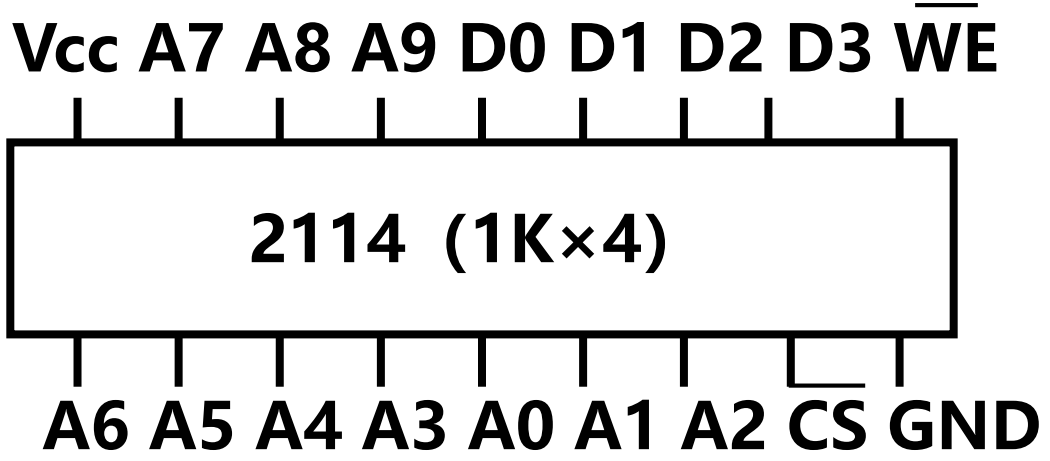
动态RAM:

DRAM（Dynamic RAM），其存储电路以电容为基础。

功耗较小，容量大，速度较快，常用作主存。

SRAM芯片Intel 2114 （1K×4位）

外特性

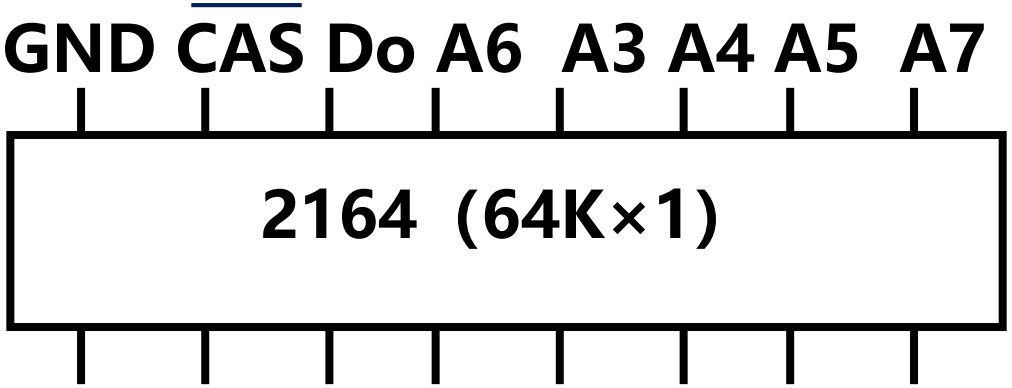


地址： A9～A0；数据： D3～D0（双向输入/输出）

控制端：
- 片选$\overline{CS}$ = 0 选中芯片；= 1 未选中芯片
- 写使能$\overline{WE}$ = 0 写；= 1 读

V_{CC}：电源，GND：接地

DRAM芯片2164（64K×1）



问题：地址增加1位，容量增加多少?

数据线：Di(输入),Do(输出)

地址线：A7～A0（分时复用，可提供16位地址)

行选$\overline{RAS}$=0时：A_7～A_0为行地址(即高8位)

列选$\overline{CAS}$=0时：A_7～A_0为列地址(即低8位)

控制线：写使能信号($\overline{WE}$)：0-写入；1-读出；

动态刷新：行选信号送达，即可实现自动刷新；

半导体只读存储器（ROM）

ROM的最大优点是具有非易失性，即使电源断电，ROM中存储的信息也不会丢失。

根据编程方法的不同，ROM通常可以分为以下几类：

(1)掩膜式ROM（MROM）：字符点阵存储器、微程序存储器等。

(2)一次可编程ROM（PROM）：结破坏型和熔丝型，可编程逻辑阵列等。

(3)可擦除可编程ROM（EPROM）：EEPROM和UVEPROM，专用编程器。

(4)闪速存储器（FLASH）： 在线可擦除，U盘、SSD固态硬盘等。

DRAM的刷新

DRAM为了维持MOS型动态记忆单元的存储信息，每隔一定时间必须对存储体中的所有记忆单元的栅极电容补充电荷，这个过程就是刷新。

刷新和重写（再生）是两个完全不同的概念。重写是随机的，某个存储单元只有在破坏性读出之后才需要重写。而刷新是定时的，即使许多记忆单元长期未被访问，若不及时补充电荷的话，信息也会丢失。重写一般是按存储单元进行的，而刷新通常以存储体矩阵中的一行为单位进行的。

刷新按行进行，常见的刷新方式有集中式、分散式和异步式三种。

(1)集中刷新方式：在允许的最大刷新间隔内，按照存储芯片容量的大小集中安排若干个刷新周期，刷新时停止读写操作。

刷新时间=存储体矩阵行数×刷新周期

这里刷新周期是指刷新一行所需要的时间，由于刷新过程就是“假读”的过程，所以刷新周期就等于存取周期。

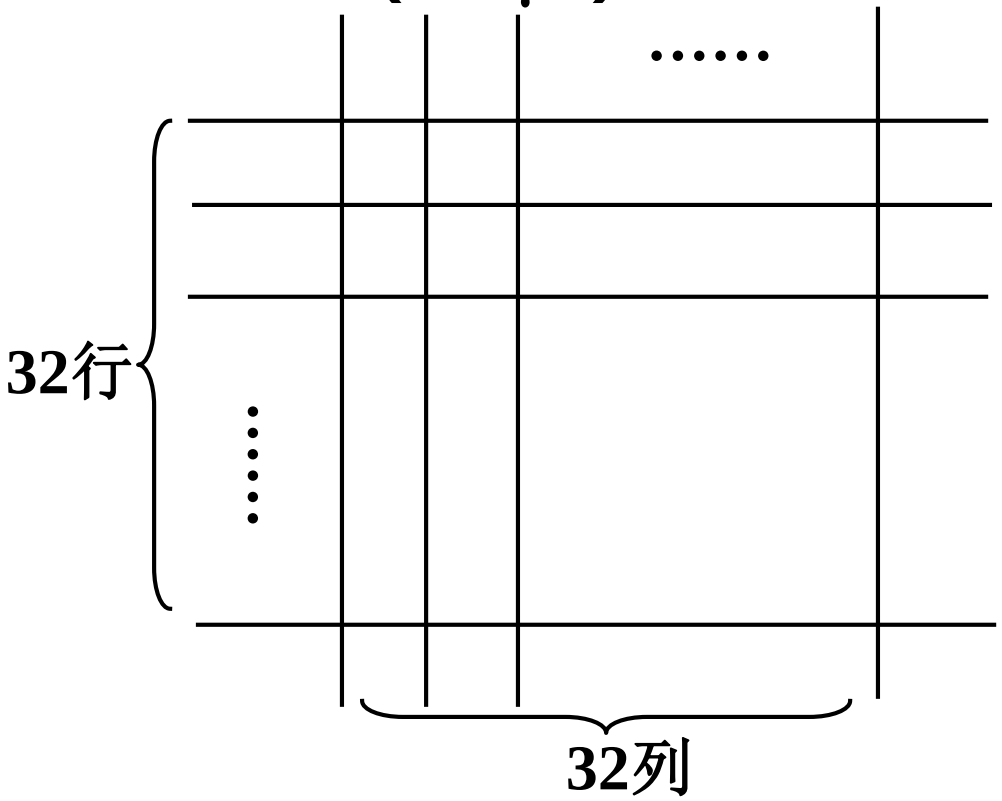
常见的刷新方式

(2)分散刷新方式

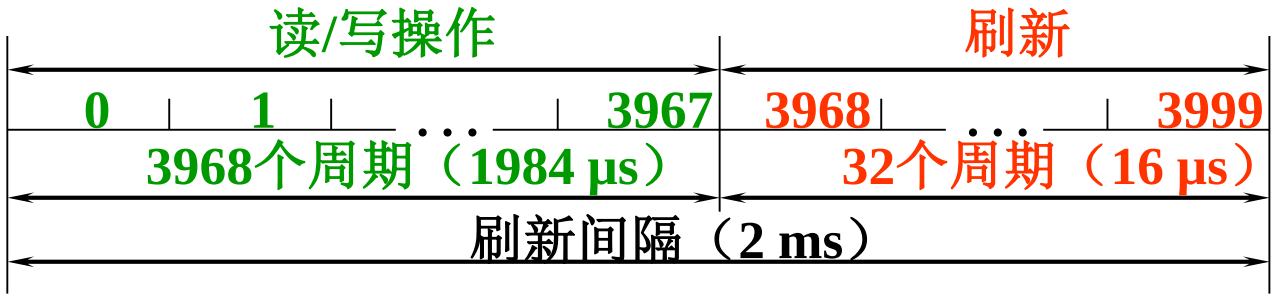
分散刷新是指把刷新操作分散到每个存取周期内进行，此时系统的存取周期被分为两部分，前一部分时间进行读/写操作或保持，后一部分时间进行刷新操作。一个系统存取周期内刷新存储矩阵中的一行。

- (3)异步刷新方式
- 异步刷新方式可以看成前述两种方式的结合，它充分利用了最大刷新间隔时间，把刷新操作平均分配到整个最大刷新间隔时间内进行，故有：
- 相邻两行的刷新间隔＝最大刷新间隔时间/行数
- 现有DDR系列内存最大刷新间隔为64ms。

假定：对具有1024个记忆单元（排列成32×32矩阵）的存储芯片进行刷新，最大刷新间隔为2ms，每刷新一行占用一个存取周期，存取周期为500ns（0.5 μs）。

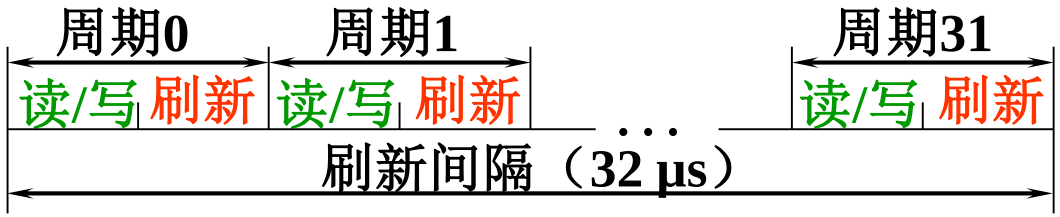


集中刷新方式：在最大刷新间隔2ms内共可以安排4000个存取周期。



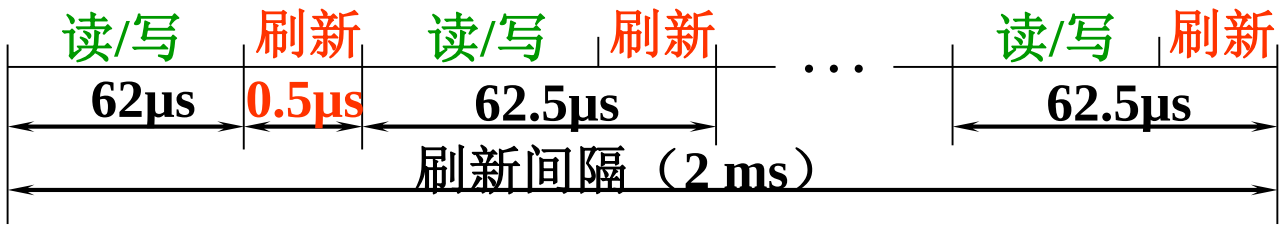
红色部分为：死区，此时CPU不能访存。

分散刷新方式：



分散刷新方式没有死区。但是，它也有很明显的缺点，第一是加长了系统的存取周期，如存储芯片的存取周期为0.5 μs，则系统的存取周期应为1 μs，降低了整机的速度；第二是刷新过于频繁（本例中每32 μs就重复刷新一遍），尤其是当存储容量比较小的情况下，没有充分利用所允许的最大刷新间隔（2ms）。

异步的刷新方式：



对于32×32矩阵，在2ms内需要将32行刷新一遍，所以相邻两行的刷新时间间隔=2ms/32=62.5 μs，即每隔62.5 μs安排一个刷新周期，在刷新时封锁读/写。

异步刷新方式虽然也有死区，但比集中刷新方式的死区小得多，仅为0.5 μs。这样可以避免使CPU连续等待过长的时间，而且减少了刷新次数，是比较实用的一种刷新方式。

MOS型DRAM刷新控制要点

① 刷新对CPU是透明的。

② 刷新通常是一行一行地进行的，每一行中各记忆单元同时被刷新，故刷新操作时仅需要行地址，不需要列地址。

③ 刷新操作类似于读出操作。

④ 因为所有芯片同时被刷新，所以在考虑刷新问题时，应当从单个芯片的存储容量着手，而不是从整个存储器的容量着手。

主存设计

设计一个主存要解决三个问题：**芯片的选用、片内地址分配与片选逻辑、信号线的连接。**

已知存储器所要求的容量和选定的存储芯片的容量，则可以计算出总的芯片数，即

$$\text{总片数} = \frac{\text{总容量}}{\text{芯片容量}}$$

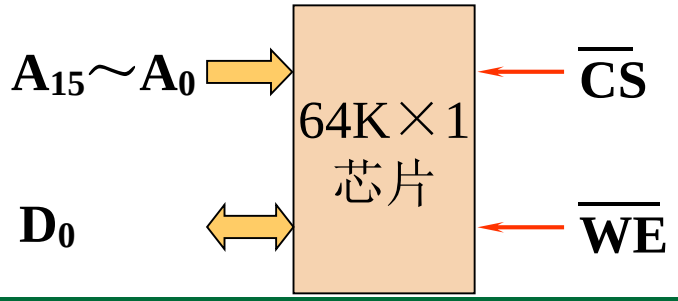
注意：不同类型的芯片分开计算，如RAM和ROM芯片。

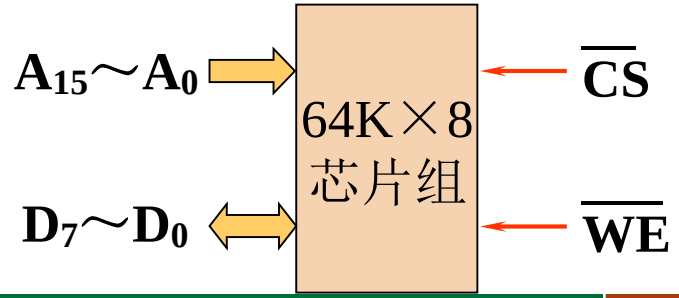
位扩展

位扩展指只**在位数方向扩展**（加大字长），而芯片的字数和存储器的字数是一致的。位扩展的连接方式是将各存储芯片的地址线、片选线和读/写线相应地并联起来，而将各芯片的数据线单独列出。

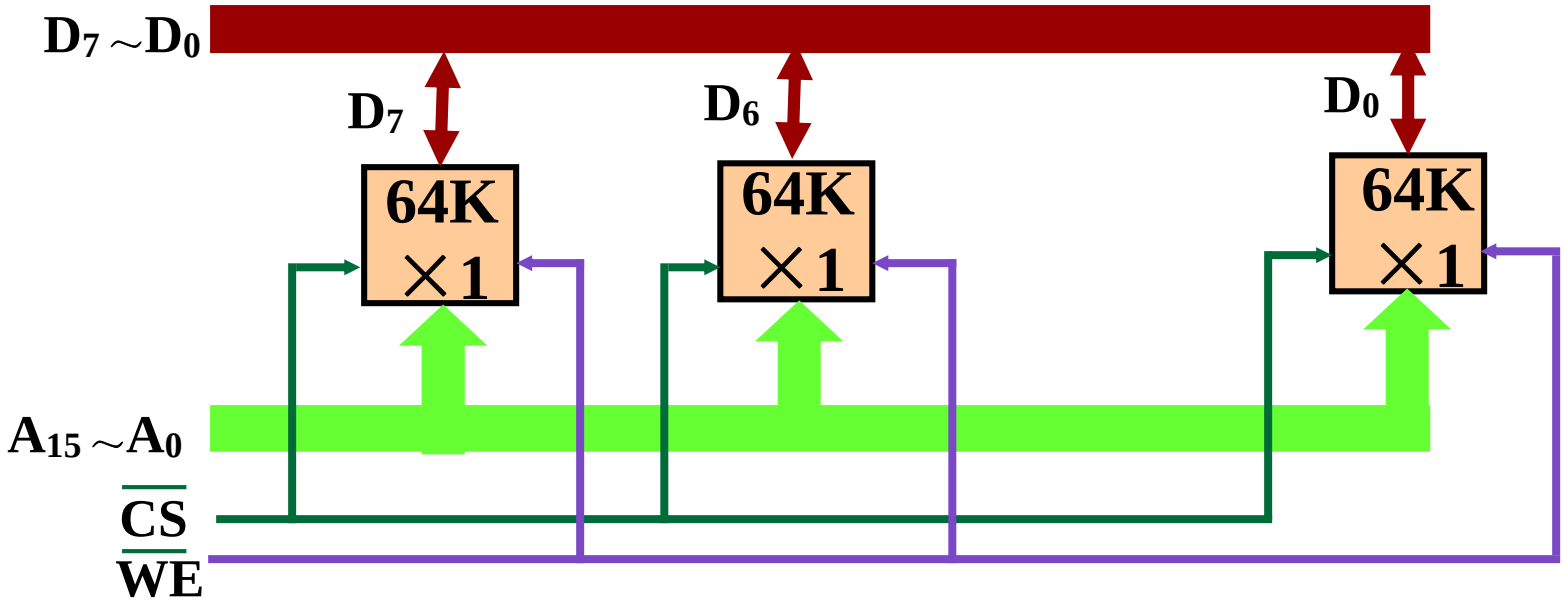
如用64K×1的SRAM芯片组成64K×8的存储器，需要8个芯片。

	容量	地址	数据
存储器	64K×8	16	8
存储芯片	64K×1	16	1





位扩展



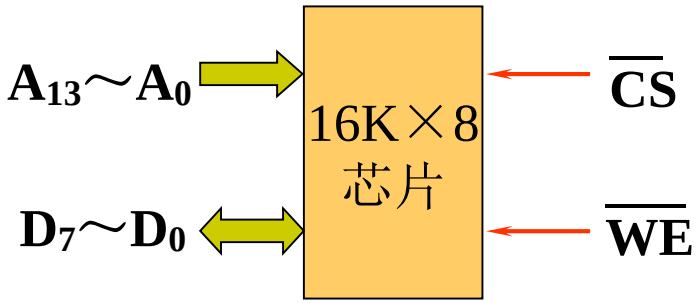
当CPU访问该存储器时，其发出的地址和控制信号同时传给8个芯片，选中每个芯片的同一单元，其单元的内容被同时读至数据总线的相应位，或将数据总线上的内容分别同时写入相应单元。

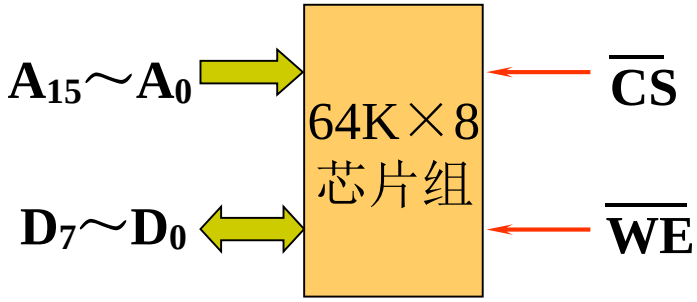
字扩展

字扩展是指仅在字数方向扩展，而位数不变。字扩展将芯片的地址线、数据线、读/写线并联，由片选信号来区分各个芯片。

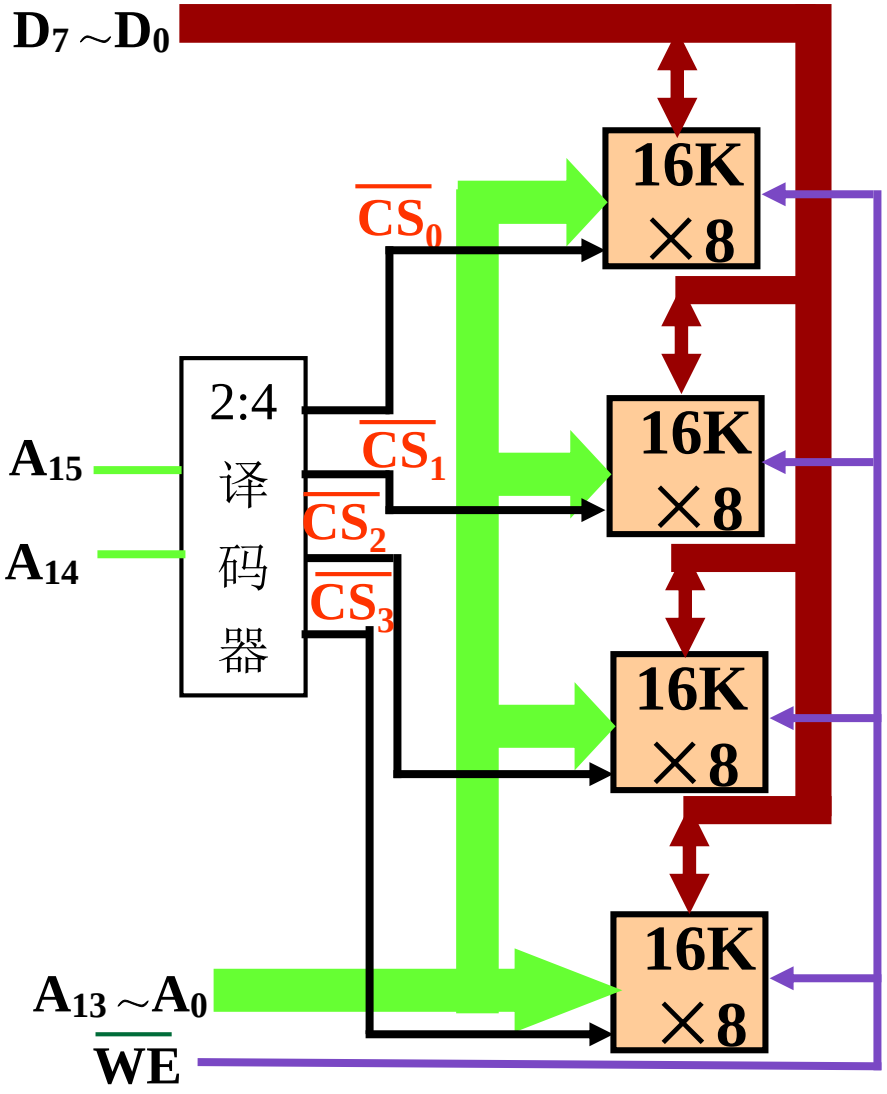
如用16K×8的SRAM组成64K×8的存储器，需要4个芯片。

	容量	地址	数据
存储器	64K×8	16	8
存储芯片	16K×8	14	8





字扩展



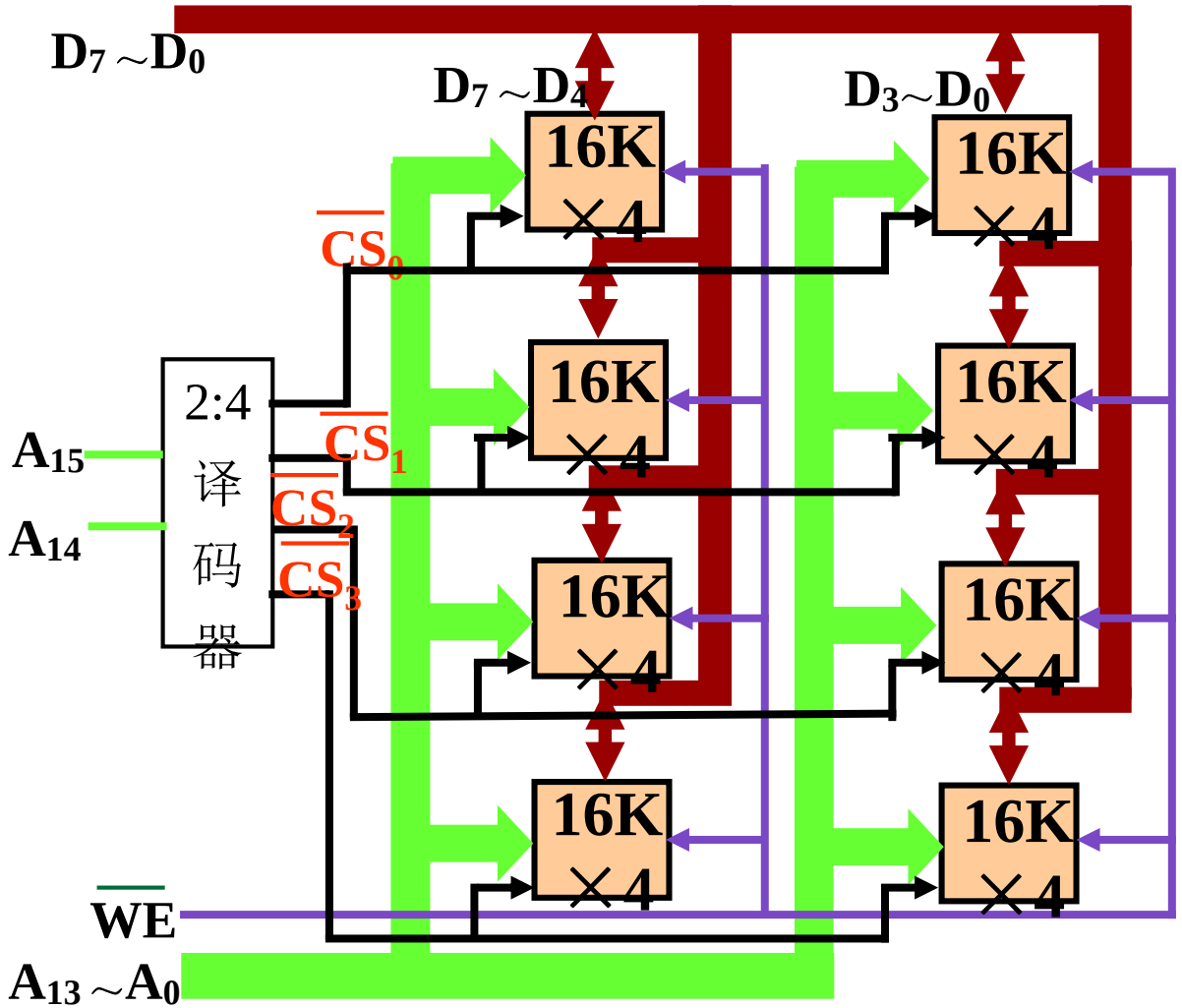
地址分配：
在同一时间内四个芯片中只能有一个芯片被选中。

第一片	最低地址	0000H
	最高地址	3FFFH
第二片	最低地址	4000H
	最高地址	7FFFH
第三片	最低地址	8000H
	最高地址	BFFFH
第四片	最低地址	C000H
	最高地址	FFFFH

字和位同时扩展

当构成一个容量较大的存储器时，往往需要**在字数方向和位数方向上同时扩展**，这将是前两种扩展的组合，实现起来也是很容易的。

如用16K×4的SRAM组成64K×8的存储器，需要8个芯片。



	容量	地址	数据
存储器	64K×8	16	8
存储芯片	16K×4	14	4

地址译码方式与片选

CPU要实现对存储单元的访问，首先要选择存储芯片，即进行片选；然后再从选中的芯片中依地址码选择出相应的存储单元，以进行数据的存取，这称为字选。片内的字选是由CPU送出的N条低位地址线完成的，地址线直接接到所有存储芯片的地址输入端（N由片内存储容量2^N 决定），而片选信号则是通过高位地址得到的。实现片选的方法可分为3种：即线选法、全译码法和部分译码法。

地址译码方式与片选

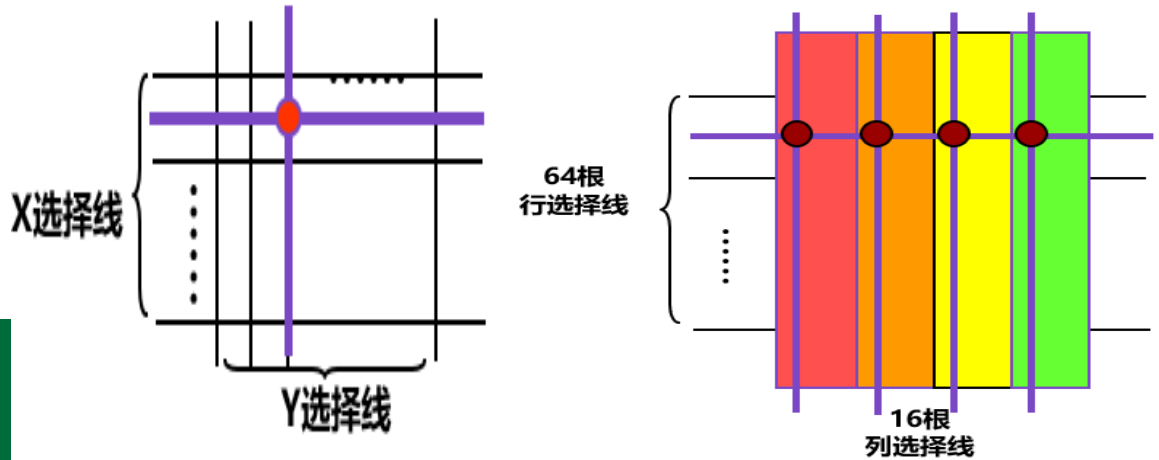
地址译码电路能把地址线送来的地址信号翻译成对应存储单元的选择信号。

(1)单译码方式，又称字选法，不常用，自己了解。

(2)双译码方式

双译码方式又称为重合法。通常是把K位地址码分成接近相等的两段，一段用于水平方向作X地址线，供X地址译码器译码；一段用于垂直方向作Y地址线，供Y地址译码器译码。X和Y两个方向的选择线在存储体内部的一个记忆单元上交叉，以选择相应的记忆单元。

双译码方式对应的存储芯片结构可以是位结构的，则在Z方向上重叠b个芯片。也可以是字段结构的。



片选信号

1.线选法

线选法就是用除片内寻址外的高位地址线直接（或经反相器）分别接至各个存储芯片的片选端，当某地址线信息为“0”时，就选中与之对应的存储芯片。请注意，这些片选地址线每次寻址时只能有一位有效，不允许同时有多位有效，这样才能保证每次只选中一个芯片（或组）。

假定CPU地址线为20位，选用2k×8的芯片构成8k×8的存储器。

芯片	A_{14}~A_{11}	A_{10}~A_0	地址范围
0#	1 1 1 0	00...0~ 11...1	7000~ 77FFH
1#	1 1 0 1	00...0~ 11...1	6800~ 6FFFH
2#	1 0 1 1	00...0~ 11...1	5800~ 5FFFH
3#	0 1 1 1	00...0~ 11...1	3800~ 3FFFH

片选信号

2.全译码法

全译码法将片内寻址外的全部高位地址线作为地址译码器的输入，把经译码器译码后的输出作为各芯片的片选信号，将它们分别接到存储芯片的片选端，以实现对存储芯片的选择。

全译码法的优点是每片（或组）芯片的地址范围是唯一确定的，而且是连续的，也便于扩展，不会产生地址重叠的存储区，但全译码法对译码电路要求较高。

片选信号

芯片	A_{19}～A_{13}	A_{12} A_{11}	A_{10}～A_0	地址范围
0#	X … X	0 0	00…0	00000～
			11…1	007FFH
1#	X … X	0 1	00…0	00800～
			11…1	00FFFH
2#	X … X	1 0	00…0	01000～
			11…1	017FFH
3#	X … X	1 1	00…0	01800～
			11…1	01FFFH

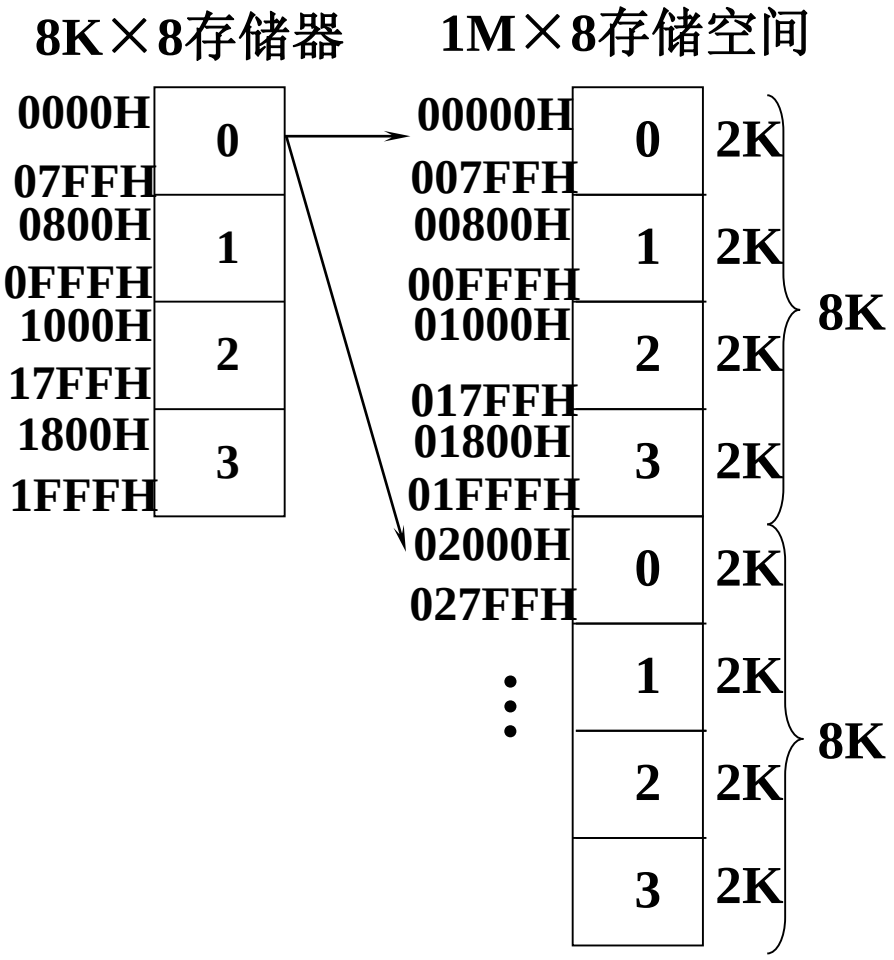
3.部分译码

所谓部分译码即用片内寻址外的高位地址的一部分来译码产生片选信号。

片选信号

从地址分布来看，这8KB存储器实际上占用了CPU全部的空间（1MB）。每片2K×8的存储芯片有1/4M=256K的地址重叠区。

令未用到的高位地址全为0，这样确定的存储器地址称为基本地址，本例中8K×8存储器的基本地址即00000H～01FFFH。



提高存储系统性能的技术

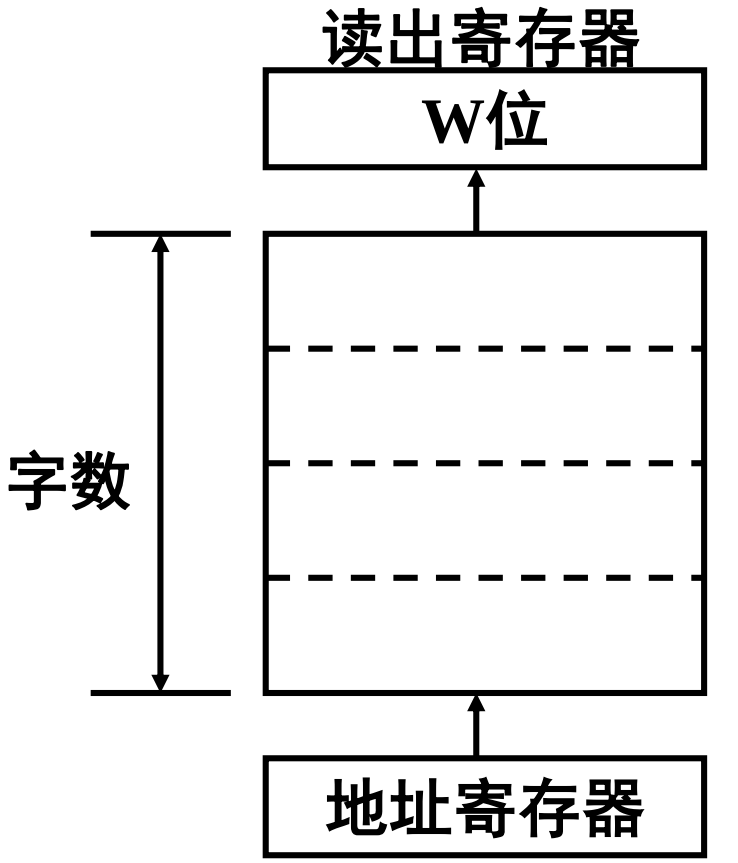
速度和容量是存储器的两大主要课题，计算机的发展不断地对存储器提出更高速度和更大容量的要求。在单机系统中，除了采用更高速度的元器件外，还能够提高存储器性能的主要技术有：双端口存储器、并行主存储器、高速缓冲存储器、虚拟存储器等。

双端口存储器：同一个存储器具有两组独立的读写控制线路，两个端口分别具有各自的地址线、数据线和控制线，可进行独立的存取操作。

当两个端口同时存取同一存储单元时，会发生端口间的读写冲突，一般需要进行仲裁。

并行主存储器

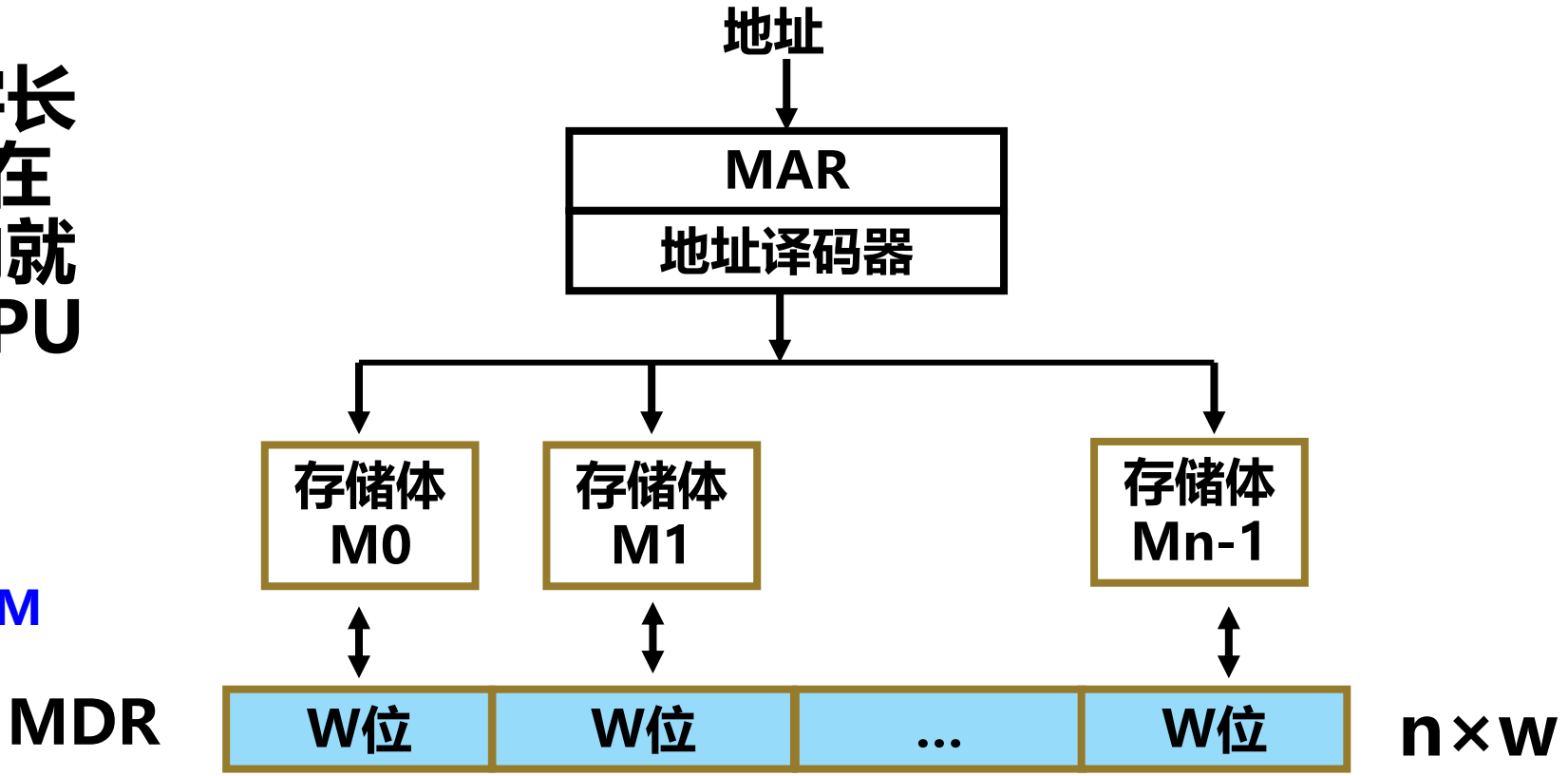
- **并行主存储器，即多体交叉存贮器，目标是在位价格基本不变的情况下，使主存的频宽得到较大的提高。**
- **单体单字存贮器**
  - **字长为W位的存贮器，一次可访问一个存贮器字。若存贮器字长与CPU字长相等，其最大频宽为：$B_M = W / T_M$**
  - **要提高频宽，只有设法提高存贮器字长 W 才行。有三种方案：**
    - **单体多字存贮器**
    - **多体单字交叉存贮器**
    - **多体多字交叉存贮器**



单体单字存贮器

单体多字存贮器

- 增加存贮器的字长（n倍），这样在一个主存周期内就可以读出多个CPU字。
- 其最大频宽为：
- $B_M = n \times W / T_M$



特点：适合指令和数据连续存放的情况。
访存冲突大。

多体单字交叉访问存储器

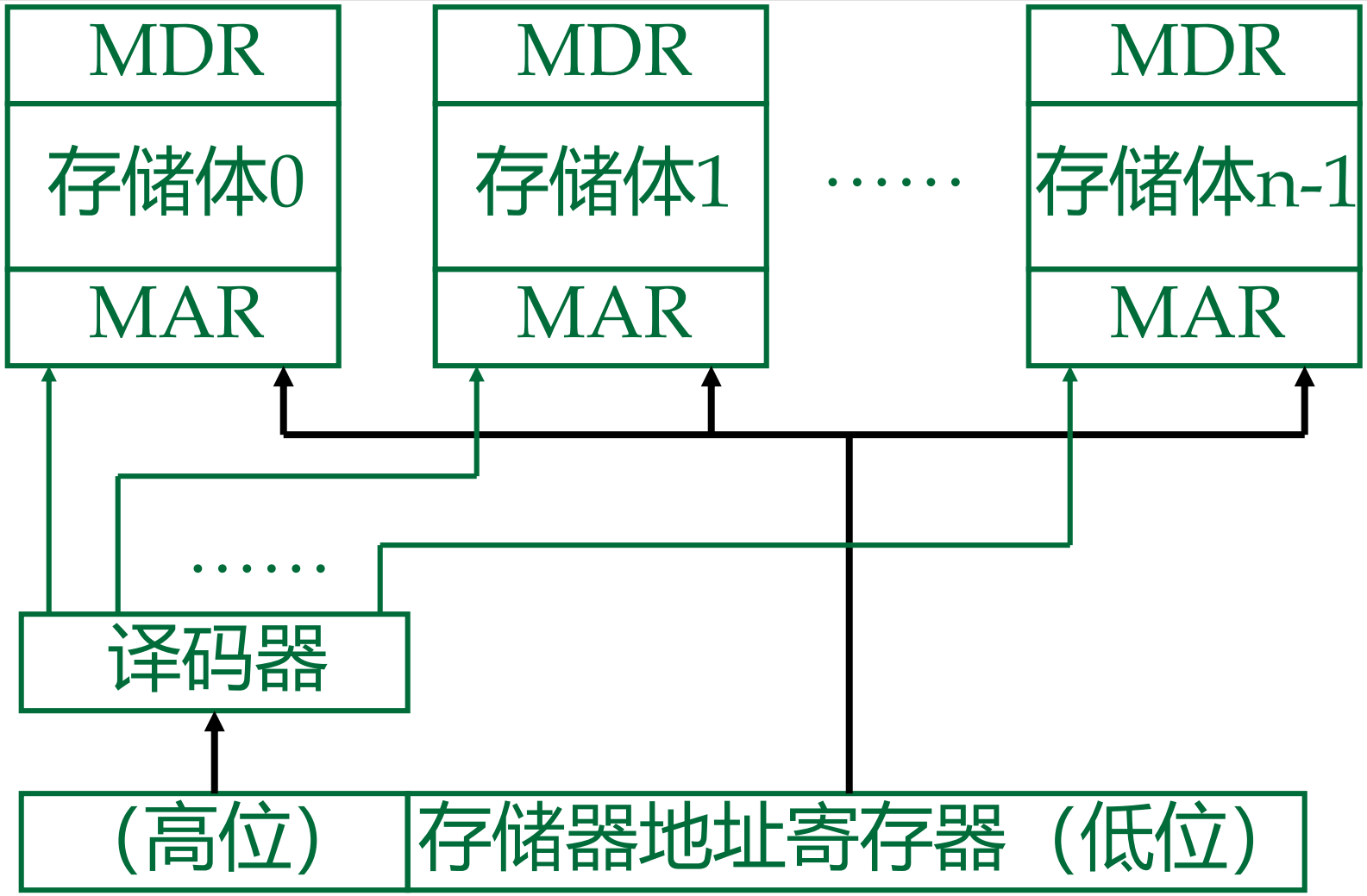
由多个容量较小、字长较短的相同存贮器芯片组成。每个芯片都有自己的地址译码、读/写驱动等外围电路。每个存贮体字长都是一个CPU字的宽度。各个存储体能并行工作，又能交叉工作。

多体交叉又分高位交叉和低位交叉两种。

- 高位交叉
  - 实现方法：用地址码的高位部分区分存储体号
  - 主要目的：扩大存储器容量
- 低位交叉
  - 实现方法：用地址码的低位部分区分存储体号
  - 主要目的：提高存储器访问速度

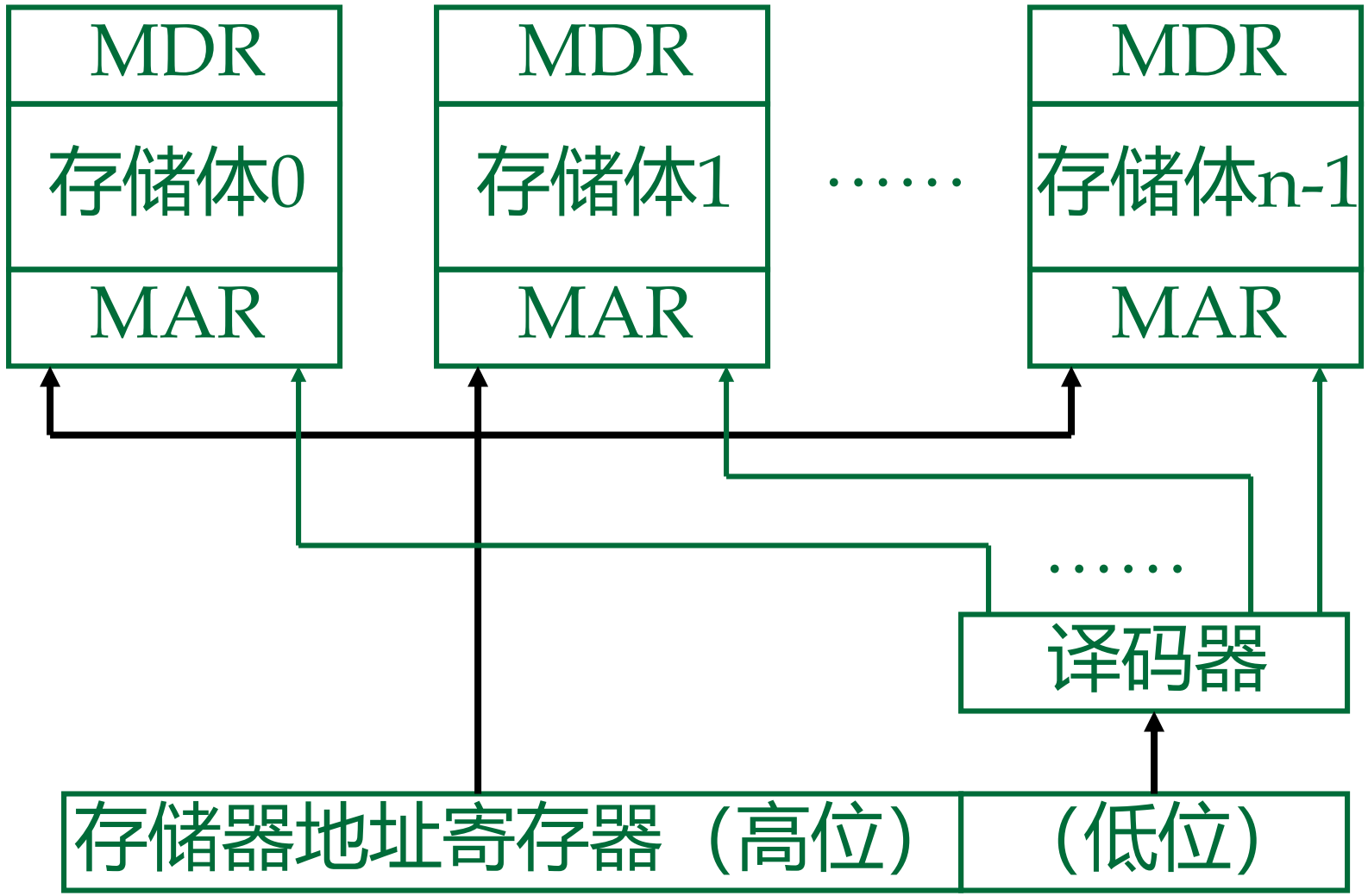
高位交叉

每个存储模块都有各自独立的控制部件，每个存储模块可以独立工作。但由于程序的局部性原理，通常只有一个存储模块在不停地忙碌，其他存储模块是空闲的。



低位交叉

n个存储体分时启动。本质上是一种采用流水线方式工作的并行存储器。理论上，存储器的速度可望提高n倍。

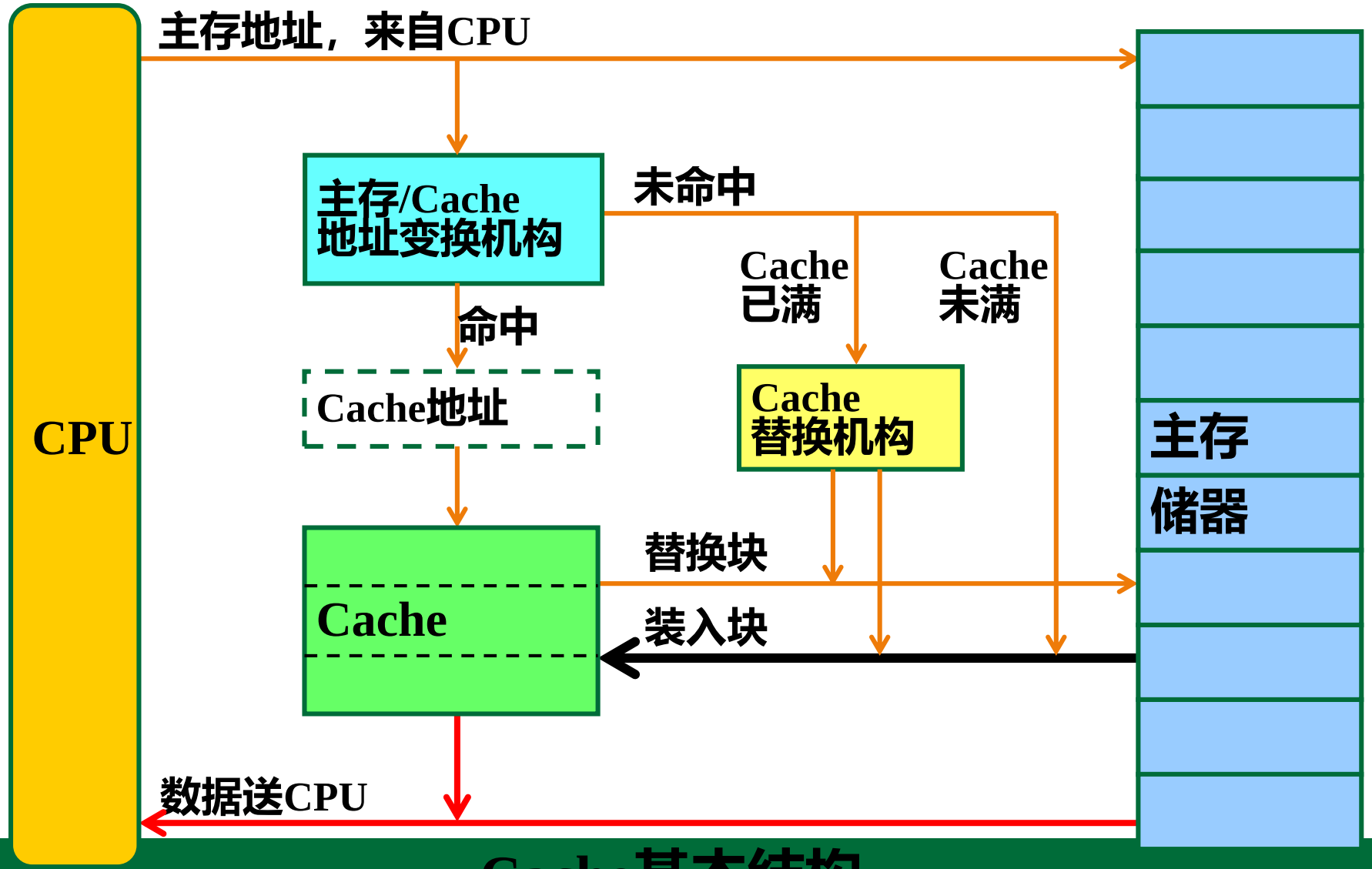


模体	地址编址序列	对应二进制地址最末两位的状态
M_0	0，4，8，…，4i+0，…	00
M_1	1，5，9，…，4i+1，…	01
M_2	2，5，10，…，4i+2，…	10
M_3	3，7，11，…，4i+3，…	11

地址的模4低位交叉编址　　横向编址，地址不连续。

多体多字交叉存贮器：将多体单字存取与单体多字存取结合，进一步提高了带宽。

5.7 Cache基本结构和工作原理



Cache基本结构

Cache基本结构和工作原理

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Cache工作原理：读取

将CPU给出的主存地址
变换为Cache地址，搜索Cache

在Cache中找到
（命中）

访问Cache，
向CPU返回
Cache中的数据副本

只要Cache的命中率足够高，就能以接近于Cache的速度访问主存。

在Cache中未找到
（未命中）

1. 从主存中读取数据块
2. 等待…
3. 向CPU返回数据，
   更新Cache（满时替换）

Cache基本结构和工作原理

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Cache工作原理：写入

将CPU给出的主存地址
变换为Cache地址，搜索Cache

在Cache中找到
（命中）

在Cache中未找到
（未命中）

写Cache，写主存
（存在一致性问题）

写主存
（与Cache无关）

为保持Cache与主存中的内容一致，
可以采取以下方法：

- 写直达法
- 写回法

Cache系统须解决三个问题

1. 定位问题

- 将主存中的数据装入Cache的哪个位置；
- 如何知道主存中的数据已经装入Cache（即是否命中）；
- 如果命中，如何形成Cache地址并访问主存；
- 如何提高地址变换速度。

2. 替换问题

- 若未命中或失效，需将数据从主存调入Cache；
- 若Cache满，则按何种算法将Cache中的数据替换出去。

3. 数据一致性问题

- 如何保证Cache内容与主存内容的一致。

地址映像规则与地址变换

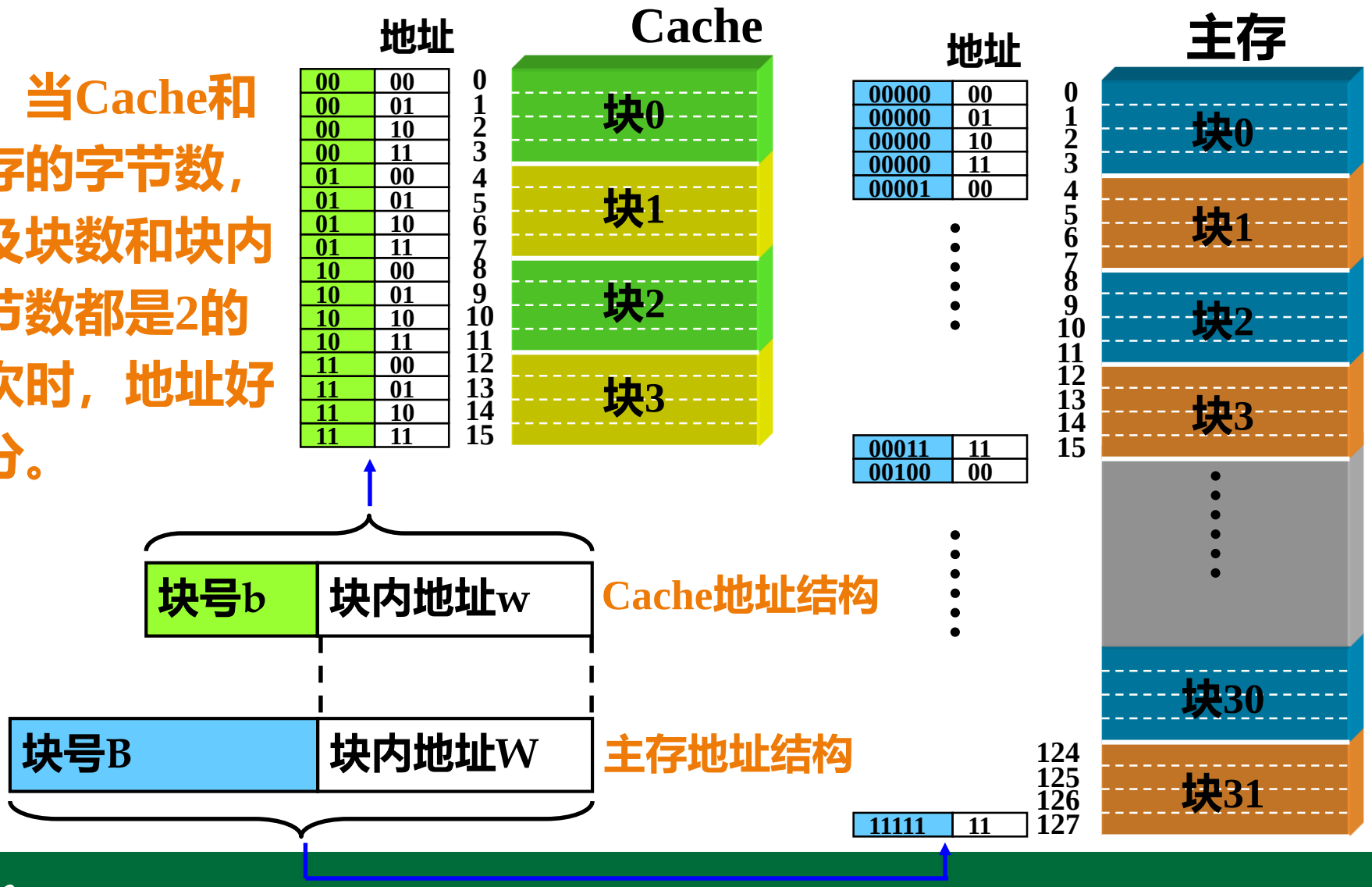
地址映像：

把主存中的数据按照某种规则装入Cache中，并建立主存地址与Cache地址之间的对应关系，进而根据主存地址，判断Cache有无命中并变换为Cache的地址。

为便于进行地址的映象和变换，也便于替换和管理，把Cache和主存等分成相同大小的块，这样，Cache—主存地址映像就演变为主存中的块如何与Cache中的块相对应。

地址映像规则与地址变换

当Cache和主存的字节数，以及块数和块内字节数都是2的幂次时，地址好划分。



地址映像规则与地址变换

可以采用的地址映像方法有很多。选择依据：

- 地址映像和变换硬件的速度是否高，价格是否低，实现是否容易；
- Cache空间的利用率是否高；
- 块冲突概率是否低；

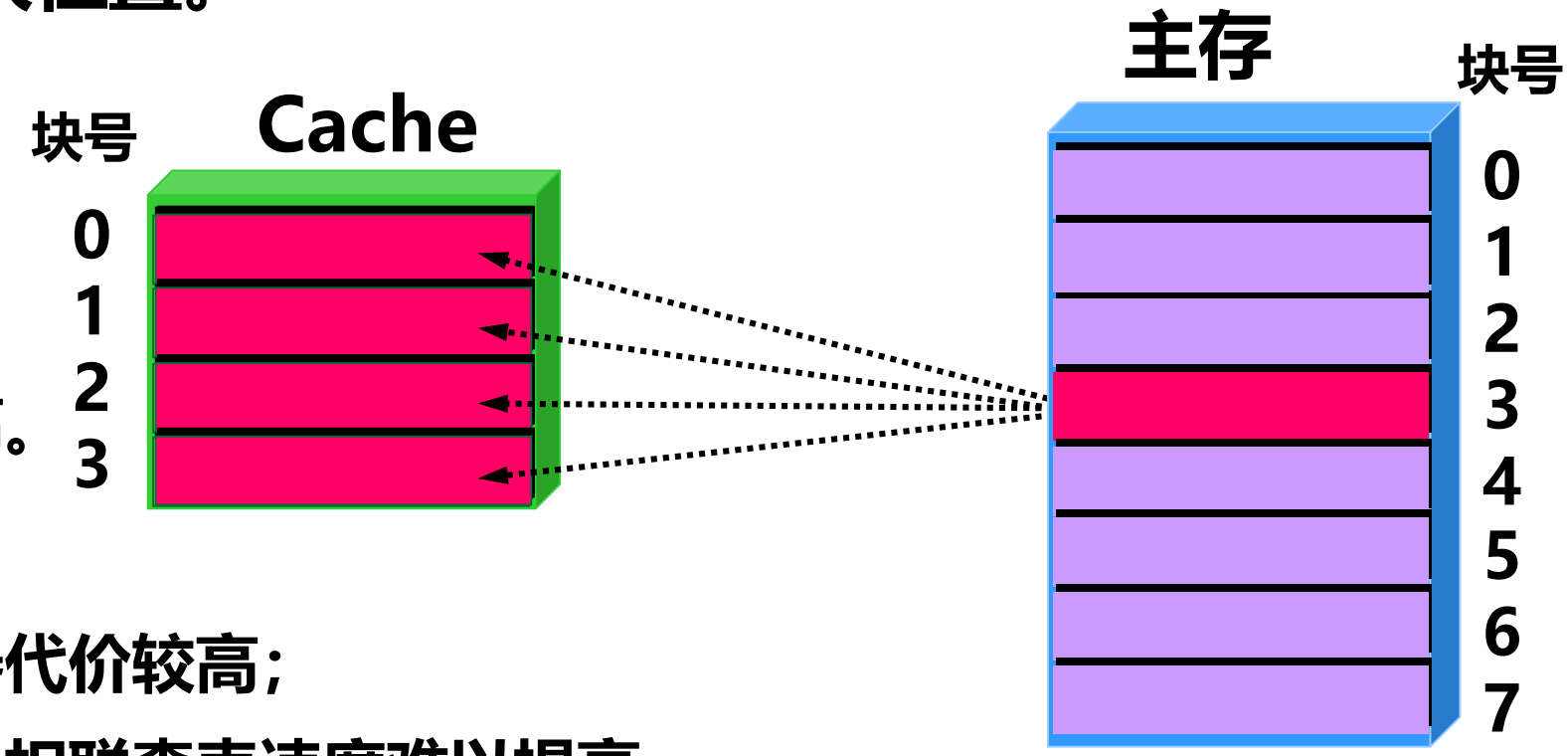
块冲突：主存中的块要调入Cache中的某个位置，但该位置已经被其他主存块所占用。

在Cache—主存存贮层次，典型的地址映像与变换方法主要有：

1. 全相联映像与变换
2. 直接映像与变换
3. 组相联映像与变换

全相联映像规则与变换

映像规则： 主存中的任意一块都可以装入到Cache中的任意一个块位置。



- 优点：
  - ☑块冲突概率最低；
  - ☑Cache空间利用率最高。
- 缺点：
  - ☑所需容量的相联存贮器代价较高；
  - ☑Cache容量已经很大，相联查表速度难以提高。

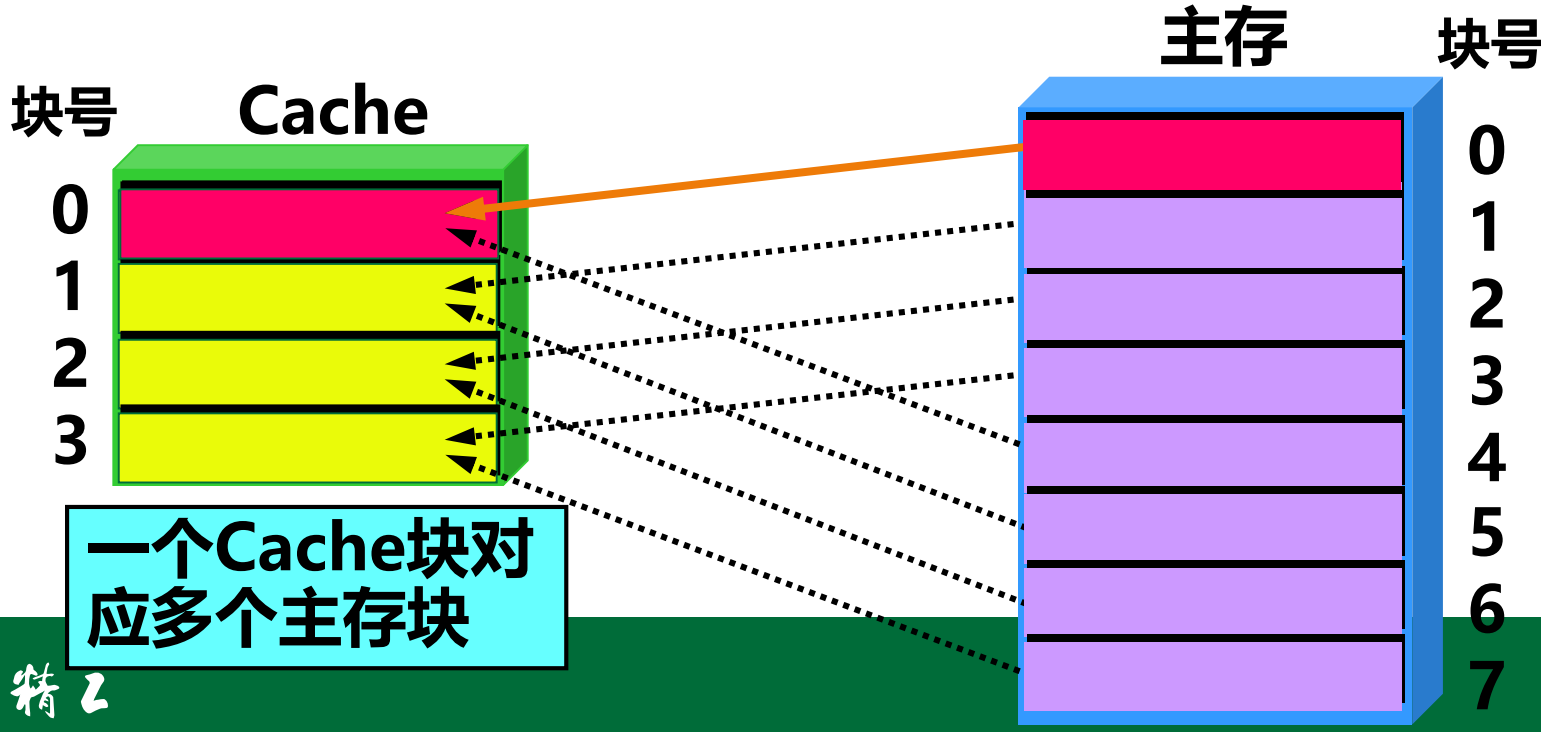
直接映像规则与变换

映像规则：主存中的每一块只能装入到Cache内唯一一个指定的块位置。

为便于地址变换，

设：Cache块号b = (主存块号B) mod (Cache块数)

则：0=0|4 mod 4, 1=1|5 mod 4, 2=2|6 mod 4, 3=3|7 mod 4



直接映像规则与变换

- 优点
  - ☑所需硬件简单，成本较低；
  - ☑访问Cache可与访问区号表、比较区号等操作同时进行，节省了地址变换时间。
- 缺点
  - ☑块冲突概率很高；
  - ☑Cache空间利用率很低。因此已经很少使用。

组相联映像规则与变换

映像规则： 主存中的每一块可以被装入到Cache中唯一 一个组中的任何一个位置。

为便于地址变换，可以采用如下映像算法：

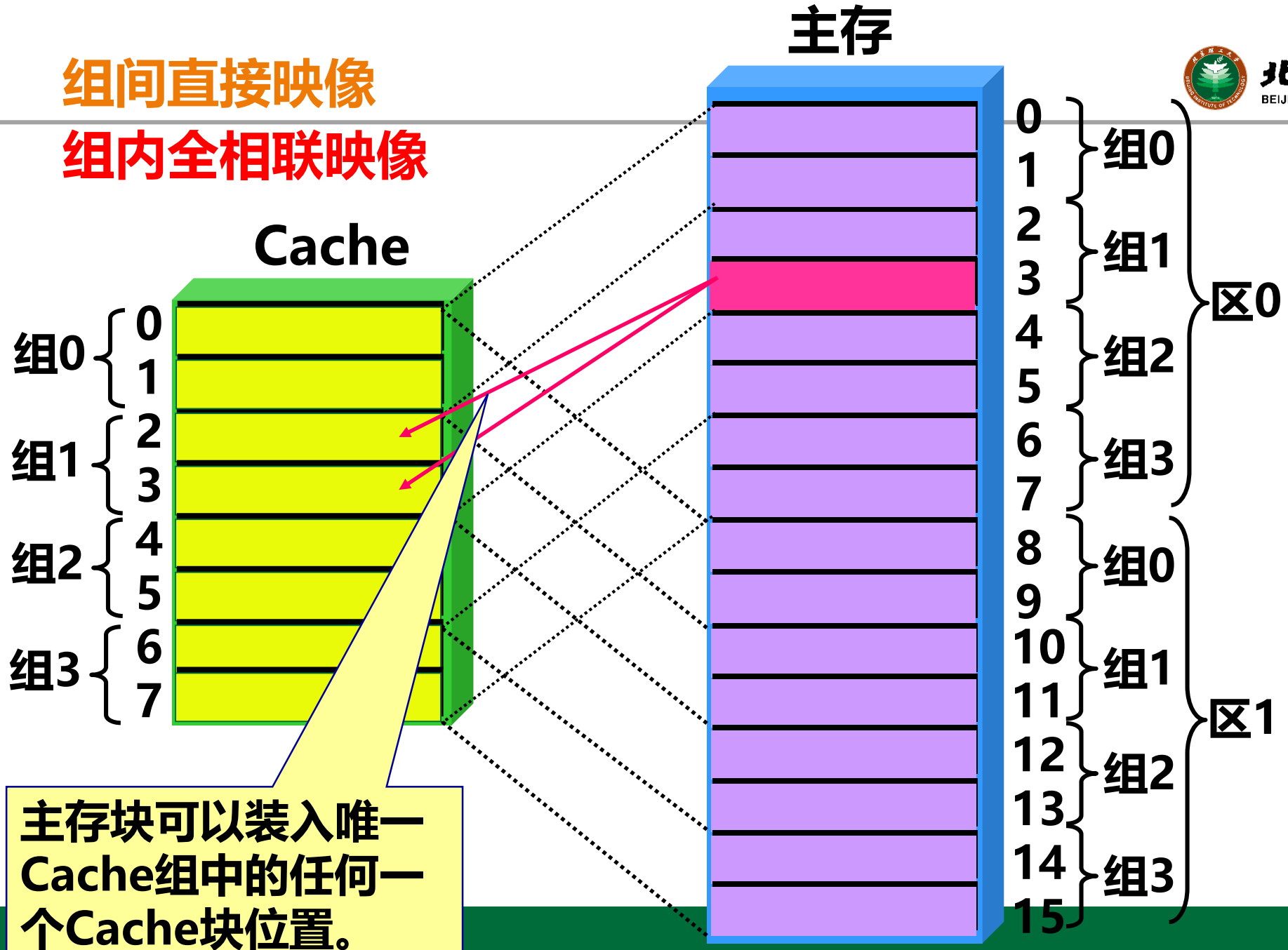
- 将整个Cache看成是一个区，将Cache分成G组（$G=2^g$），每个组S块（$S=2^s$）；

Cache地址	组号(g位)	块号(s位)	块内地址

- 将主存分成2^e个与Cache大小相同的区，每个区分成G组（$G=2^g$），每个组S块（$S=2^s$）；

主存地址	区号(e位)	组号(g位)	块号(s位)	块内地址

组间直接映像
组内全相联映像



组相联映像规则与变换

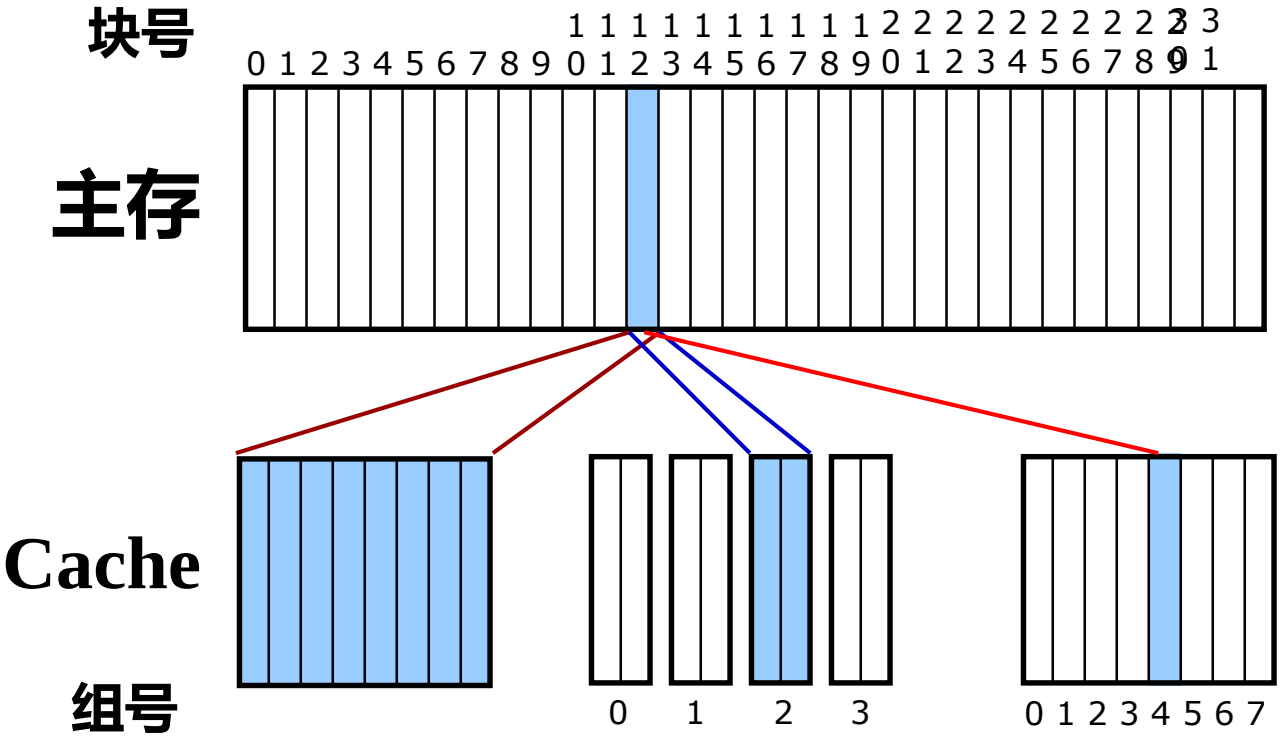
组相联实际上是全相联映像和直接映像的折衷方案。

- 当 S=Cache块数，组相联就变成了全相联；
- 当 S=1，组相联变成了直接映像；
- S越大，冲突越少，地址变换就越复杂；
- S越小，冲突越多，地址变换就越容易。

优点：

- ☑块冲突概率比直接映像低得多；
- ☑Cache空间利用率也比直接映像提高；
- ☑实现成本比全相联映像要低得多；
- ☑性能接近于全相联映像。

组相联映像规则与变换



主存块12可以装入Cache中的

任意位置

组2中的任意位置

只能是块4位置

替换算法与实现

所要解决的问题：当所要访问的主存块不在Cache中（称为块失效），需新调入一个主存块，而该块能够占用的Cache块位置已被占满时（称为块冲突），将哪一个位置的主存块替换出去。

典型替换算法：

1. 随机算法：随机选取一块进行替换。
2. FIFO算法：选取最早装入的块进行替换。
3. LRU算法：选取近期被CPU访问次数最少的块进行替换。

   优点：比较正确地反映了程序的局部性，失效率低。 ➔ 常用

写策略

解决一致性问题的关键：

CPU写入Cache时如何更新主存的内容。

两种写策略：

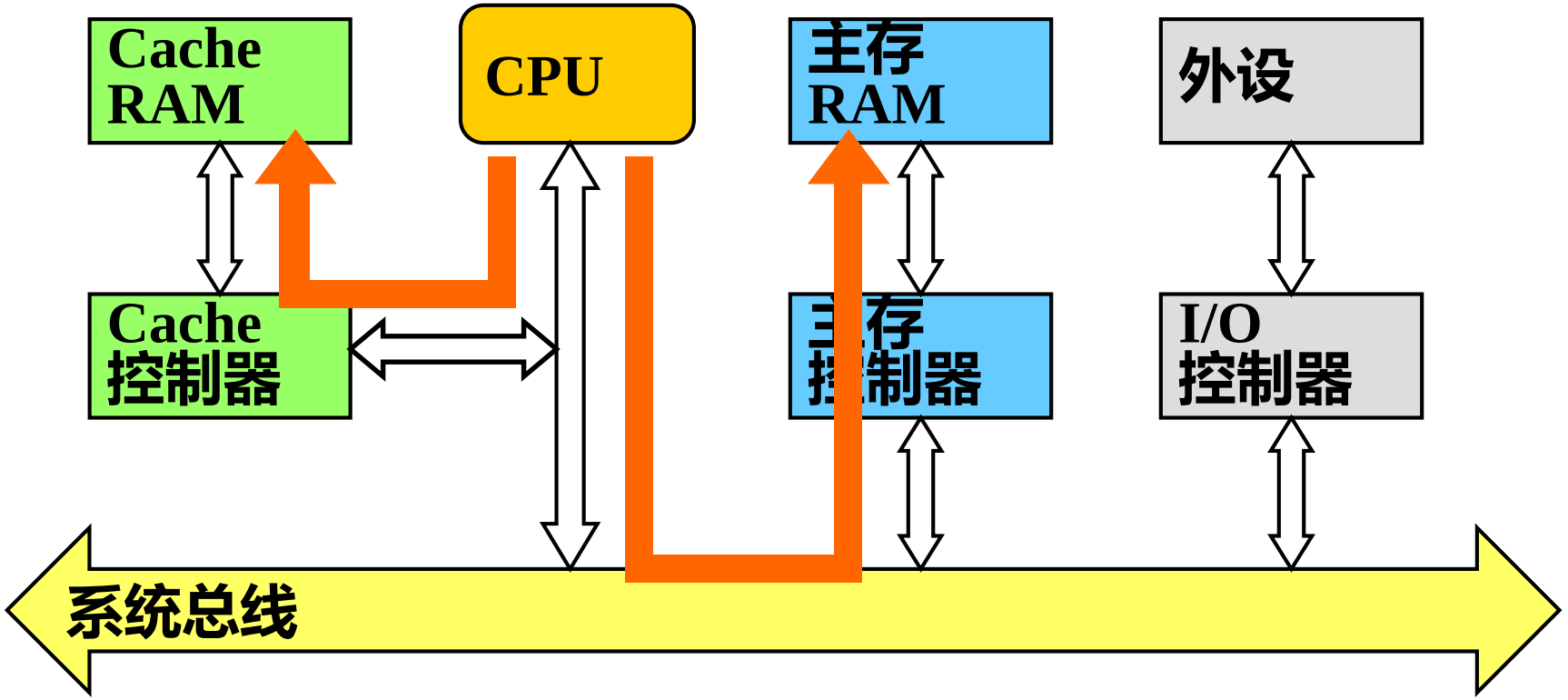
●写直达法（Write-through）

- **CPU在执行写操作时，利用直接通路，把数据同时写入Cache和主存。**

●写回法 （Write-Back）

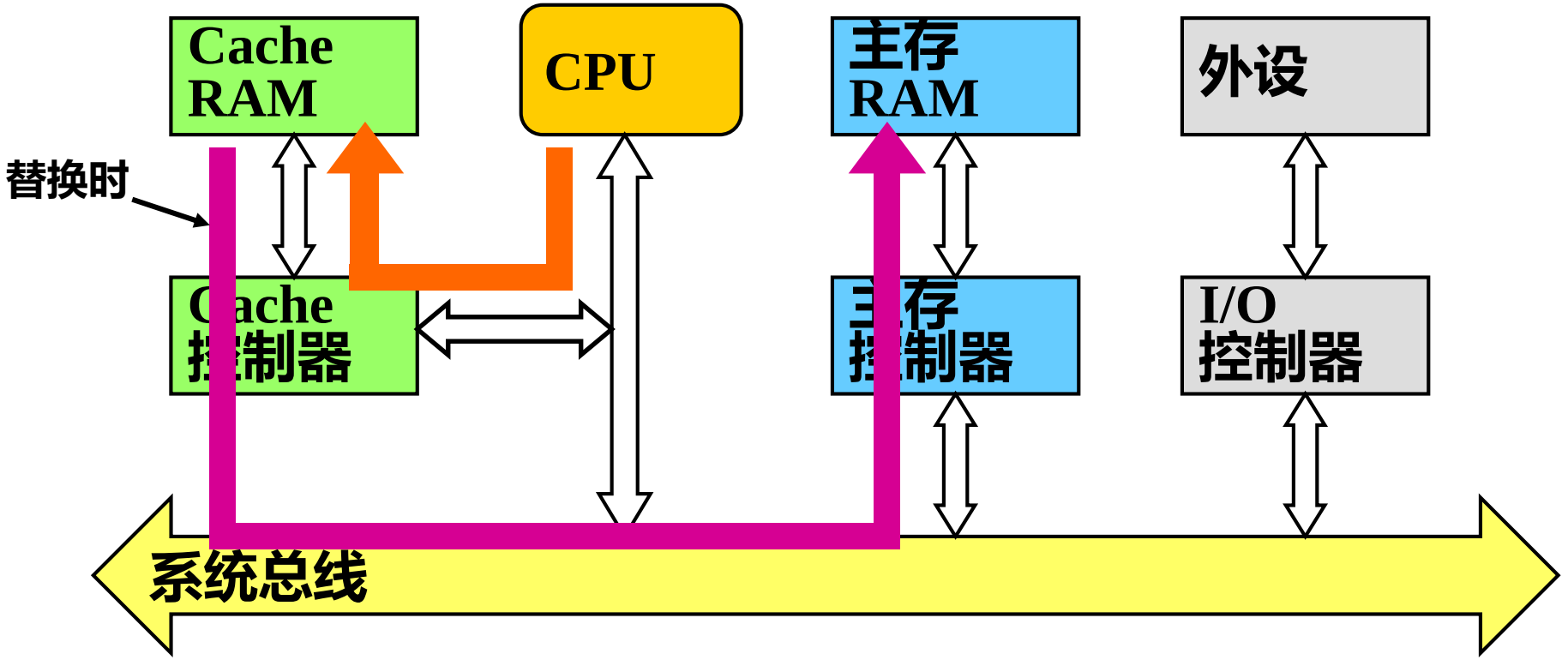
- **CPU数据只写入Cache，不写入主存；**
- **为每个Cache块设置“修改位”；**
- **仅当替换时，才把修改过的Cache块写回到主存。**

写策略



写直达法：数据同时写入Cache和主存

写策略



写回法：数据只写入Cache，替换时才写入主存

Cache地址映象举例

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

(1)主存地址多少位？如何分配？

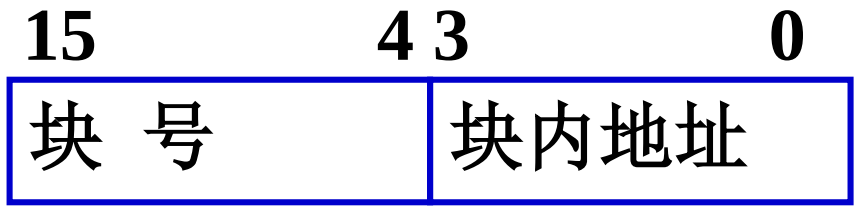
(2)Cache地址多少位？如何分配？

Cache地址映象举例

●解

主存地址格式：

Cache地址格式：



Cache地址映象举例

- **例：主存容量为1MB，Cache容量为32KB，每块为64个字节，共分128组。请写出主存与Cache的地址格式。**
- **解：**

主存地址：

19 ~ 15	14 ~ 8	7 ~ 6	5 ~ 0
区号	组号	块号	块内地址

Cache地址：

14 ~ 8	7 ~ 6	5 ~ 0
组号	块号	块内地址

Cache的平均访问时间计算

- 1、**贯通查找式（Look-Through）**
- CPU首先向Cache发读命令和地址。
- Cache命中，则从Cache中读出数据；
- Cache未命中，再将读命令和地址传给主存并读主存。
- Cache平均访问时间= Cache访问时间+（1-命中率）×未命中时主存访问时间

- 优点：降低了CPU对主存的请求次数。
- 缺点：不命中时延迟了CPU对主存的访问时间。

Cache的平均访问时间

- 2、旁路式 (Look-Aside)
- CPU向Cache和主存同时发读命令和地址。
- Cache命中，则Cache回送数据并中断读主存命令；
- Cache未命中，则直接访问主存读取数据。
- Cache的平均访问时间=命中率×Cache访问时间+（1-命中率）×未命中时主存访问时间
- 优点：不命中时，Cache不会增加额外的时间延迟。
- 缺点：CPU的每次请求都会发送给主存，CPU的系统总线占用率较高。

Cache命中率计算

- 例：CPU执行一段程序，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的平均访问时间。

  解：

  Cache的命中率=1900/（1900+100）=95%

  贯通查找式：

  Cache访问时间+（1-命中率）×未命中时主存访问时间

  =50 +(1-95%) ×250 = 62.5 ns

  旁路式

  命中率×Cache访问时间+（1-命中率）×未命中时主存访问时间

  =95% ×50+(1-95%) ×250=60 ns

虚拟存储器

虚拟存储概念1951年由英国曼彻斯特大学Kilbrn等人提出。

虚拟存储器由主存储器和联机工作的辅助存储器（通常为磁盘存储器）共同组成，这两个存储器在硬件和系统软件的共同管理下工作，对于应用程序员，可以把它们看作是一个单一的存储器。

70年代广泛地应用于大中型计算机系统中，目前许多微型机也使用虚拟存储器。

原理和Cache-主存层次类似，请自学。

常见的虚拟存储器：页式虚拟存储器、段式虚拟存储器、段页式虚拟存储器。

感谢聆听