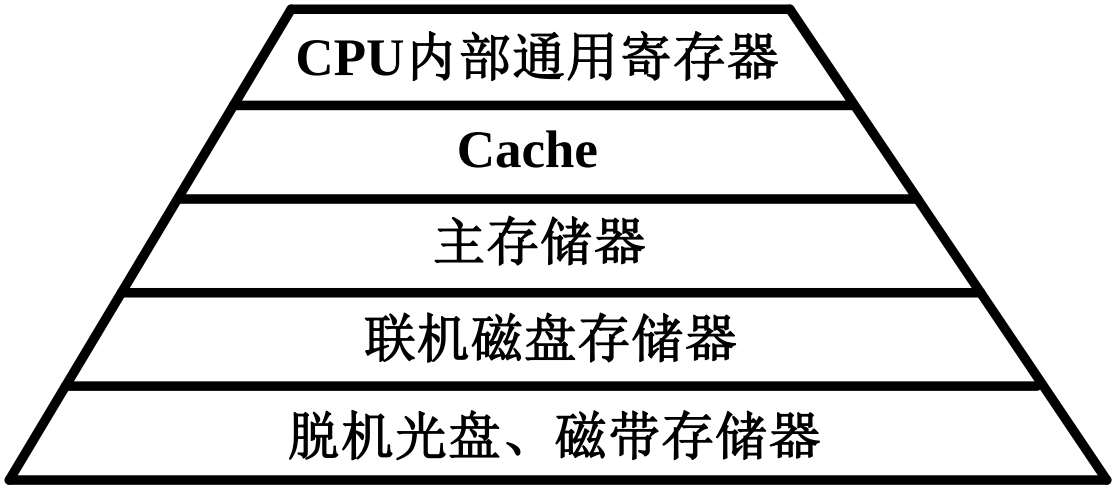
第7章　存储系统

7.1　存储系统概述

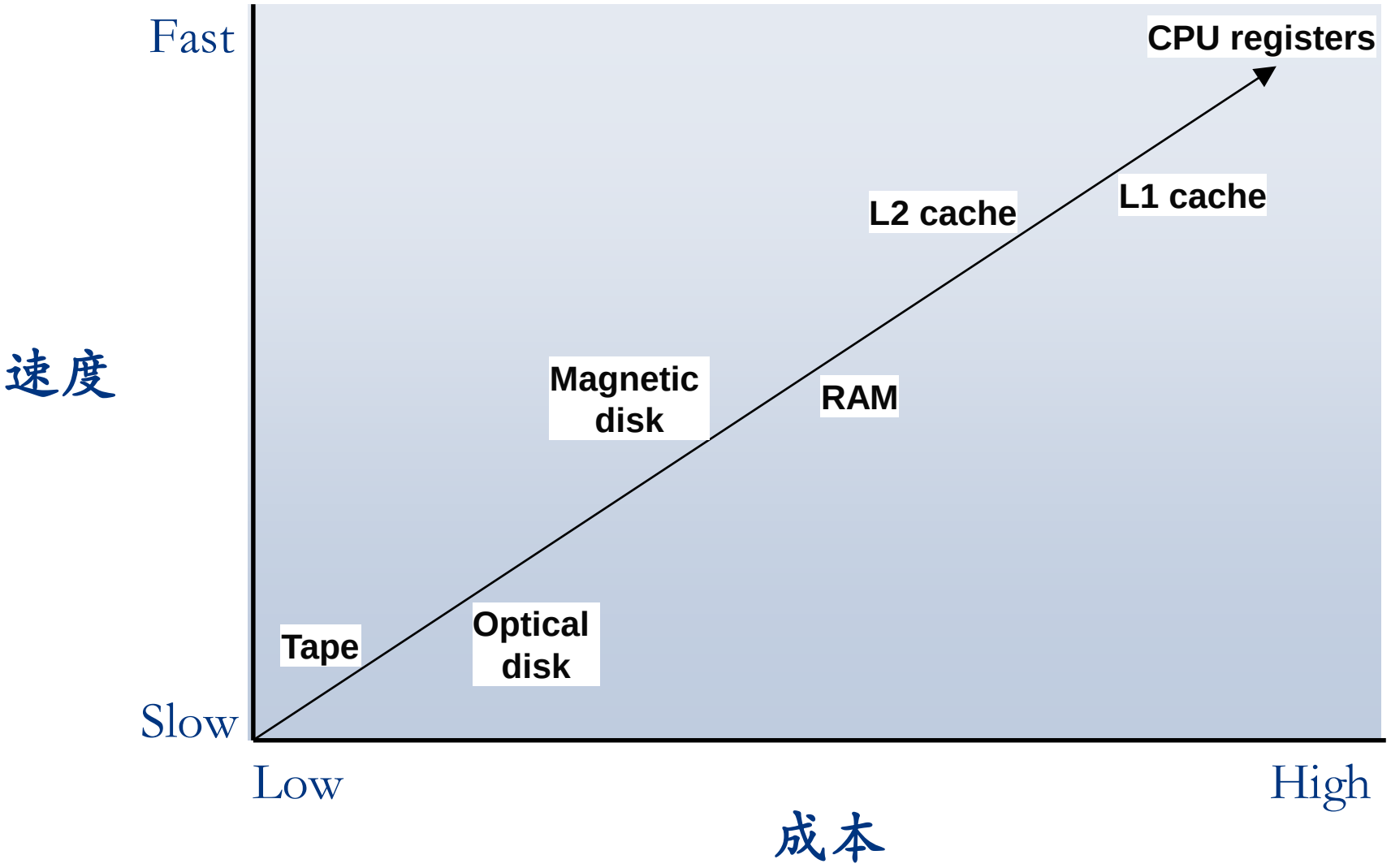
- 存储系统是指计算机中由存放程序和数据的各种存储设备、控制部件及管理信息调度的设备（硬件）和算法（软件）所组成的系统。
- 存储系统是计算机的重要组成部分之一。存储系统提供写入和读出计算机工作需要的信息（程序和数据）的能力，实现计算机的信息记忆功能。
- 现代计算机系统中常采用寄存器、高速缓存、主存、外存的多级（层）存储体系结构。

7.1.1 存储系统的层次结构

计算机层次结构中，越是靠近上层，其速度越快、容量越小而单位存储容量的价格也越高（图）。



存储层次速度与成本曲线



7.1.2 存储器分类

（1）按存储信息介质分

a. 半导体存储器

b. 磁存储器

c. 激光存储器

（2）按与 CPU的关系分

a. 主存储器MM

b. 控制存储器CM

c. 高速存储器Cache

d. 外存储器

（3）按读写功能分

a.读写存储器RAM（随机存储器）

---- SRAM、DRAM

b. 只读存储器ROM

---- ROM、PROM、EPROM、EEPROM（FM）

7.1.3 存储器主要性能指标

1. 存储容量

- 用字数×位数表示，以位（bit）为单位。

常用来表示存储芯片的容量，如1K×4位，表示该芯片有1K（1024）个单元，每个存储单元的长度为4位。

- 用字节数表示容量，以字节（Byte，B）为单位

常用来表示存储系统的容量，如1KB，表示该芯片有1K（1024）个单元，每个存储单元的长度为8位。

$$1KB = 1K \times 8位 = 2^{10} \times 8位$$

显然，存储容量越大，所能存储的信息越多，计算机系统的功能便越强。

现代计算机存储系统的容量很大，常用KB、MB、GB和TB等为单位表示存储容量的大小。

1B=8b　　b=bit　　B=Byte

1KB = 2^{10} Byte　　KB=KiloByte

1MB = 2^{20} Byte　　MB=MegaByte

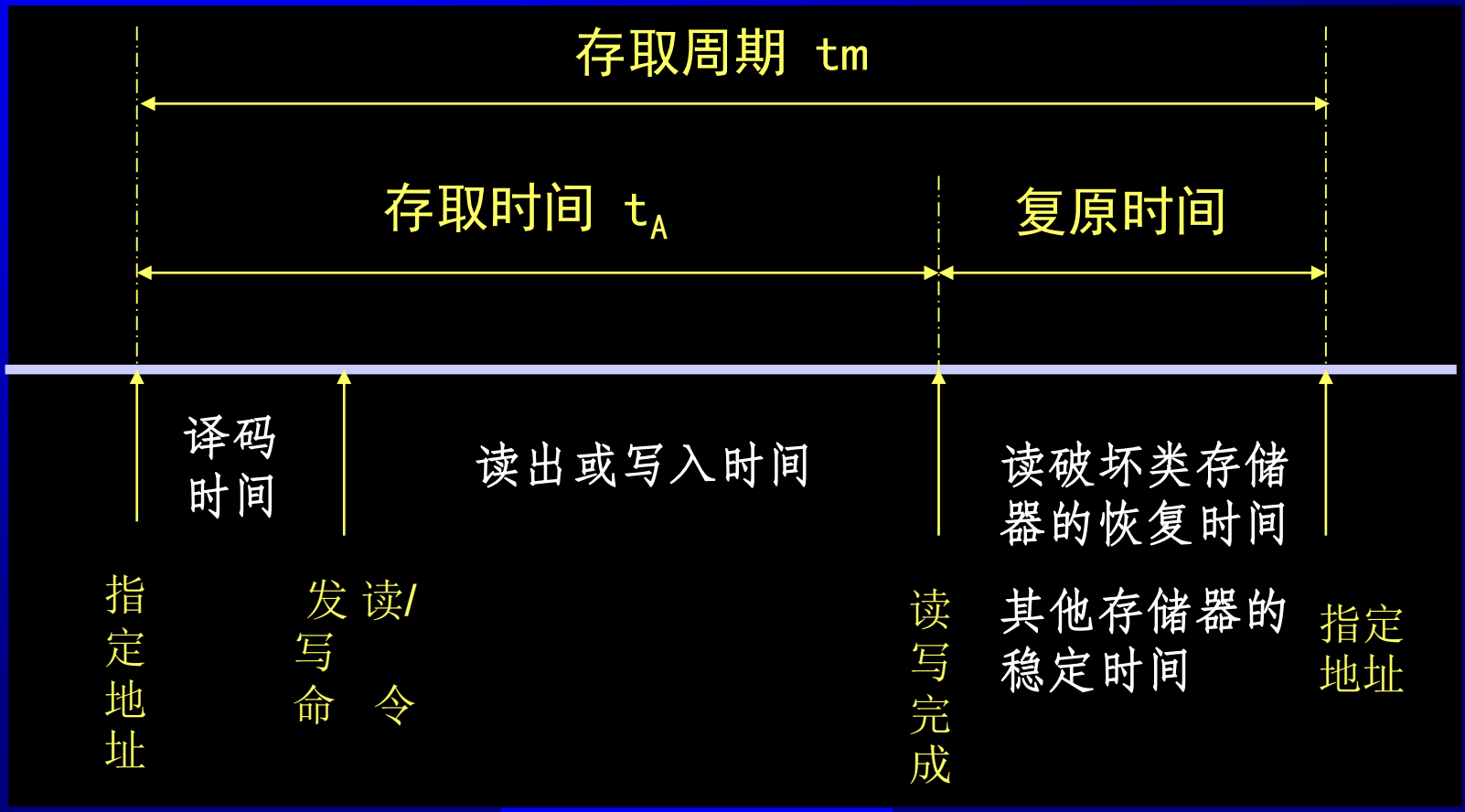
1GB = 2^{30} Byte　　GB=GigaByte

1TB = 2^{40} Byte　　TB=TeraByte

1PB = 2^{50} Byte　　PB=PetaByte

7.1.3 存储器主要性能指标

2. 存储器速度： 存取周期、存取时间



存储器访问过程

7.1.3 存储器主要性能指标

3. 可靠性
   - ✓ 可维修部件的可靠性可用平均故障间隔时间（MTBF）
   - ✓ 不可维修部件的可靠性常用平均无故障时间或平均故障前的时间（MTTF）
4. 功耗
4. 价格(成本)

存储器的单位成本 c = C/ S 元/位

其中： S是存储容量

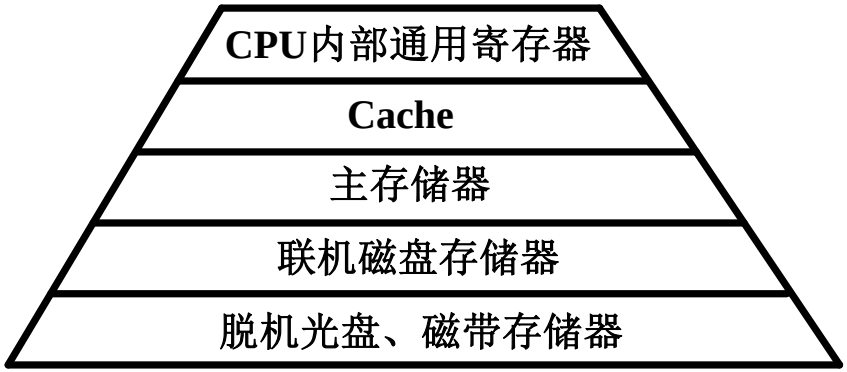
C为整个存储器的价格

理解：存储系统的层次结构

协调存储系统的三个指标：容量、速度、价格/位

- 计算机层次结构中，越是靠近上层，其速度越快、容量越小而单位存储容量的价格也越高。

解决存储系统大容量、高速度、低成本相互制约的矛盾！



常用的层次结构

1、主存-辅存 存储层次：　大容量、低成本；

软硬件结合完成

cache-主存 存储层次：　高速度、低成本；

纯硬件完成

2、cache-主存-辅存　三级存储体系

本章主要内容说明

- **主存储器** *-----存储系统（上），此处学习*

  主存的构成，各种半导体存储器基本原理、连接使用的问题，如SRAM、DRAM、ROM等

- **高速缓存** *-----存储系统（下），后面学习*

  Cache的基本原理、映射方式、替换算法以及性能分析

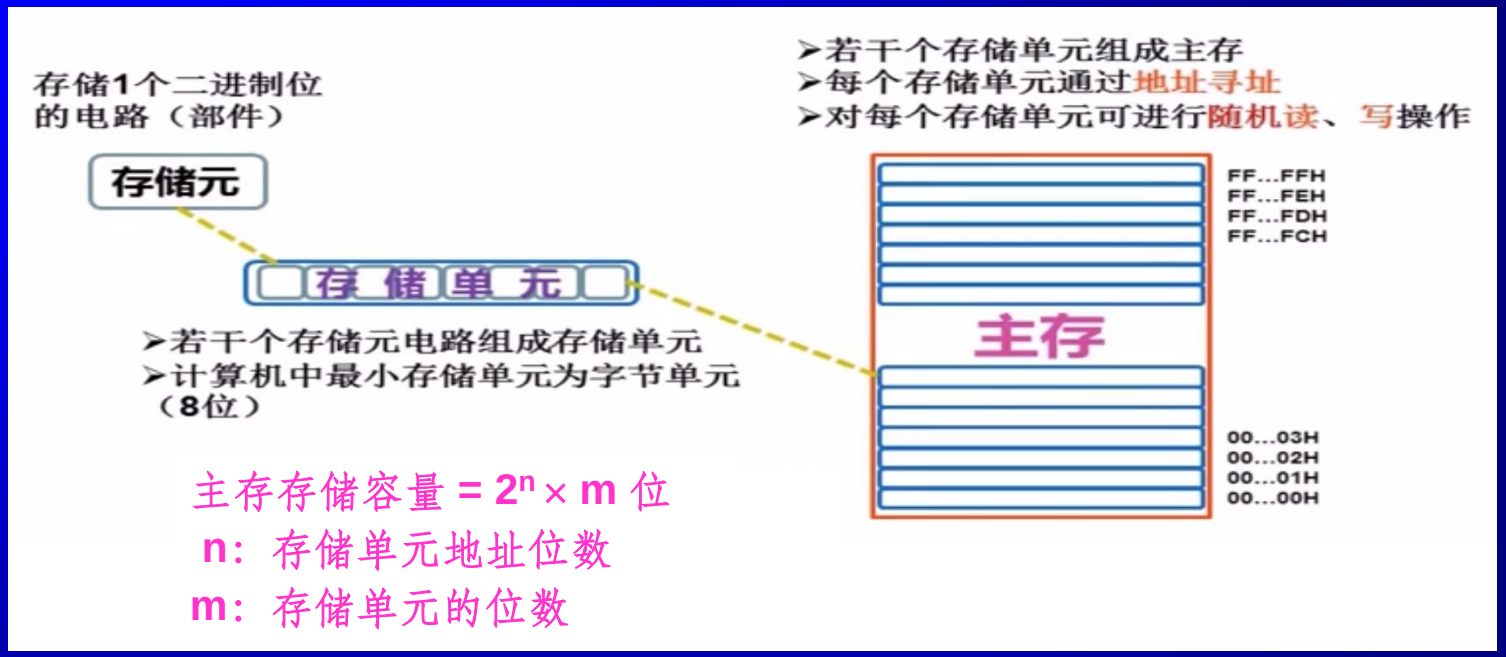
- **虚拟存储器** *-----操作系统课程里学习*

  基本原理、映射方式、实例等

7.2 主存储器

- 主存(Main Memory，MM) 用来存放当前运行的程序和所用的数据（二进制）
- 主存是计算机存储体系中最早出现的存储层次，是从冯·诺依曼计算机开始到目前所有计算机中不可缺少的功能部件
- 内部存储器，狭义是指主存储器，广义是包括主存、高速缓存、虚拟存储器在内的存储层次
- 主存容量和速度对计算机性能（速度）有直接影响
- 现代计算机的主存毫无例外地采用半导体存储器集成芯片构成

• 主存的逻辑结构



- ✓ 存储元是用于存储1位二进制信息的电路，又称为存储位或基本存储单元。
- ✓ 若干个存储元组成一个存储单元。许多个存储单元组成主存。
- ✓ 每个存储单元用二进制编码的地址标识，并依据地址寻址访问
- ✓ 对每个存储单元可进行随机读写
- ✓ 计算机的最小存储单元为字节单元（8位），现代计算机基本按字节单元编址
- ✓ 主存的地址空间由CPU地址线数决定

7.2.1 随机读写存储器RAM

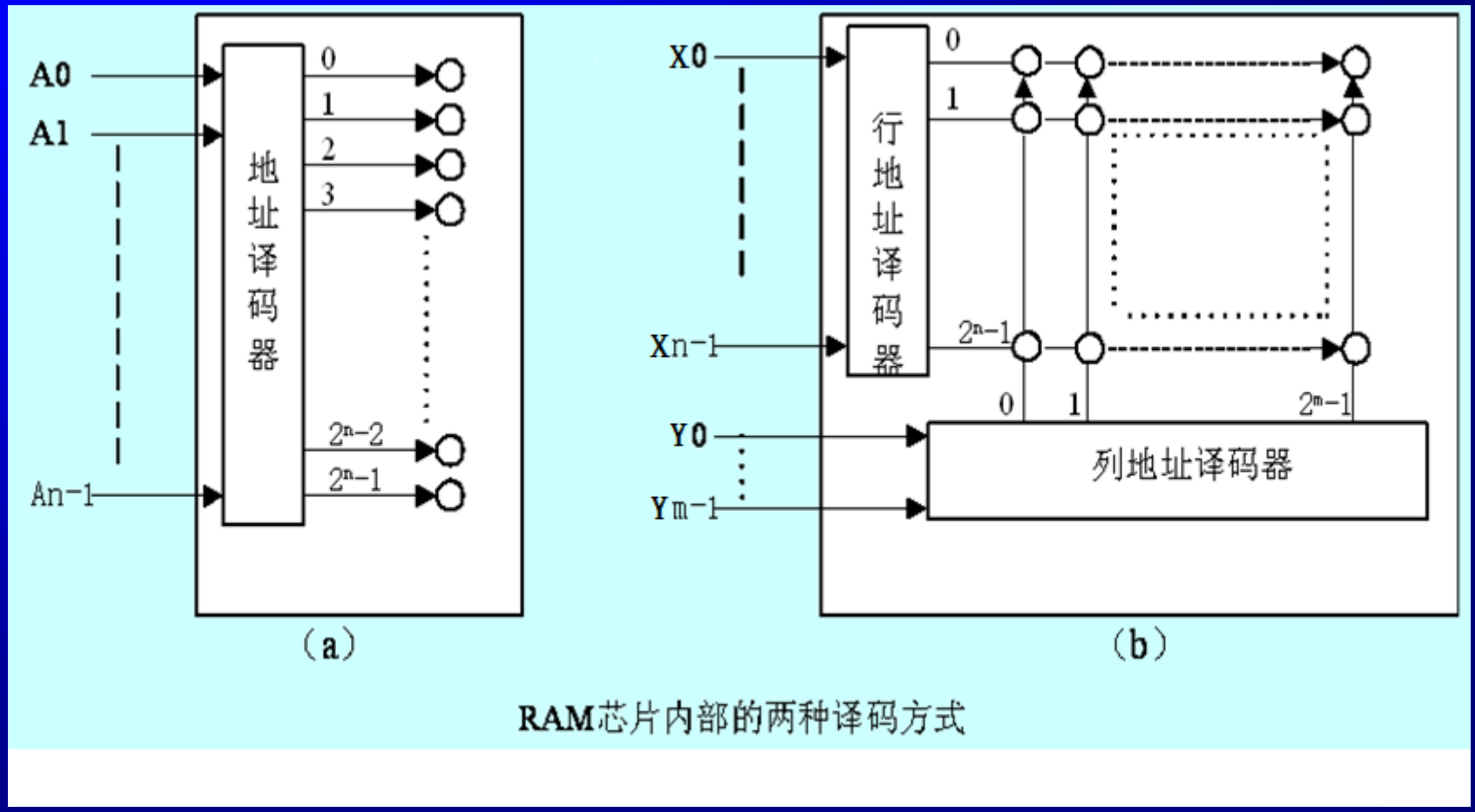
---- RANDOM ACCESS MEMORY

- SRAM 静态随机读写存储器
- DRAM 动态随机读写存储器

1、静态随机读写存储器SRAM 的组成

- 常规RAM芯片，其外部有地址引线、数据引线和控制信号引线。
- 地址引线在芯片内部译码，选中芯片内部的相应存储单元。
- 某静态RAM芯片上有n条地址线时，地址线上所能表示的地址编码就有2^n种。这就意味着该芯片内部有2^n个存储单元。

(1) 内部译码结构



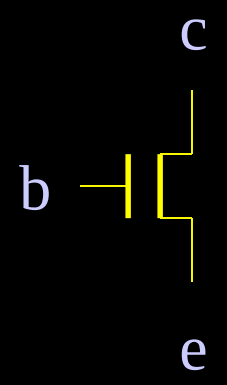
RAM芯片内部的两种译码方式

(a) 一维译码

(b) 二维译码

（2）单元电路

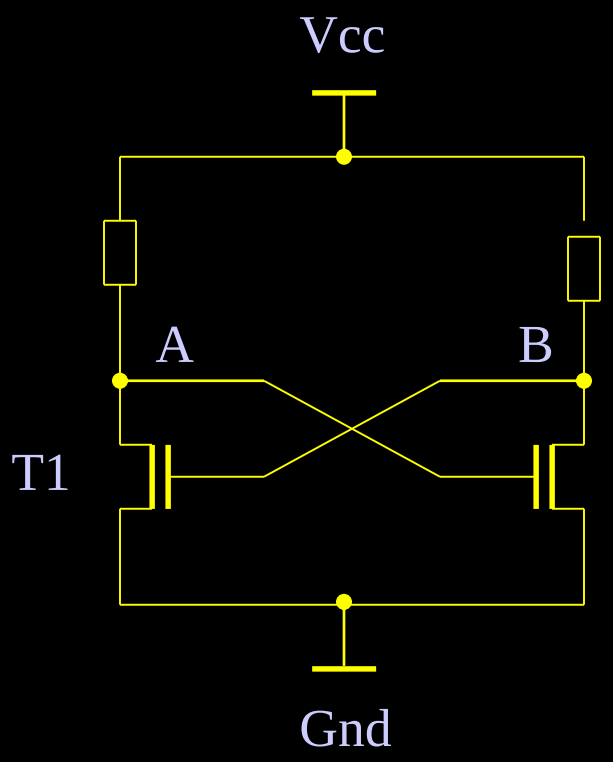
- 1位SRAM （MOS型）结构及工作原理：



开关状态时：

b 高电平，e－c 饱和

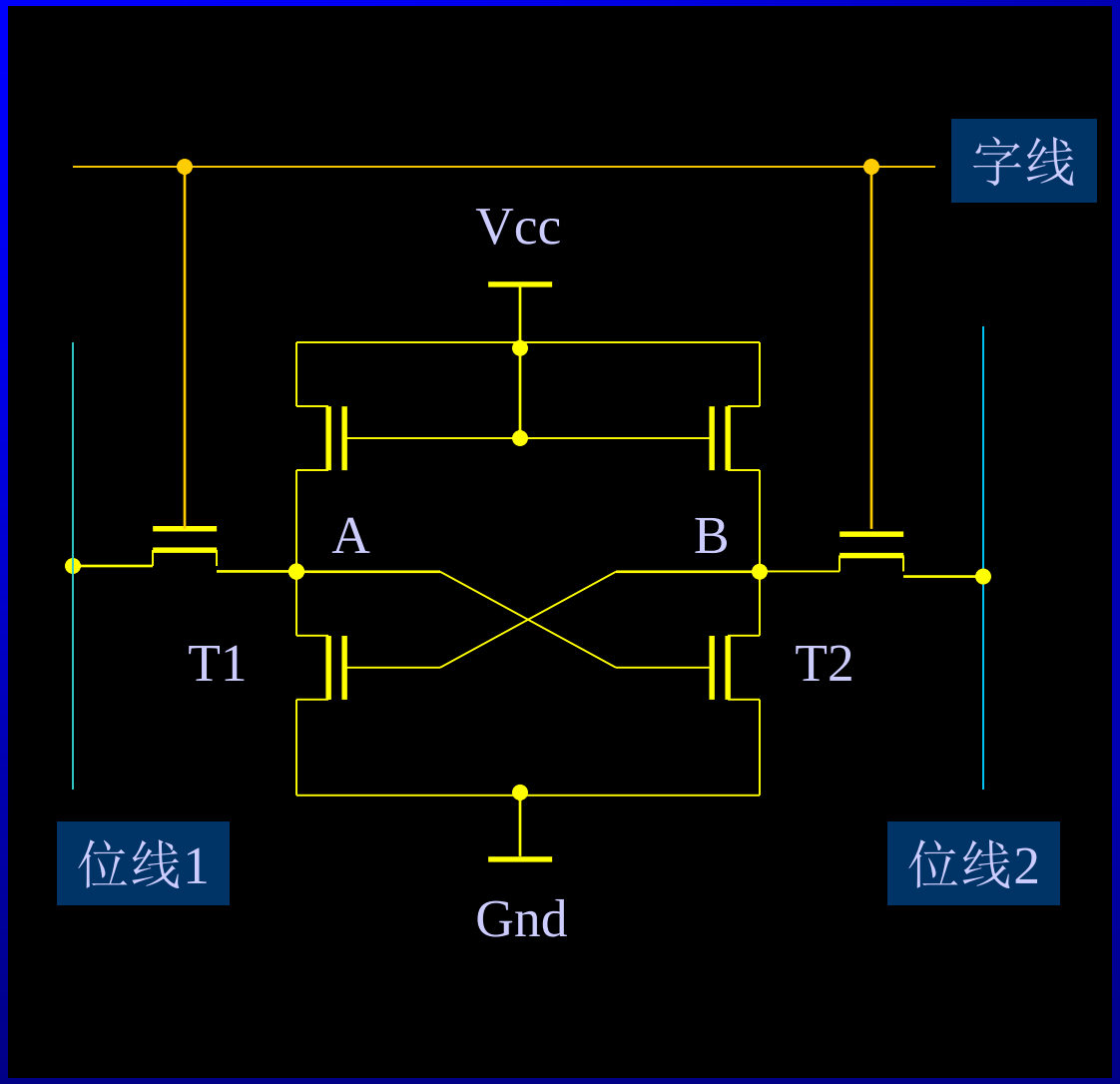
b 低电平，e－c 截止



状态 ‘1’：
T1 饱和、T2截止
A点低　　B点高

状态 ‘0’：
T1截止、T2饱和
A点高　　B点低

双稳态电路，可以分别存放 0 和 1



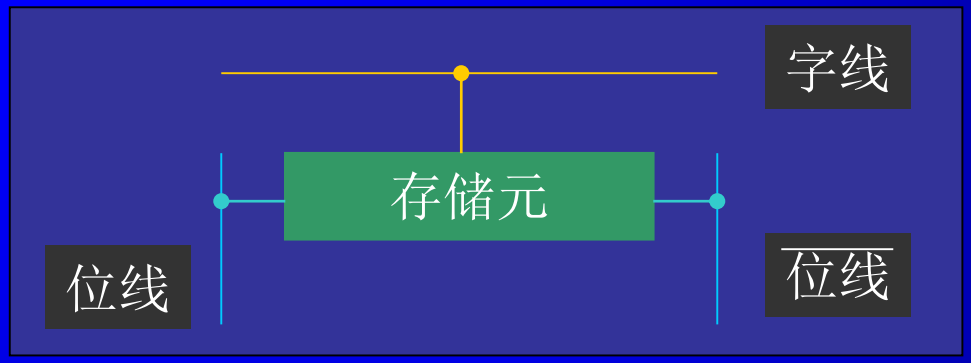
6管SRAM 存储元

写入‘1’： 字线 高
位线1 低、位线2 高
A点低 B点高
T1 饱和、T2截止

写入‘0’： 字线 高
位线1 高、位线2 低
A点高 B点低
T1截止、T2 饱和

保持： 字线 低
T1、T2 状态维持

读出： 字线 高
位线1低、位线2高 -- ‘1’
位线1高、位线2低 – ‘0’

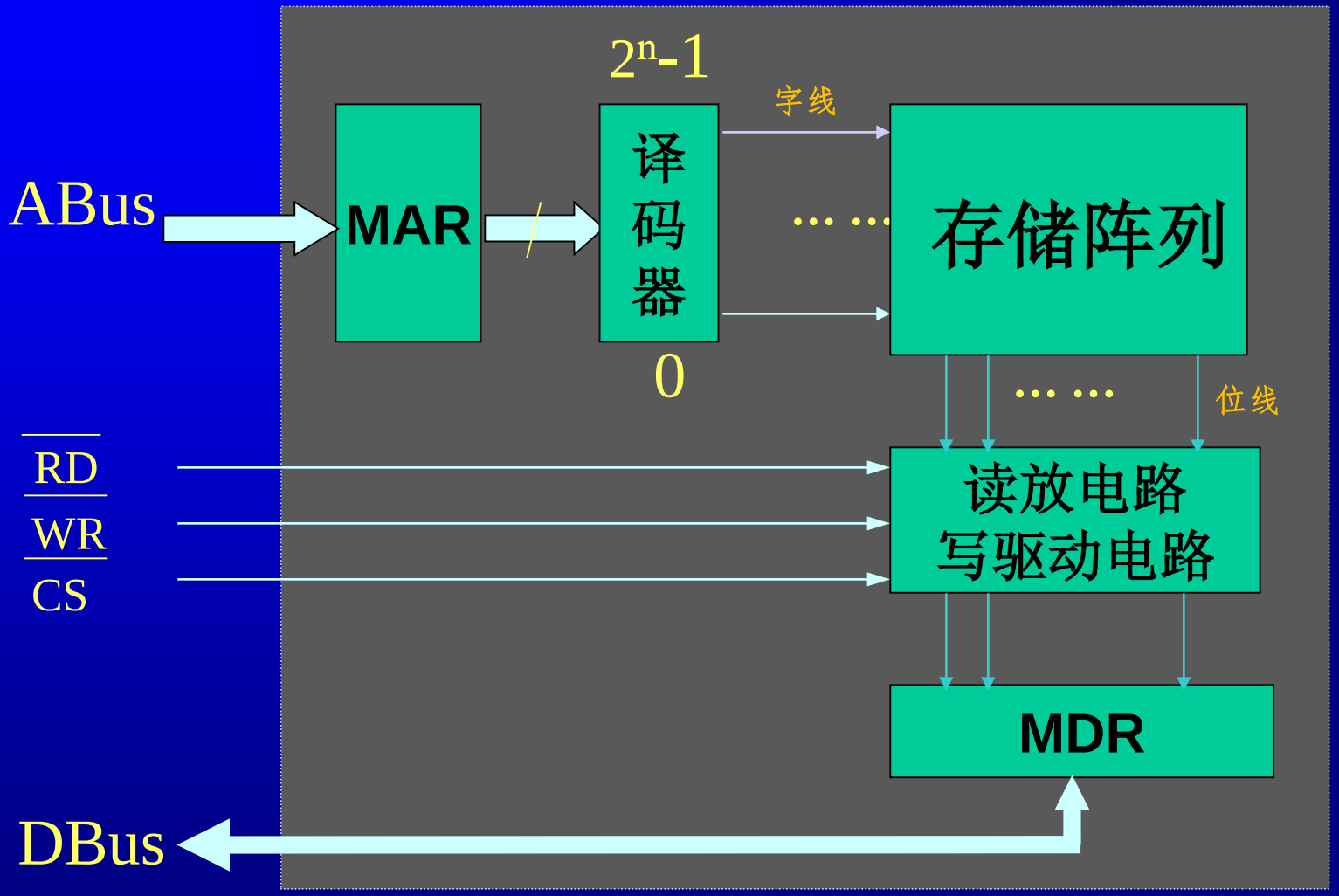


SRAM存储元简图

分析1位SRAM存储元电路，可知：

- ✓ 用n个1位存储元电路同时工作可构成n位存储单元
- ✓ 数据一旦写入，其信息就稳定地保存在电路中并等待读出无论读出多少次，只要不断电，此信息会一直保持下去，这也许就是“静态”一词的由来
- ✓ SRAM初始加电时，其状态是随机的。写入新的状态，原来的旧状态就消失了，新状态一直维持到写入更新的状态为止
- ✓ 在电路工作时，即使不进行读写操作，只是保持在加电状态下，电路中就一定有晶体管导通，也就一定有电流流过，从而一定会有功率消耗
- ✓ 每存储一个二进制位,平均需要用到6个晶体管，因此，与DRAM比较，SRAM的功耗大，集成度不能做得很高

存储器基本组成

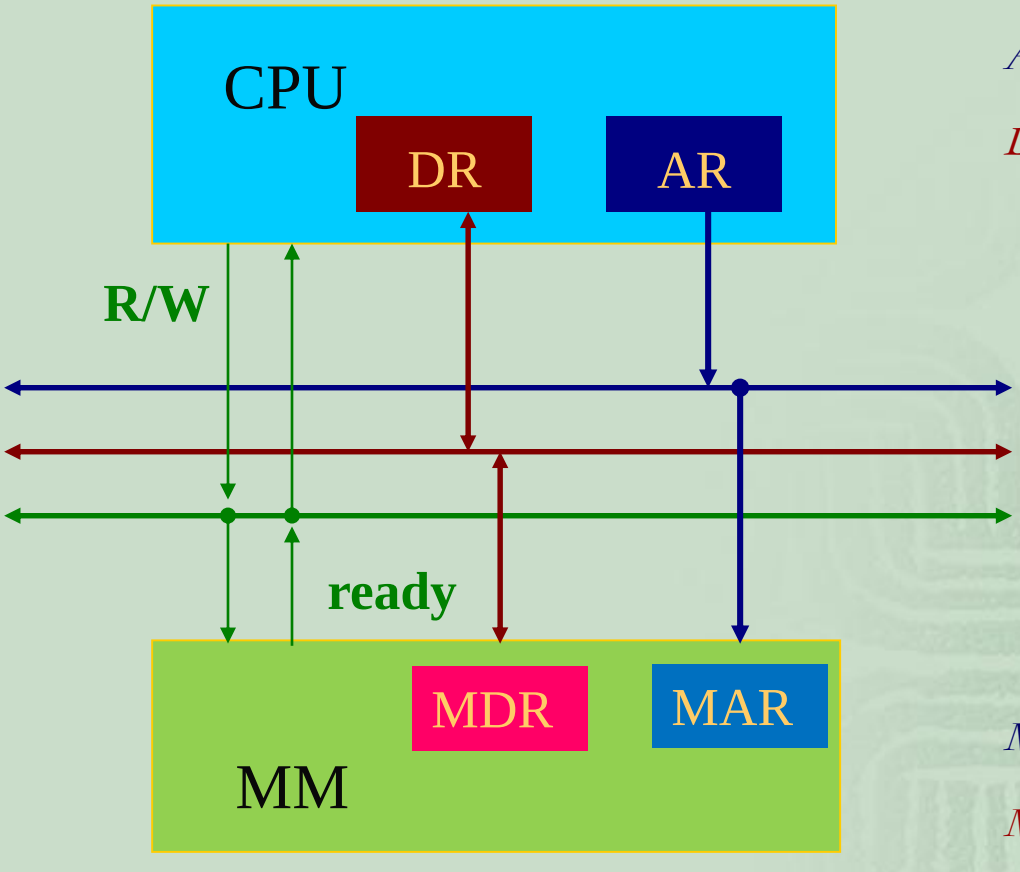


MAR 存储器地址寄存器

MDR 存储器数据寄存器

7.2.1 随机读写存储器RAM

2、主存储器的组成及接口



AR 地址寄存器

DR 数据寄存器

Abus 地址总线

DBus 数据总线

CBus 控制总线

MAR 存储器地址寄存器

MDR 存储器数据寄存器

主存接口相关的CPU总线信号

例：某CPU的 地址总线 A0～A15，数据总线 D0～D7，
控制信号 /RD、/WE 等

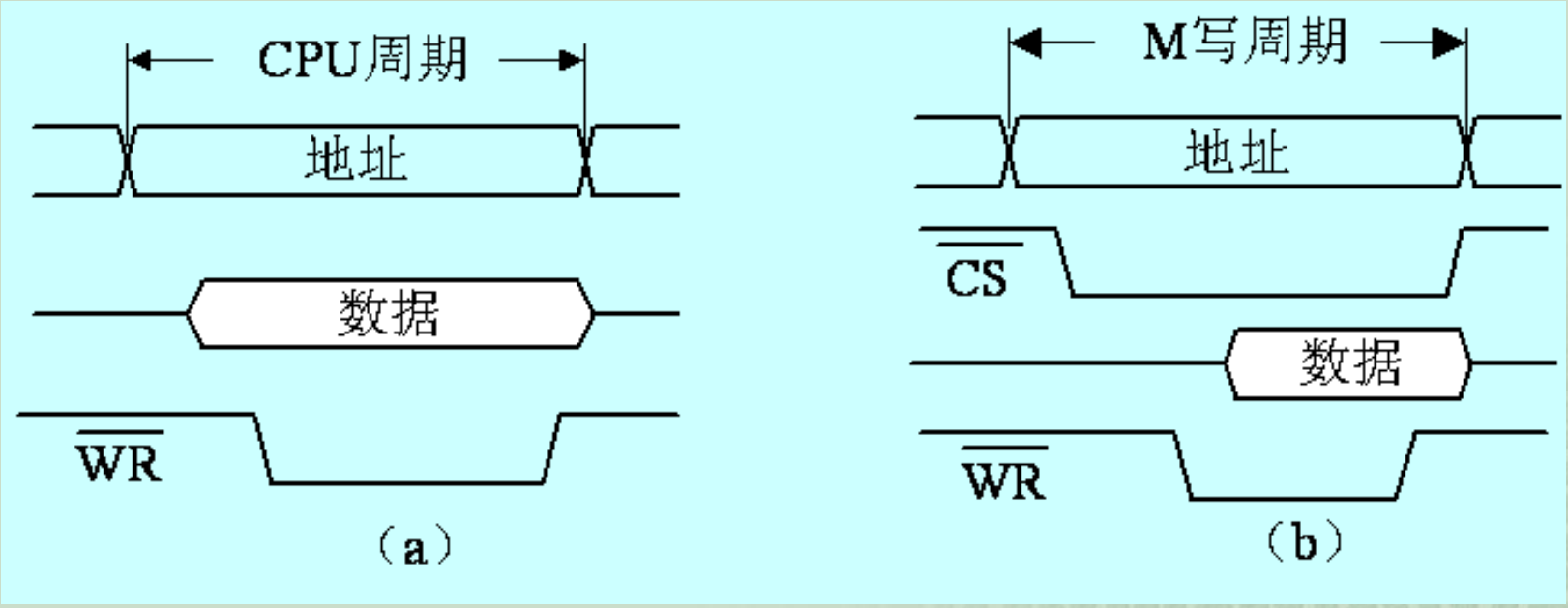
例：8088CPU的 地址总线 A0～A19，数据总线 D0～D7，
内存读信号 /MEMR、内存写信号 /MEMW 等

其主存的地址空间 1MB = 1M×8位 = 2^{20}×8位

例：8086CPU的 地址总线 A0～A19，数据总线 D0～D15，
内存读信号 /MEMR、内存写信号 /MEMW 等

其主存的地址空间 1MB = 1M×8位 = 2^{20}×8位

(1) 存储器与CPU的速度协调

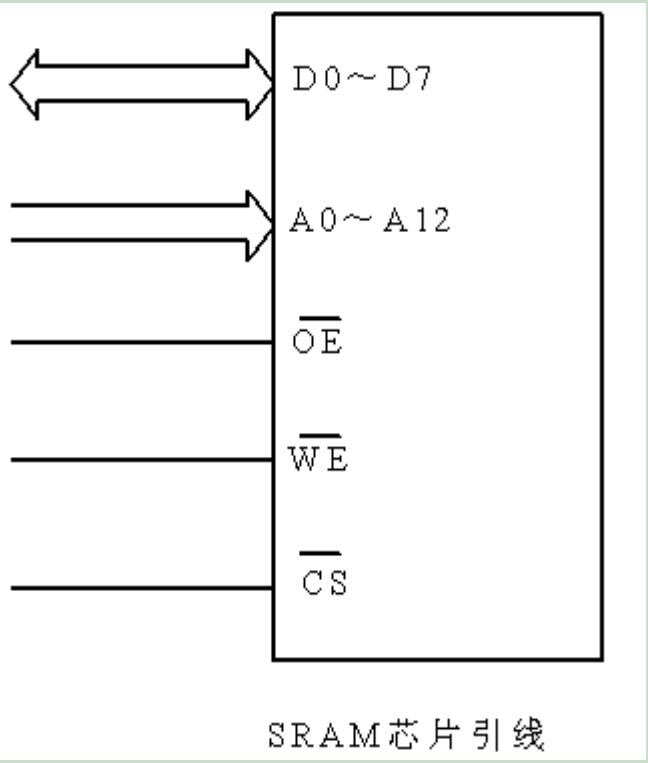


(a) CPU写内存的时序　　(b) 存储器的写时序

要求：CPU的写（或读）内存的周期应大于等于存储器芯片所要求的写（或读）时间。也就是说CPU提供的信号持续时间必须大于等于存储器芯片所要求的信号时间。

(2) 内存构成

- *单片存储器的连接*



SRAM芯片引线

A0～A12为13条地址信号线

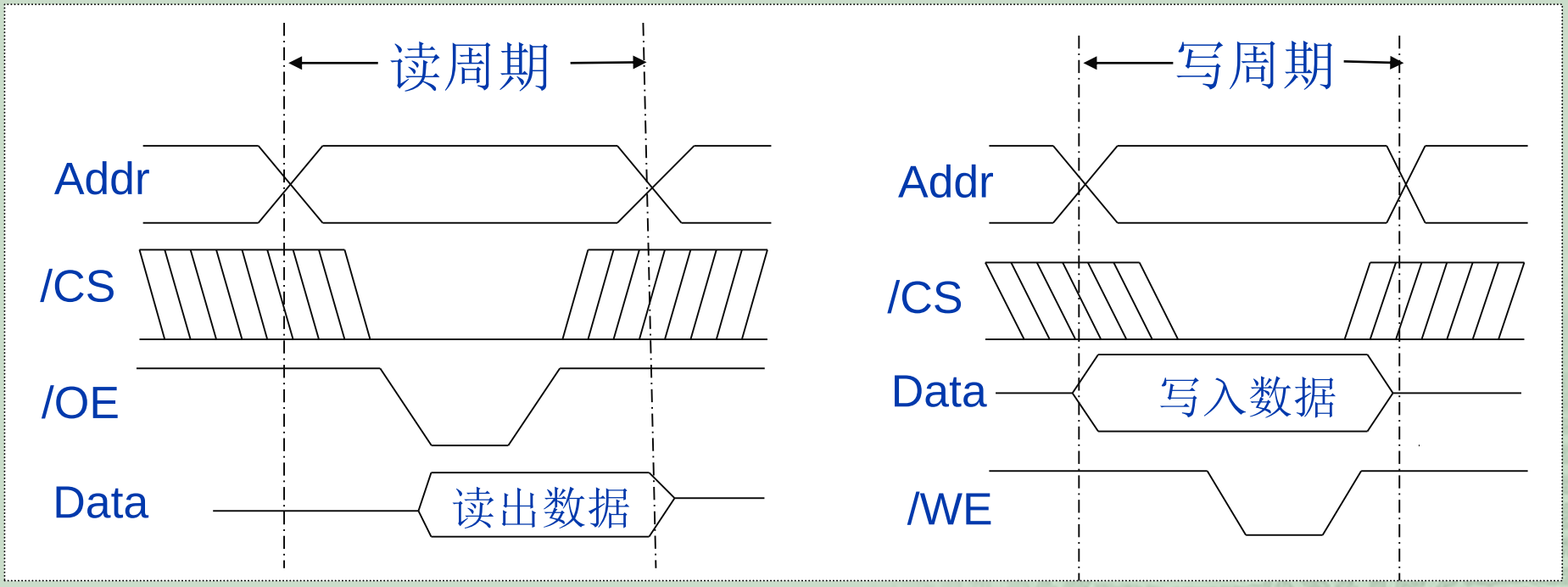
D0～D7为8条双向数据线

$\overline{OE}$为输出允许信号。只有当$\overline{OE}$=0，即其有效时，才允许该芯片将某单元的数据送到芯片外部的D0～D7上。

$\overline{WE}$是写允许信号。
当$\overline{WE}$=0时，允许将数据写入芯片；
当$\overline{WE}$=1时，允许芯片的数据读出。

$\overline{CS}$片选信号的引线，当片选信号同时有效时，即$\overline{CS}$=0，时，才能选中该芯片。

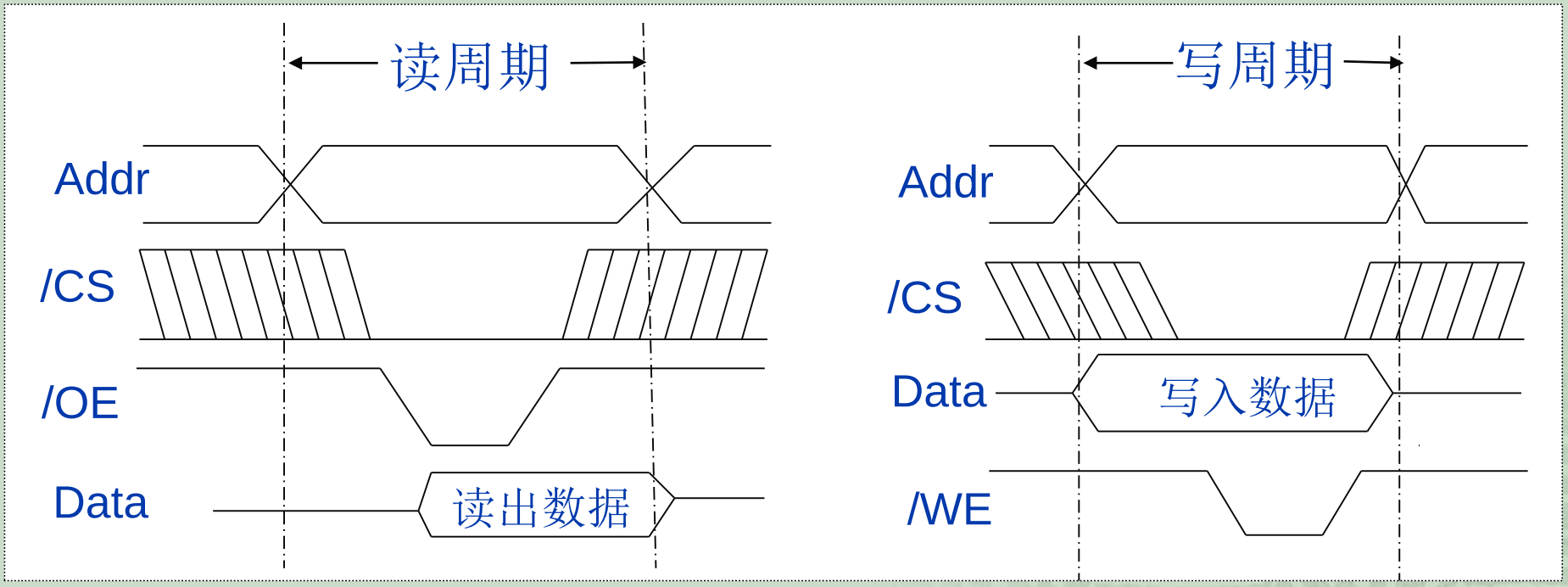
● 工作过程



读出数据的过程：

A0～A12加上要读出单元的地址；使CS有效；使OE有效（为低电平）；使WE为高电平，这样即可读出数据。

● 工作过程

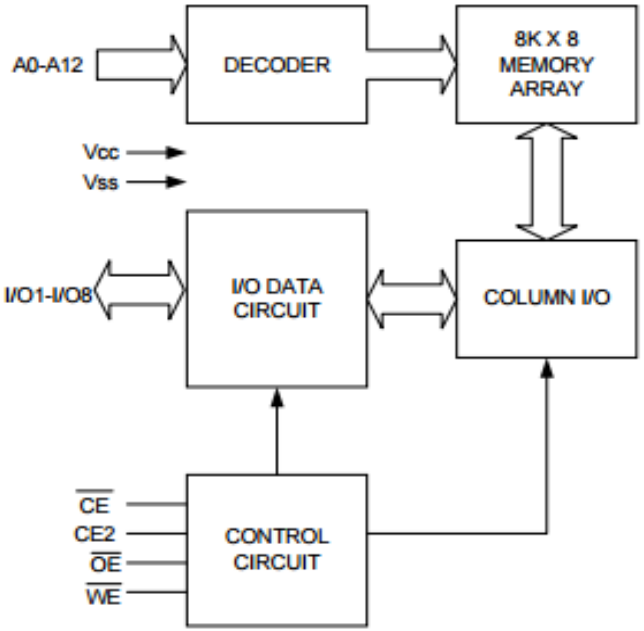


写入数据的过程：

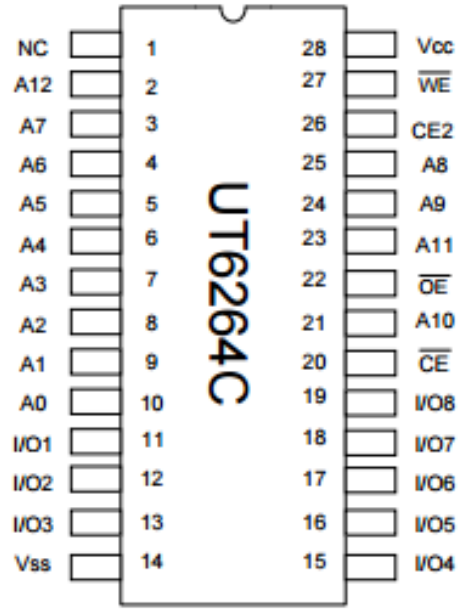
A0～A12上加上要写入单元的地址；在D0～D7上加上要写入的数据；使CS有效；在WE上加上有效的低电平（OE可为高也可为低），这样就将数据写到了地址所选中的单元中。

实例：SRAM6264

FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATION

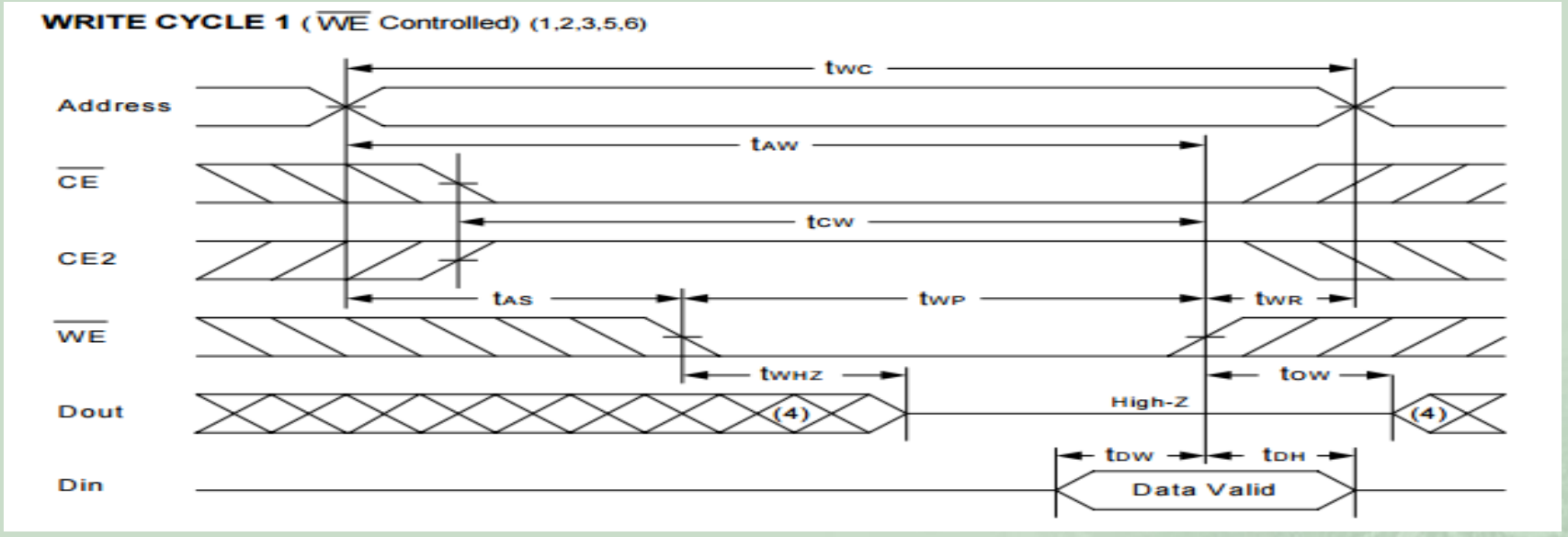


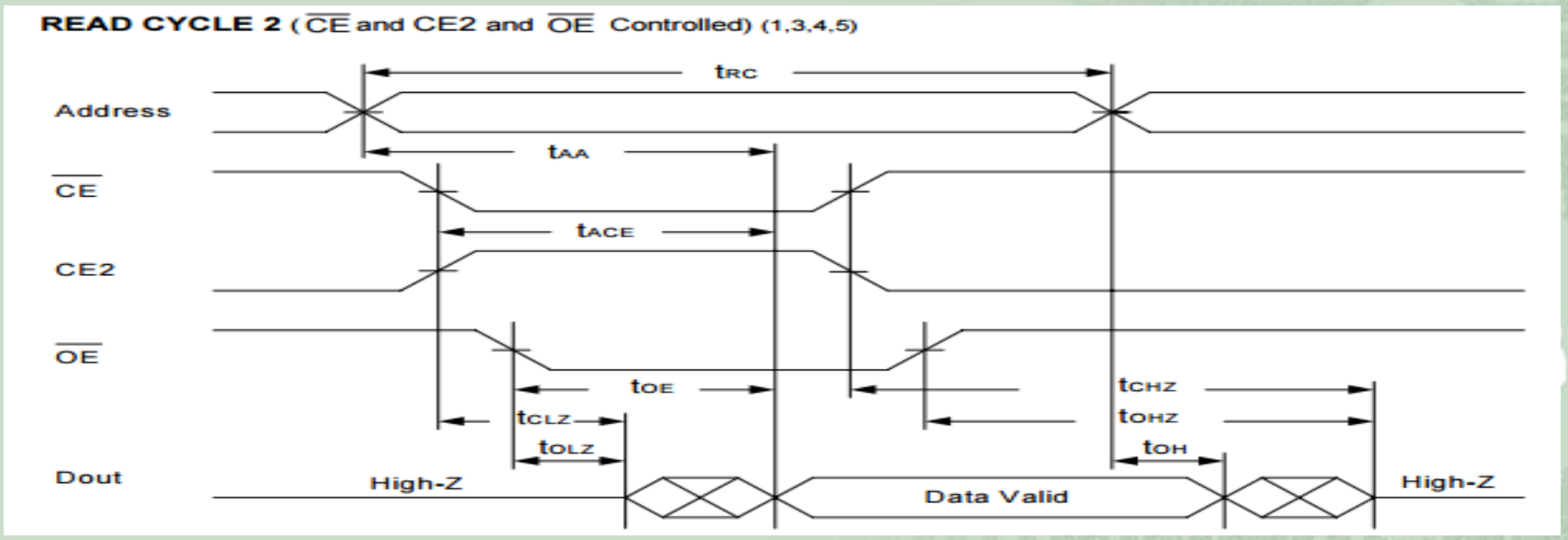
PDIP/SOP

TRUTH TABLE

MODE	/CE1	CE2	/WE	/OE	DQ0~7	Vcc Current
Standby	H	X	X	X	High Z	I_{CCSB}, I_{CCSB1}
	X	L	X	X		
Output Disable	L	H	H	H	High Z	I_{CC}
Read	L	H	H	L	D_{OUT}	I_{CC}
Write	L	H	L	X	D_{IN}	I_{CC}

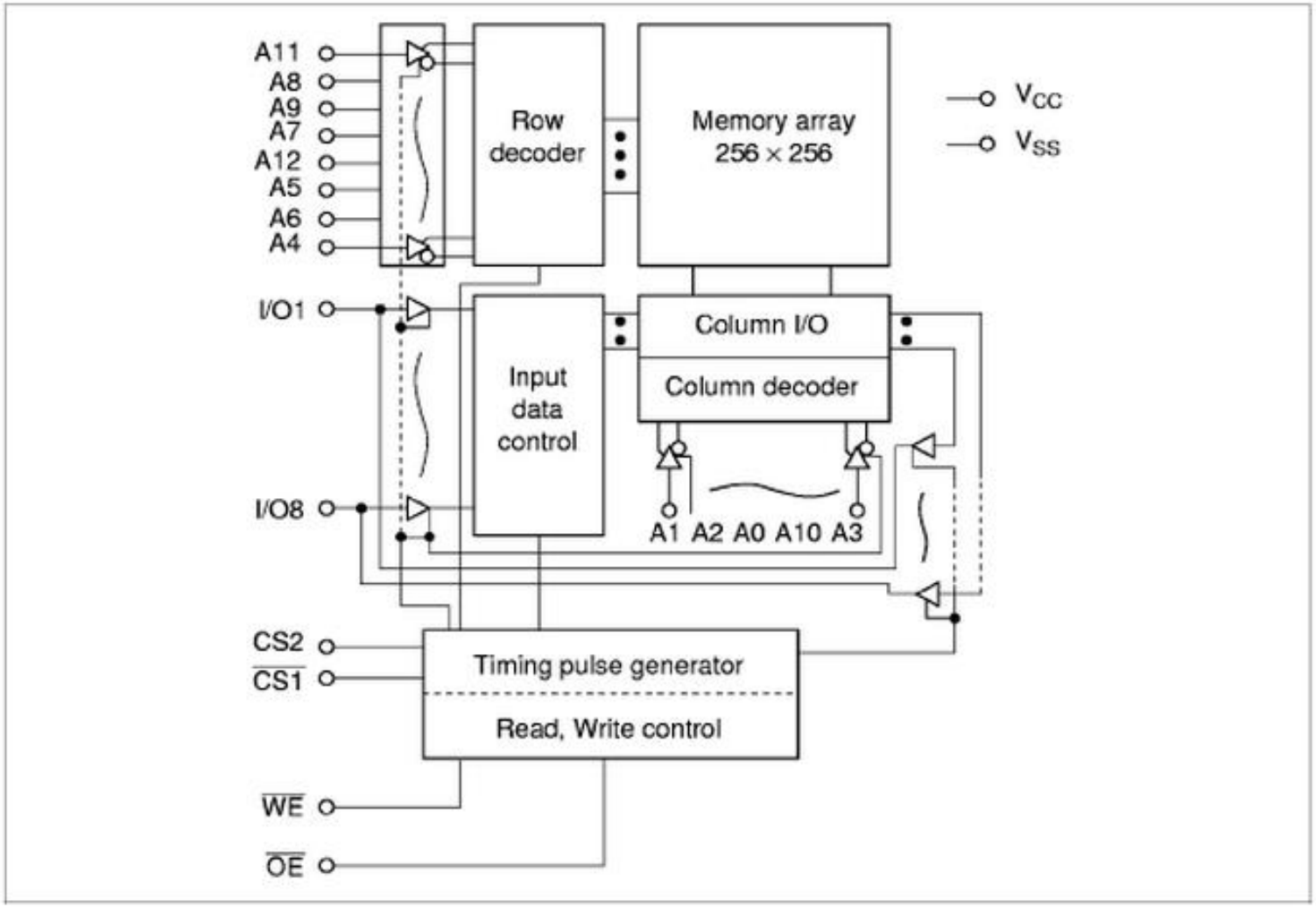
实例：SRAM6264


WRITE CYCLE 1 (WE Controlled) (1,2,3,5,6)
tWC
Address
tAW
CE
tCW
CE2
tAS
tWP
tWR
WE
tWHZ
tOW
Dout
(4)
High-Z
(4)
tDW
tDH
Din
Data Valid


READ CYCLE 2 (CE and CE2 and OE Controlled) (1,3,4,5)
tRC
Address
tAA
CE
tACE
CE2
OE
tOE
tCHZ
tCLZ
tOHZ
tOLZ
tOH
Dout
High-Z
Data Valid
High-Z

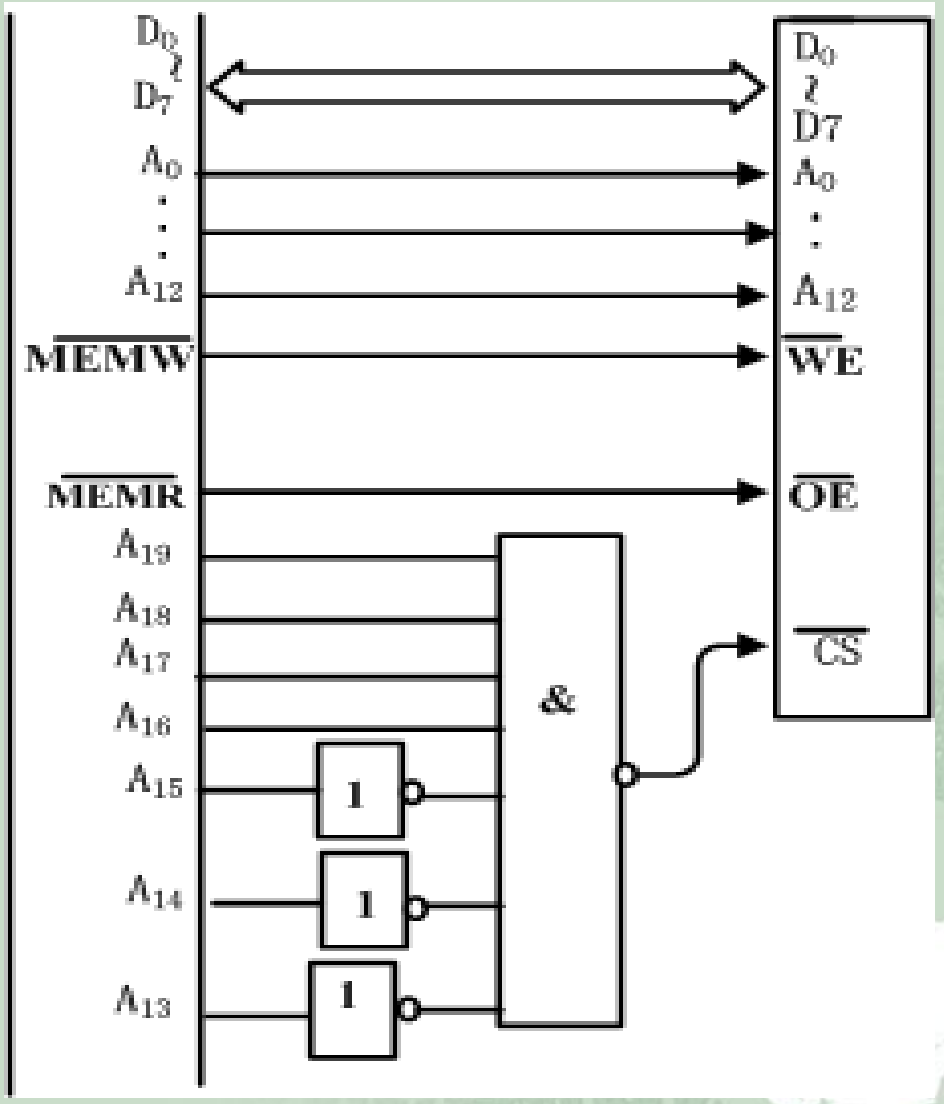
实例：HITACHI 公司的SRAM6264

Block Diagram



- 连接使用

地址分析

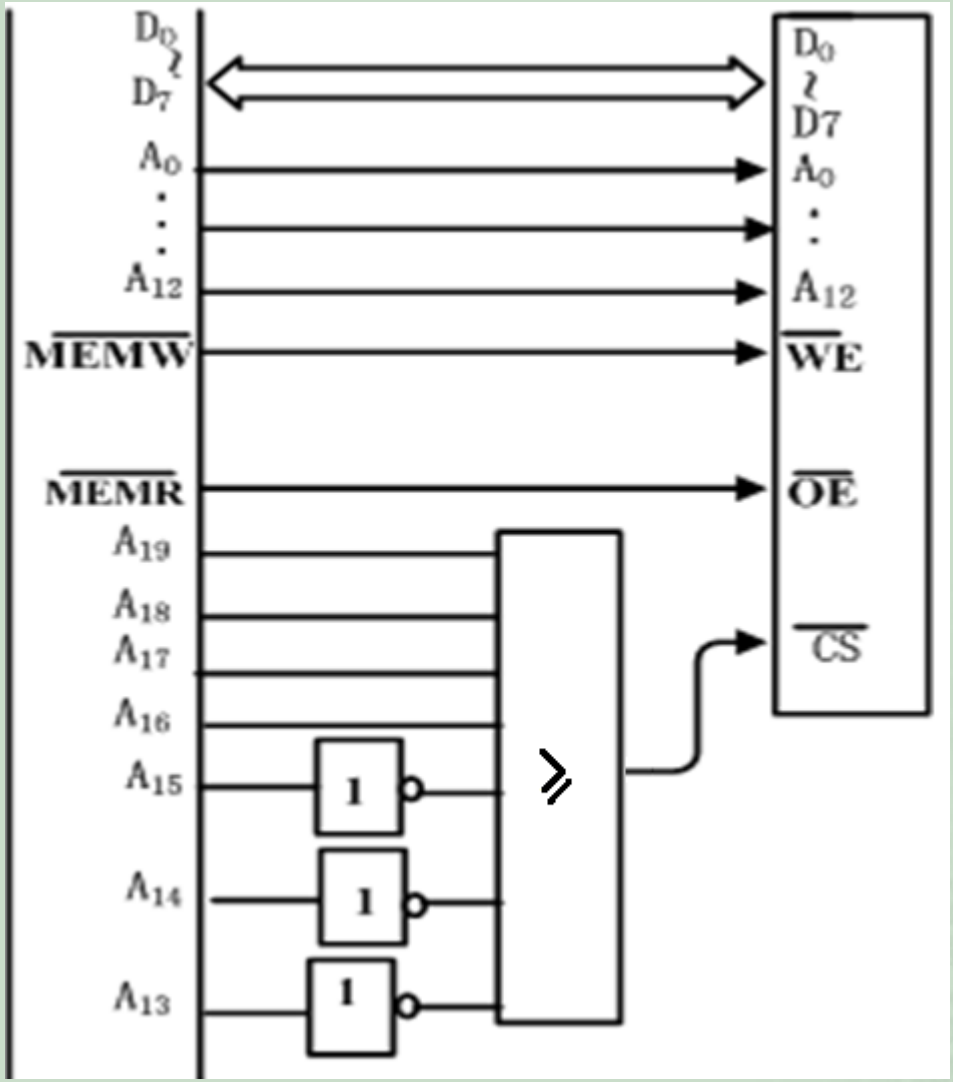


芯片惟一地占据从F0000H～F1FFFH，共8KB内存空间

内存地址分析

高位地址		低位地址（片内地址）				内存地址
A_{19} A_{18} A_{17} A_{16}	A_{15} A_{14} A_{13}	A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
1 1 1 1	0 0 0	0	0 0 0 0	0 0 0 0	0 0 0 0	F0000H
		0	0 0 0 0	0 0 0 0	0 0 0 1	F0001H
		0	0 0 0 0	0 0 0 0	0 0 1 0	F0002H
		0	0 0 0 0	0 0 0 0	0 0 1 1	F0003H
		⋮	⋮	⋮	⋮	⋮
		1	1 1 1 1	1 1 1 1	1 1 0 0	F1FFCH
		1	1 1 1 1	1 1 1 1	1 1 0 1	F1FFDH
		1	1 1 1 1	1 1 1 1	1 1 1 0	F1FFEH
		1	1 1 1 1	1 1 1 1	1 1 1 1	F1FFFH

内存地址空间 F0000H～F1FFFH，共8KB



问题：
分析该芯片占用的内存地址范围？

内存地址分析

高位地址	低位地址（片内地址）				内存地址
$A_{19}A_{18}A_{17}A_{16}$ $A_{15}A_{14}A_{13}$	A_{12}	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	
0 0 0 0 1 1 1	0	0 0 0 0	0 0 0 0	0 0 0 0	0E000H
	0	0 0 0 0	0 0 0 0	0 0 0 1	0E001H
	⋮	⋮	⋮	⋮	⋮
	1	1 1 1 1	1 1 1 1	1 1 1 0	0FFFEH
	1	1 1 1 1	1 1 1 1	1 1 1 1	0FFFFH

内存地址空间 0E000H～0FFFFH，共8KB

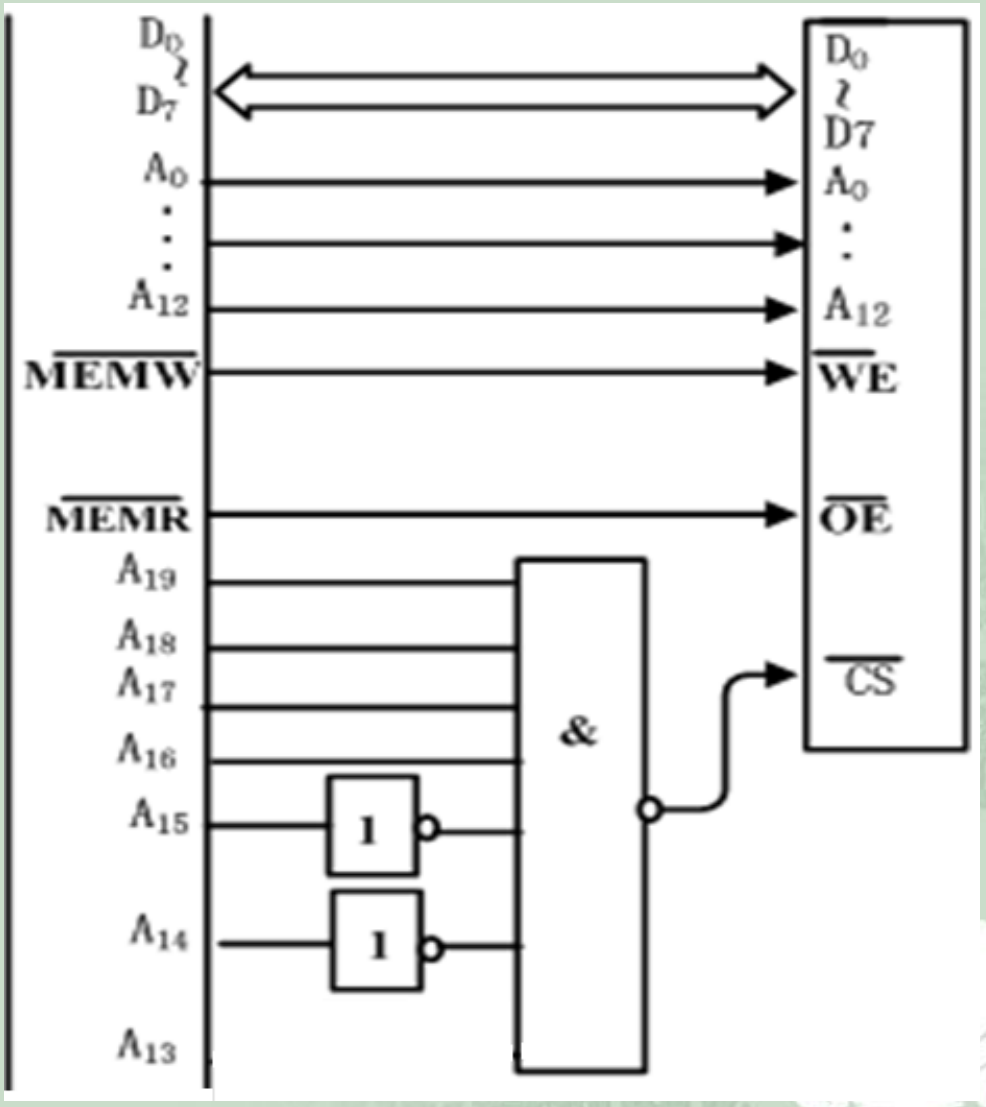
内存地址分析（简化）

高位地址	低位地址（片内地址）				内存地址
$A_{19}A_{18}A_{17}A_{16}$ $A_{15}A_{14}A_{13}$	A_{12}	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	
0 0 0 0 1 1 1	0	0 0 0 0	0 0 0 0	0 0 0 0	0E000H
	0	0 0 0 0	0 0 0 0	0 0 0 1	0E001H
	⋮	⋮	⋮	⋮	⋮
	1	1 1 1 1	1 1 1 1	1 1 1 0	0FFFEH
	1	1 1 1 1	1 1 1 1	1 1 1 1	0FFFFH

高位地址	低位地址（片内地址）				内存地址
$A_{19}A_{18}A_{17}A_{16}$ $A_{15}A_{14}A_{13}$	A_{12}	$A_{11}A_{10}A_9A_8$	$A_7A_6A_5A_4$	$A_3A_2A_1A_0$	
0 0 0 0 1 1 1					

内存地址空间 0E000H～0FFFFH，共8KB

问题：
分析该芯片占用的内存地址范围？



- 常用地址译码方法

全地址译码

- ✓ 全地址译码是除了地址总线中参与片内寻址的低位地址线外，其余所有高位地址线全部参与片间地址译码的方法。
- ✓ 全地址译码法不会产生地址码重叠的存储区域，对译码电路要求较高。

部分地址译码

- ✓ 部分地址译码是线选法和全译码相结合的方法，即利用高位地址线译码产生片选信号时，有的地址线未参加译码。
- ✓ 部分地址译码会产生地址码重叠的存储区域。

内存地址分析（部分地址译码）

A13未参加译码，分为A13=0 和 A13=1 两种情况分析

高位地址			低位地址（片内地址）				内存地址
A_{19} A_{18} A_{17} A_{16}	A_{15} A_{14} A_{13}	A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0		
1 1 1 1	0 0	X					

0 *F0000H～F1FFFH*

1 *F2000H～F3FFFH*

产生两个重叠区

内存地址空间 F0000H～F1FFFH 和 F2000H～F3FFFH，各8KB，共16KB

- ***存储器的扩展***

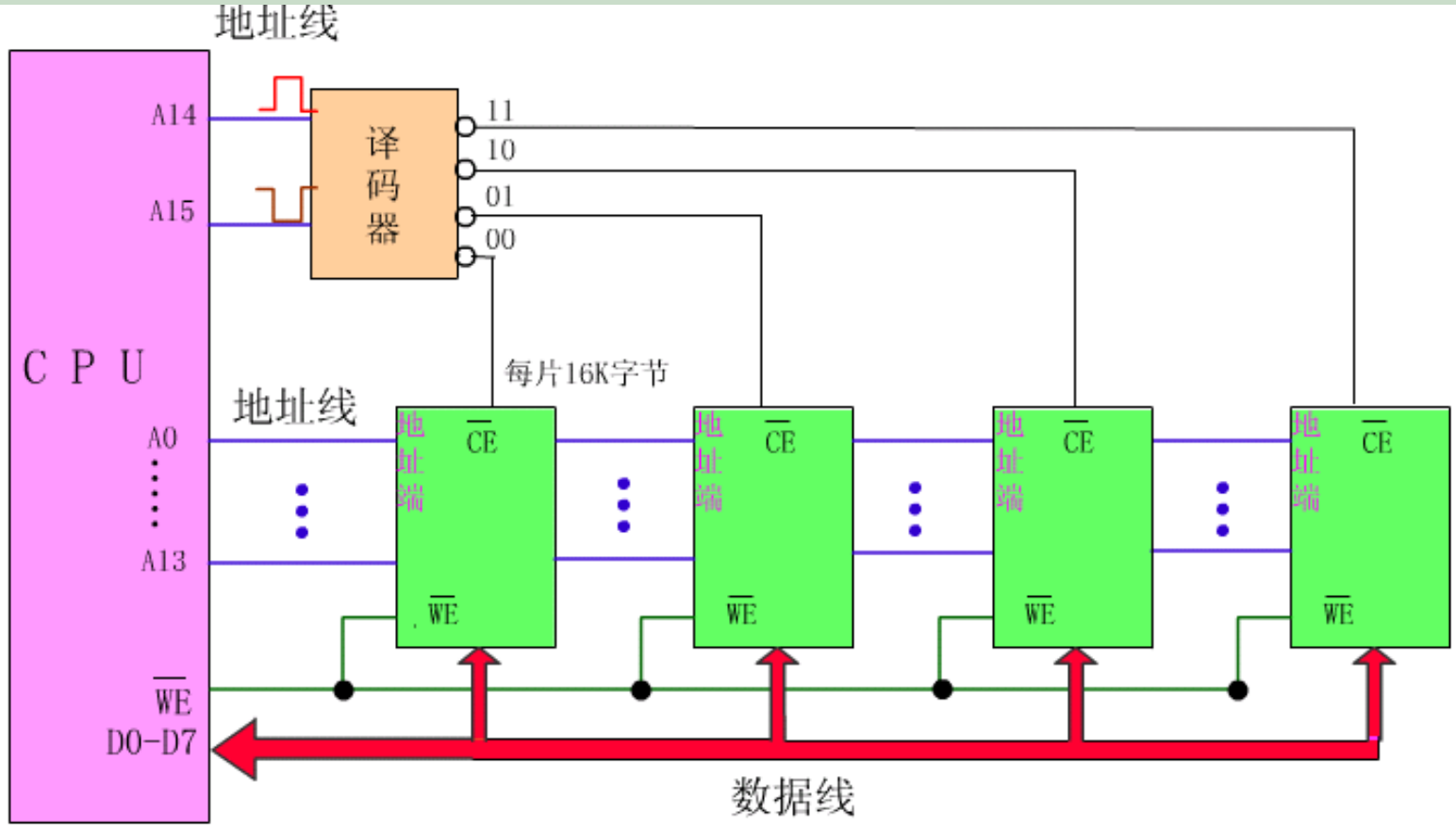
存储器容量扩展 ---- 用多片存储器芯片构成主存储器

(1)**字扩展**，即存储单元数的扩展

(2)**位扩展**，即每个存储单元二进制位数的扩展

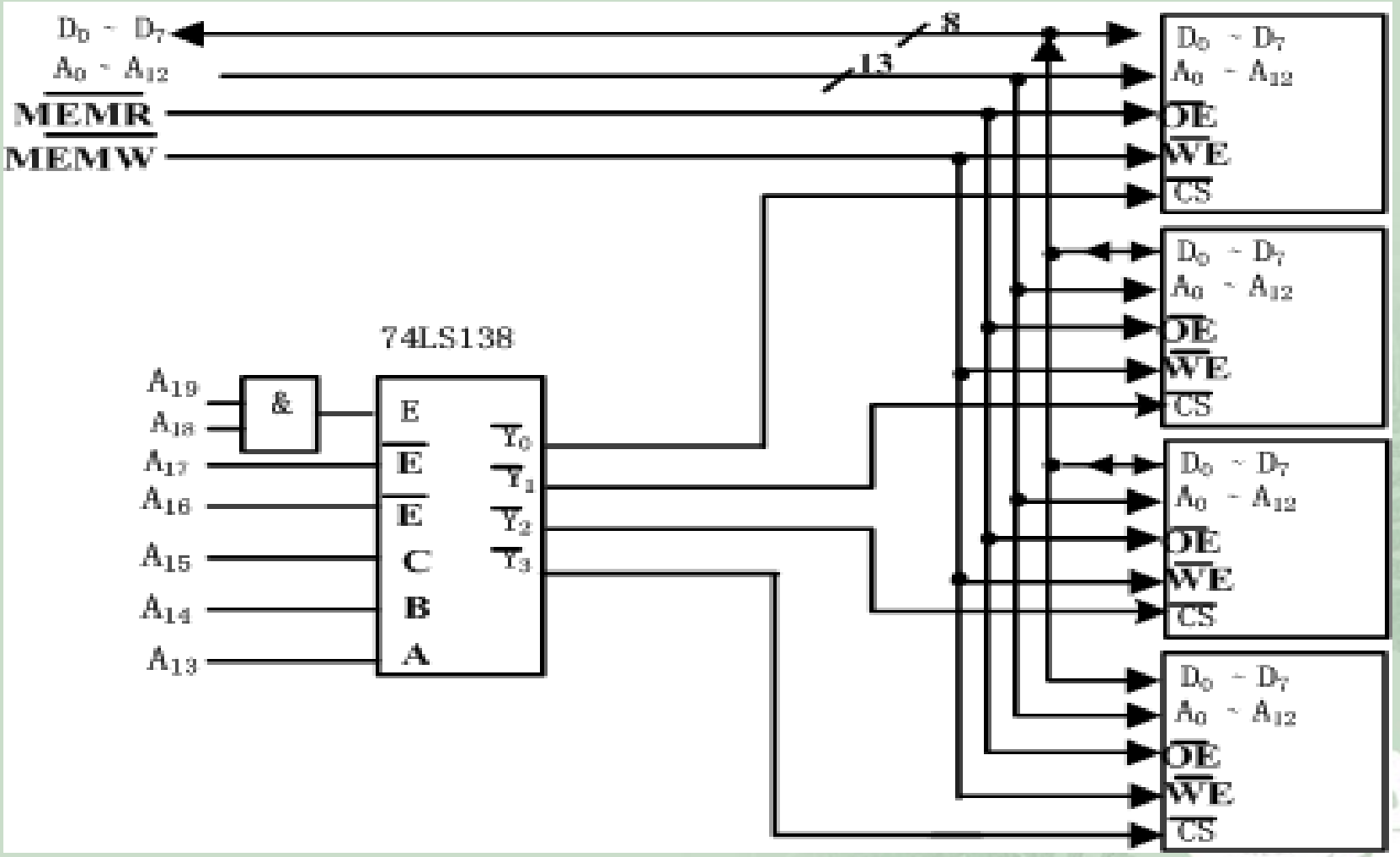
(3) 字位同时扩展

① 字扩展



字扩展法组成64K RAM

例：4片8K×8b SRAM的字扩展



占据内存地址C0000H ～ C7FFFH，共32KB内存空间

内存地址分析（字扩展）

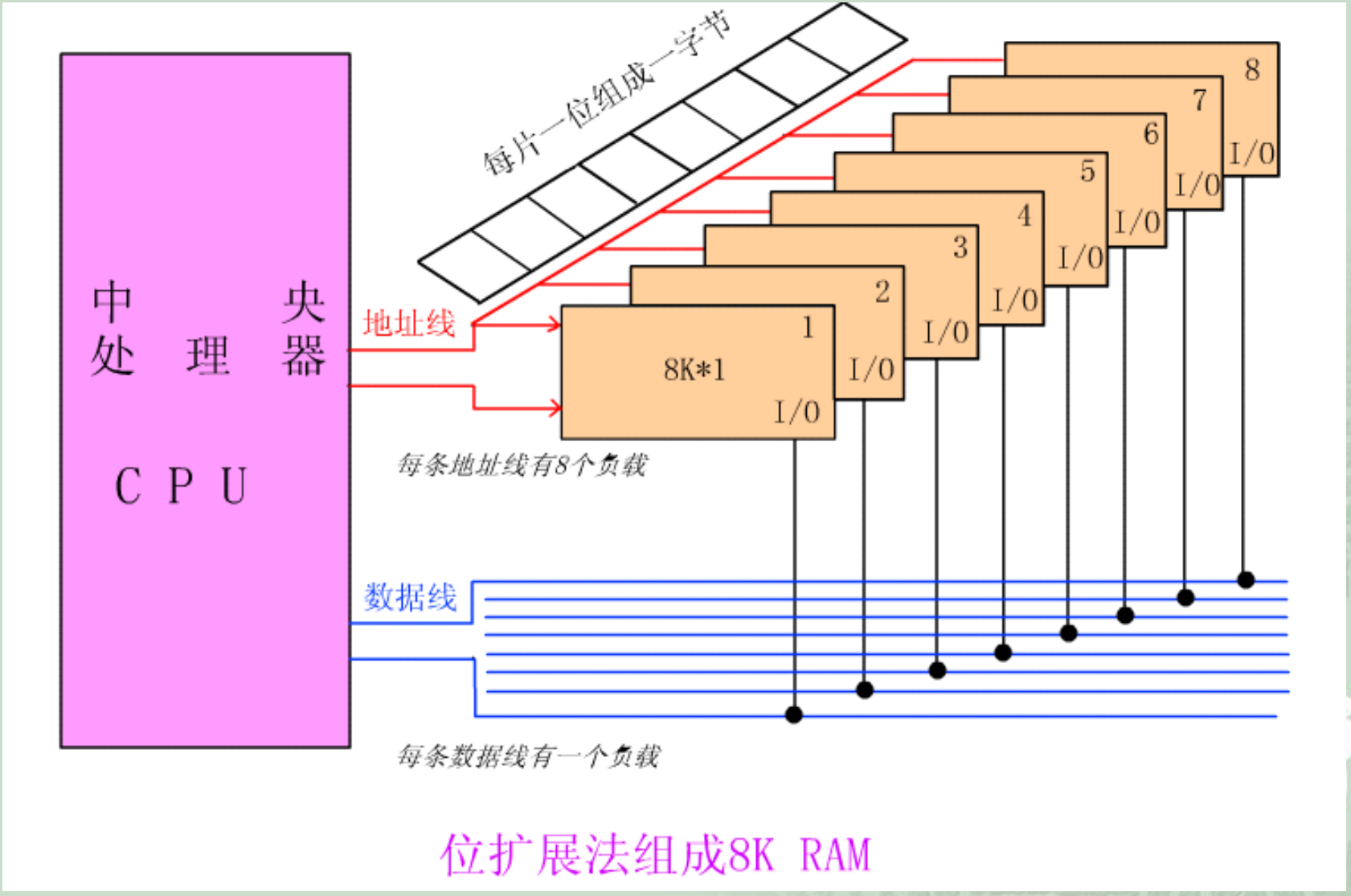
高位地址		低位地址（片内地址）	内存地址
$A_{19}A_{18}A_{17}A_{16}$	$A_{15}A_{14}A_{13}$	A_{12} $A_{11}A_{10}A_9A_8$ $A_7A_6A_5A_4$ $A_3A_2A_1A_0$	
1 1 0 0	0 0 0		/Y0
	0 0 1		/Y1
	0 1 0		/Y2
	0 1 1		/Y3

片选择

/Y0　C0000H～C1FFFH，8KB
/Y1　C2000H～C3FFFH，8KB
/Y2　C4000H～C5FFFH，8KB
/Y3　C6000H～C7FFFH，8KB

占据内存地址 C0000H ～ C7FFFH，共32KB内存空间

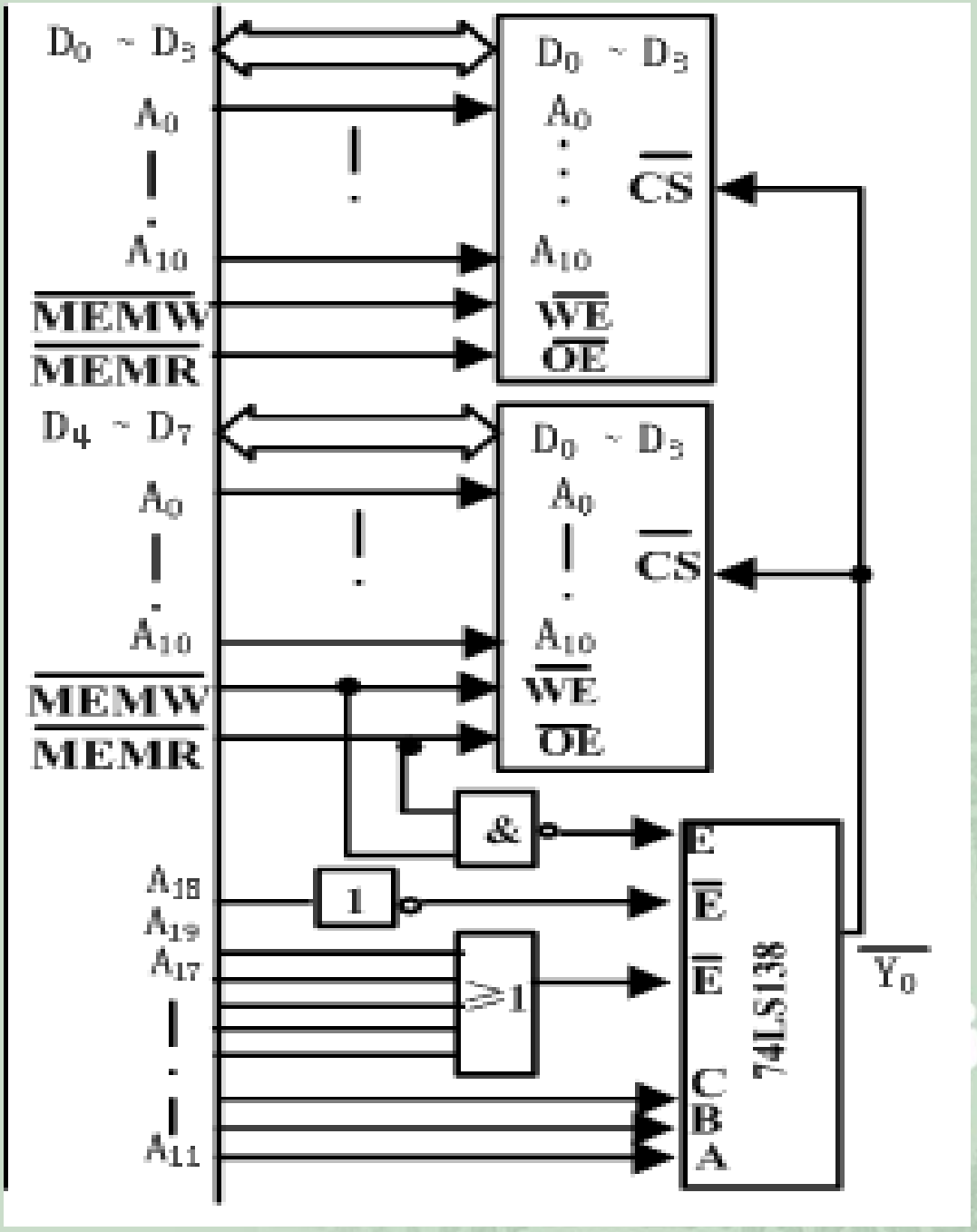
② 位扩展



位扩展法组成8K RAM

例：

2片2K×4b SRAM的位扩展



占据内存地址 40000H～407FFH，共2KB内存空间

内存地址分析

高位地址		低位地址（片内地址）		内存地址
A_{19} A_{18} A_{17} A_{16}	A_{15} A_{14} A_{13} A_{12} A_{11}	A_{10} A_9 A_8	A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0	
0 1 0 0	0 0 0 0 0			

内存地址空间 40000H～407FFH，共2KB

例：2片2K×4b SRAM的位扩展，占据40000H ～ 407FFH，共2KB内存空间，编程对该内存进行检测。

分析：对内存单元40000H的检测

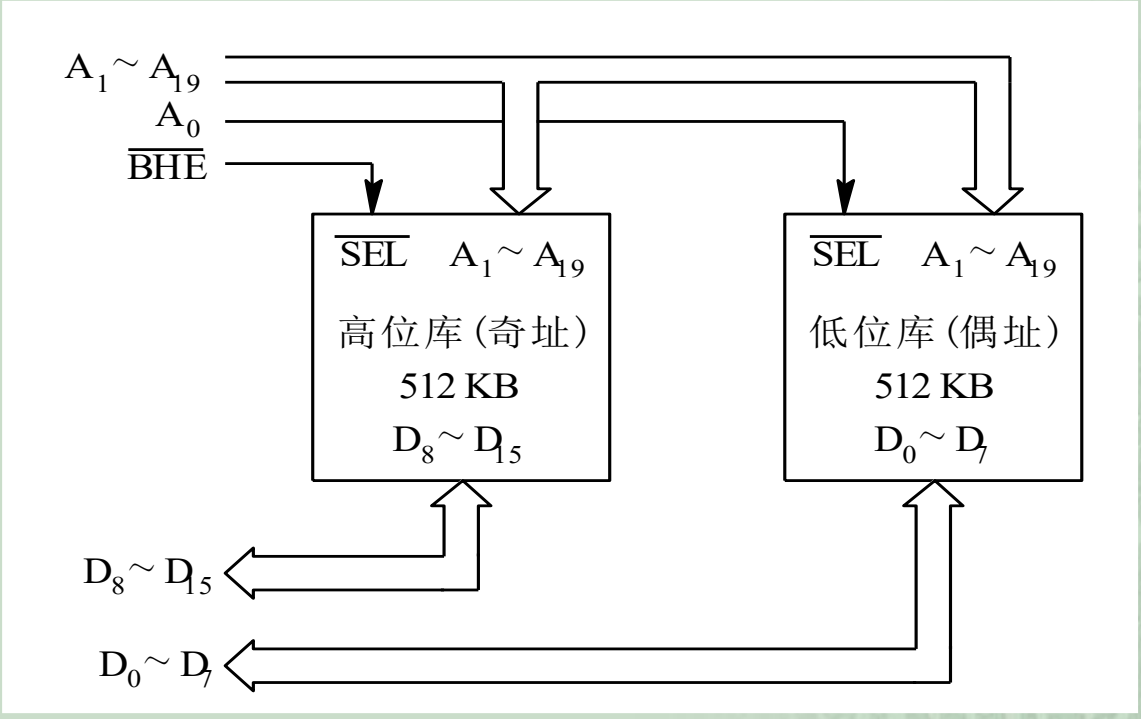
```
MOV  AX, 4000H
MOV  DS, AX
MOV  SI, 0000H  ; 初始化 DS:SI 为内存地址40000H

MOV  [SI], 55H   ; 将 55H 写入40000H内存单元
MOV  AL, [SI]    ; 读出 40000H内存单元的内容
CMP  AL, 55H     ; 将读出值与55H进行比较
JNE  ERROR       ; 比较结果不相等，则为出错
```

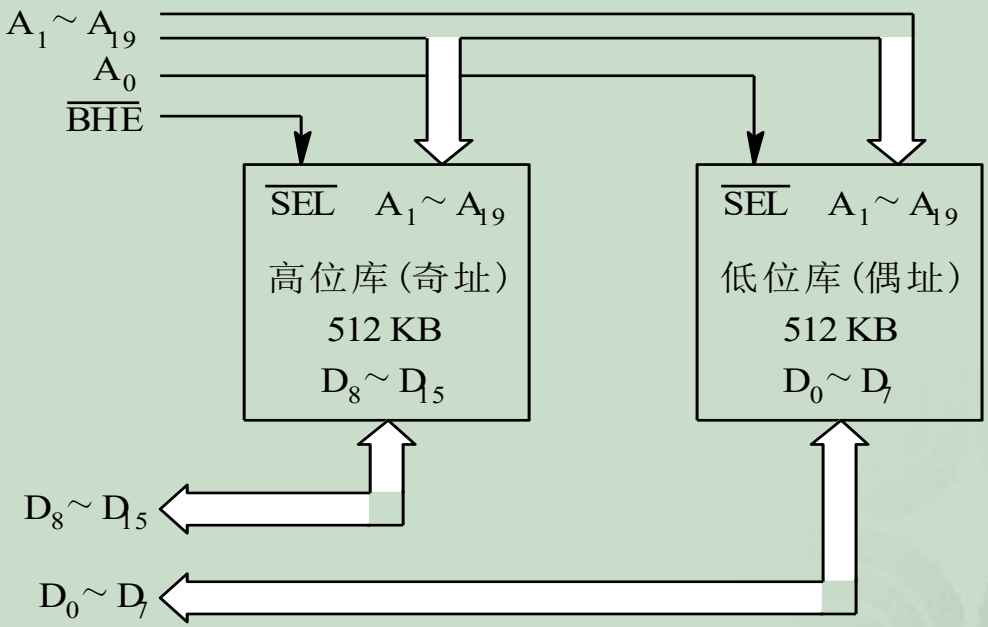
- *80x86内存的连接*

16位机、32位机的存储器结构比较复杂，内存按字节编址，必须分为2个或4个体，在存储器的连接上也有许多不同。

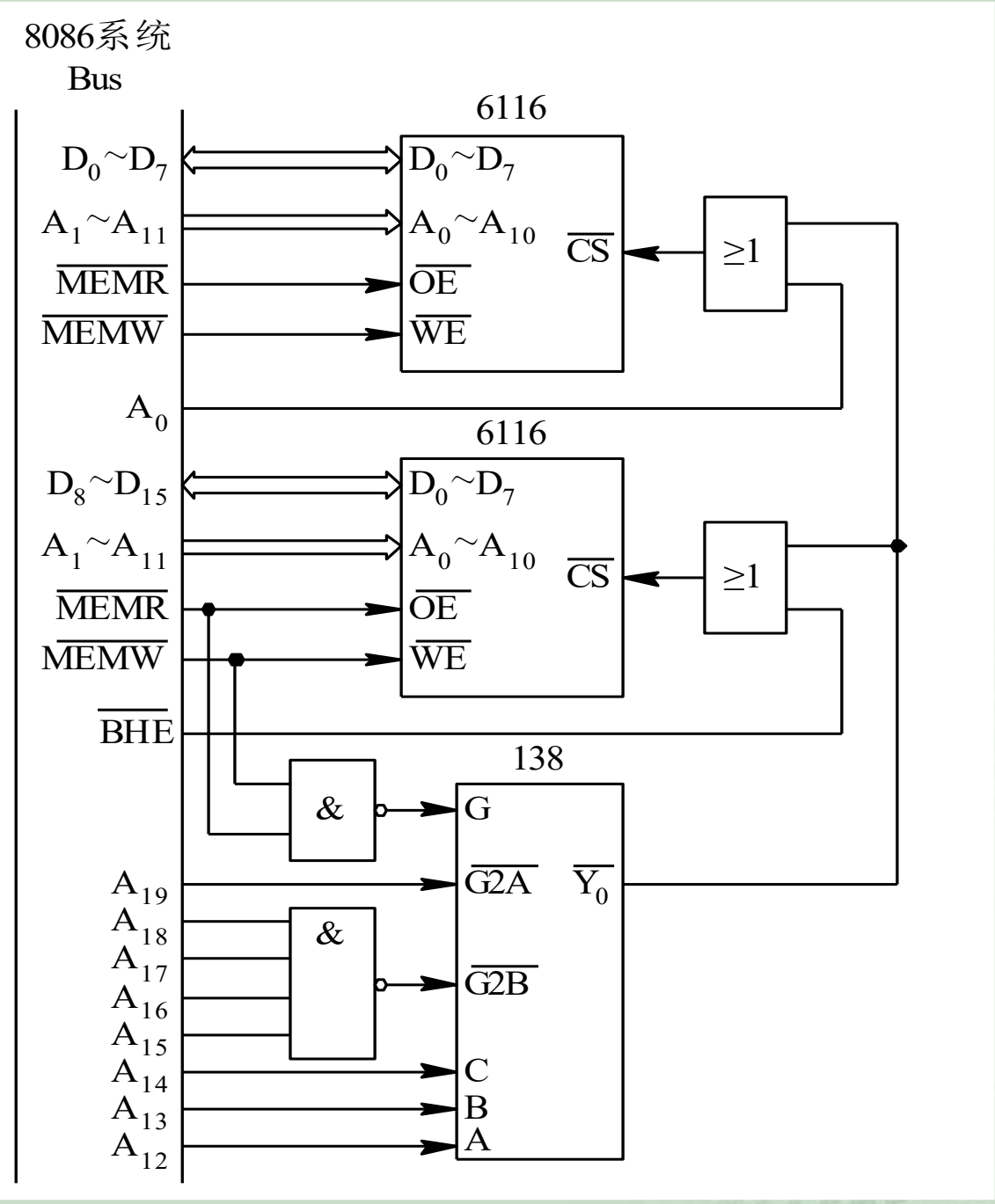
例：16位机的内存组织（分为2个体）



例：8086存贮器的高低位库

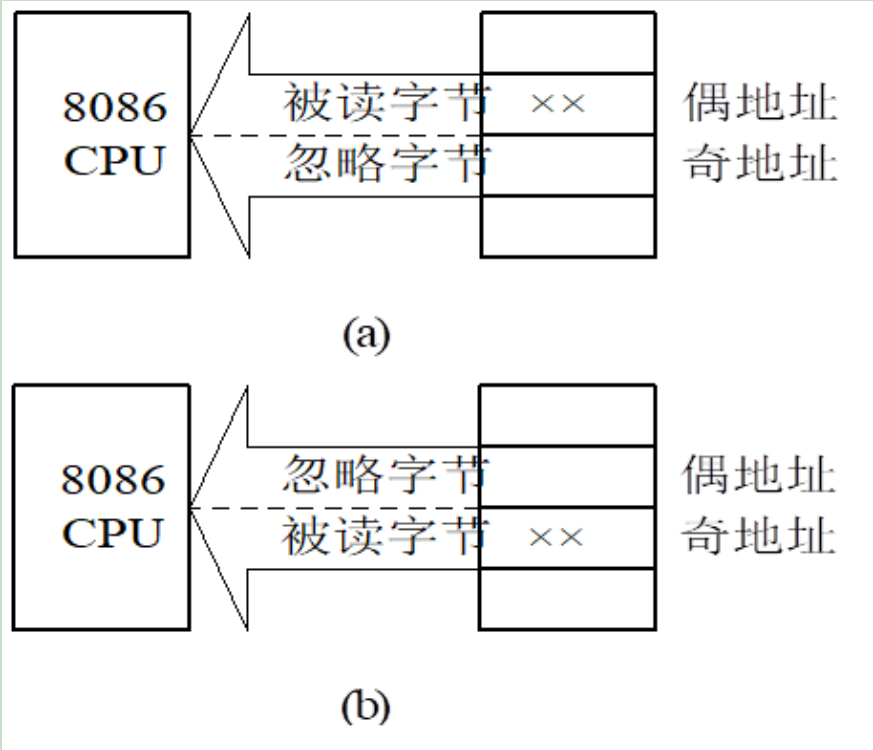


$\overline{BHE}$	A_0	读/写的字节
0	0	读/写高低两个字节
0	1	读/写高位字节(奇地址)
1	0	读/写低位字节(偶地址)
1	1	不传送



两片 6116 与 16 位总线的连接图

- 在8086系统中，存储器这种分体结构对用户来说是透明的。
- 当用户需要访问存储器中**某个存储单元**，以便进行字节型数据的读/写操作时，该地址可能是偶地址，也可能是奇地址。
  - ✓ 如果是**偶地址**(A_0=0)，这时系统将自动产生A_0=0，选定偶地址存储体（*同时,$\overline{BHE}$=1奇地址存储体不会被选中，也就不会启动它*），通过A_{19}~A_1从偶地址存储体中选中某个单元，并启动该存储体，读/写该存储单元中一个字节信息，通过数据总线的低8位传送数据；
  - ✓ 如果是**奇地址**(A_0=1)，*则偶地址存储体不会被选中，也就不会启动它*。启动奇地址存储体，系统自动产生$\overline{BHE}$=0，作为奇地址存储体的选体信号，与A_{19}~A_1一起选定奇地址存储体中的某个存储单元，并读/写该单元中的一个字节信息，通过数据总线的高8位传送数据。



- 可以看出，对于字节型数据，不论它存放在偶地址的低位库，还是奇地址的高位库，都可通过一个总线周期完成数据的读/写操作（8086CPU实际进行的是一次字操作）。

- 如果用户需要访问存储器中某两个存储单元，以便进行字型数据的读/写时，分两种情况。
  - ✓ 一种情况是，访问从偶地址开始的两个连续存储单元（即字的低字节在偶地址单元，高字节在奇地址单元），这种存放称为规则存放，这样存放的字称为规则字。

    对于规则字可通过一个总线周期完成读/写操作，这时 $A_0=0$，$\overline{BHE}=0$；
  - ✓ 另一种情况是。访问从奇地址开始的两个存储单元（即字的低字节在奇地址单元，高字节在偶地址单元），这种存放称为非规则存放，这样存放的字称为非规则字。

对于非规则字需要通过两个总线周期才能完成读/写操作，即第一次访问存储器时读/写奇地址单元中的字节，第二次访问存储器时读/写偶地址单元中的字节 。

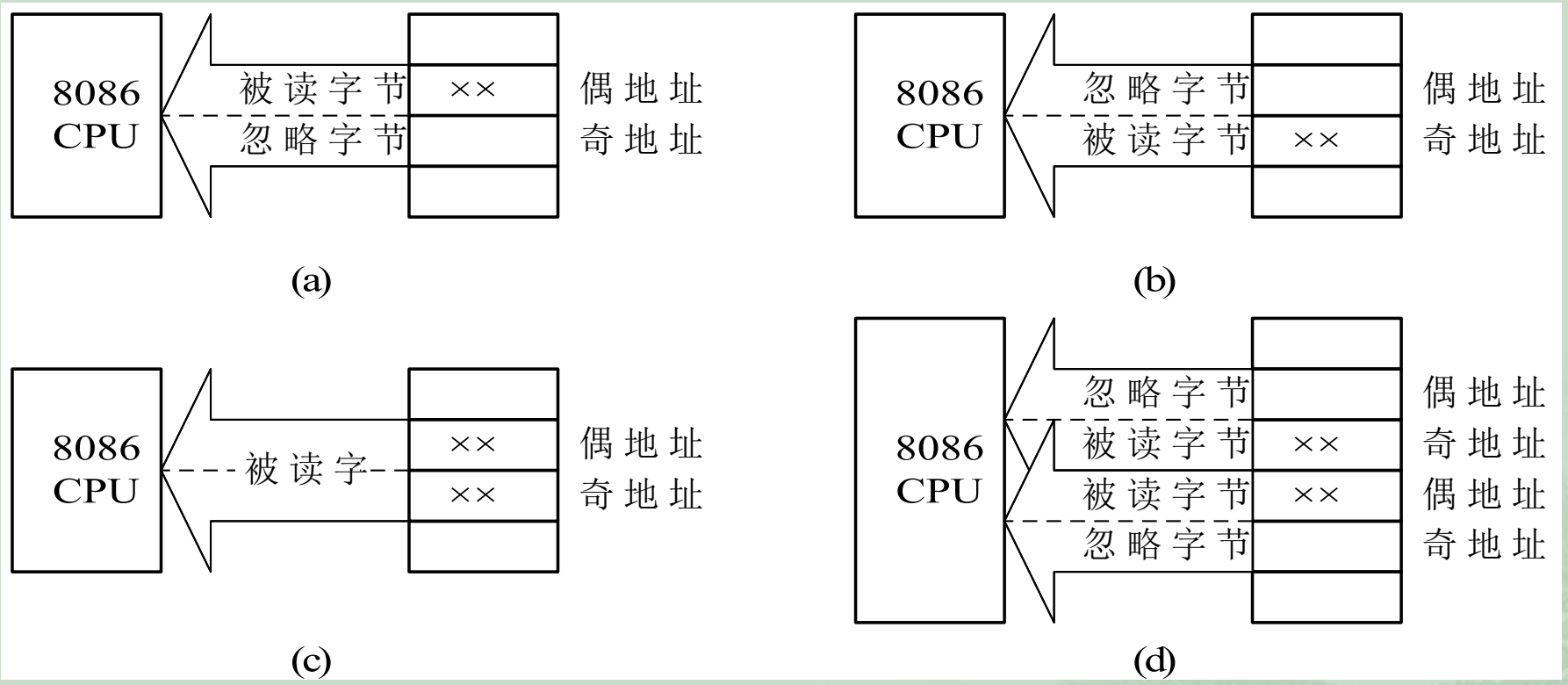


图　8086存储器的偶地址和奇地址读字节和字

（a）读偶地址单元中的字节；　（b）读奇地址单元中的字节；

（c）读偶地址单元中的字；　　（d）读奇地址单元中的字

- 显然，为了加快程序的运行速度，希望字型数据在存储器中规则存放。

- 在8088系统中，可直接寻址的存储空间同样也是1 MB，但其存储器的结构与8086有所不同，它的1MB 存储空间同属于一个单一的存储体，即存储体为1 M×8位。
- 它与总线之间的连接方式很简单，其20根地址线A_{19}~A_0与8根数据线分别与8088 CPU对应的地址线和数据线相连。
- 8088 CPU每访问一次存储器只能读/写一个字节信息，因此在8088系统的存储器中，字型数据需要两次访问存储器才能完成读/写操作。

例：32位机的内存组织（分为4个体）

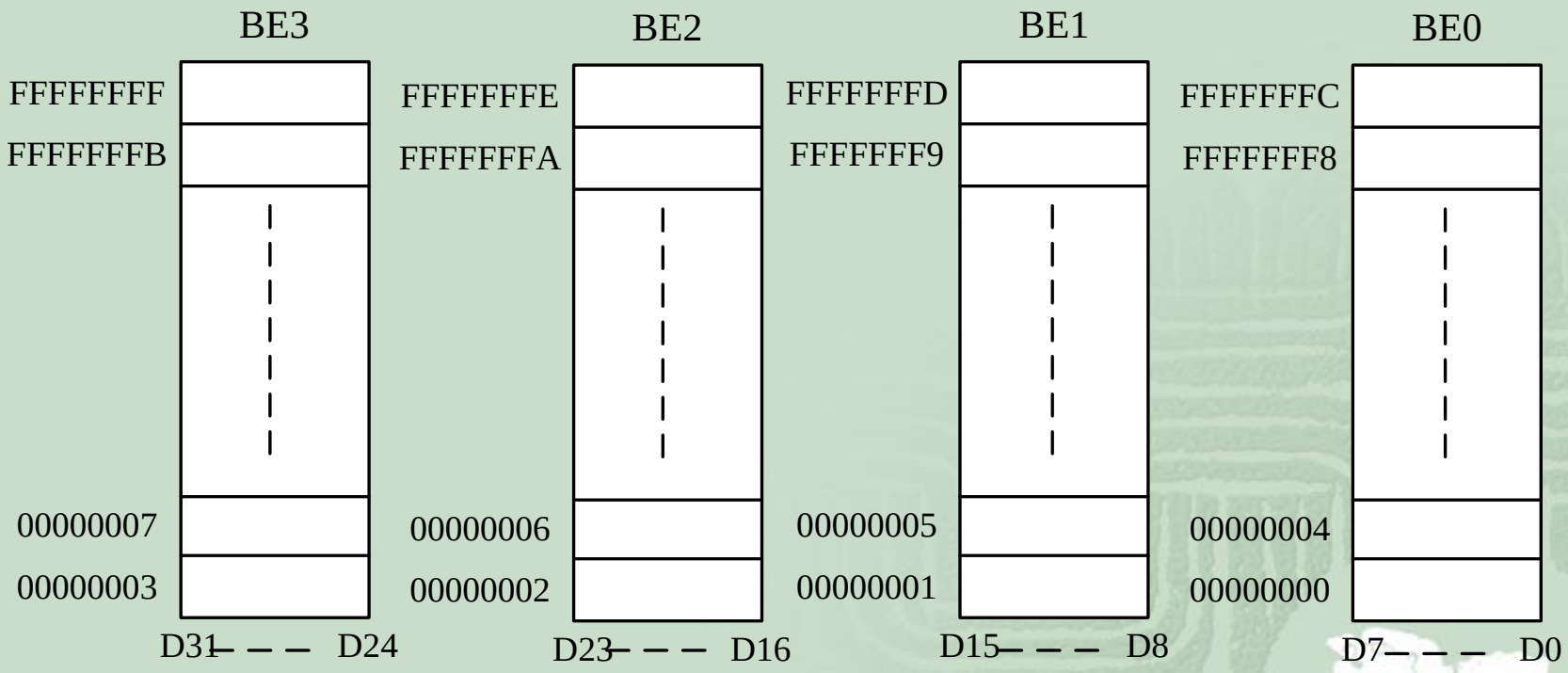
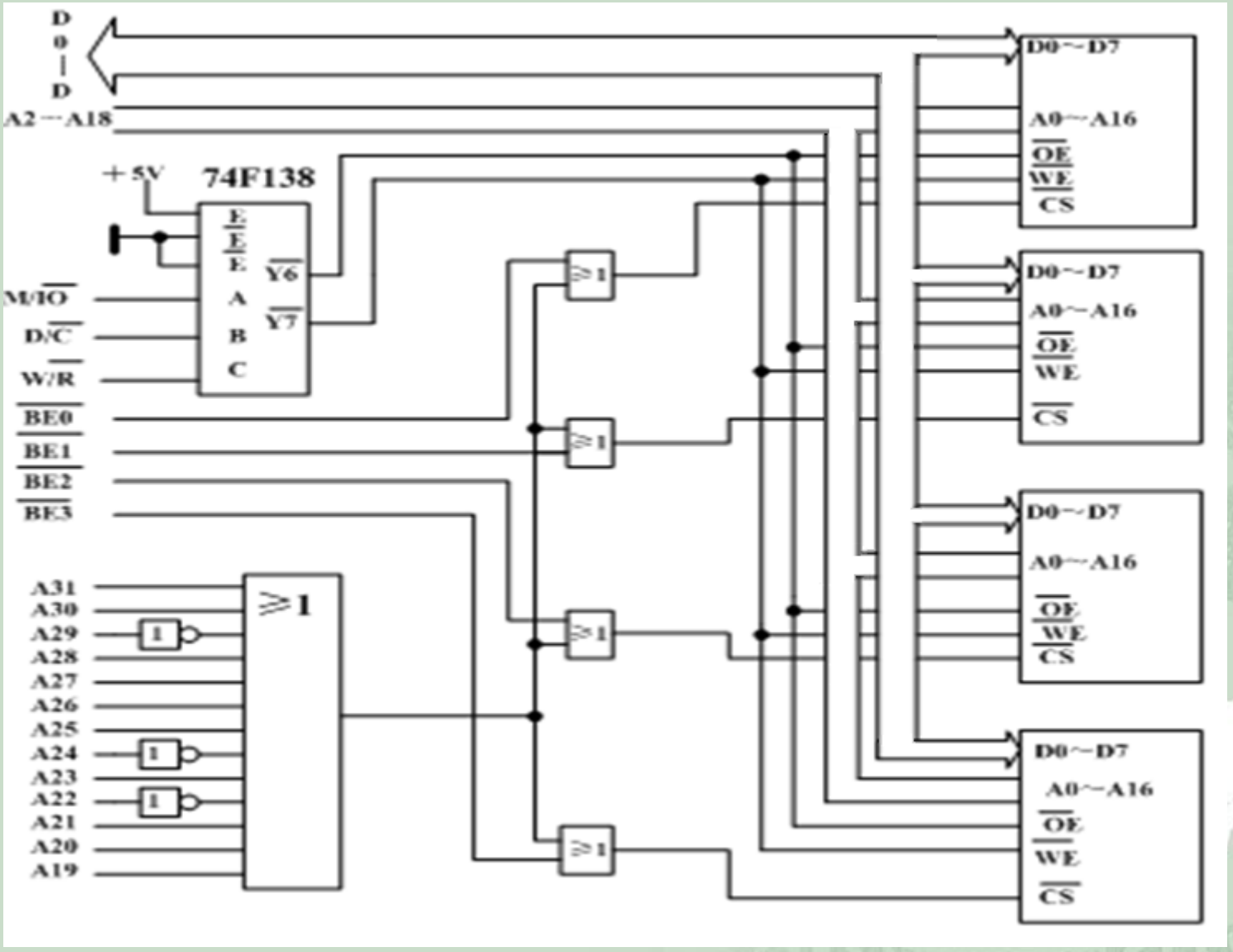


图7.11 32位机的内存组织

例：32位机的内存体选择信号$\overline{BE0}$~$\overline{BE3}$

字节允许				读/写的字节			
$\overline{BE_3}$	$\overline{BE_2}$	$\overline{BE_1}$	$\overline{BE_0}$	D_{24}~D_{31}	D_{16}~D_{23}	D_8~D_{15}	D_0~D_7
1	1	1	0				D_0~D_7
1	1	0	1			D_8~D_{15}	
1	0	1	1		D_{16}~D_{23}		
0	1	1	1	D_{24}~D_{31}			
1	1	0	0			D_8~D_{15}	D_0~D_7
1	0	0	1		D_{16}~D_{23}	D_8~D_{15}	
0	0	1	1	D_{24}~D_{31}	D_{16}~D_{23}	D_{24}~D_{31}	D_{16}~D_{23}
1	0	0	0		D_{16}~D_{23}	D_8~D_{15}	D_0~D_7
0	0	0	1	D_{24}~D_{31}	D_{16}~D_{23}	D_8~D_{15}	
0	0	0	0	D_{24}~D_{31}	D_{16}~D_{23}	D_8~D_{15}	D_0~D_7

例：四片SRAM与32位总线的连接图



- Pentium处理器的内存组织

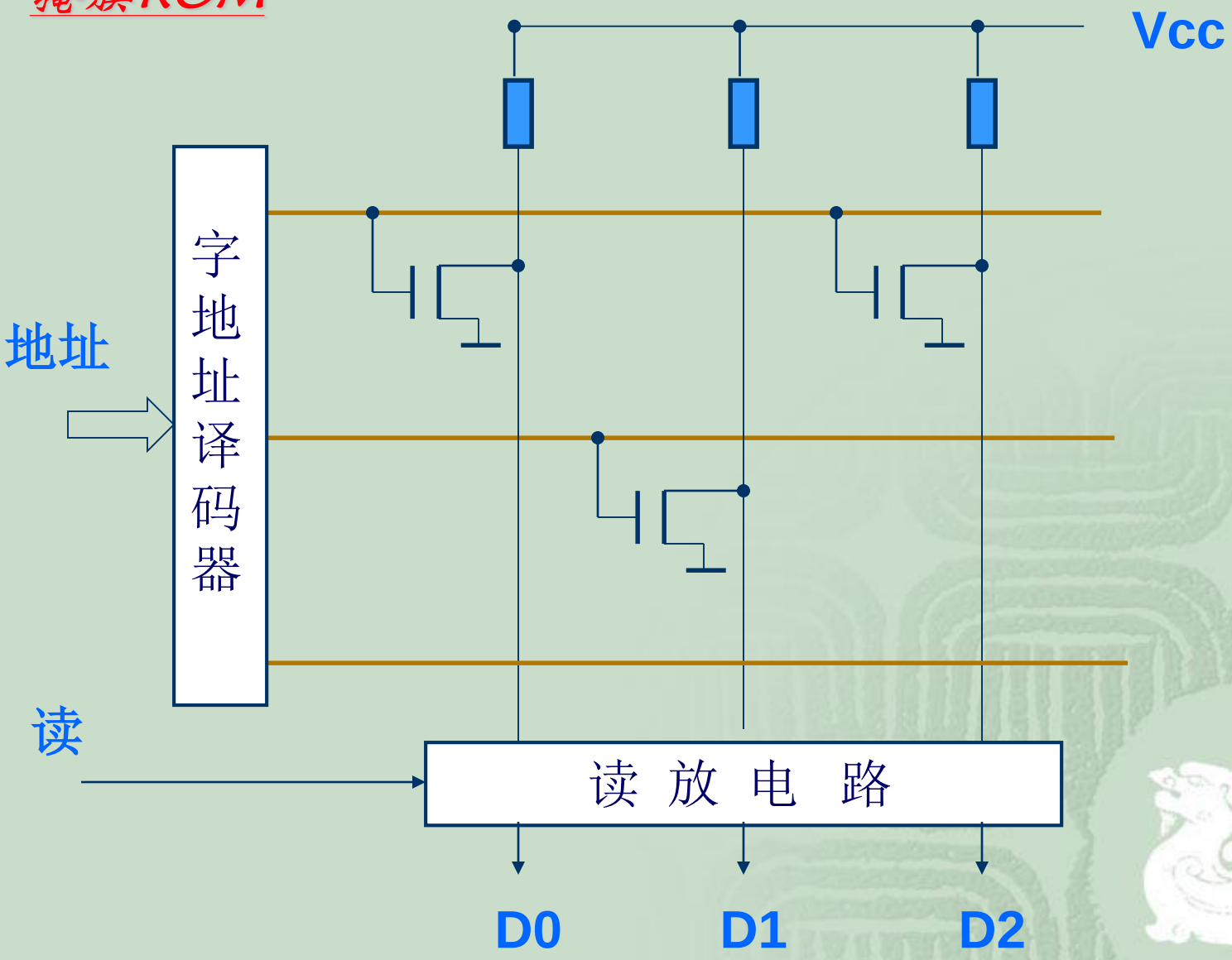
  Pentium处理器有几百条引线，其中与访问内存有关的是地址信号A3～A31、体选择信号$\overline{BE0}$～$\overline{BE7}$、64位数据信号D0～D63、及控制信号M/$\overline{IO}$、D/$\overline{C}$和W/$\overline{R}$。

7.2.2 只读存储器ROM

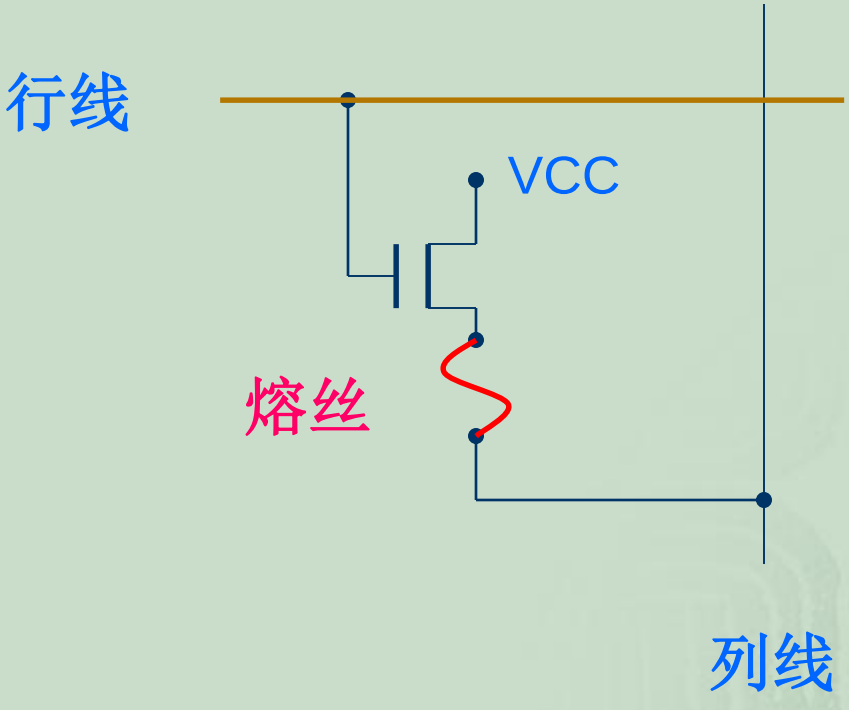
（非易失性半导体存储器）

1. 只读存储器ROM：掩膜ROM
2. 一次性编程只读存储器PROM
3. 可擦可编程只读存储器EPROM
4. 电可擦可编程只读存储器E^2PROM
5. 快擦除读写存储器（Flash Memory)： 闪存

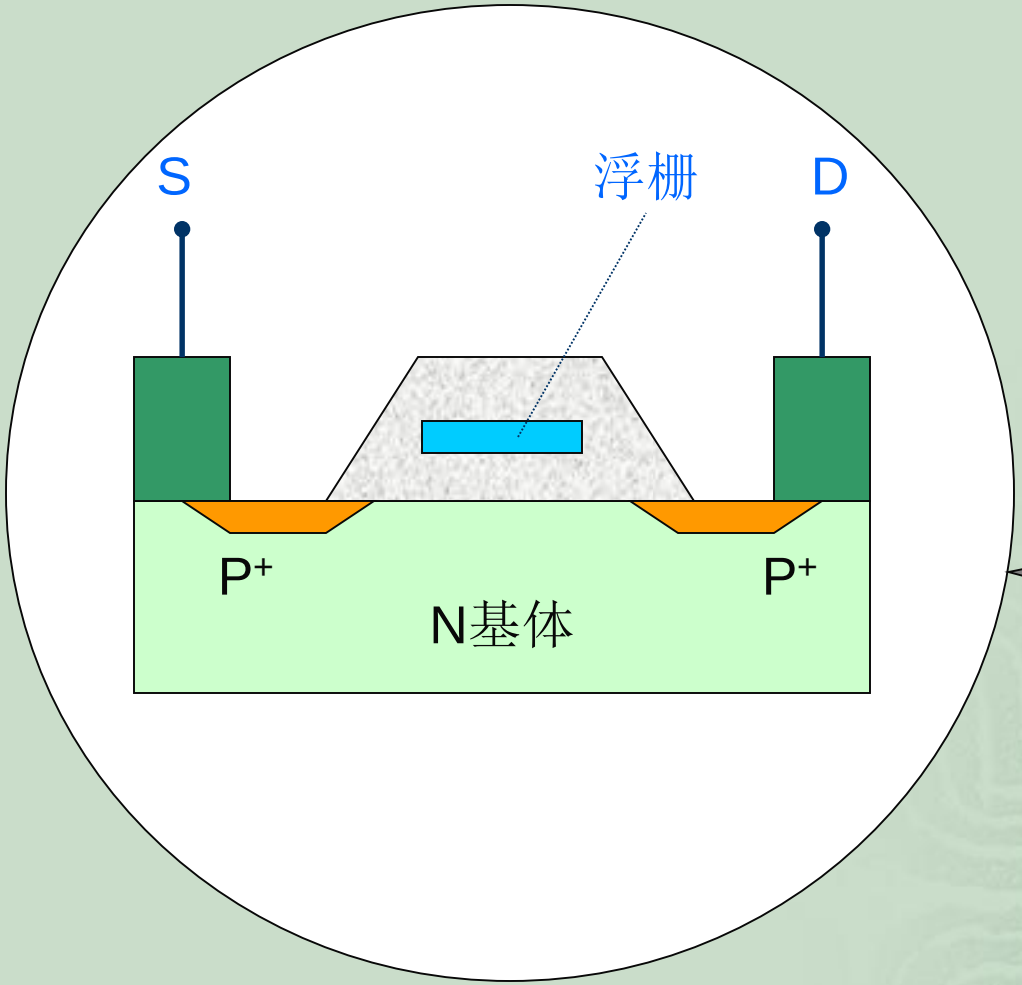
掩膜ROM

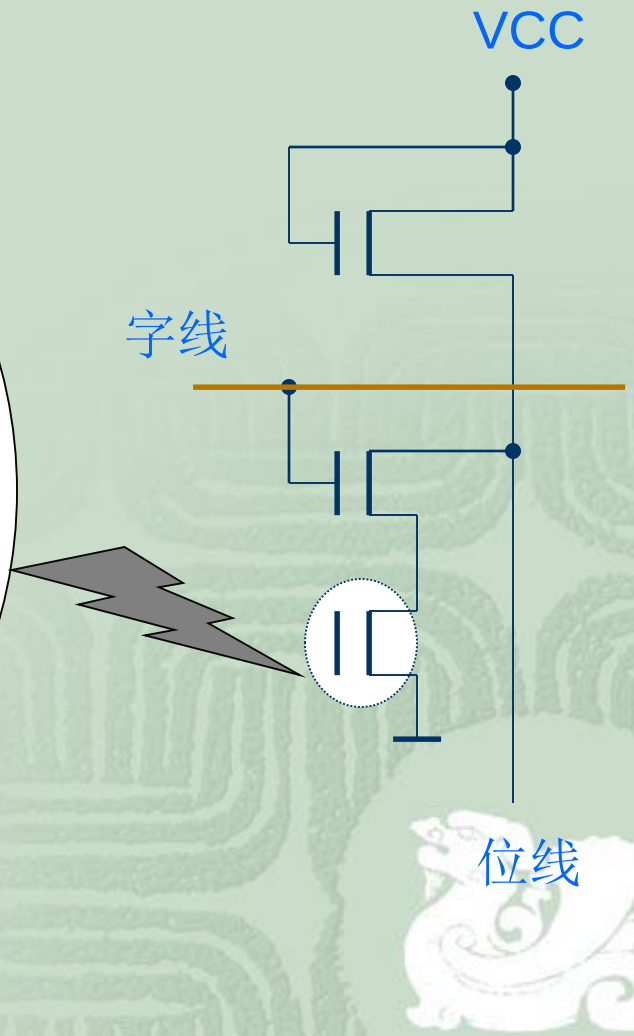


PROM

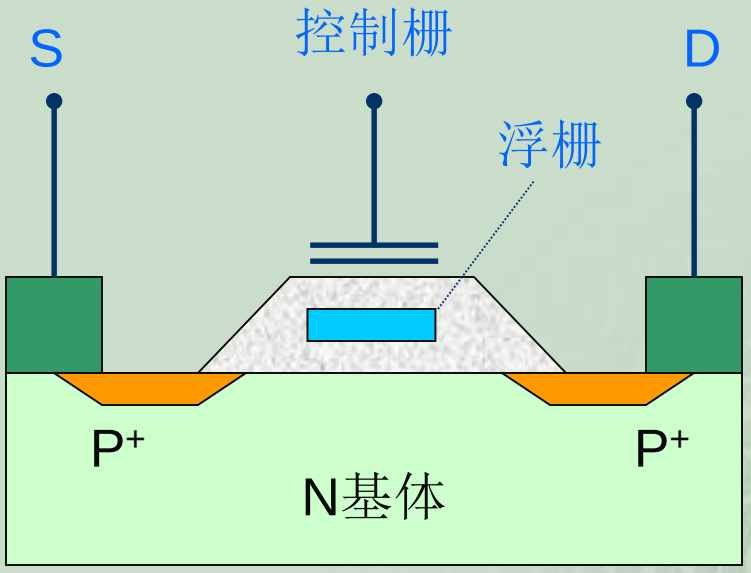


EPROM

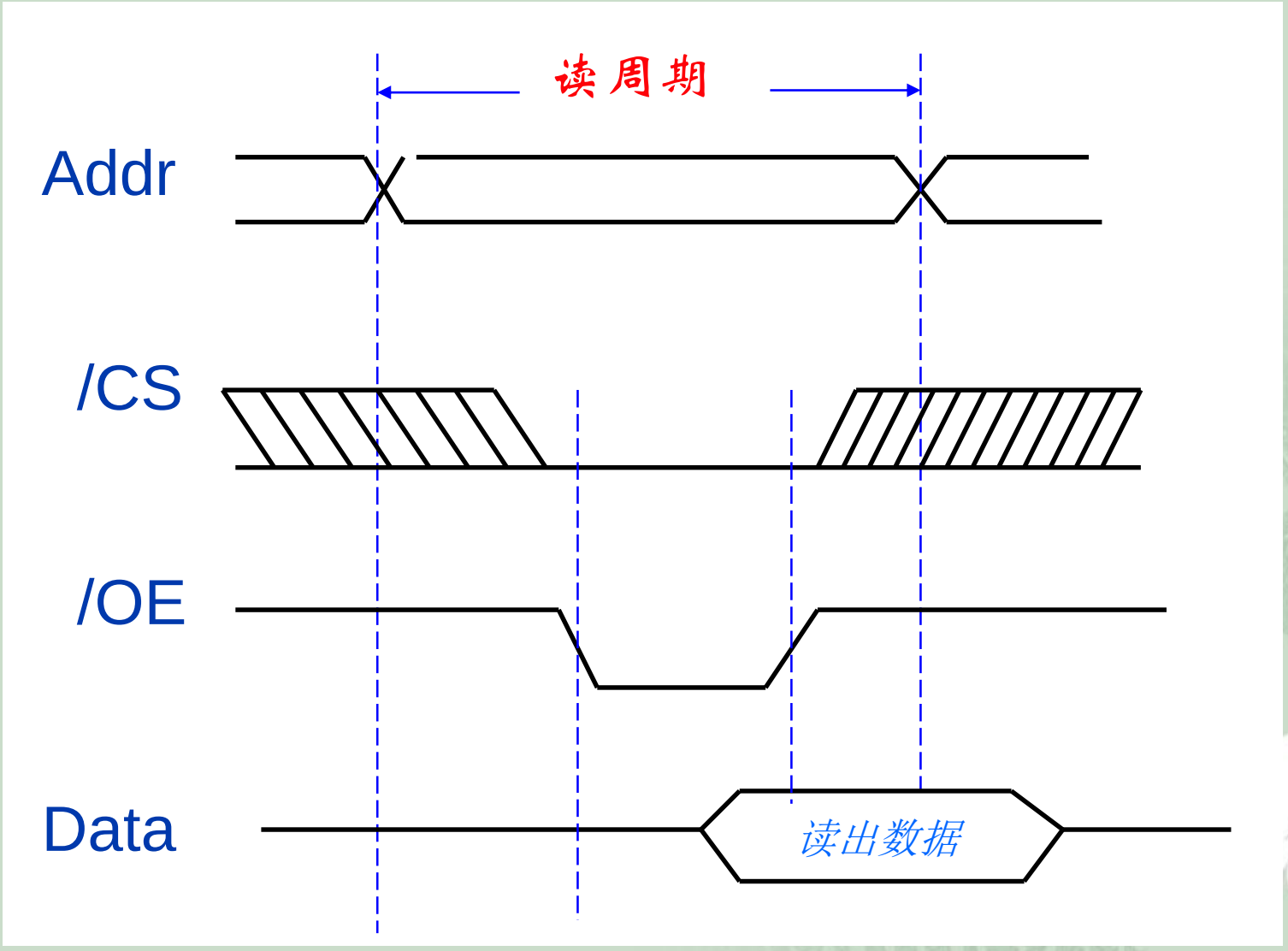




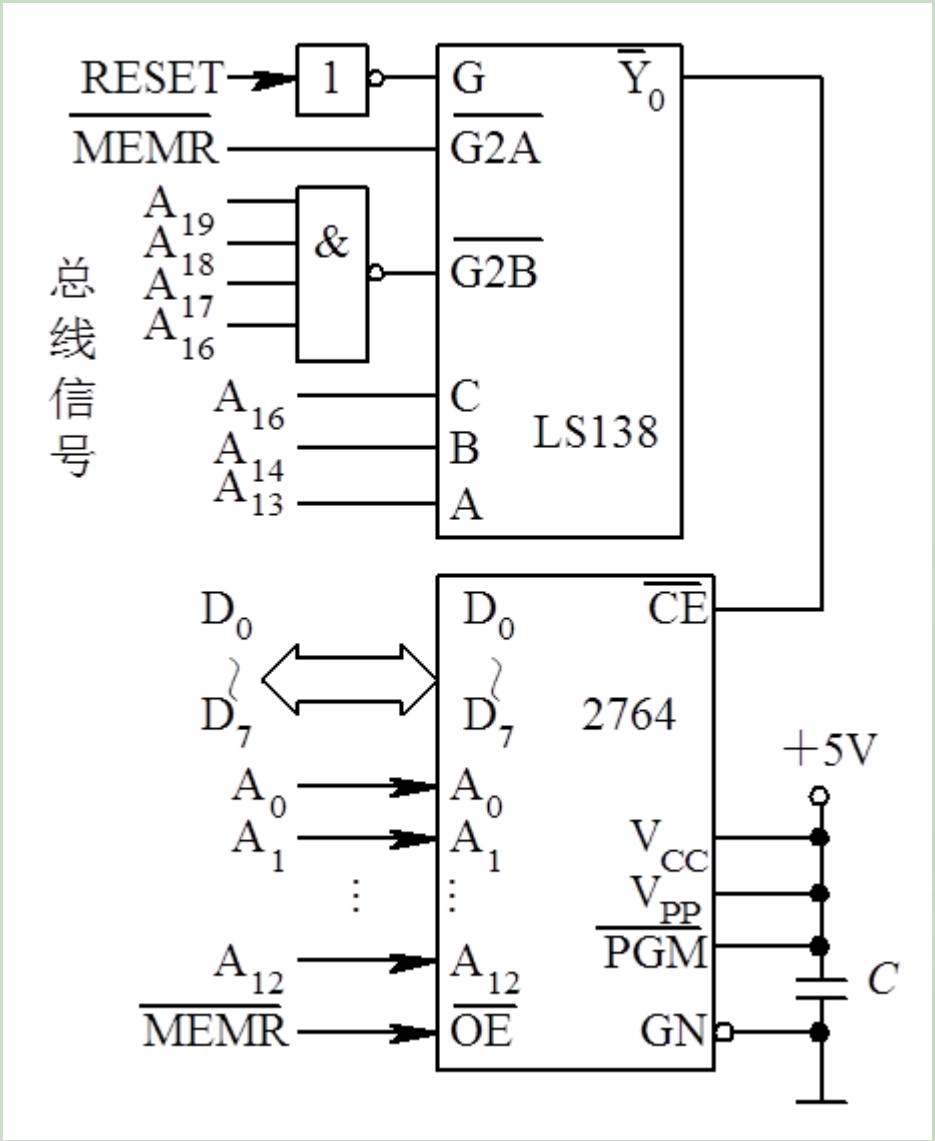
EEPROM

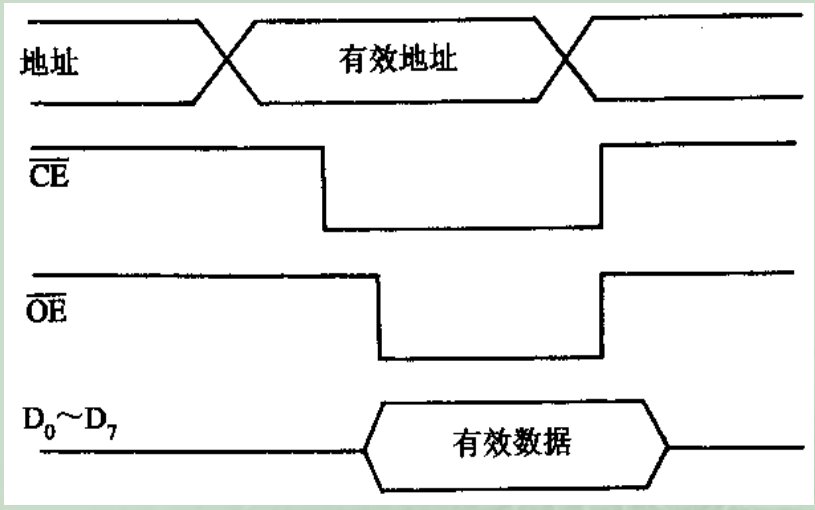


只读存储器 ROM 工作时序

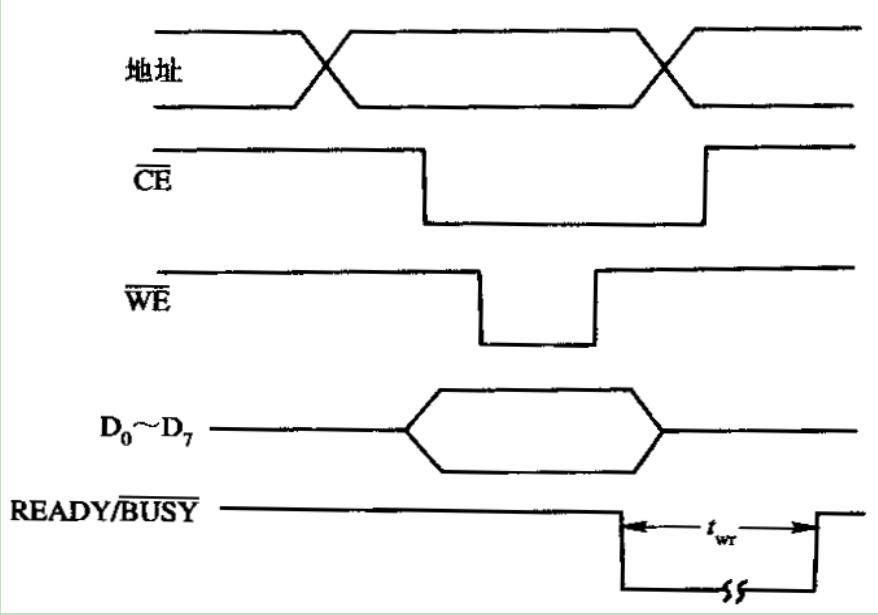


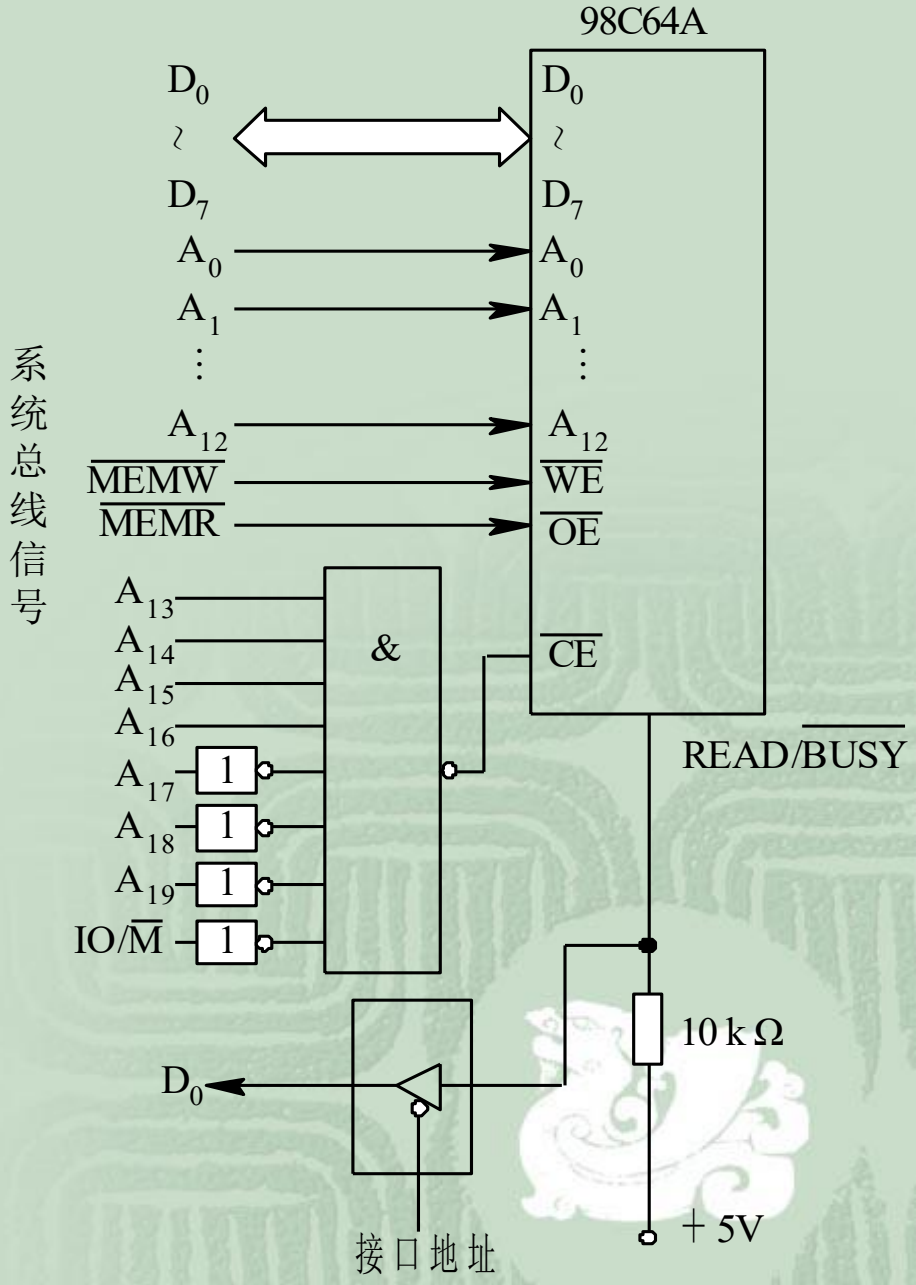
EPROM存储器的应用





EEPROM存储器的应用



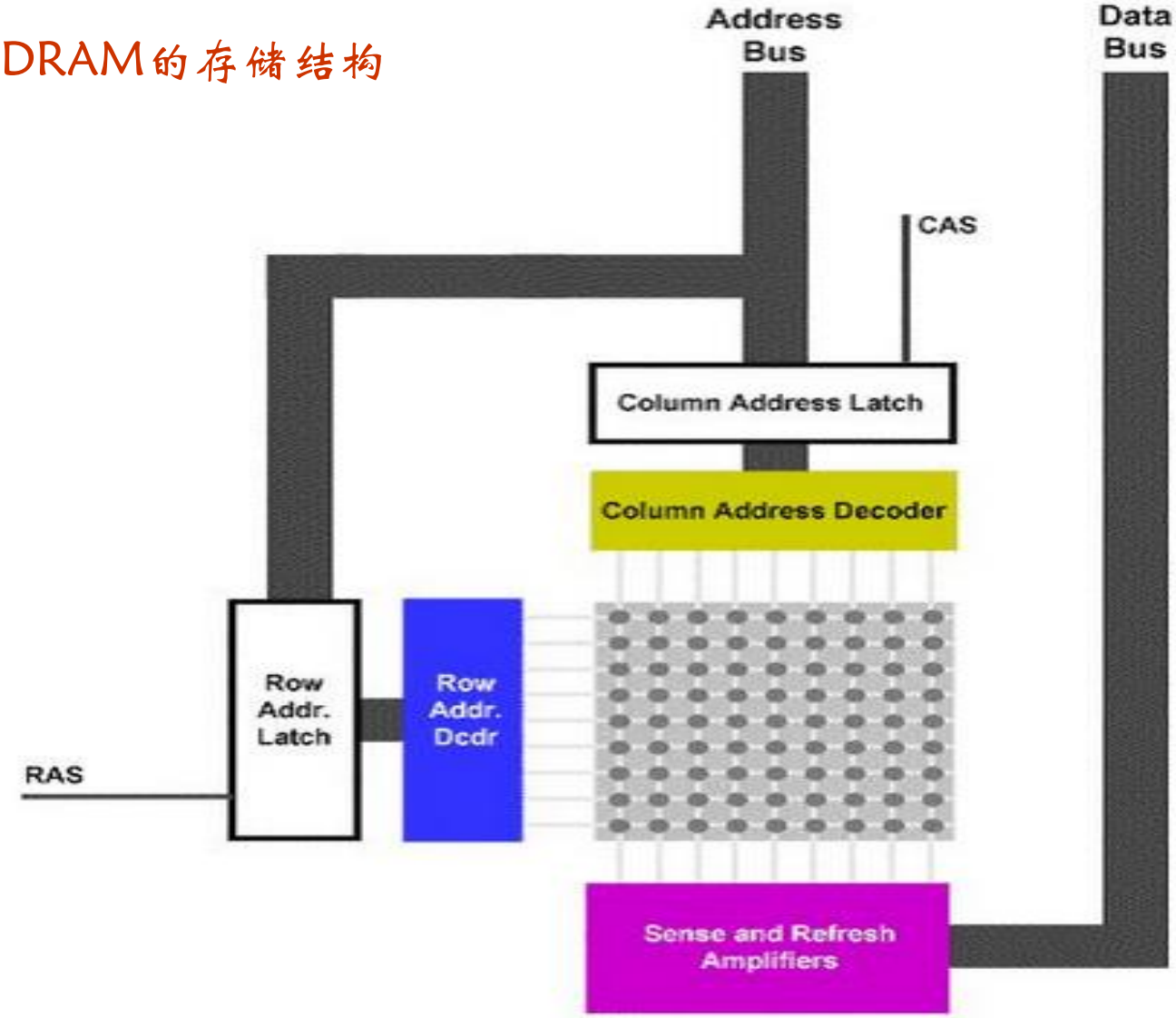


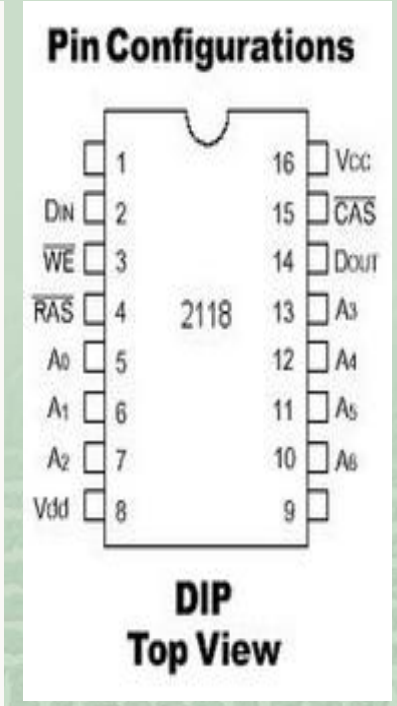
Flash Memory：闪存

- 闪存是电子可擦除只读存储器(EEPROM)的变种，闪存与EEPROM不同的是，EEPROM能在字节水平上进行删除和重写而不是整个芯片擦写，而闪存的大部分芯片需要块擦除。
- NOR型与NAND型闪存，区别很大
- *NOR型*有独立的地址线和数据线，基本存储单元是bit，按存储单元进行删除和访问，容量较小，价格较贵；
- NAND型的地址线和数据线是共用的I/O线，基本存储单元是页(Page)，类似硬盘的扇区；成本要低一些，而容量大得多。
- NOR型闪存比较适合频繁随机读写的场合，通常用于存储程序代码并直接在闪存内运行，如手机的“内存”，容量通常不大;NAND型闪存主要用来存储资料，常用的闪存产品，如闪存盘、数码存储卡都是用NAND型闪存。

7.2.3 动态读写存储器DRAM

DRAM的存储结构





7.2.3 动态读写存储器DRAM

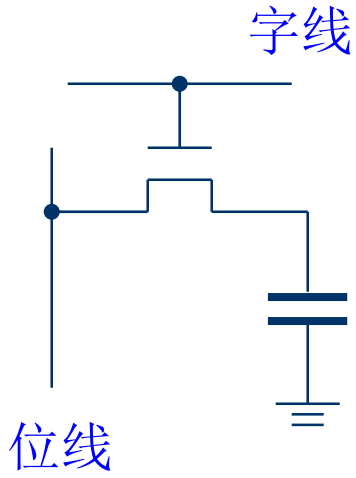
- ✓ 1位动态存储单元及工作原理

  单管基本存储单元

  * 写入：
  * 保持：
  * 读出：

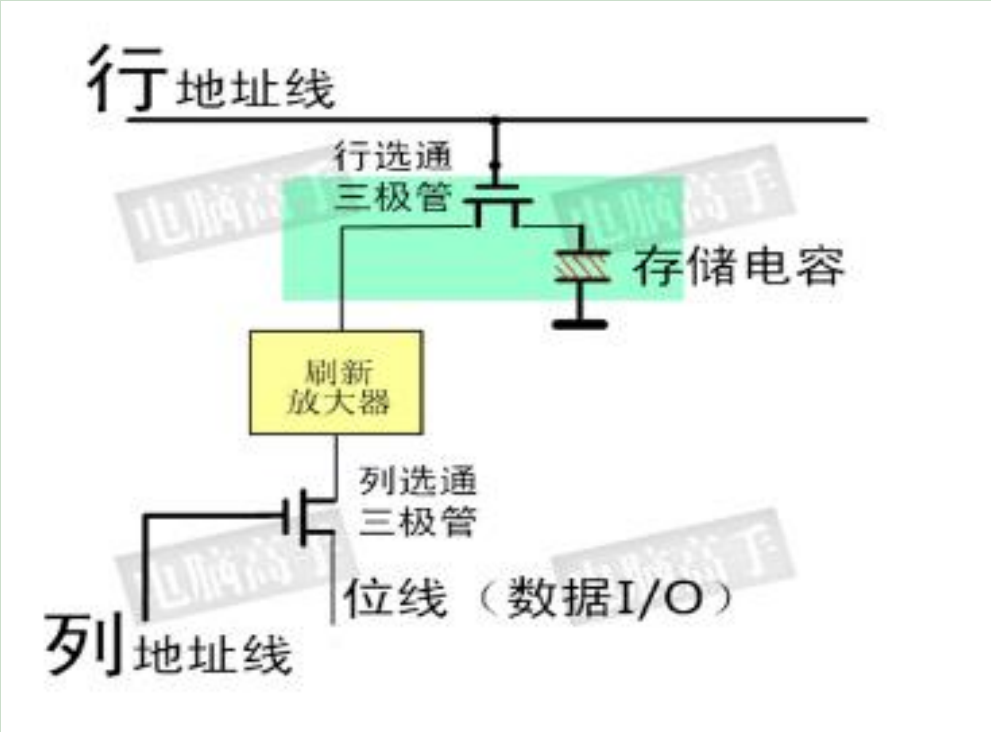
- ✓ 再生（刷新）：

  按行刷新



单管DRAM基本存储单元

单管DRAM存储原理示意图：



行选与列选信号将使存储电容与外界间的传输电路导通，从而可进行放电（读取）与充电（写入）。另外，图中刷新放大器的设计并不固定，目前这一功能被并入读出放大器（Sense Amplifier ,简称S-AMP）

4管DRAM存储原理图： 图7.4

动态读写存储器DRAM

- 动态存储器芯片的引线

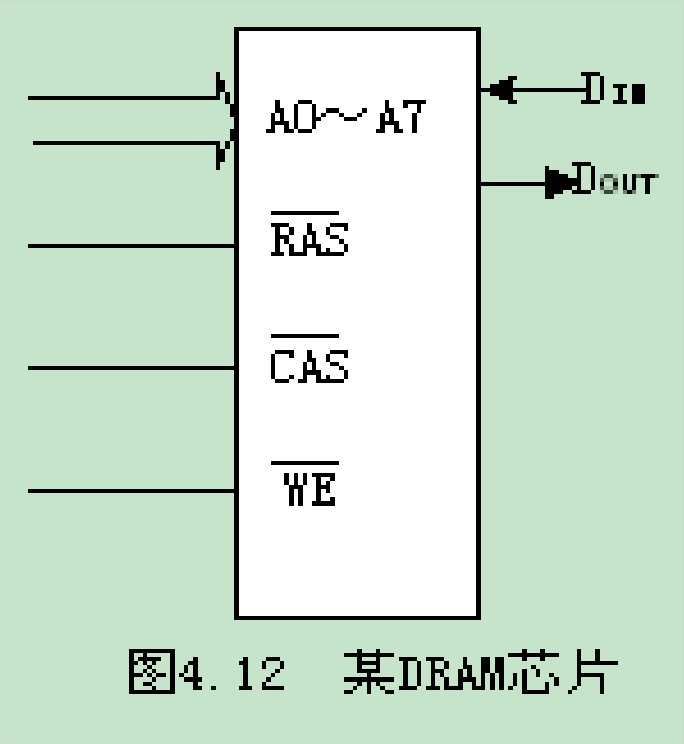


图4.12　某DRAM芯片

A0～A7为地址输入端

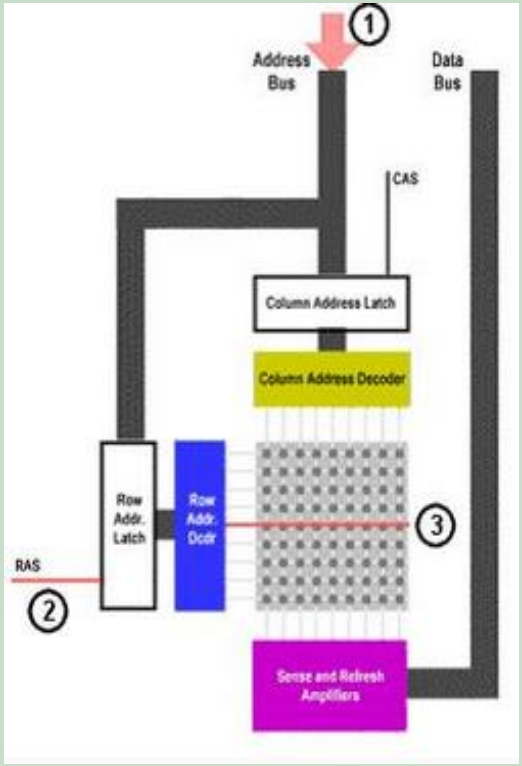
DIN和DOUT是芯片上的数据线

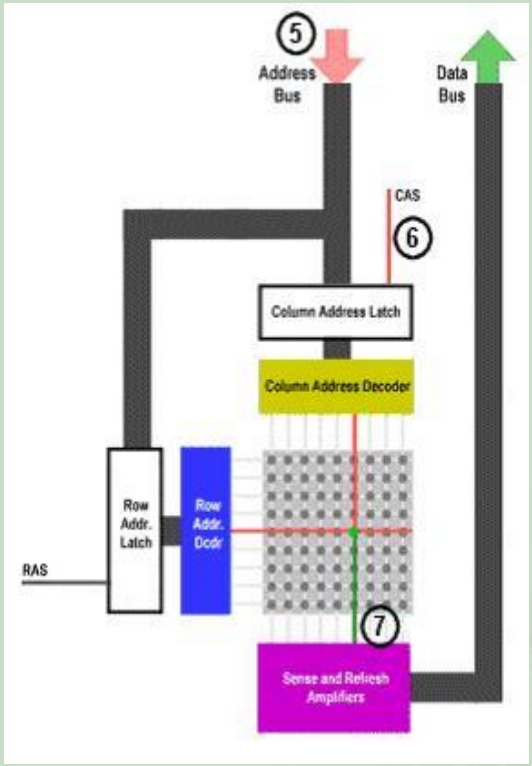
$\overline{RAS}$为行地址锁存信号

$\overline{CAS}$为列地址锁存信号

$\overline{WE}$为写允许信号

动态读写存储器DRAM



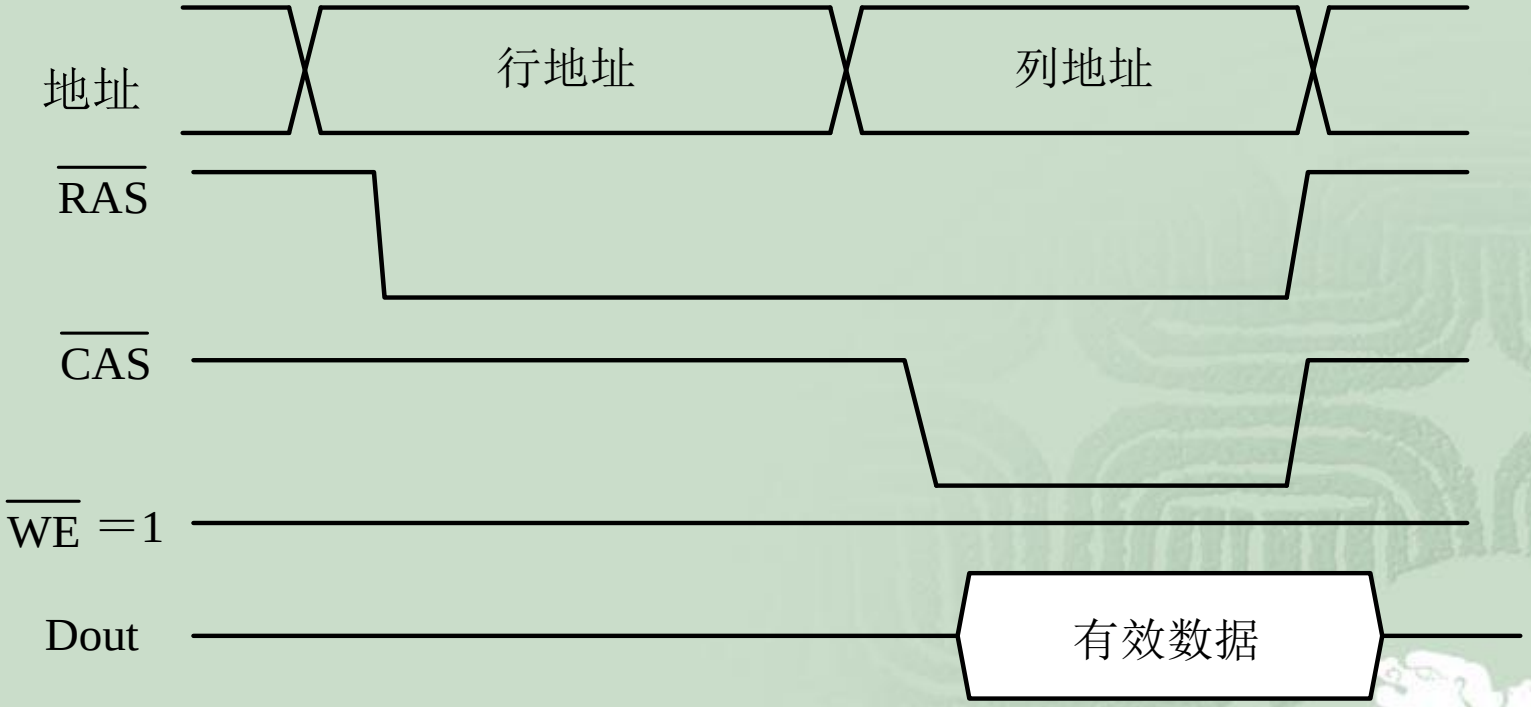


DRAM读取过程

1）通过地址总线将行地址传输到地址引脚
2）/RAS引脚被激活，这样行地址被传送到行地址锁存器中
3）行地址解码器根据接收到的数据选择相应的行
4）/WE引脚被确定不被激活，所以DRAM知道它不会进行写入操作

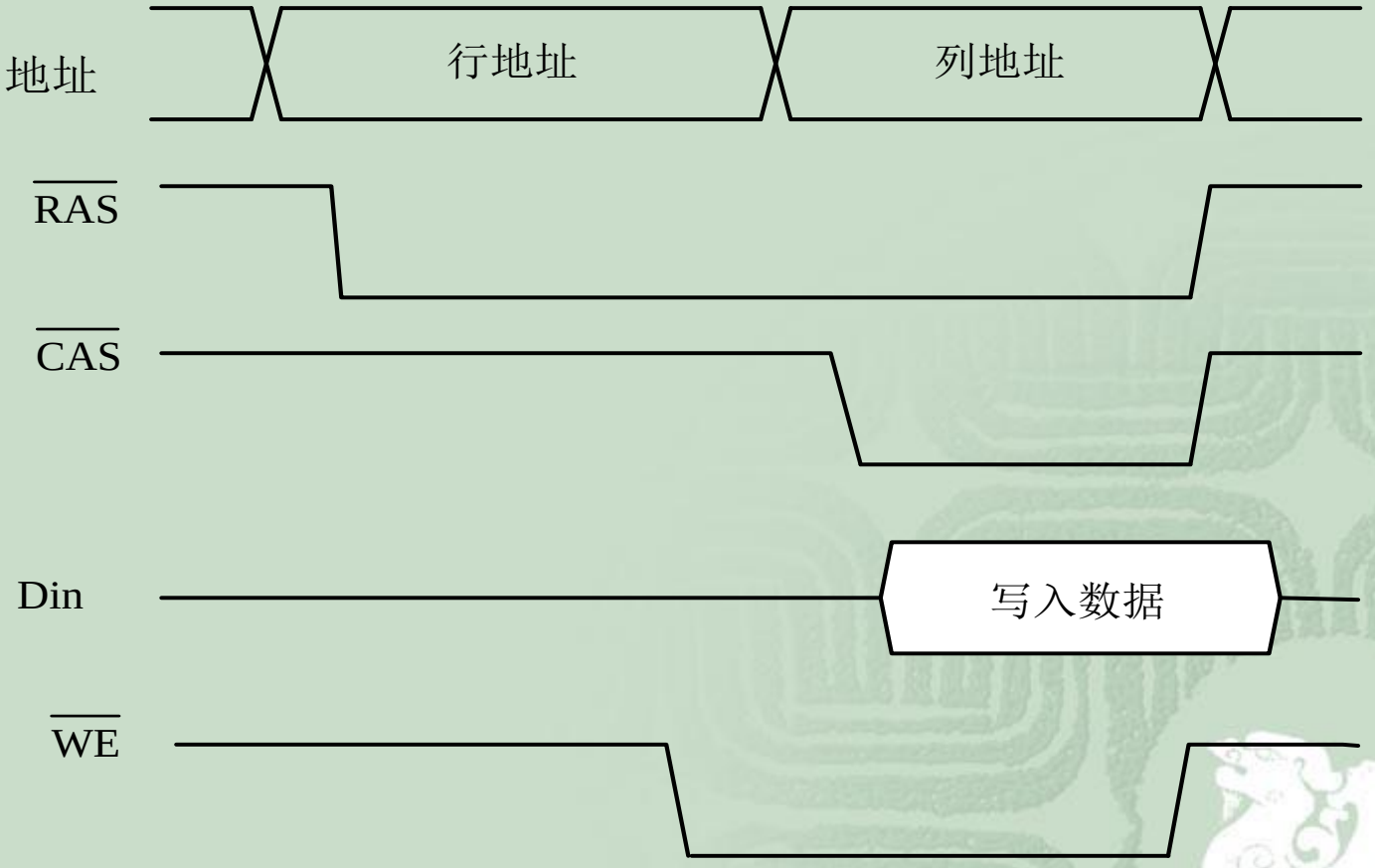
5）列地址通过地址总线传输到地址引脚
6）/CAS引脚被激活，这样列地址被传送到列地址锁存器中
7）/CAS引脚同样还具有/OE引脚的功能，所以这个时候Dout引脚知道需要向外输出数据。
8）/RAS和/CAS都不被激活，这样就可以进行下一个周期的数据操作了。

动态读写存储器DRAM



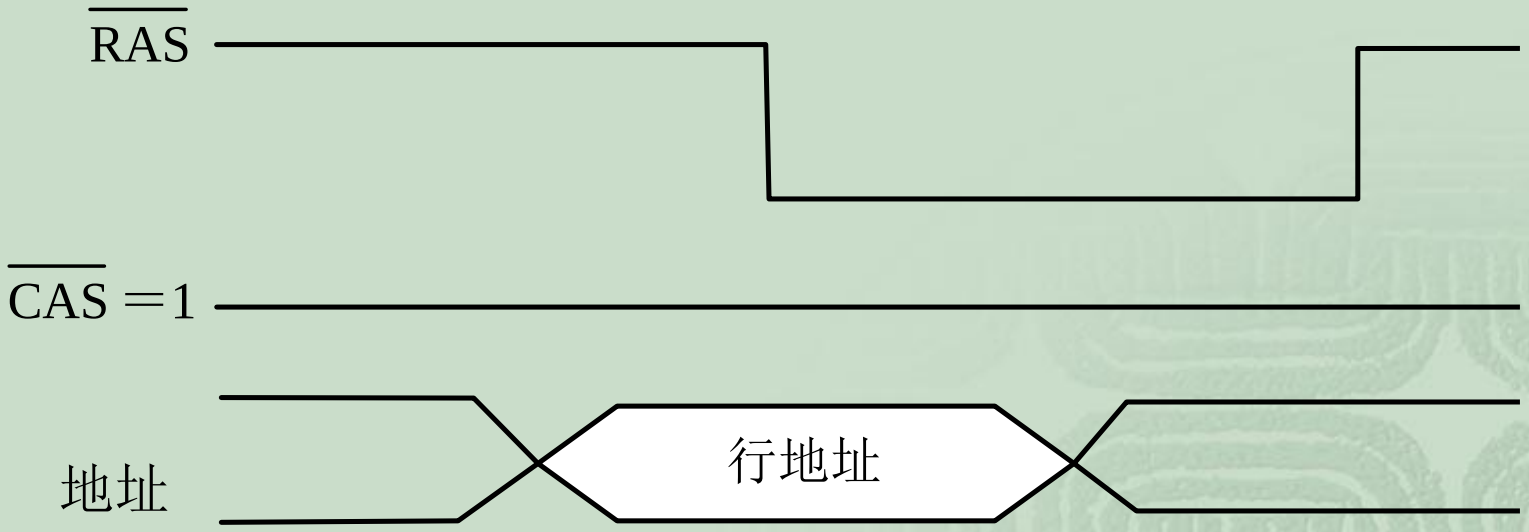
读周期

动态读写存储器DRAM



写周期

动态读写存储器DRAM



刷新周期

SRAM与DRAM的对比

- **SRAM**
  - 用稳态电路存信息，不需刷新
  - 相对于**DRAM**，速度快、集成度低、功耗大、成本高
- **DRAM**
  - 用暂态电路（电容）存信息，需刷新
  - 地址采用复用技术，分行列两次加载，$\overline{RAS}$ 、$\overline{CAS}$
  - 相对于**SRAM**，速度慢、集成度高、功耗小、成本低
  - 刷新（再生）方法：集中式、分散式、异步式
  - 刷新控制电路：刷新计数器、刷新/访存裁决、刷新控制逻辑，是**CPU**和**DRAM**的接口

DRAM内部改善存储性能措施（课外资料）

- 快速页模式（Fast Page Mode）
- 同步DRAM（SDRAM）

  Synchronous Dynamic Random Access Memory
  （同步动态随机存储器）

  - ✓ 同步是指SDRAM与CPU共享同一个时钟，同步工作，内部的命令的发送与数据的传输都以它为基准

  时钟被用来驱动一个有限状态机，对进入的指令进行流水线操作。流水线意味着芯片可以在处理完之前的指令前，接受一个新的指令。

  在一个写的流水线中，写命令在另一个指令执行完之后可以立刻执行，而不需要等待数据写入存储队列的时间。在一个读的流水线中，需要的数据在读指令发出之后固定数量的时钟频率后到达，而这个等待的过程可以发出其它附加指令。

同步DRAM（SDRAM）

- ✓ SDRAM基于双存储体结构

内含两个交错的存储阵列，当CPU从一个存储体或阵列访问数据时，另一个就已为读写数据做好了准备，通过这两个存储阵列的紧密切换，读取效率就能得到成倍的提高。

- ✓ SDRAM从发展到现在已经经历了多代

第一代 SDR SDRAM，第二代DDR SDRAM，第三代DDR2 SDRAM，第四代DDR3 SDRAM，第五代DDR4 SDRAM

目前，（市场上）习惯上把第一代 SDR SDRAM 叫做 SDRAM，第二代DDR SDRAM之后的叫做DDR。*（实质是一种错误的认识）*

- **双倍数据传输（DDR）**

  DDR（Double Data Rate）SDRAM

  （双倍速率SDRAM）

  - ✓ DDR在时钟信号上升沿与下降沿各传输一次数据，使得DDR的数据传输速度为传统SDRAM的两倍

  由于仅多采用了下降沿信号，因此并不会造成能耗增加。

  地址与控制信号则与传统SDRAM相同，仅在时钟上升缘传输。

  - ✓ DDR内存采用数据传输频率作为命名标准，在前面加上DDR代数的符号，如 DDR3 1333

  PC-即DDR，PC2=DDR2，PC3=DDR3。

  如：PC2700是DDR333，其工作频率（时钟频率）是333/2=166MHz，
  *2700表示带宽为2.7G，（333*8 = 2.66GB/sec）。*

双倍数据传输（DDR）

- DDR内存带宽计算公式

  带宽 = 内存核心频率 × 内存总线位数 × 倍增系数

DDR2仍然采用时钟脉冲上升、下降沿各传一次数据的技术，一次预读4bit数据，是DDR一次预读2bit的2倍，因此，倍增系数是2X2=4;

DDR3一次预读 8bit，是DDR2的2倍，倍增系数是2X2X2=8;

DDR4一次预读16bit，是DDR3的2倍，倍增系数是2X2X2X2=16

- DDR内存有三种不同的频率指标，它们分别是核心频率、时钟频率和数据传输频率。

核心频率即为内存Cell阵列(Memory Cell Array)的工作频率，它是内存的真实运行频率；时钟频率即I/O Buffer（输入/输出缓存）的传输频率；而有效数据传输频率则是指数据传送的频率。

数据传输频率 = 时钟频率 * 2

时钟频率 = 核心频率 * n *(DDR～DDR4的n不同)*

双倍数据传输（DDR）

- 例如

DDR3内存一次从存储单元预取8Bit的数据，在I/OBuffer（输入/输出缓存）上升和下降中同时传输，因此有效的数据传输频率达到了存储单元核心频率的8倍。同时DDR3内存的时钟频率提高到了存储单元核心的4倍。也就是说DDR3-800内存的核心频率只有100MHz，其时钟频率为400MHz，有效数据传输频率则为800MHz。

从SDRAM-DDR时代，数据总线位宽始终没有改变，都为64bit，但是采用双通道技术，可以获得64X2=128bit的位宽。

简单算法：

带宽 = 内存标称频率 × 内存总线位数

也就是 带宽 = 数据传输频率 × 内存总线位数

参见：

- https://blog.csdn.net/chenzhen1080/article/details/82951122

  DDR的发展简史

- https://blog.csdn.net/chenzhen1080/article/details/82951181

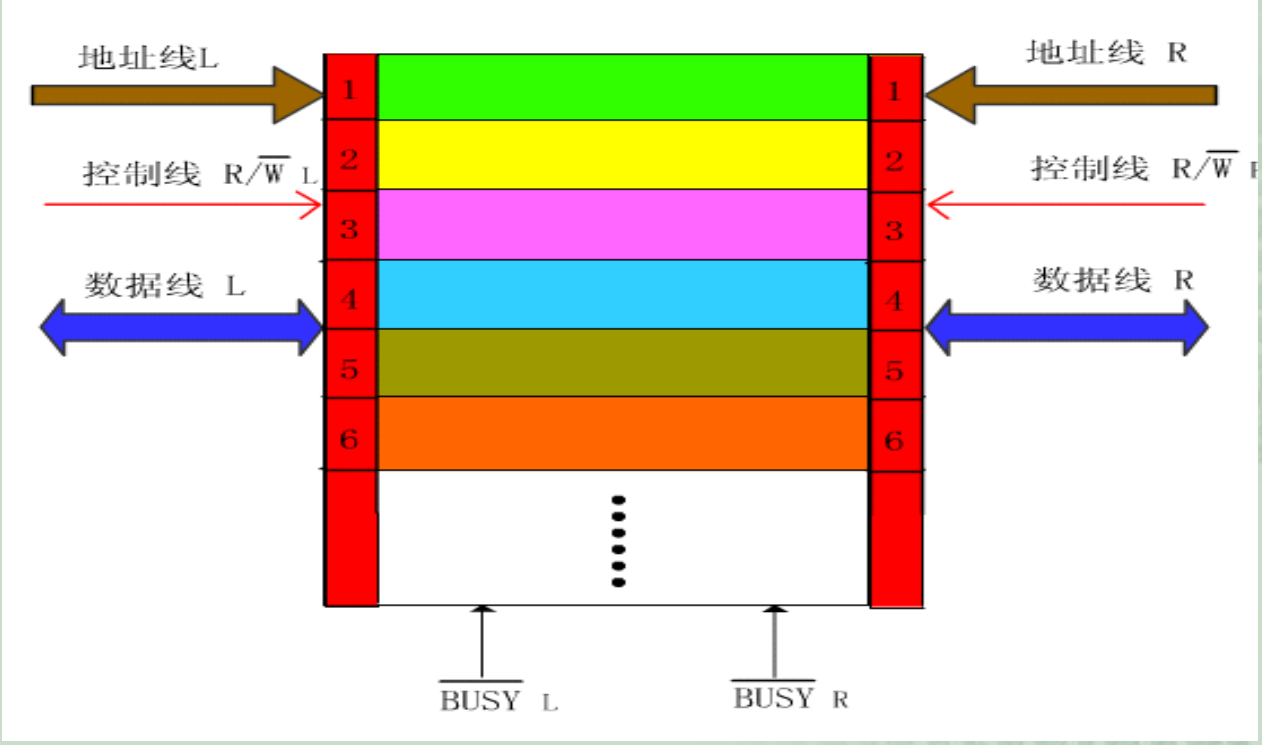
  DDR的特性分析

- ……

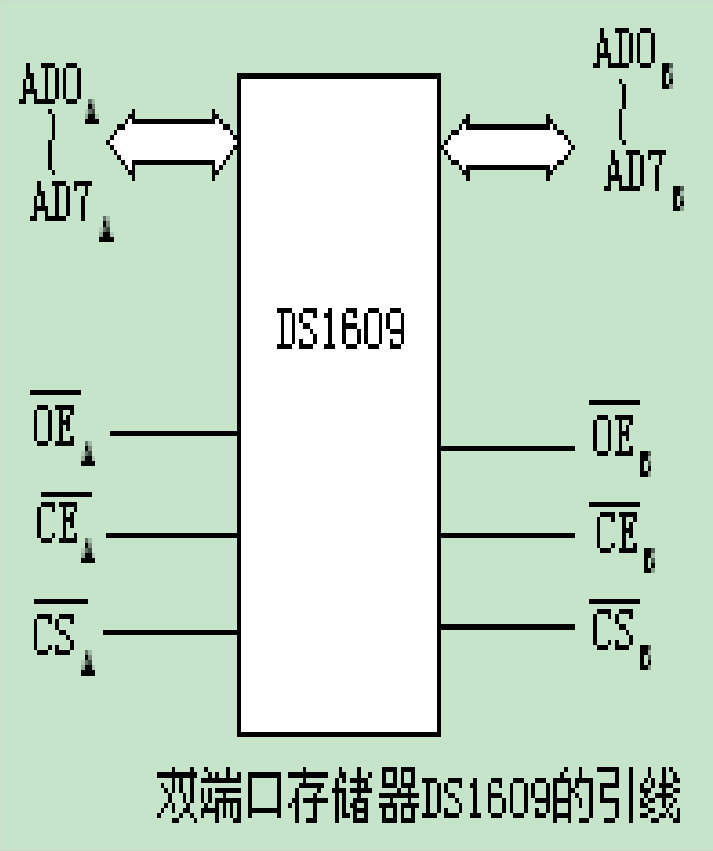
7.2.4 其他存储器

---- 提高存储器速度的技术

1、双端口存储器

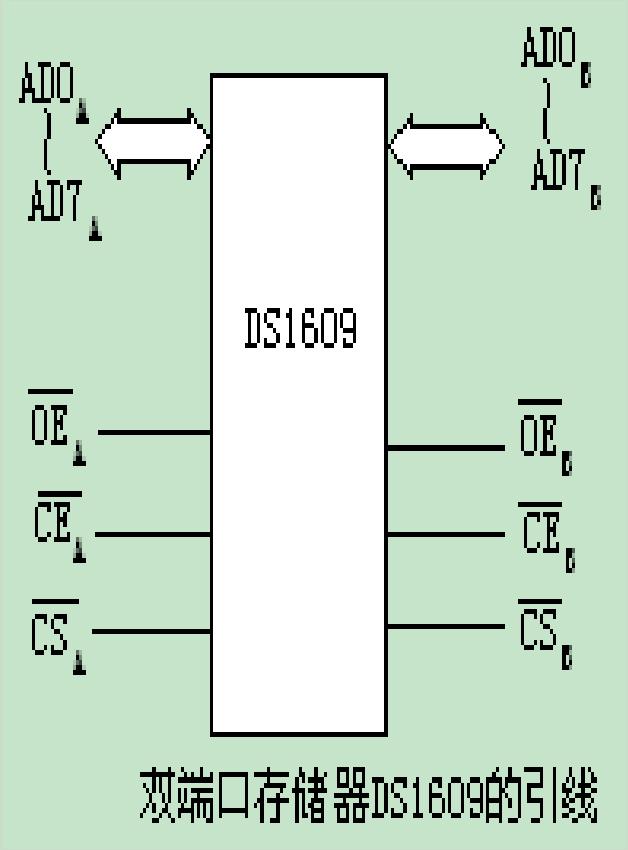


双端口存储器的例子



双端口存储器DS1609的引线

- 用于提高访存速度
- 多端口：时间并行
- 结构特点
  - 具有多(两)组独立的读写端口
  - 允许多(两)个CPU或控制器从多(两)个端口同时异步访问存储单元



双端口存储器DS1609的引线

(a) 对不同存储单元允许同时读或写。
(b) 允许对同一单元同时读。
(c) 当一个端口写某单元而另一端口同时读该单元时。
(d) 当两个端口同时对同一单元写数据时，就会引起竞争，产生错误。

访问冲突

从多(两)个端口同时写同一个存储单元，发生冲突。

解决方法

芯片内电路裁决

外部电路与写状态控制程序

2、多体（模块）交叉存储器

- 用于提高访存速度
- 多模块：空间并行

（1）多体存储器的模块化组织

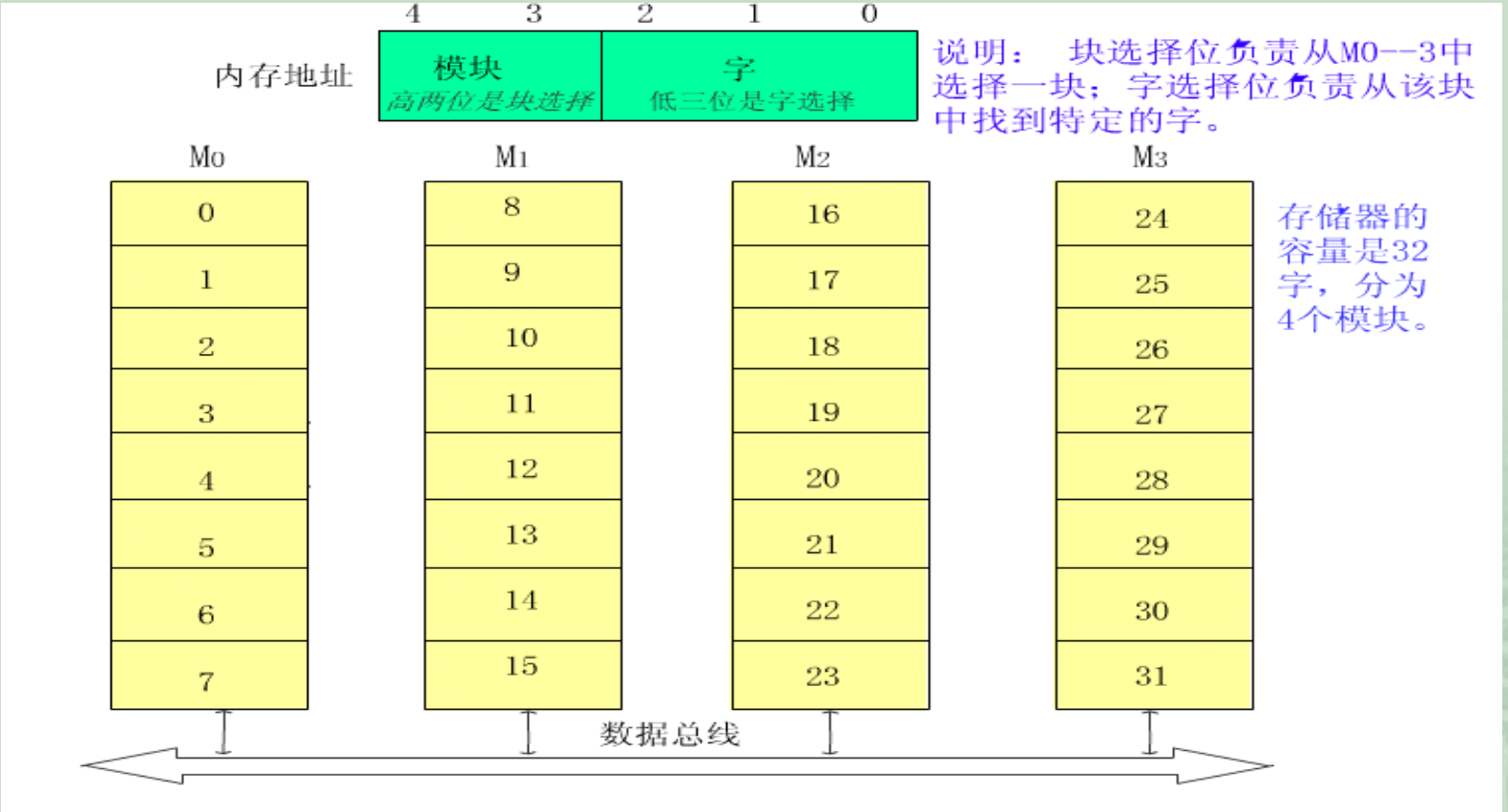
顺序组织方式

交叉组织方式

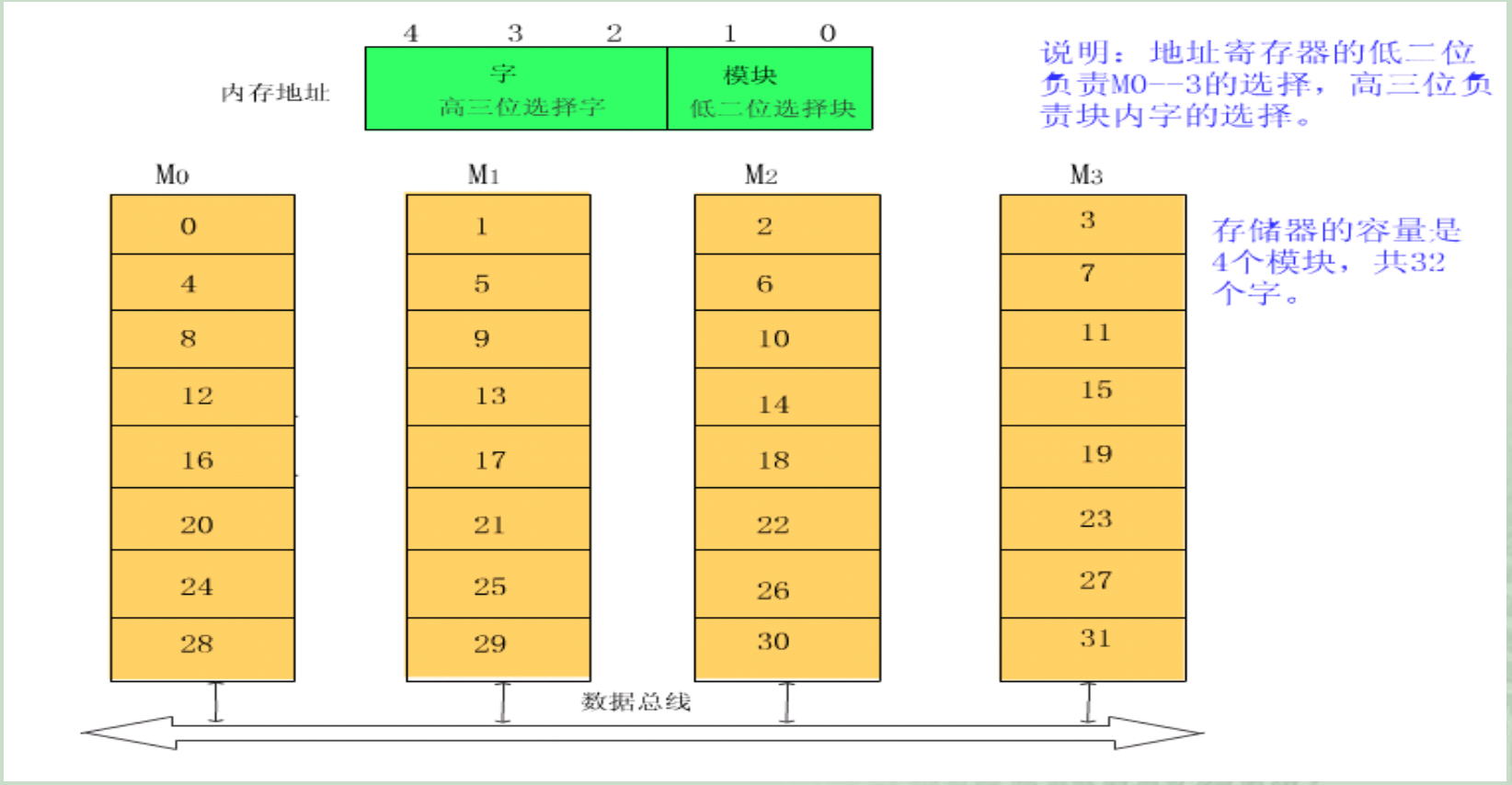
（2）多体交叉存储器的访问方式

多体并行访问

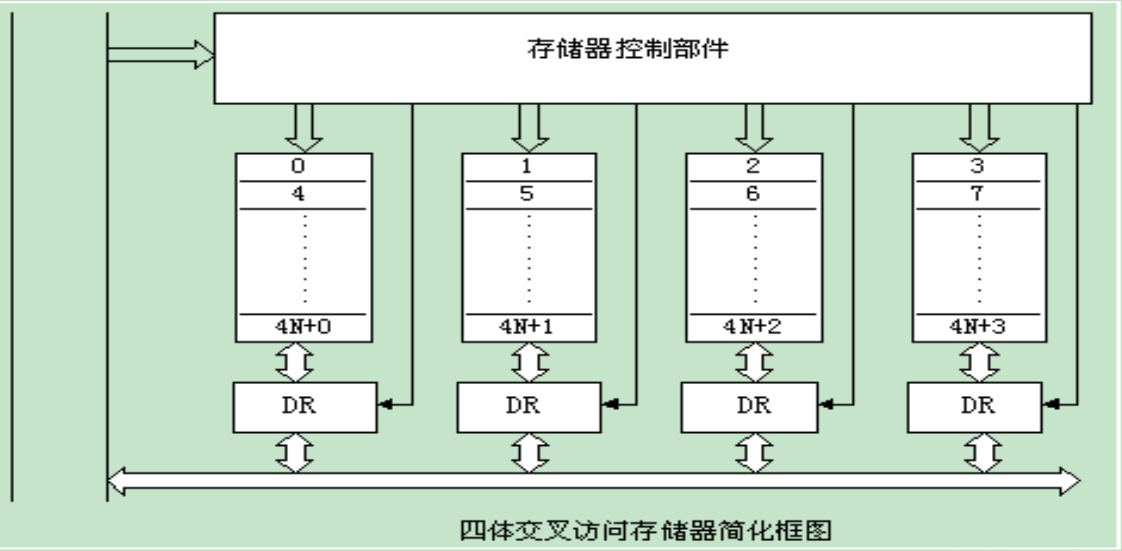
多体交叉访问



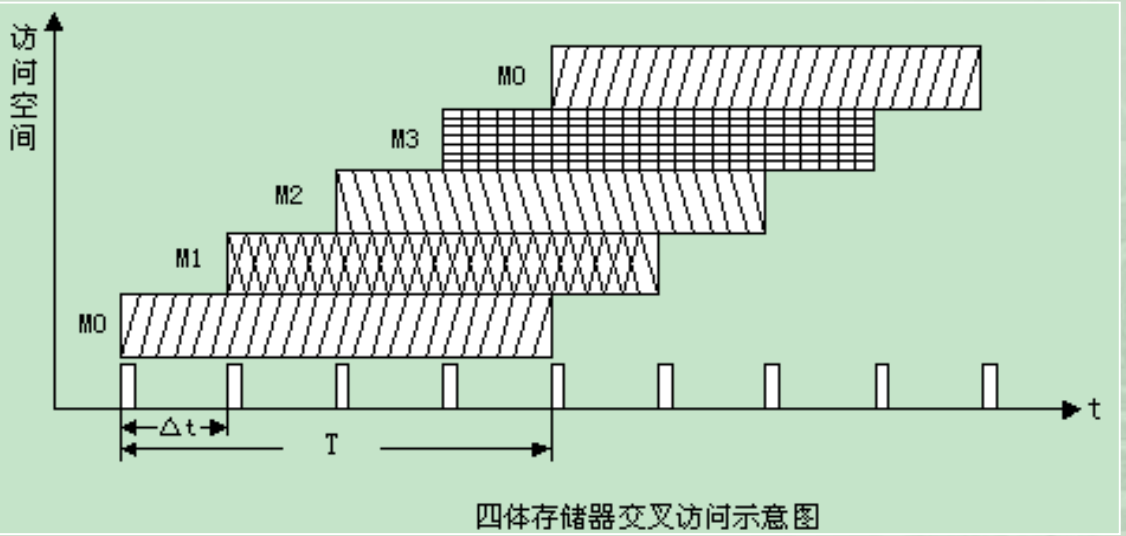
顺序组织方式：某个模块进行存取时，其他模块不工作，某一模块出现故障时，其他模块可以照常工作，通过增添模块来扩充存储器容量比较方便。但各模块串行工作，存储器的带宽受到了限制。



交叉组织方式：地址码的低位字段经过译码选择不同的模块，而高位字段指向相应模块内的存储字。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。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。



四体交叉访问存储器简化框图



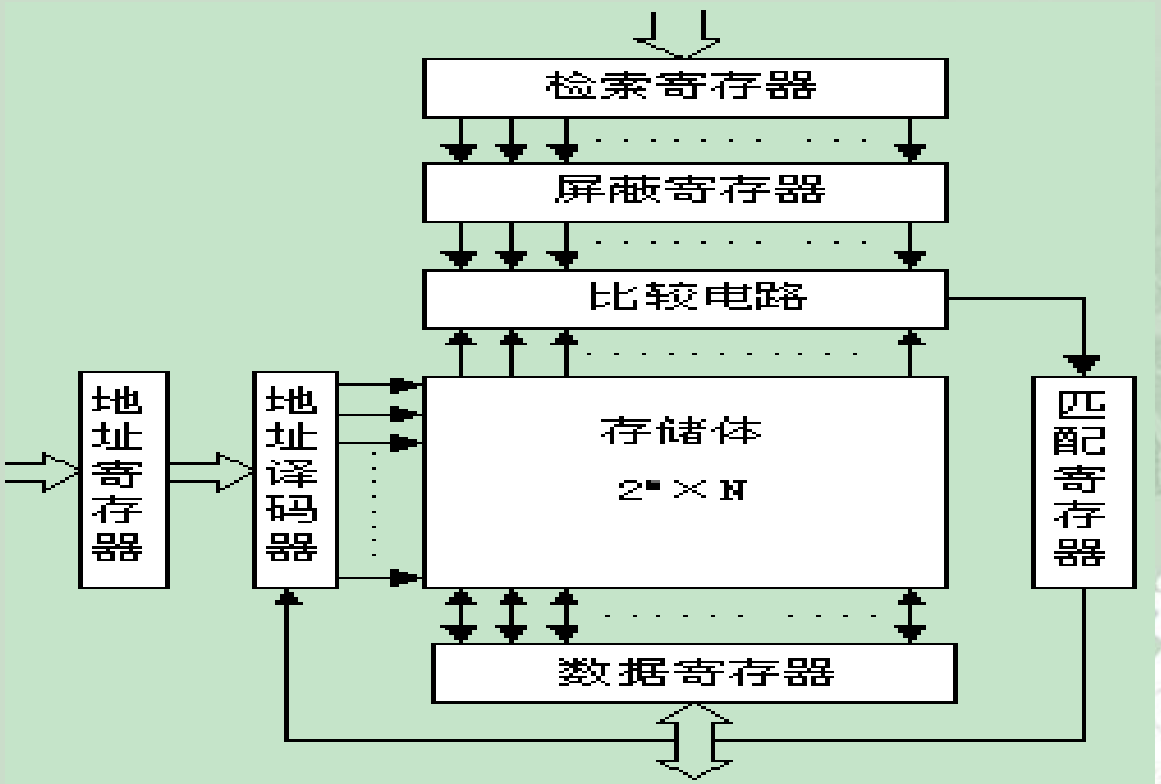
四体存储器交叉访问示意图

m体流水读写n个字用时：$T_{交叉} = T + (n-1) \times \triangle t$，　$\triangle t = T/m$，

$m\triangle t \leq T$

3、相联存储器

- 依据内容决定内容的地址或者寻找与其相关的内容
- 用于许多领域，如高速缓冲存储器及虚拟存储器中。



相联存储器的构成图

本章作业-1

第 2、6、7、16 题