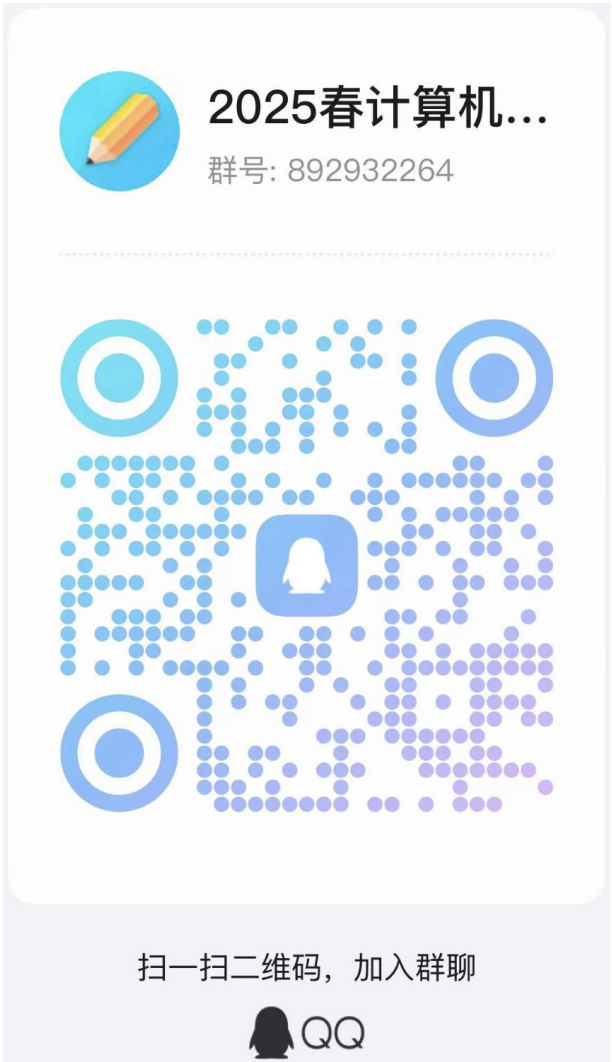
计算机组成原理



花忠云
https://huazhongyun.github.io/
http://faculty.hitsz.edu.cn/huazhongyun

计算机科学与技术学院

第六章 存储器

6.1存储器概述

一、存储器分类：存储介质、存取方式、计算机中的作用

1. 按存储介质分类

(1) 半导体存储器　　晶体管　金属氧化物半导体场效应管 TTL 、MOS　　易失

(2) 磁表面存储器　　磁头、载磁体

(3) 磁芯存储器　　硬磁材料、环状元件

(4) 光盘存储器　　激光、磁光材料

非易失

2. 按存取方式分类

(1) 存取时间与物理地址无关（随机访问）

- 随机存储器　在程序的执行过程中 可 读 可 写
- 只读存储器　在程序的执行过程中 只 读

(2) 存取时间与物理地址有关（串行访问）

- 顺序存取存储器　　磁带
- 直接存取存储器　　磁盘

3. 按在计算机中的作用分类

- 存储器
  - 主存储器
    - RAM
      - 静态 RAM
      - 动态 RAM
    - ROM
      - MROM（Mask ROM，掩膜编程的只读存储器，它只能写入一次数据）
      - PROM（可编程只读存储器，一般只能写入一次数据）
      - EPROM（可擦除可编程只读存储器，可以多次写入数据）
      - EEPROM（电可擦除可编程只读存储器，与EPROM运行原理类似，但抹除数据的方式不同，可以多次写入数据）
  - Flash Memory U盘、固态硬盘（SSD）
  - 高速缓冲存储器（Cache） CPU寄存器和主存储器之间
  - 辅助存储器 磁盘、磁带、光盘

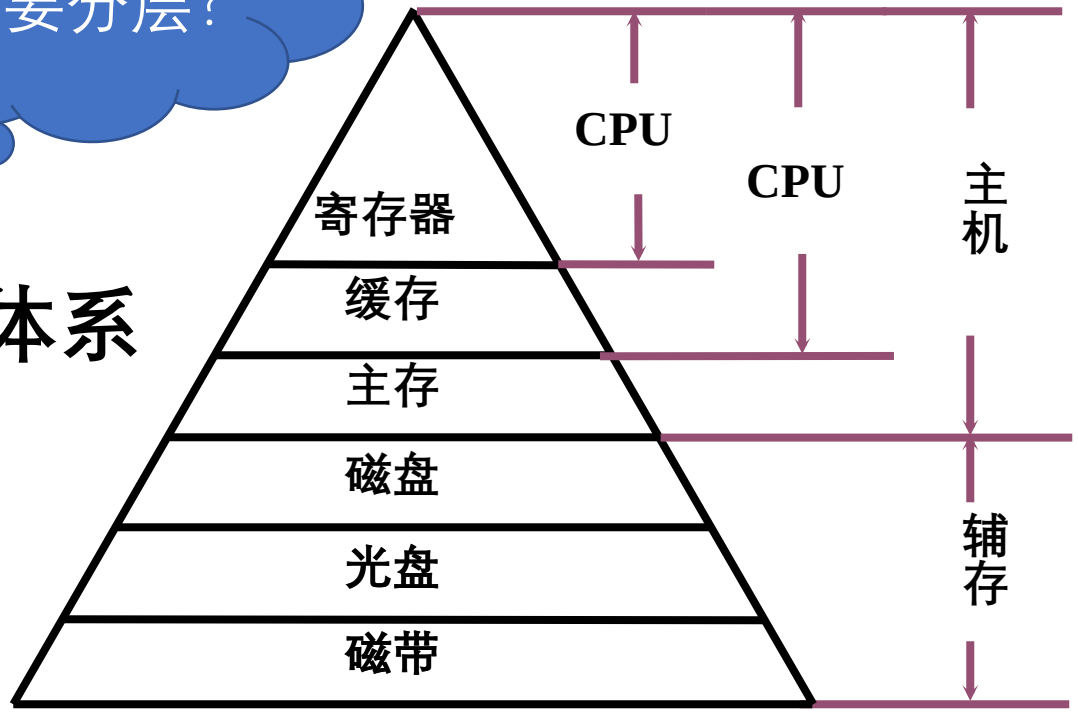
存储器的层次结构

存储器三个主要特性的关系

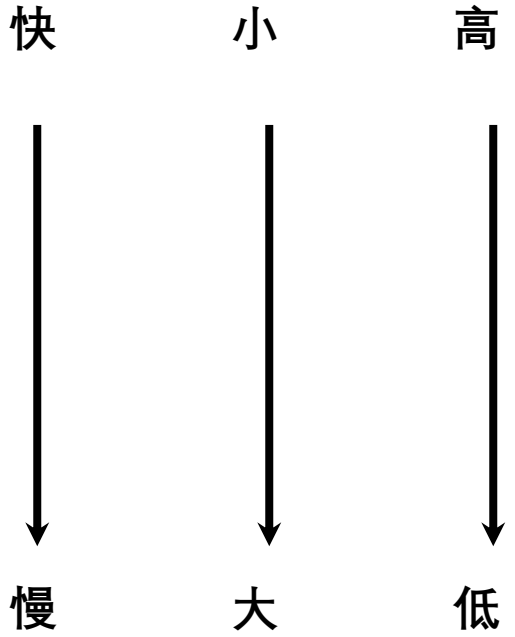
速度：存储和读取数据的速度
容量：存储器能够存储的最大数据量
价格：存储一个bit（位）的数据需要的成本

为何要分层？

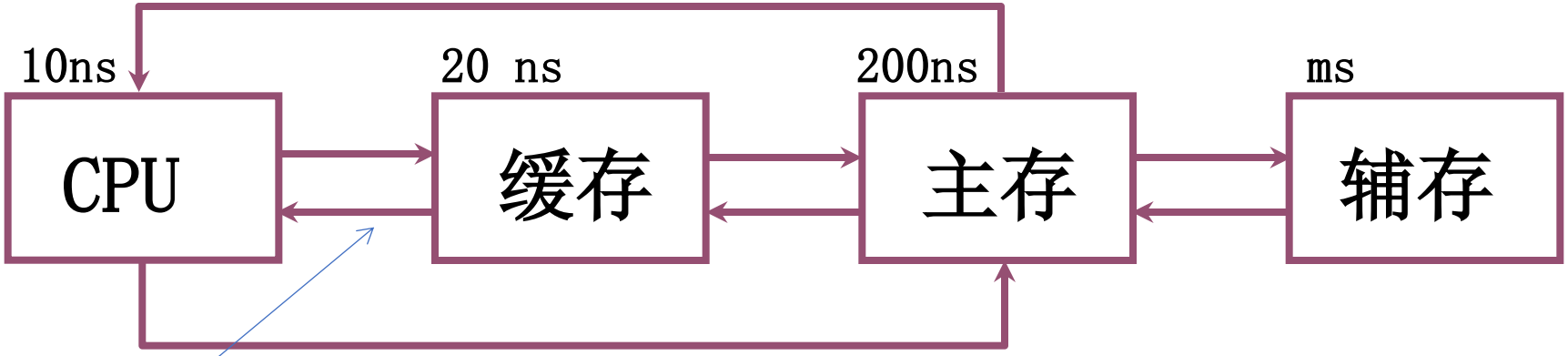
存储体系



速度　容量　价格/位



2. 缓存—主存层次和主存—辅存层次



程序运行的局部性原理

（速度）　　（容量）

缓存－主存　　主存－辅存

主存储器　　虚拟存储器

实地址　　虚地址

物理地址　　逻辑地址

程序运行的局部性原理

- 在一小段时间内，最近被访问过的程序和数据很可能再次被访问
- 在空间上这些被访问的程序和数据往往集中在一小片存储区
- 在访问顺序上，指令顺序执行比转移执行的可能性大(大约**5:1**)
- 合理地把程序和数据分配在不同存储介质中

可以将最近被访问过的程序或数据放入高速缓存中，当CPU再次访问这些程序或数据时，访问速度就可以大大提高

第六章 存储器

6.1 存储器概述

6.2 主存储器

6.3 高速缓冲存储器

6.4 虚拟存储器

6.5 辅助存储器

6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

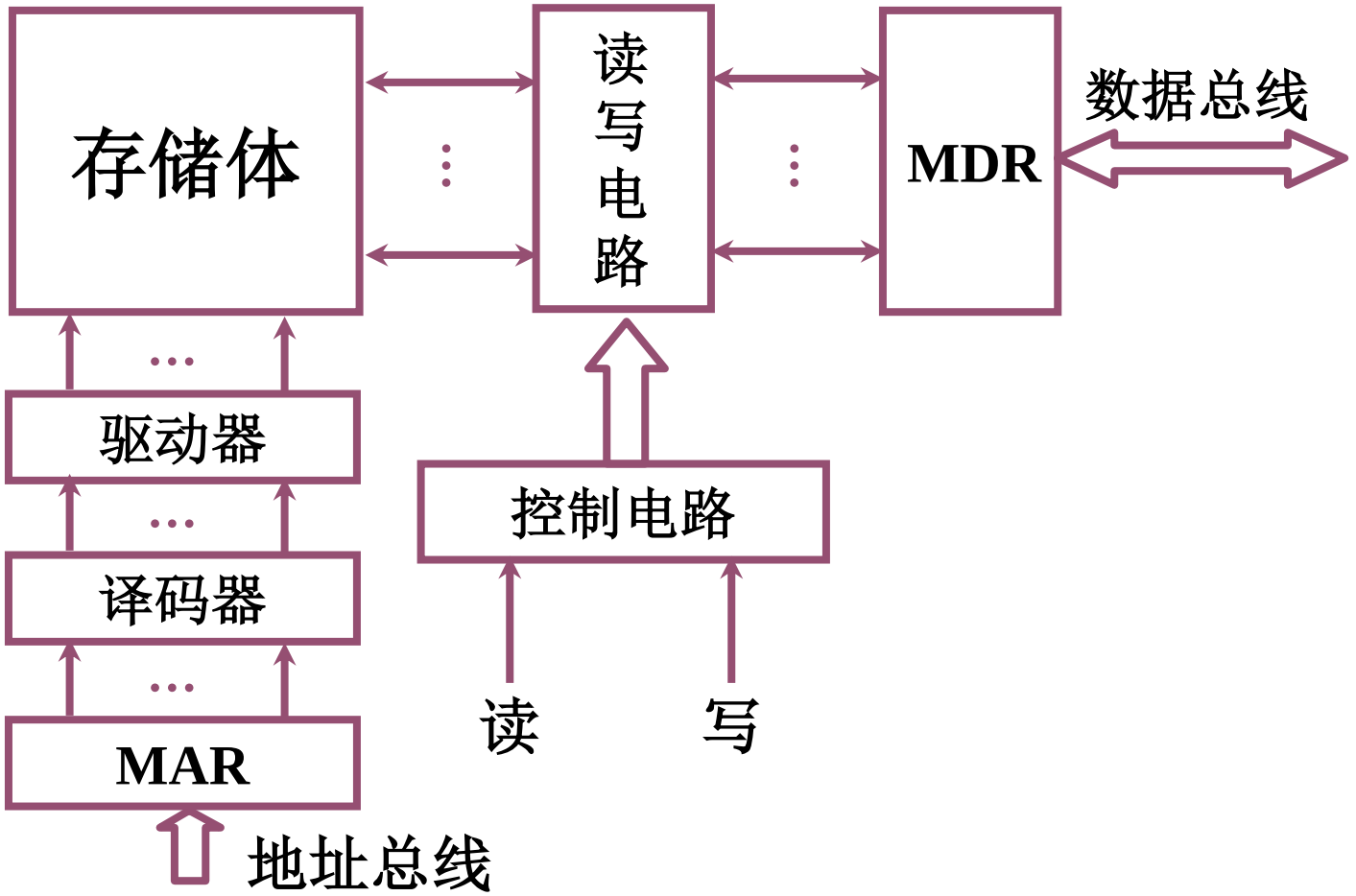
六、存储器的校验

七、提高访存速度的措施

6.2 主存储器——主存的基本组成

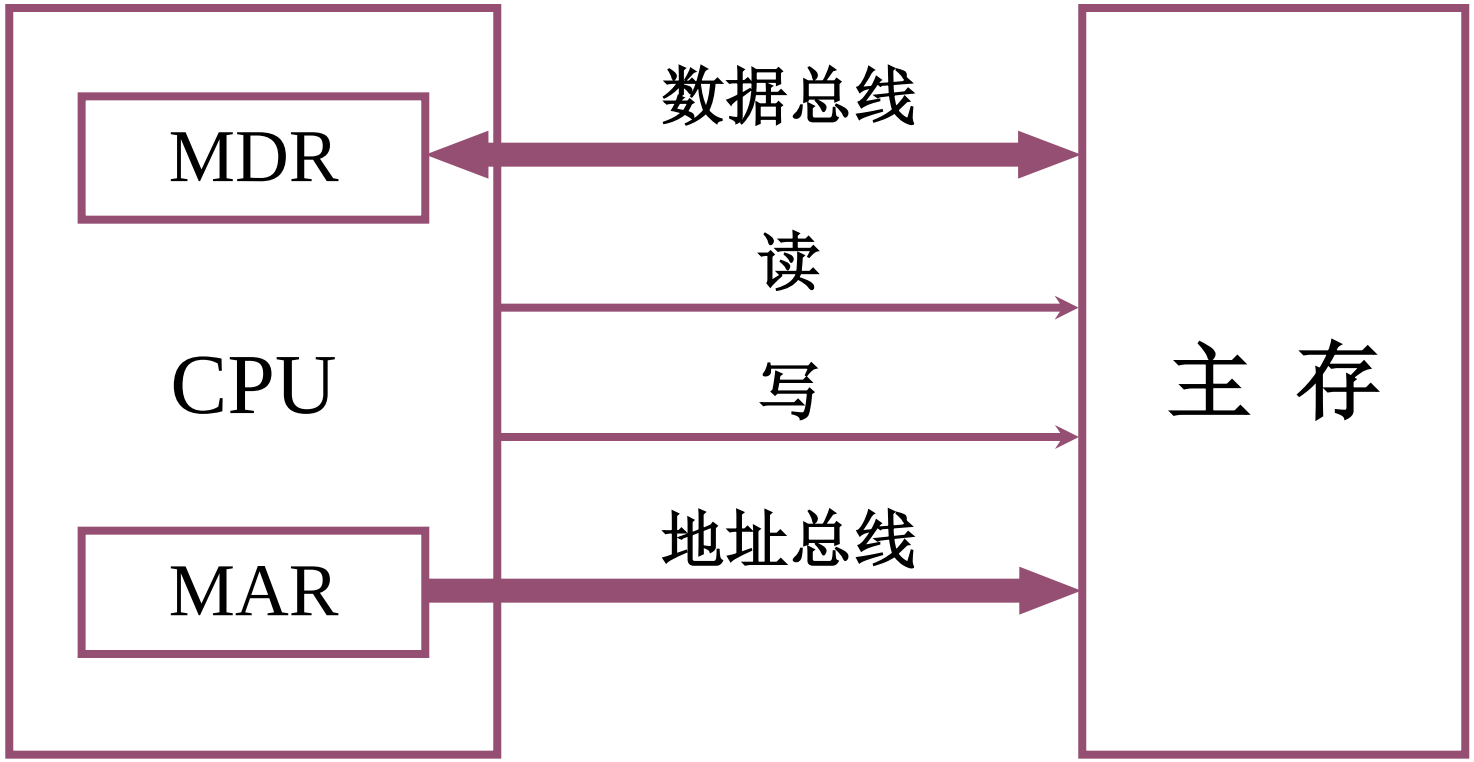
- **MDR（Memory Data Regisiter，数据寄存器）：保存要被写入/读出地址单元的数据**

- **MAR（Memory Address Regisiter，地址寄存器）：访问数据的地址**



MAR和MDR从功能上看属于主存，但通常放在CPU内。

6.2主存储器—主存和CPU的联系



6.2主存储器—主存中存储单元地址的分配

- 大端模式Big-Endian:　低地址存放高位
- 小端模式Little-Endian:低地址存放低位
- unsigned int value = 0x12345678

内存地址	小端模式	大端模式
0x4000	0x78	0x12
0x4001	0x56	0x34
0x4002	0x34	0x56
0x4003	0x12	0x78

如 16 MB （2^{27}位）的存储器	寻址范围	容量
按 字节 寻址	2^{24}= 16 M	$2^{24} \times 2^{3} = 2^{27}$位
按 字（16位）寻址	2^{23}= 8 M	$2^{23} \times 2^{4} = 2^{27}$位
按 字 （32位）寻址	2^{22}= 4 M	$2^{22} \times 2^{5} = 2^{27}$位

4. 主存的技术指标

(1) 存储容量 主存 存放二进制代码的总位数或字节总数

(2) 存储速度

- 存取时间 存储器的 访问时间

  读出时间 写入时间

- 存取周期 连续两次独立的存储器操作

  （读或写）所需的 最小间隔时间

  读周期 写周期

(3) 存储器的带宽 位/秒

6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

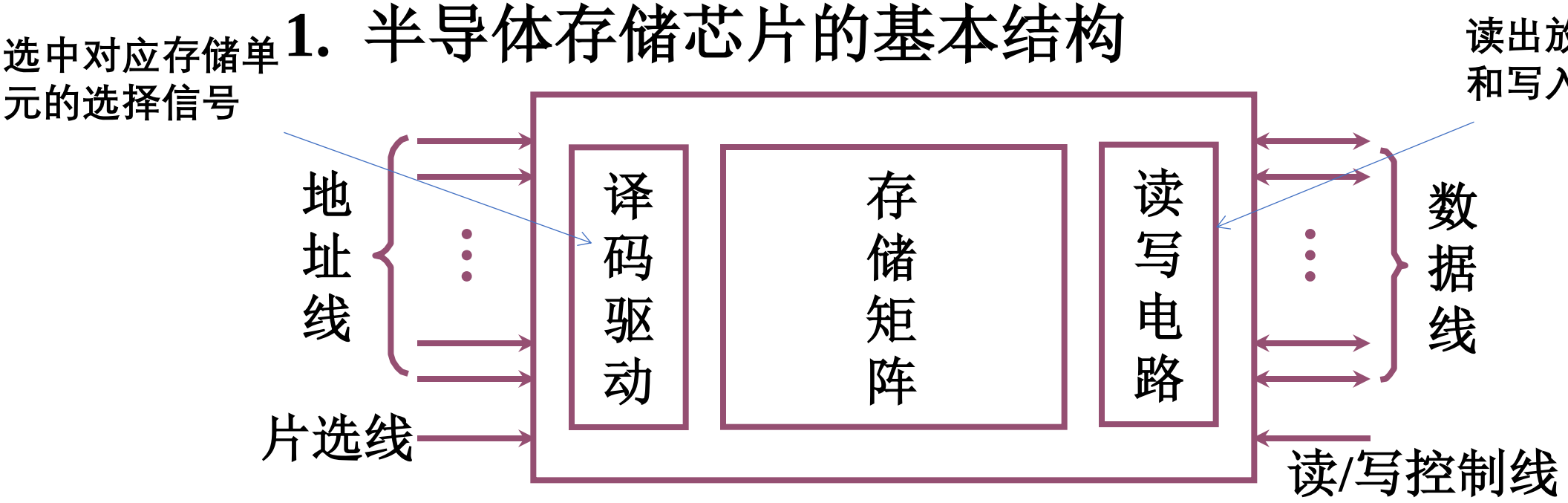
六、存储器的校验

七、提高访存速度的措施

6.2主存储器（Memory）

二、半导体存储芯片简介（基本结构和数据读写）

1. 半导体存储芯片的基本结构



地址线（单向）	数据线（双向）	芯片容量
10	4	1K×4位
14	1	16K×1位
13	8	8K×8位

6.2主存储器

二、半导体存储芯片简介

1. 半导体存储芯片的基本结构

片选线 $\overline{CS}$ $\overline{CE}$ 选择有效的存储芯片

读/写控制线 $\overline{WE}$（低电平写 高电平读）

$\overline{OE}$（允许读）$\overline{WE}$（允许写）

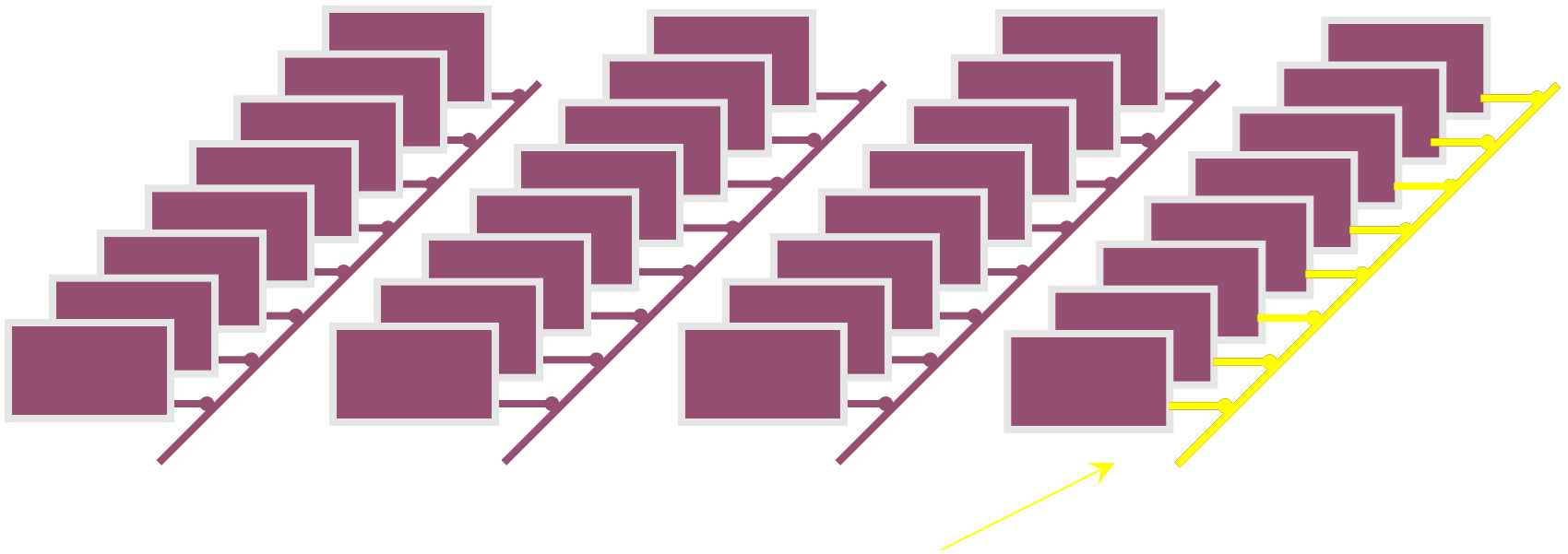
6.2主存储器

- 存储芯片片选线的作用

用 16K × 1位 的存储芯片组成 64K × 8位 的存储器

32片

8片 16K × 1位　8片 16K × 1位　8片 16K × 1位　8片 16K × 1位



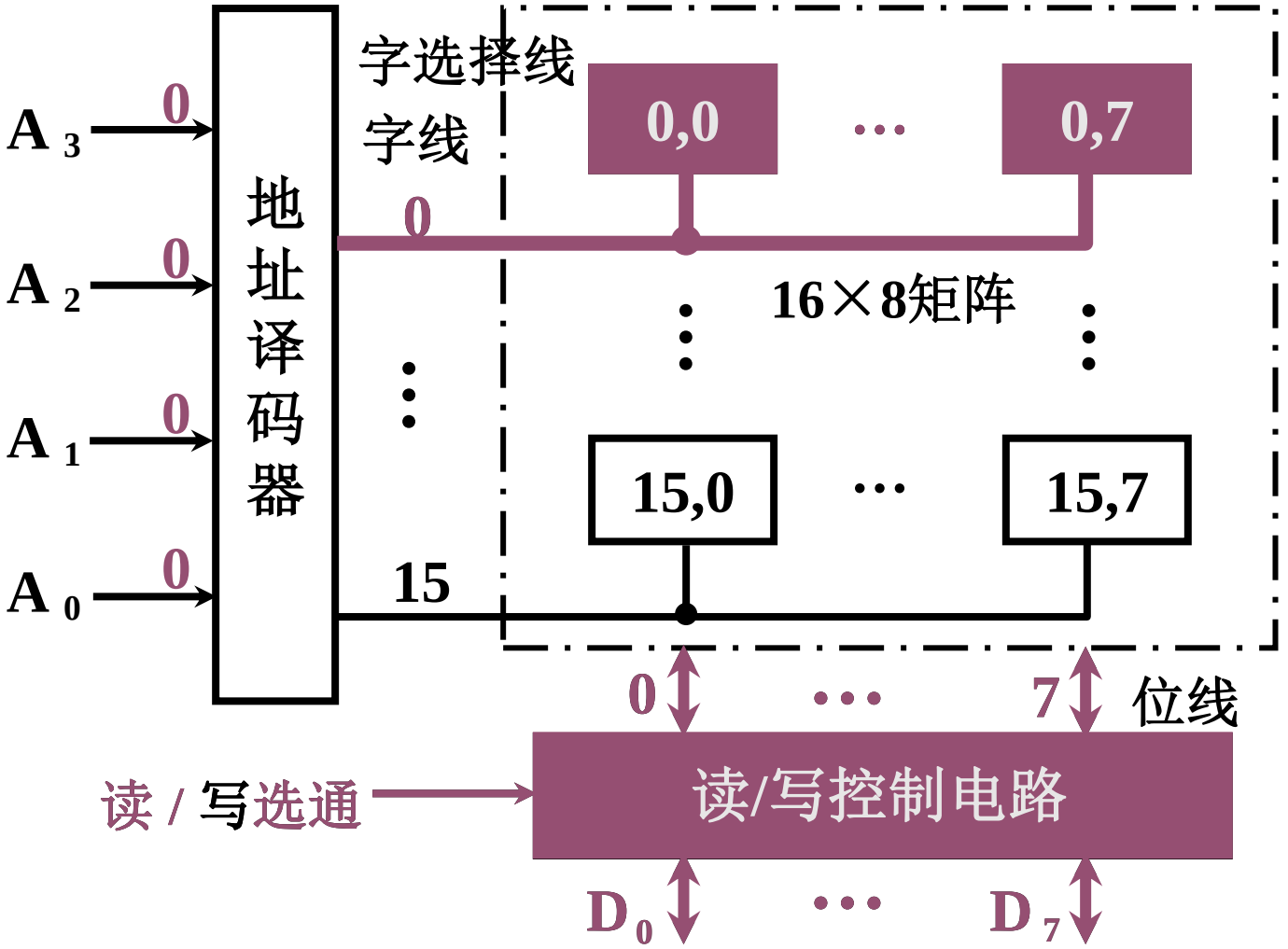
当地址为 65 535 时，此 8 片的片选有效

6.2主存储器

2. 半导体存储芯片的译码驱动方式（两种）

(1) 线选法

主要思想是用一根字选择线（字线），直接同时选中一个存储单元中的8位数据

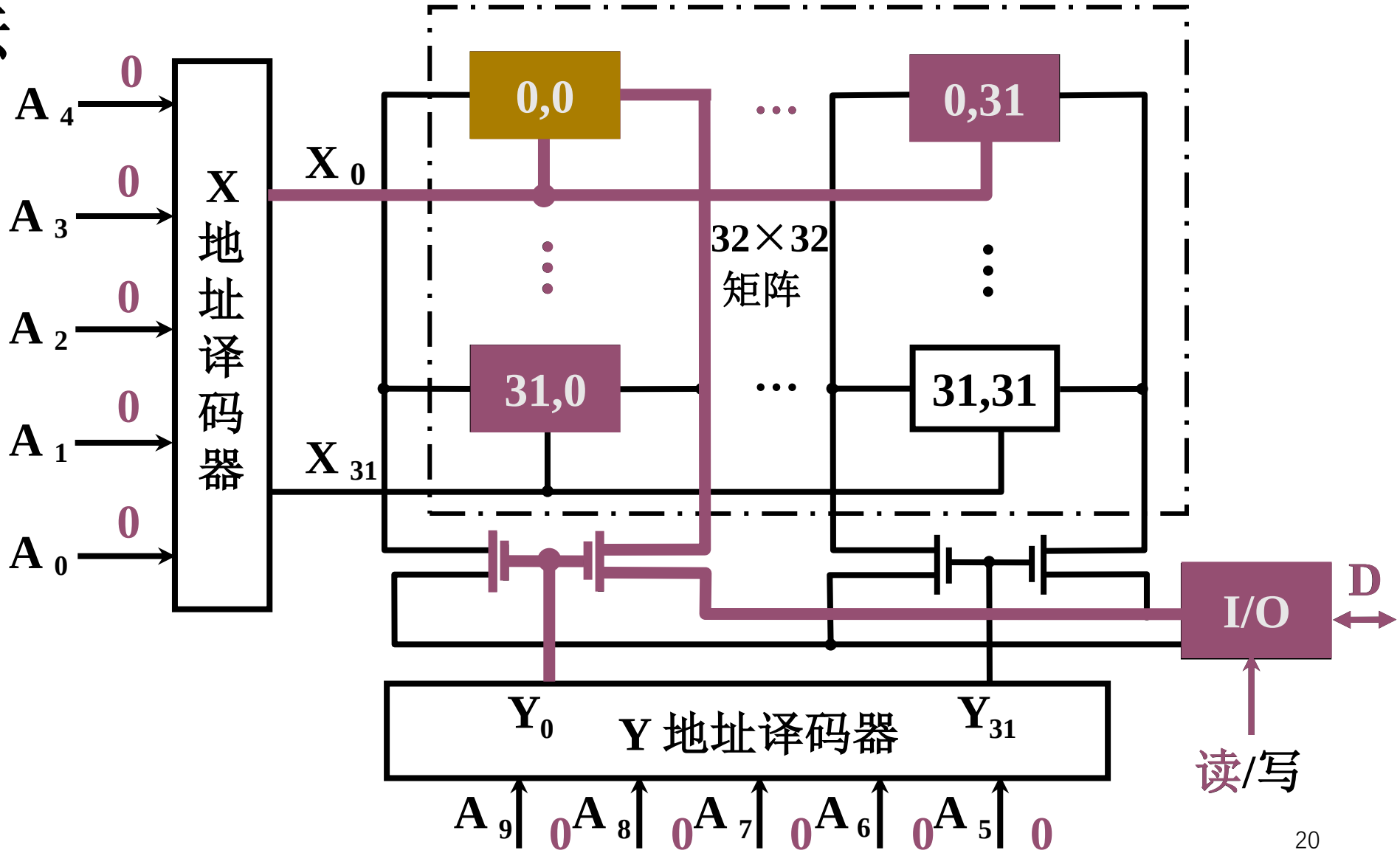


只适用于容量不大的存储芯片

6.2主存储器

(2) 重合法

主要思想是使用64根选择线，X方向和Y方向各32根。当X方向的选择线有效时，表明某一行被选中，当Y方向的选择线有效时，表明某一列被选中，同时被X行和Y列选中的这一位就是我们需要读取或写入的位。



6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

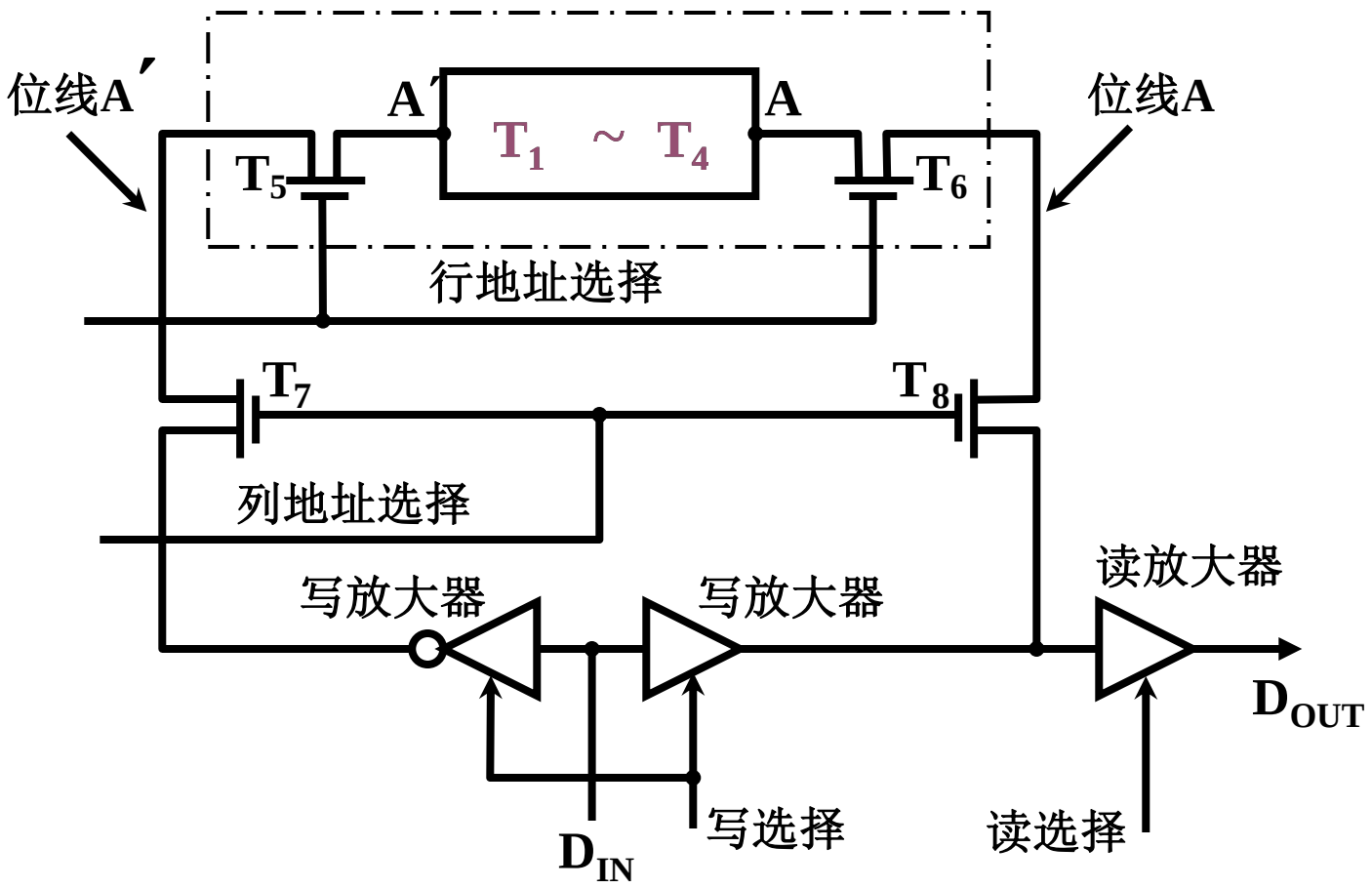
六、存储器的校验

七、提高访存速度的措施

三、随机存取存储器(Random Access Memory)

1. 静态 RAM (SRAM)

(1) 静态 RAM 基本电路：存取1位二进制数据



$T_1 \sim T_4$ 触发器

T_5、T_6 行开关

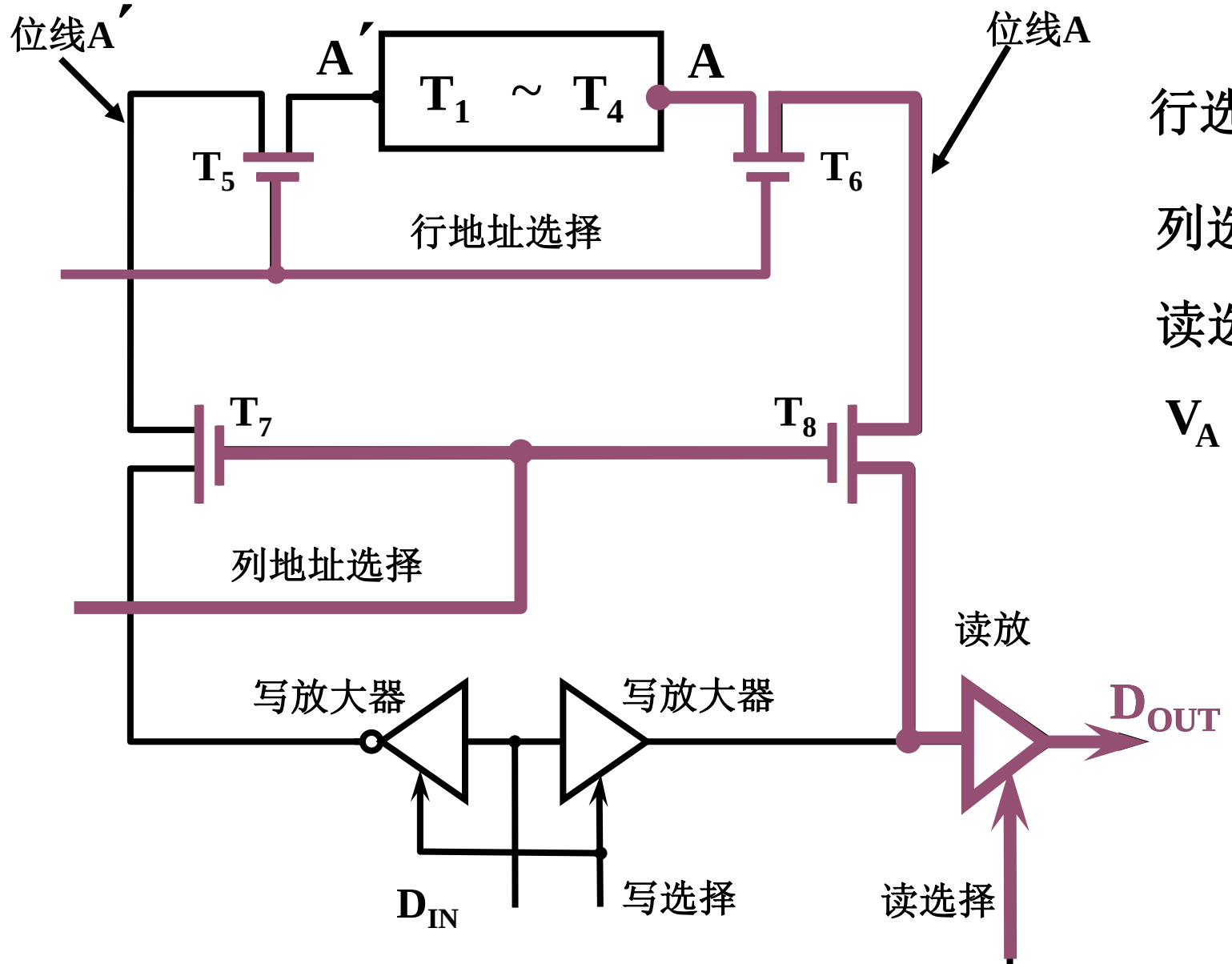
T_7、T_8 列开关

T_7、T_8 一列共用

A 触发器原端

A′ 触发器非端

① 静态RAM基本电路的读操作



行选 ⟶ T_5、T_6 开

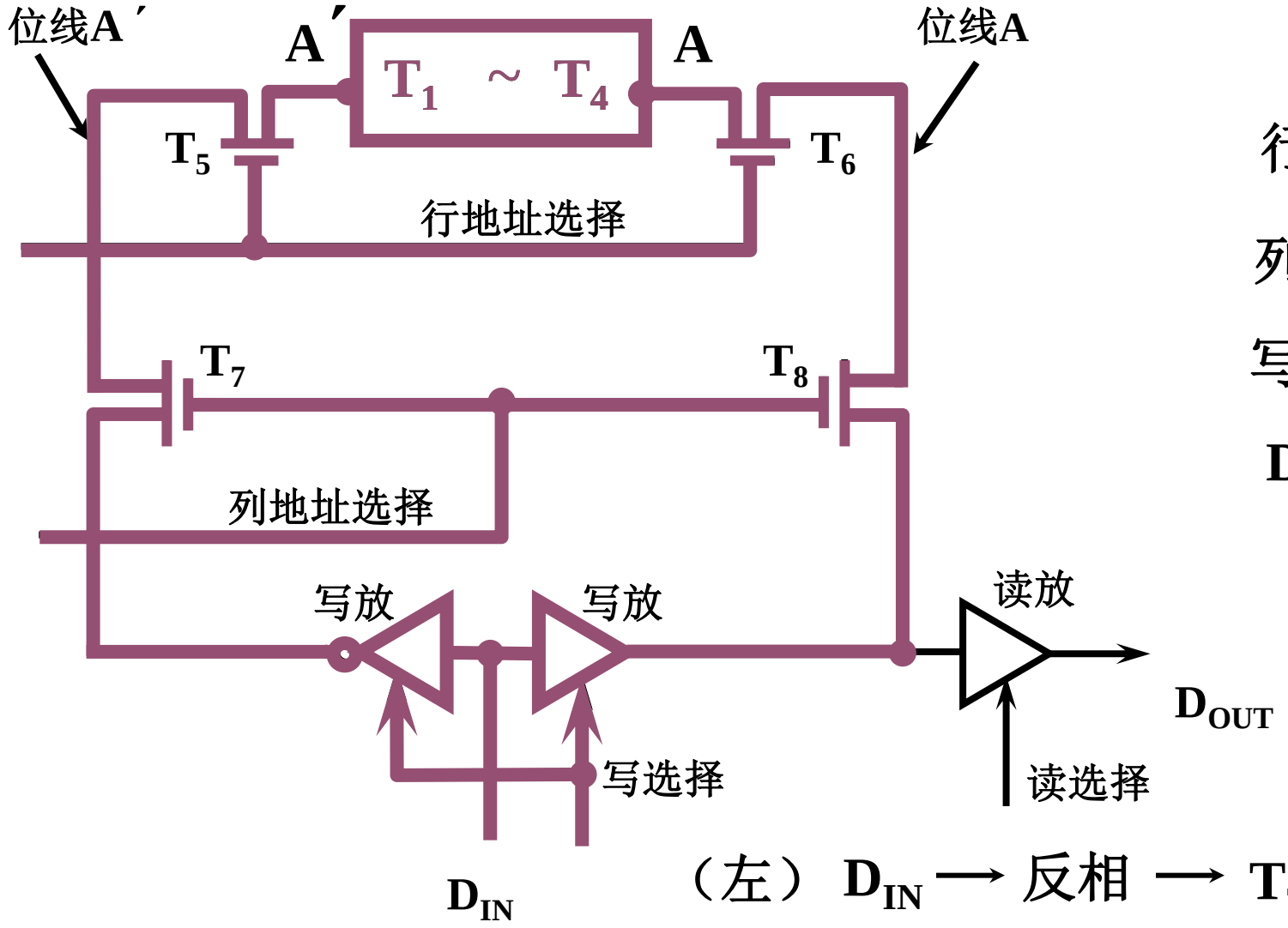
列选 ⟶ T_7、T_8 开

读选择有效

V_A ⟶ T_6 ⟶ T_8

⟶ 读放 ⟶ D_{OUT}

②静态RAM基本电路的写操作



行选 → T_5、T_6 开

列选 → T_7、T_8 开

写选择有效

D_{IN} → 两个写放大器

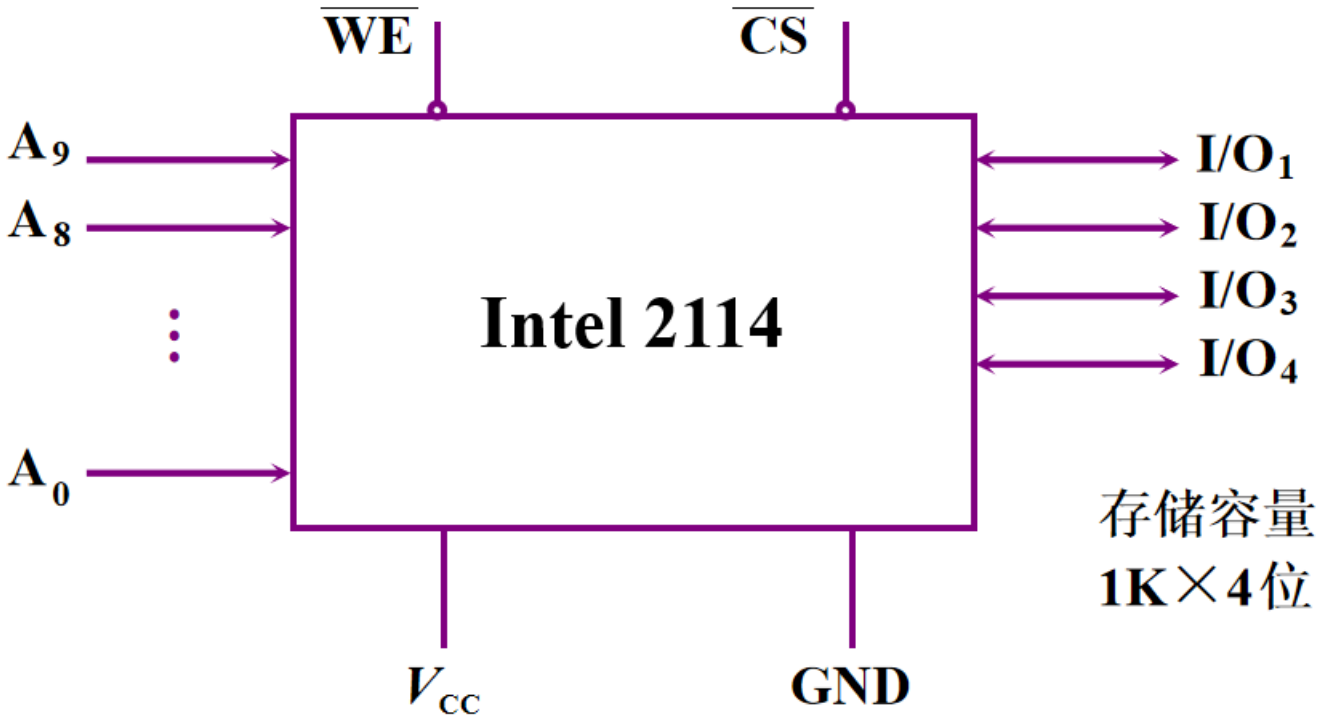
（左） D_{IN} → 反相 → T_7 → T_5 → A´

（右） D_{IN} → T_8 → T_6 → A

6.2主存储器

(2) 静态RAM芯片举例：由多个基本电路组成的静态RAM存储芯片

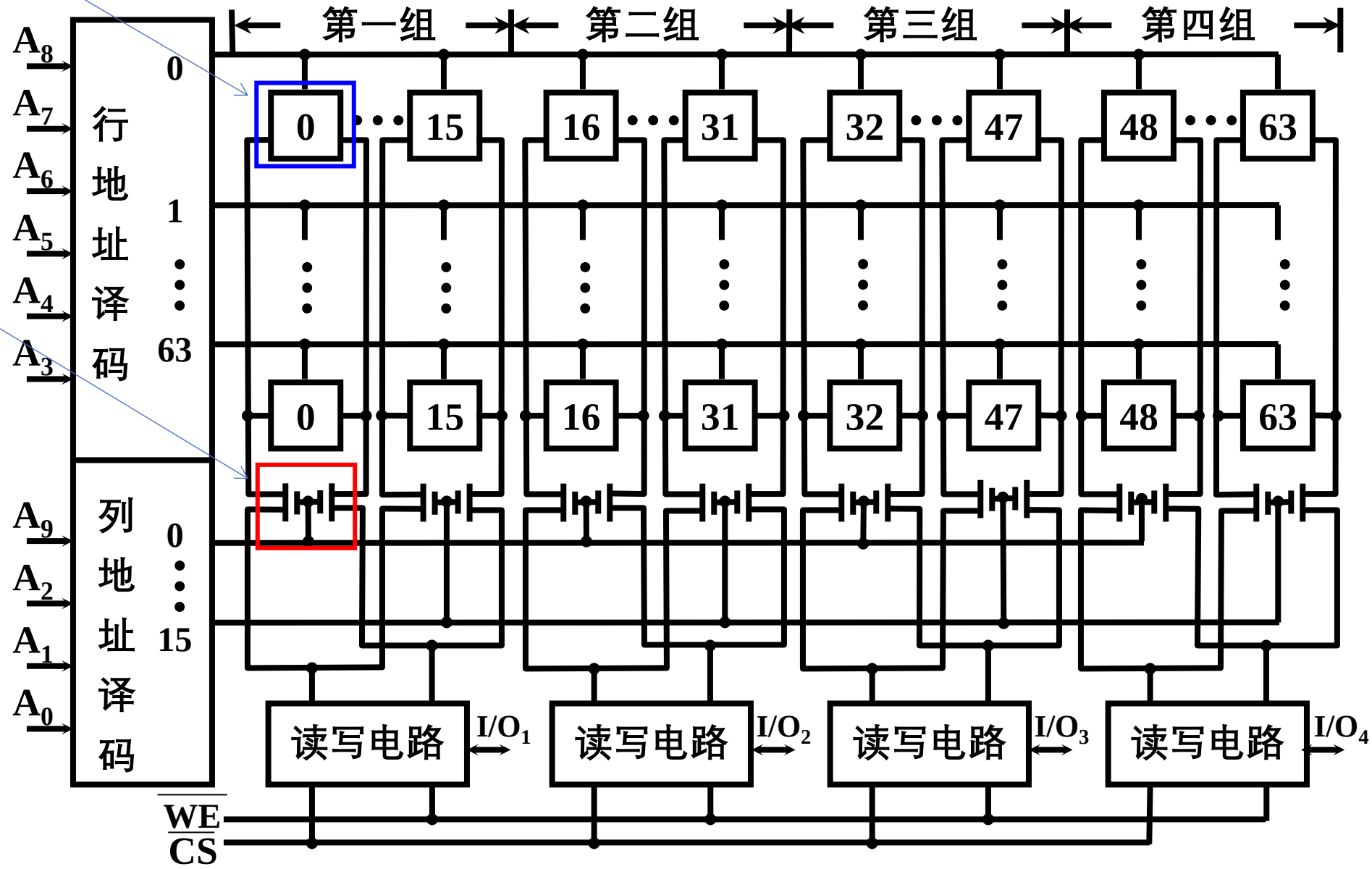
① Intel 2114 外特性



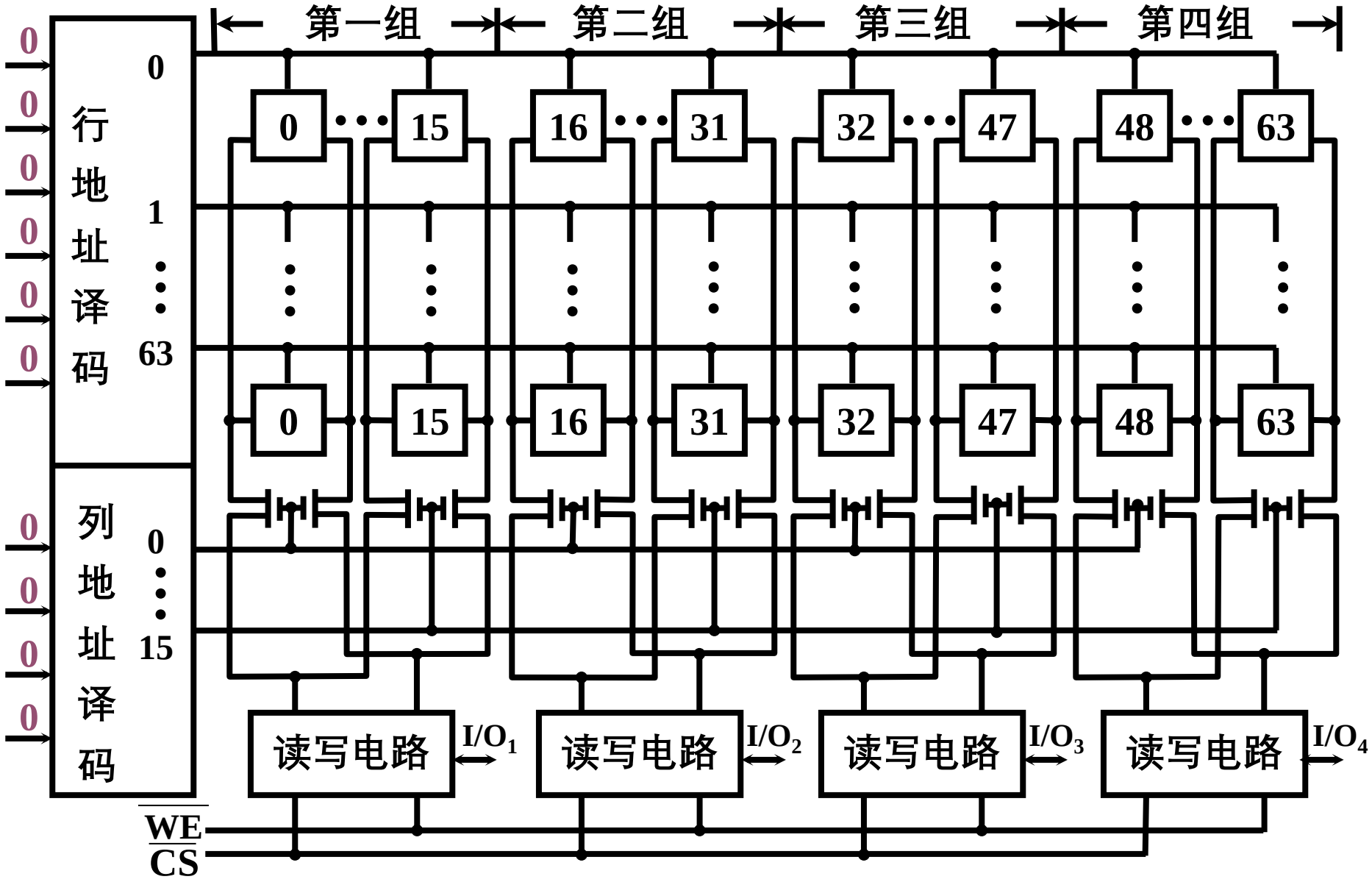
Intel 2114 RAM 矩阵(64×64)读

基本电路

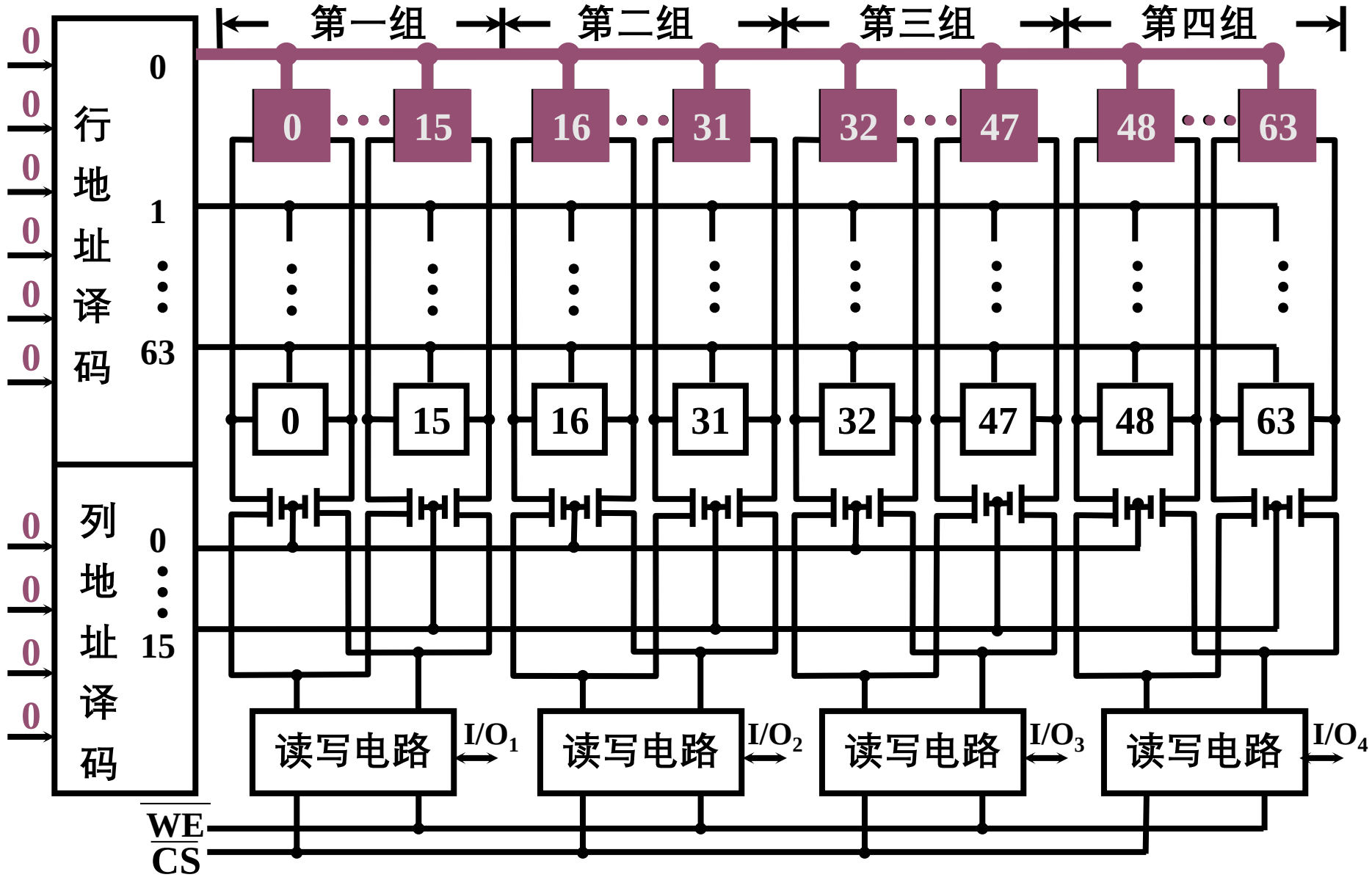
T7/T8



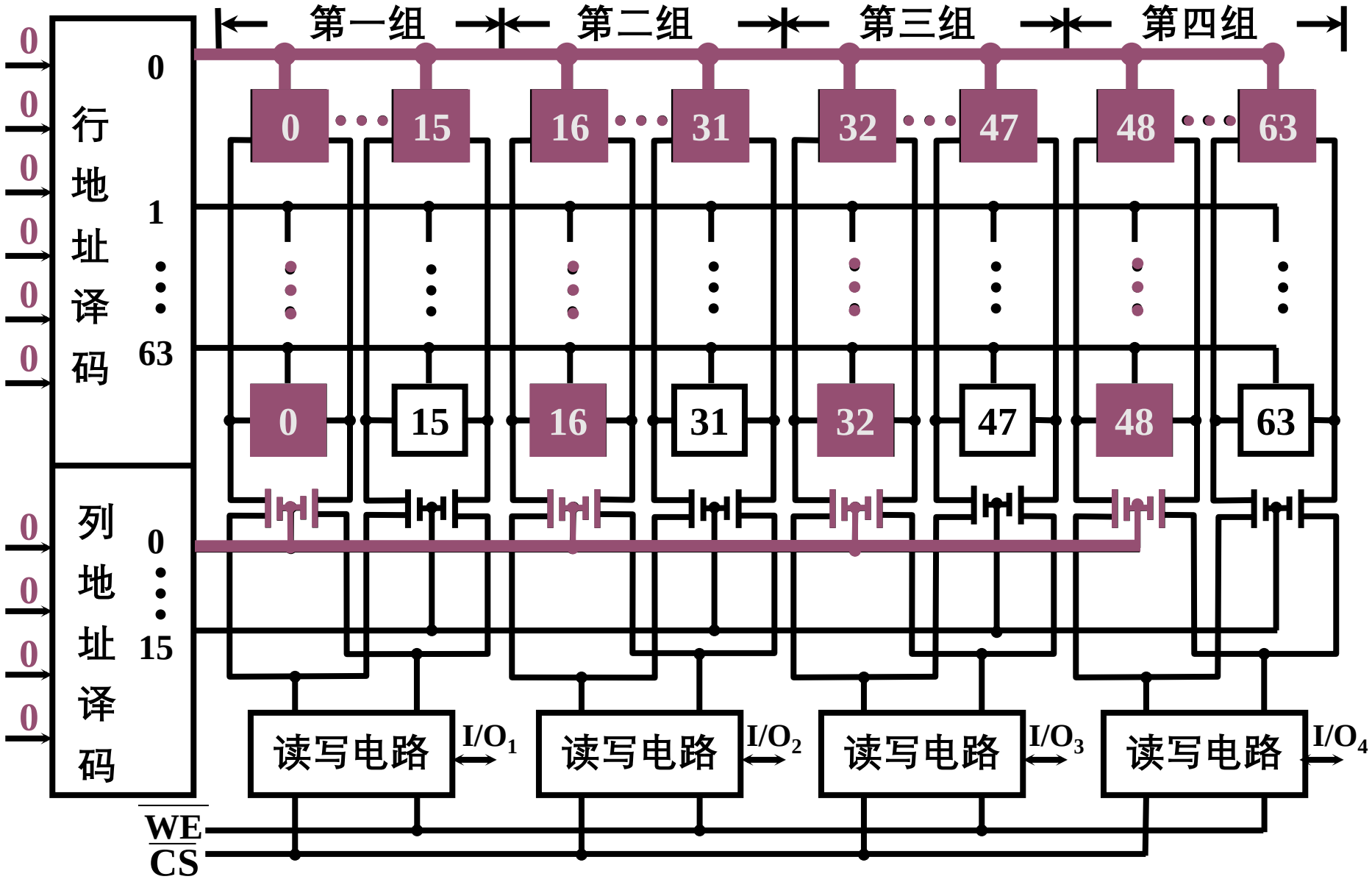
Intel 2114 RAM 矩阵(64×64)读



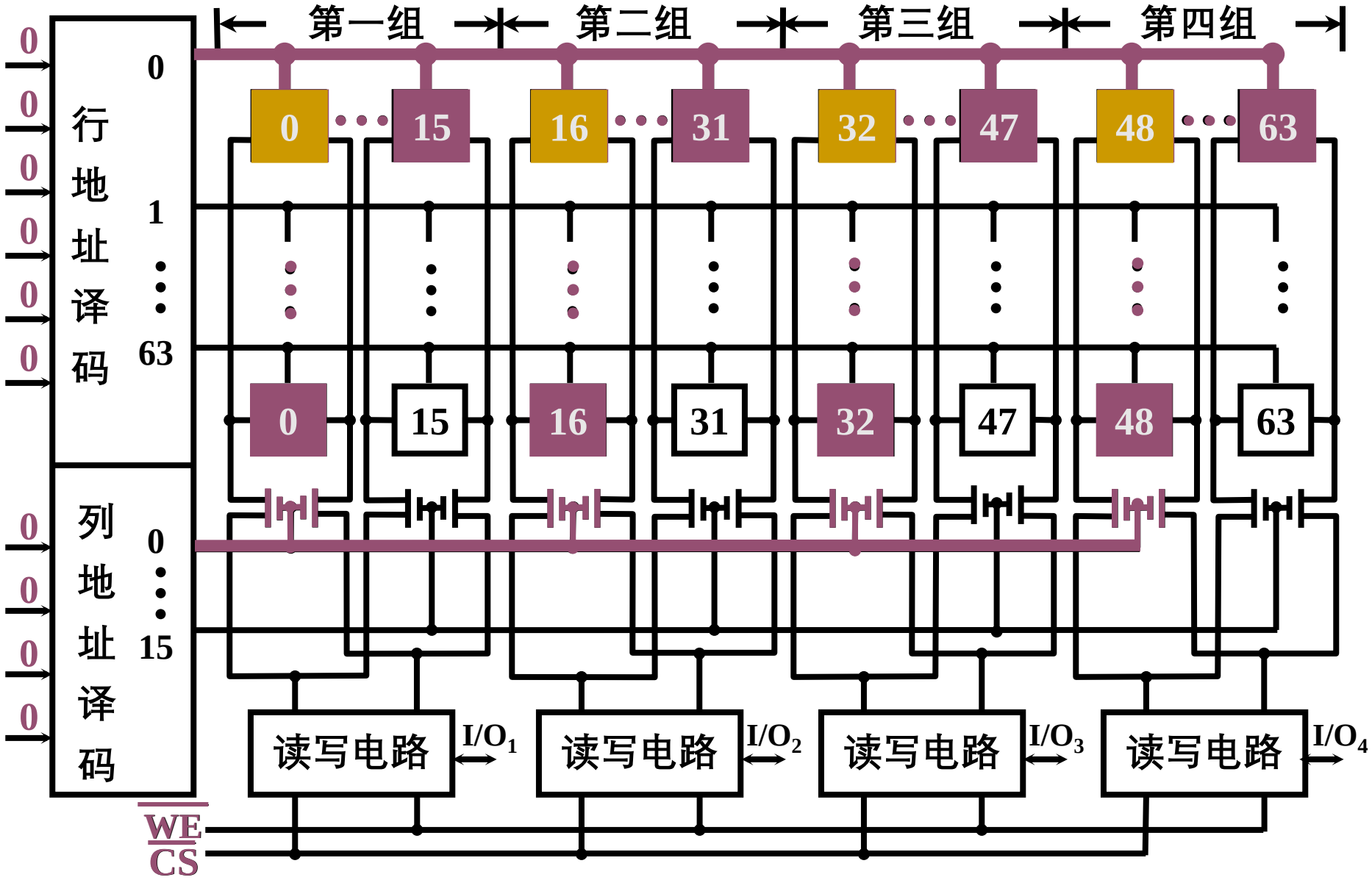
Intel 2114 RAM 矩阵(64×64) 读



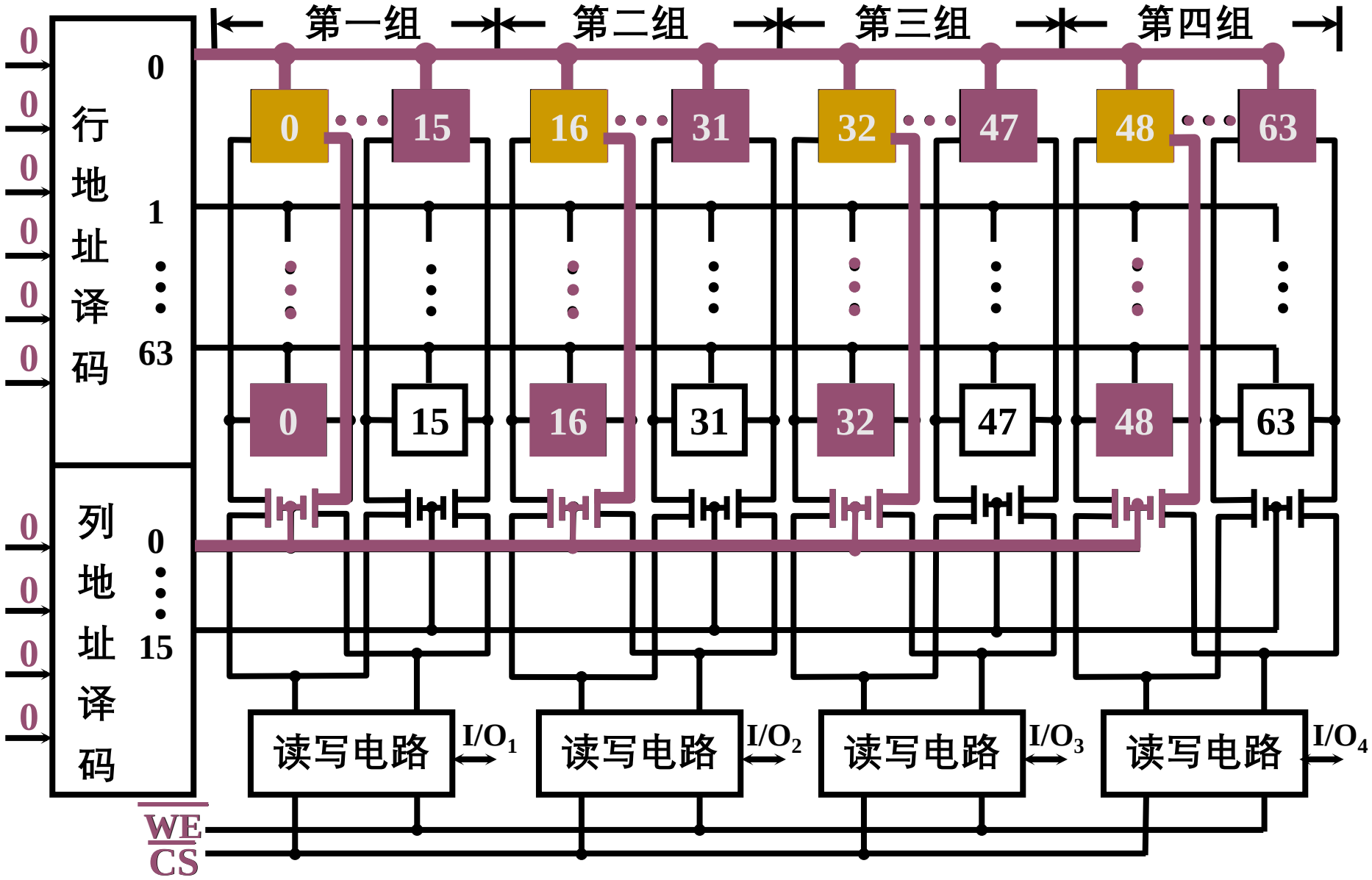
Intel 2114 RAM 矩阵(64×64) 读



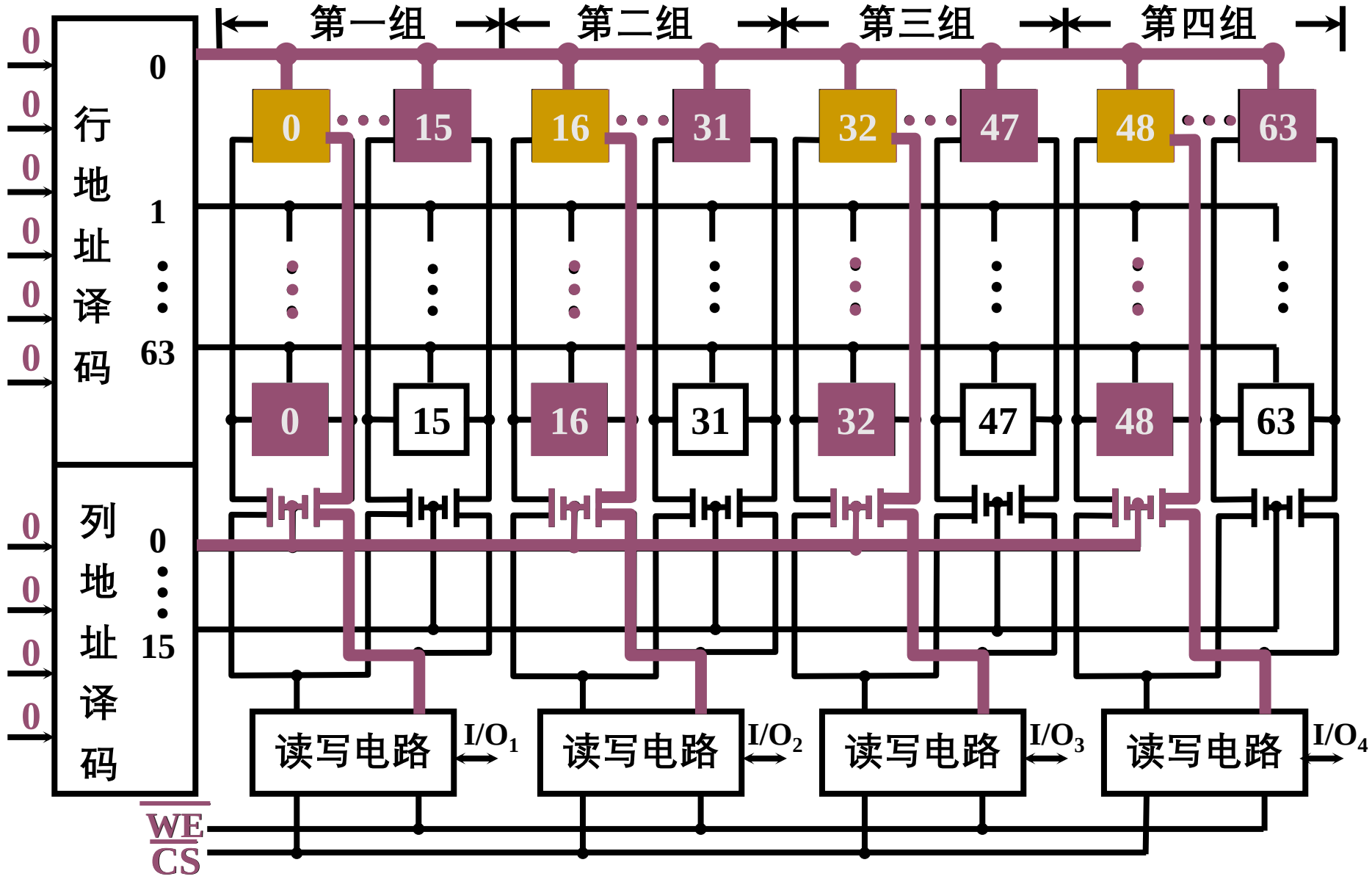
Intel 2114 RAM 矩阵(64×64) 读



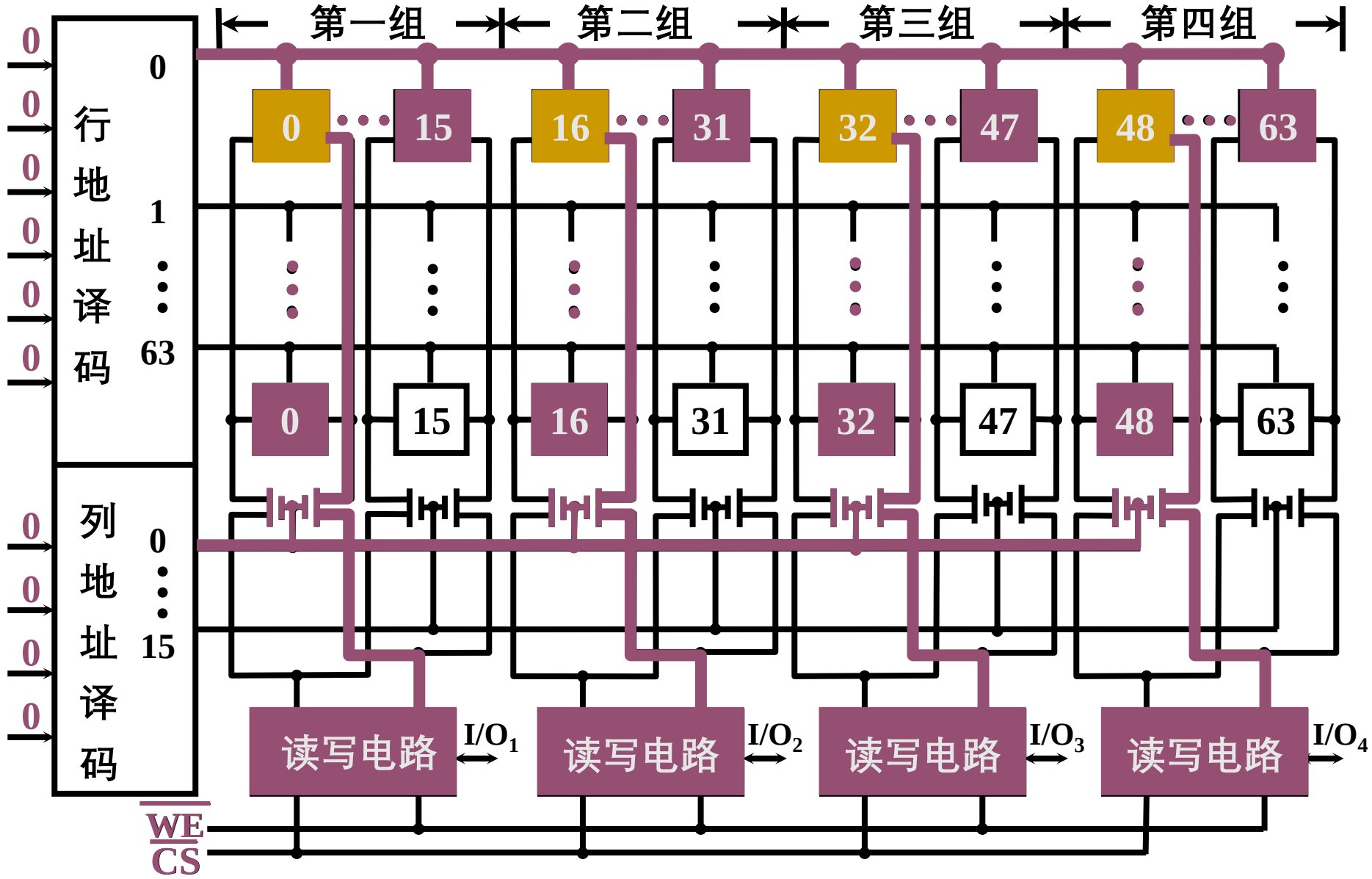
Intel 2114 RAM 矩阵(64×64) 读



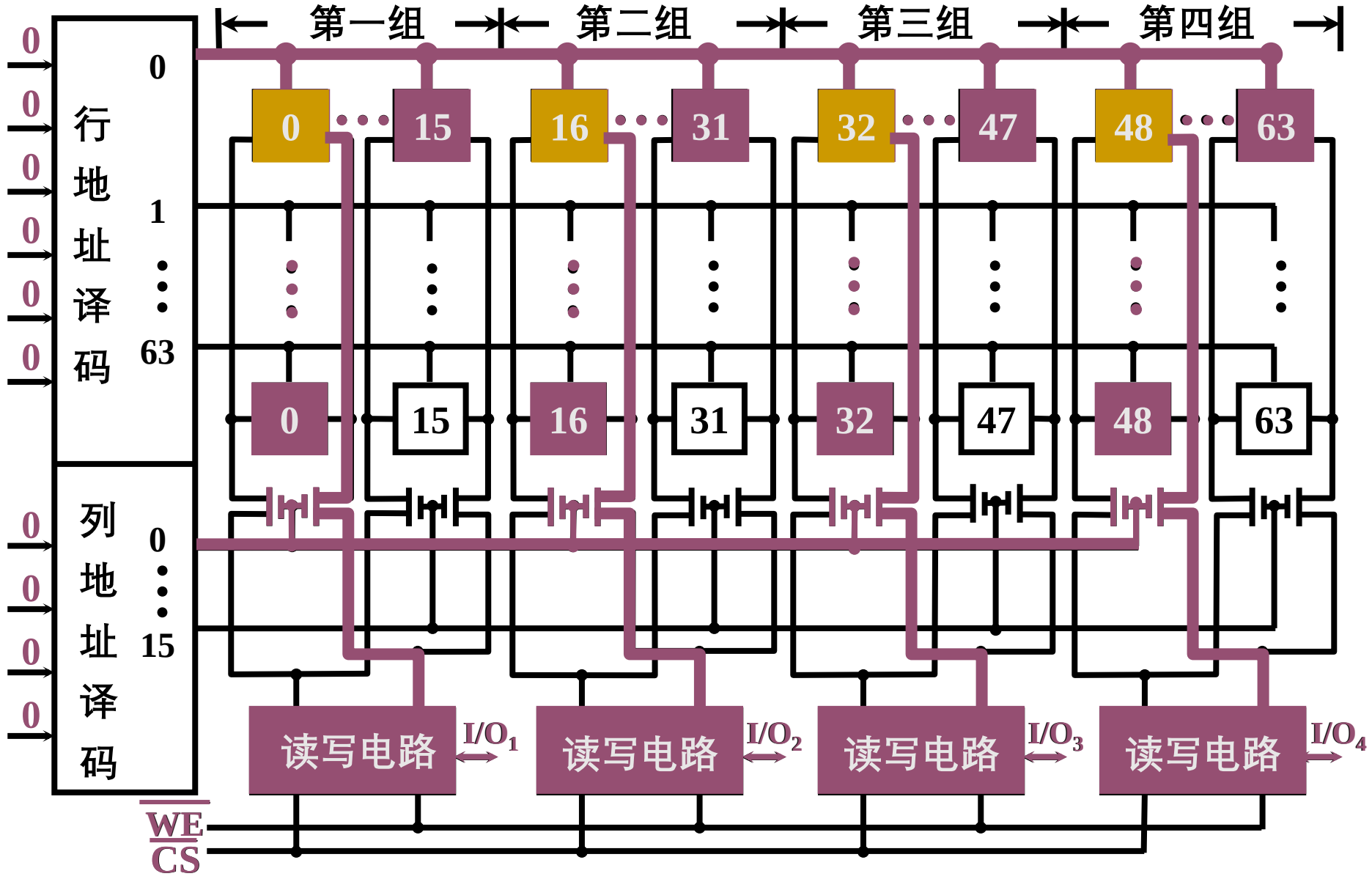
Intel 2114 RAM 矩阵(64×64) 读



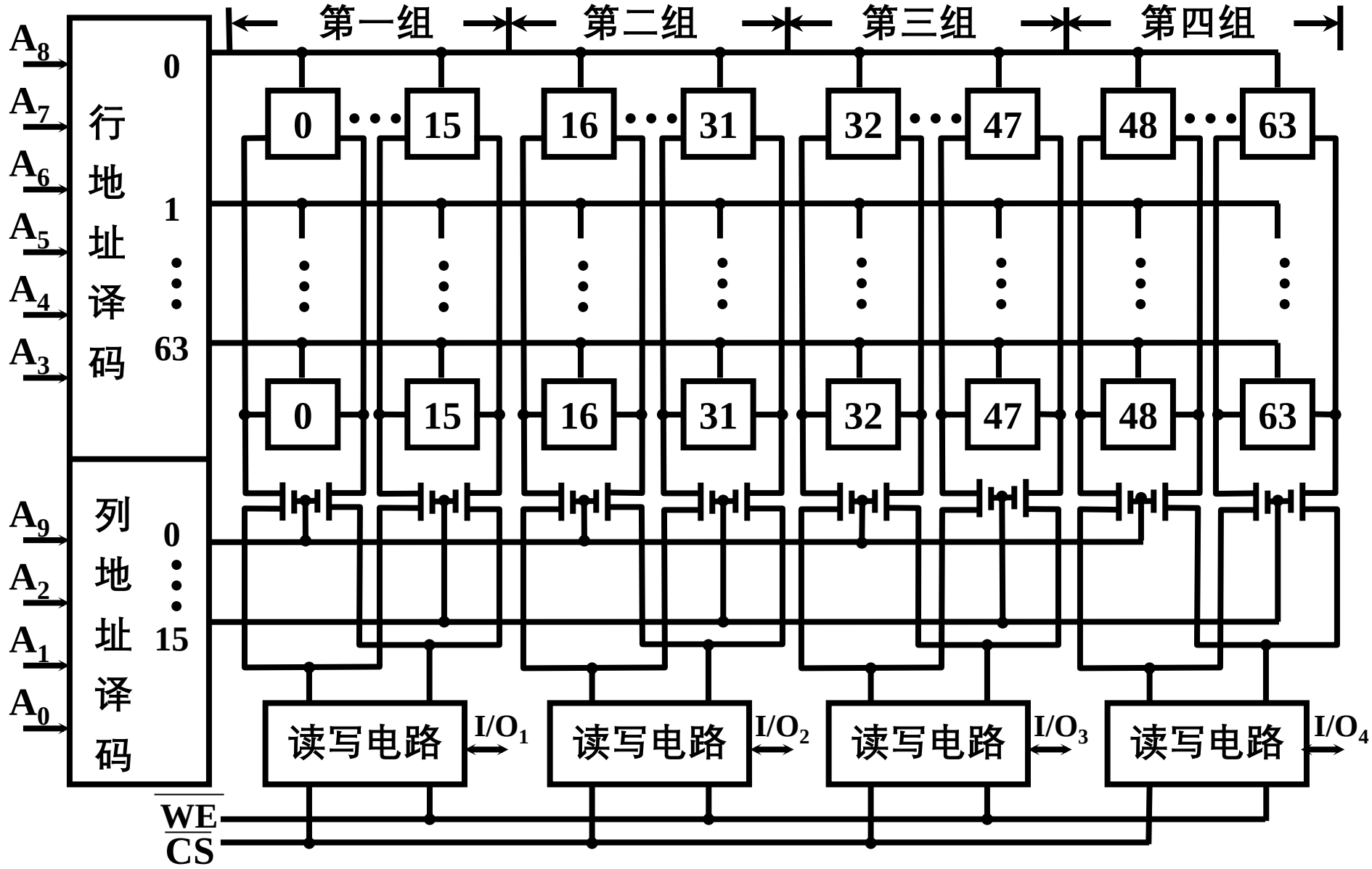
Intel 2114 RAM 矩阵(64×64) 读



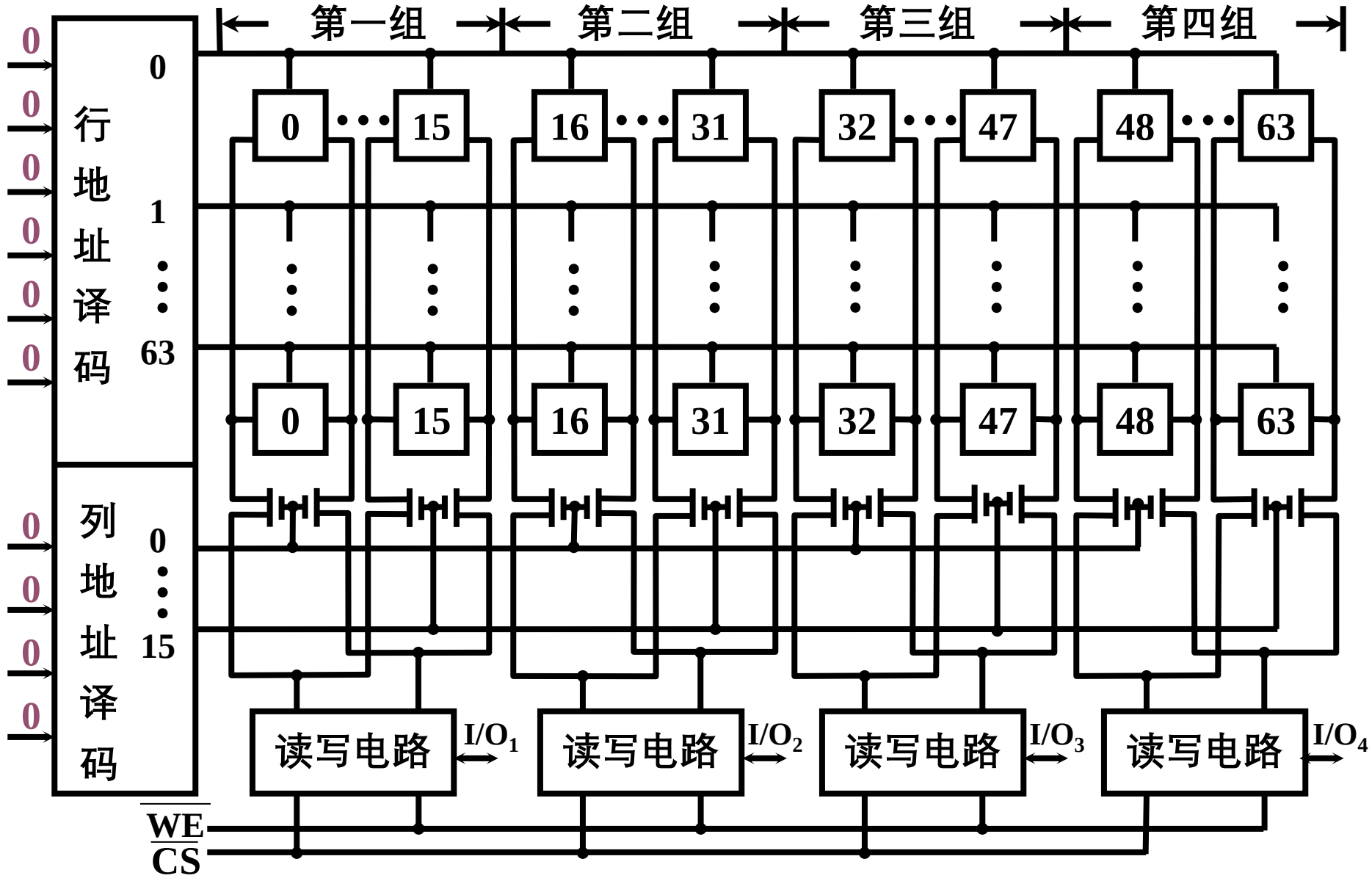
Intel 2114 RAM 矩阵(64×64) 读



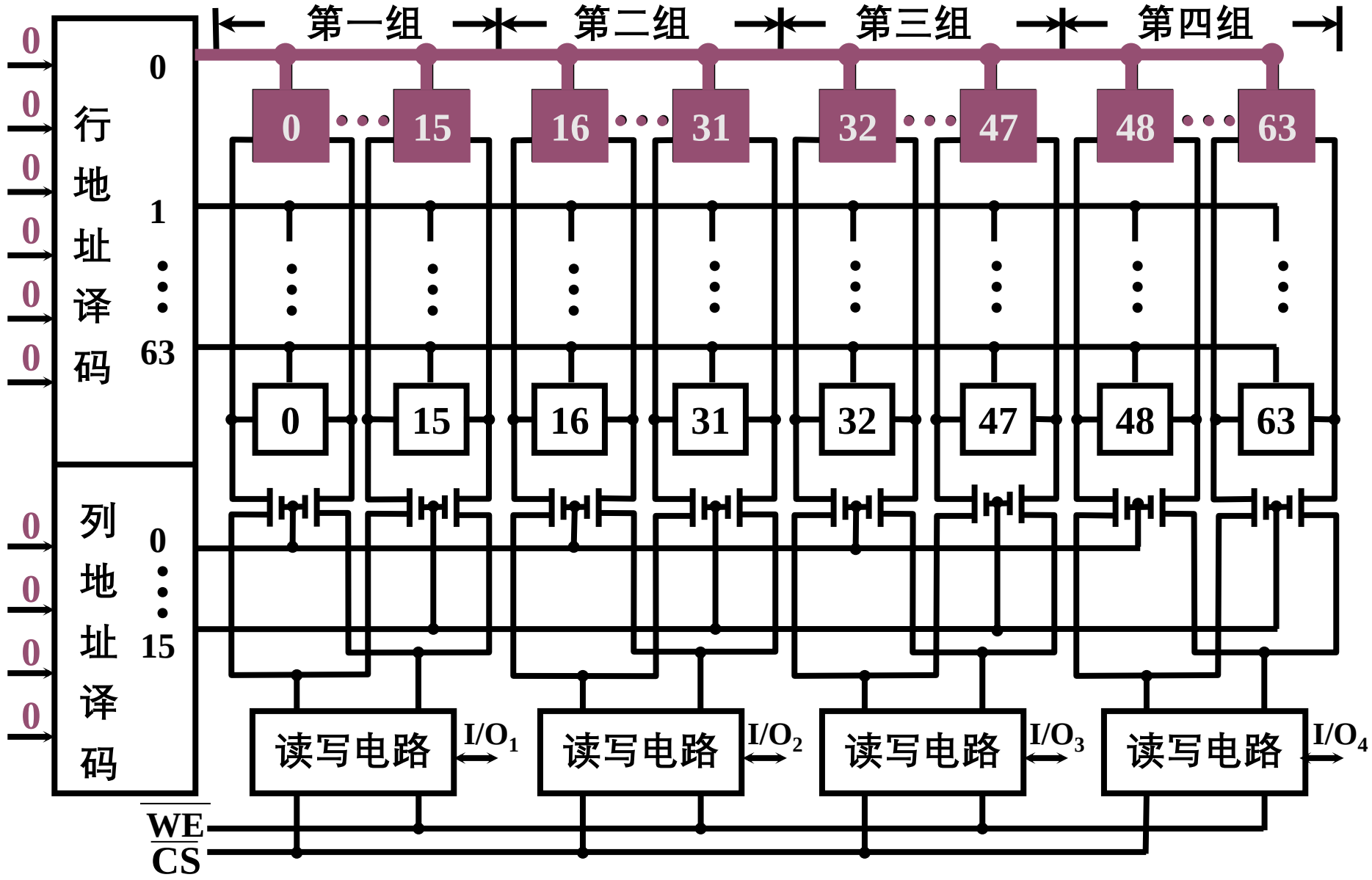
Intel 2114 RAM矩阵 (64 × 64)写



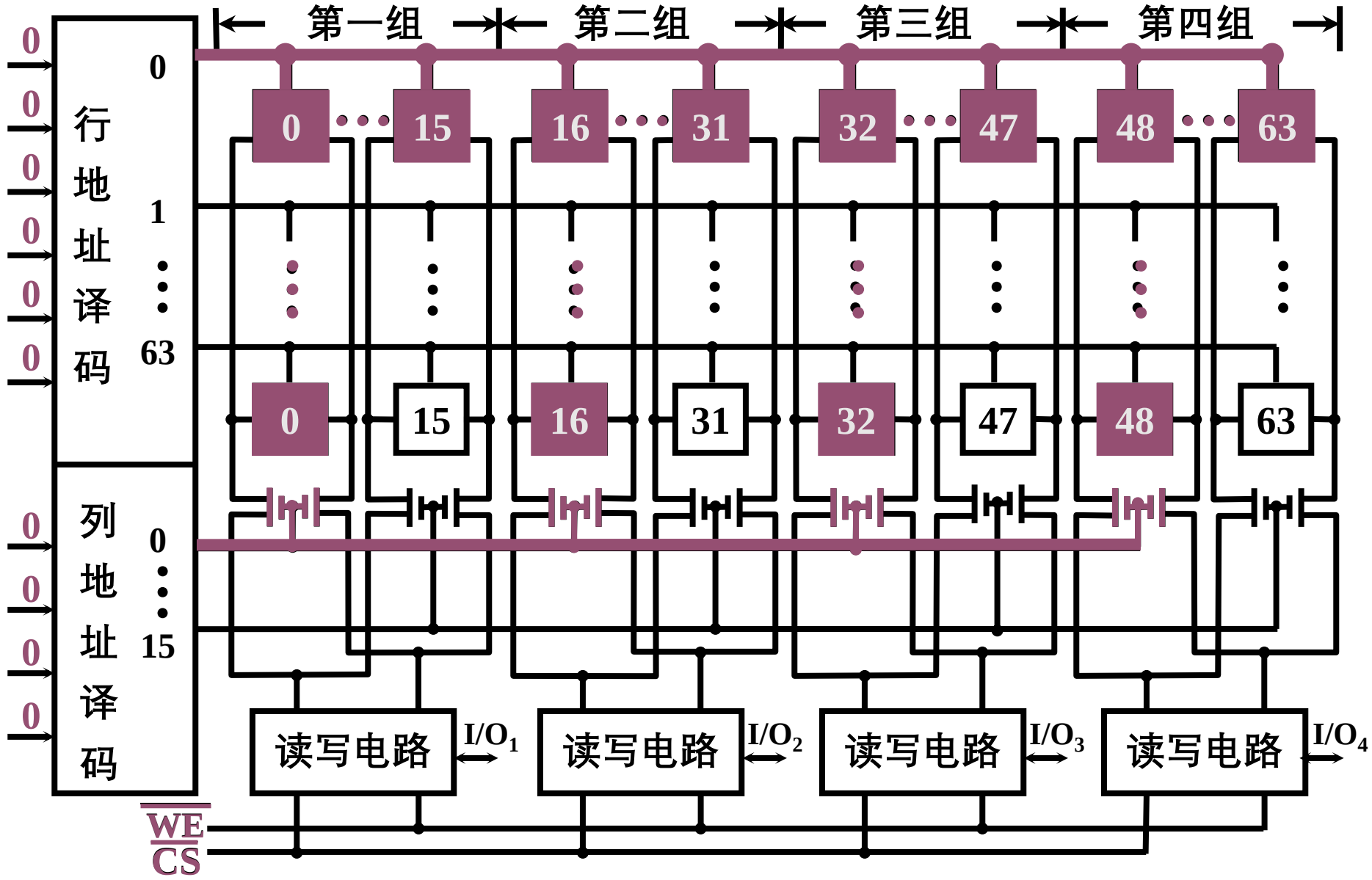
Intel 2114 RAM 矩阵(64×64)写



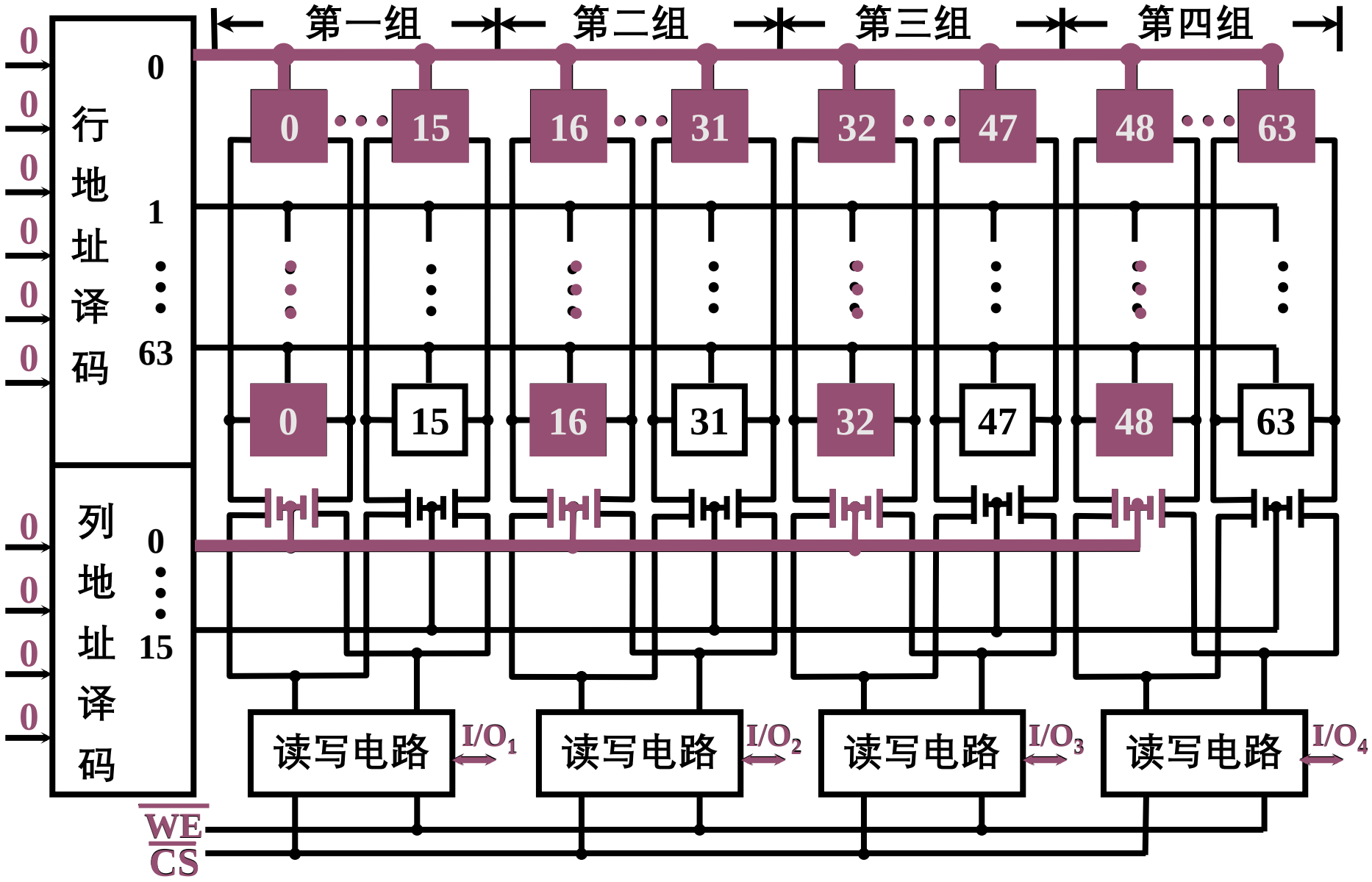
Intel 2114 RAM 矩阵(64×64)写



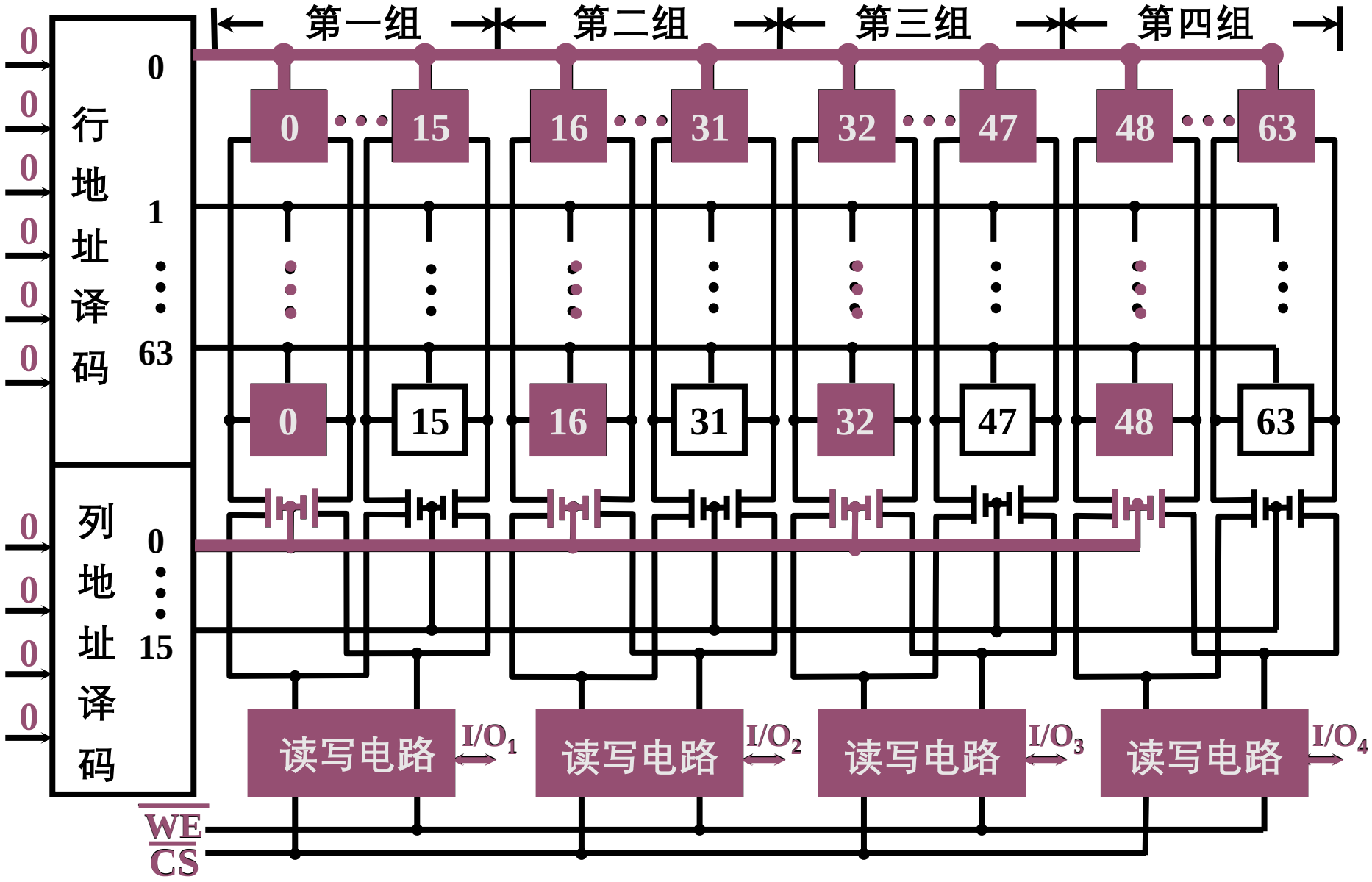
Intel 2114 RAM 矩阵(64×64)写



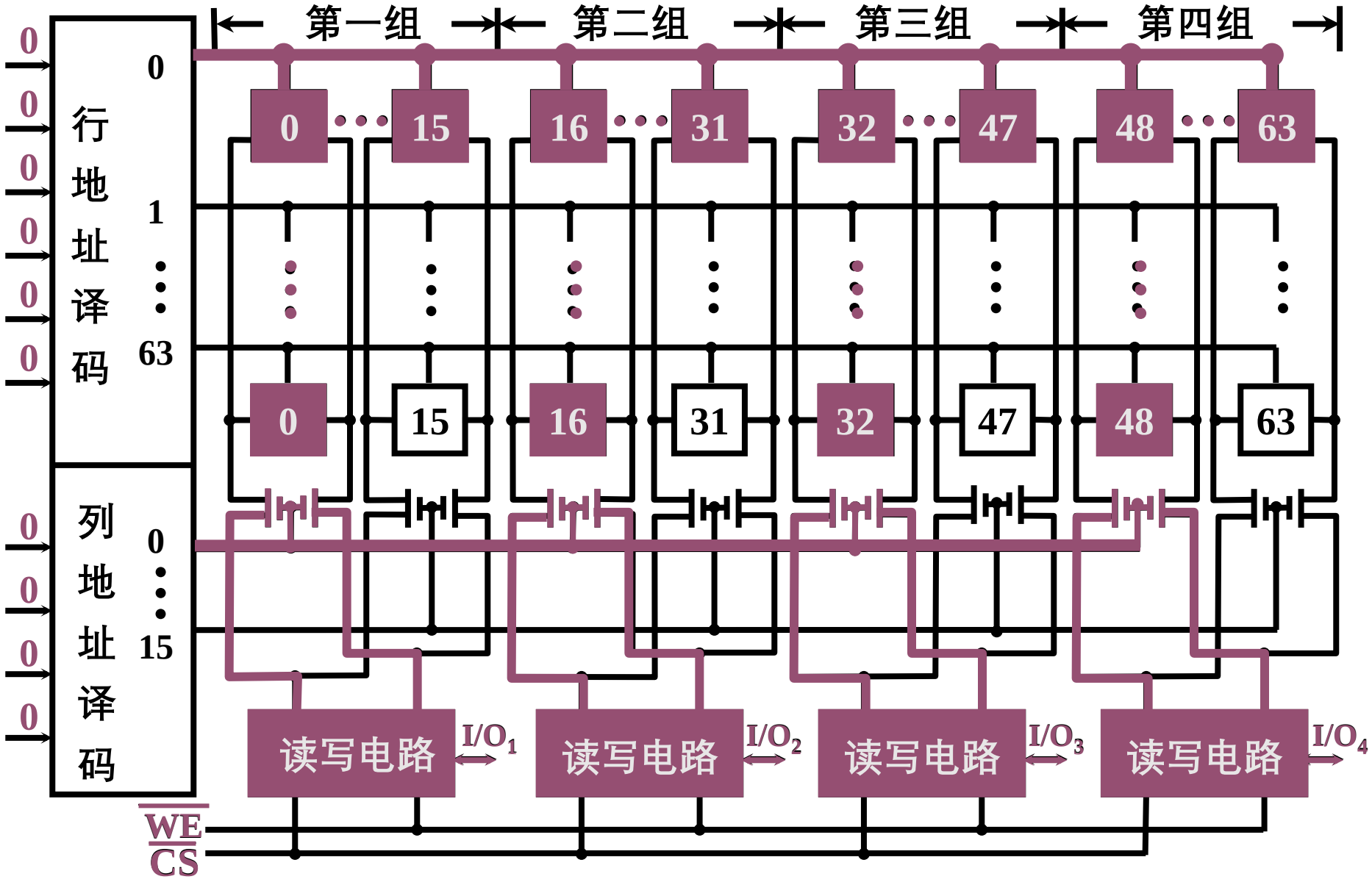
Intel 2114 RAM 矩阵(64×64)写



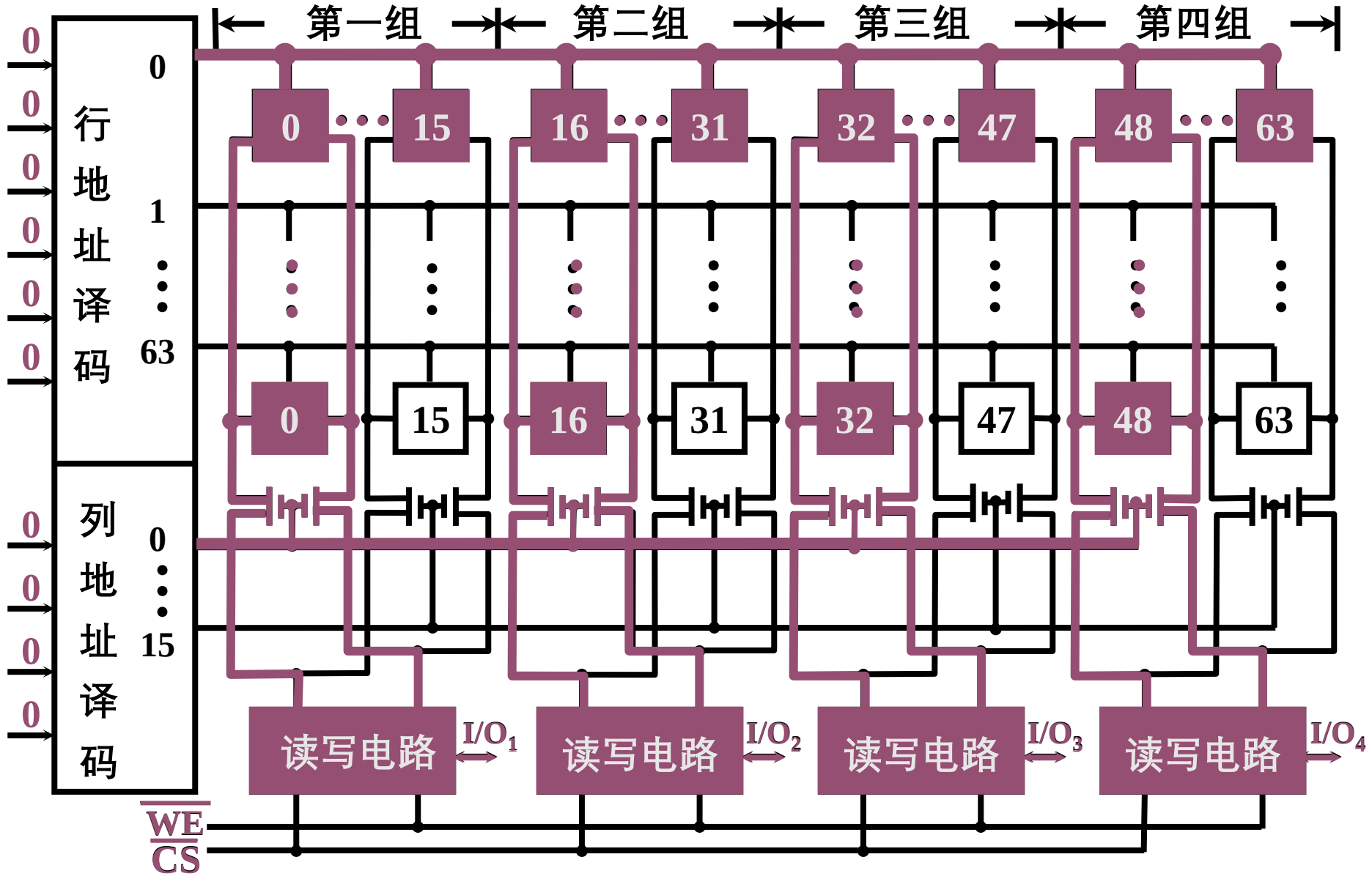
Intel 2114 RAM 矩阵(64×64)写



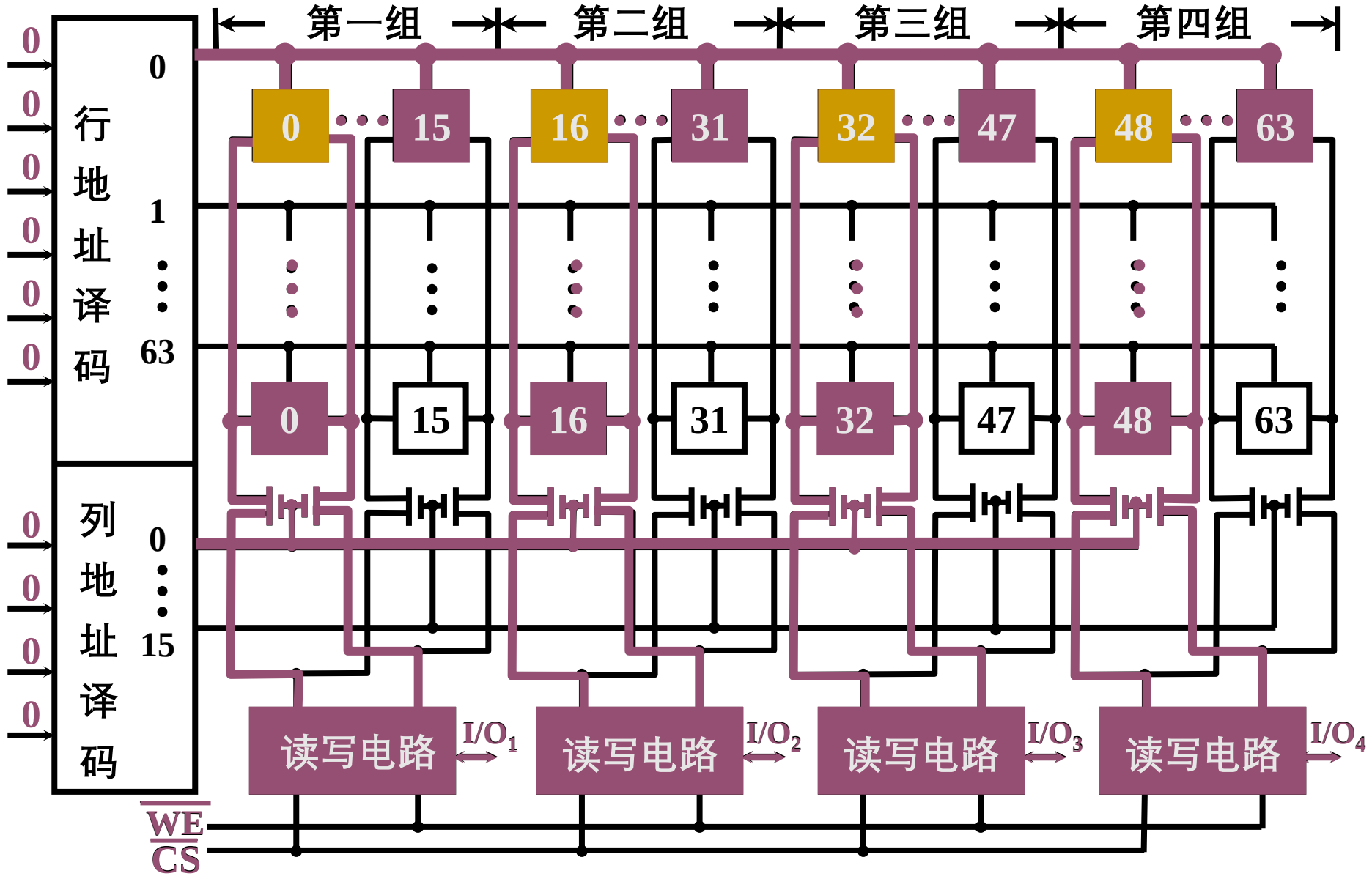
Intel 2114 RAM 矩阵(64×64)写



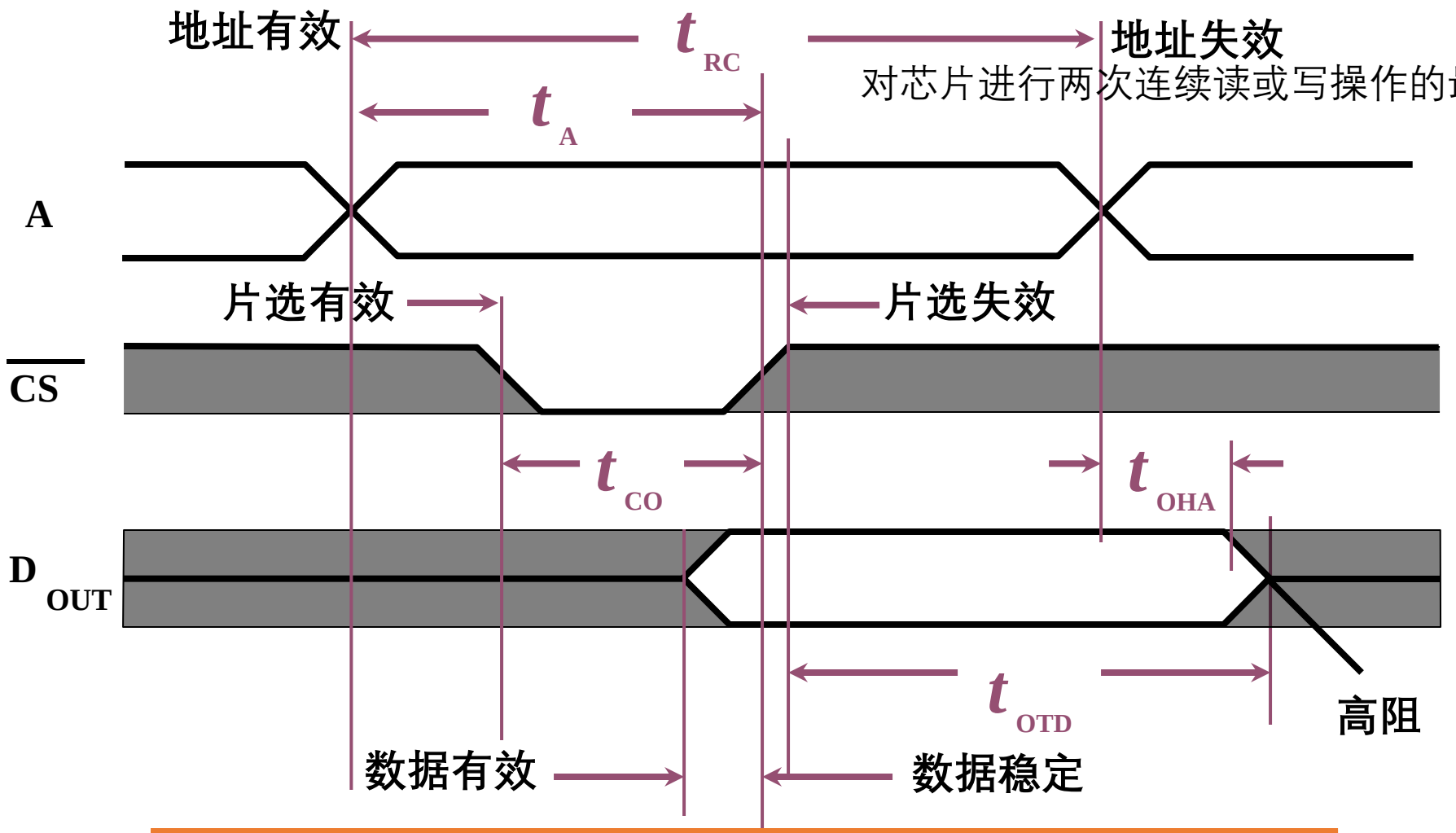
Intel 2114 RAM 矩阵(64×64)写



Intel 2114 RAM 矩阵(64×64)写



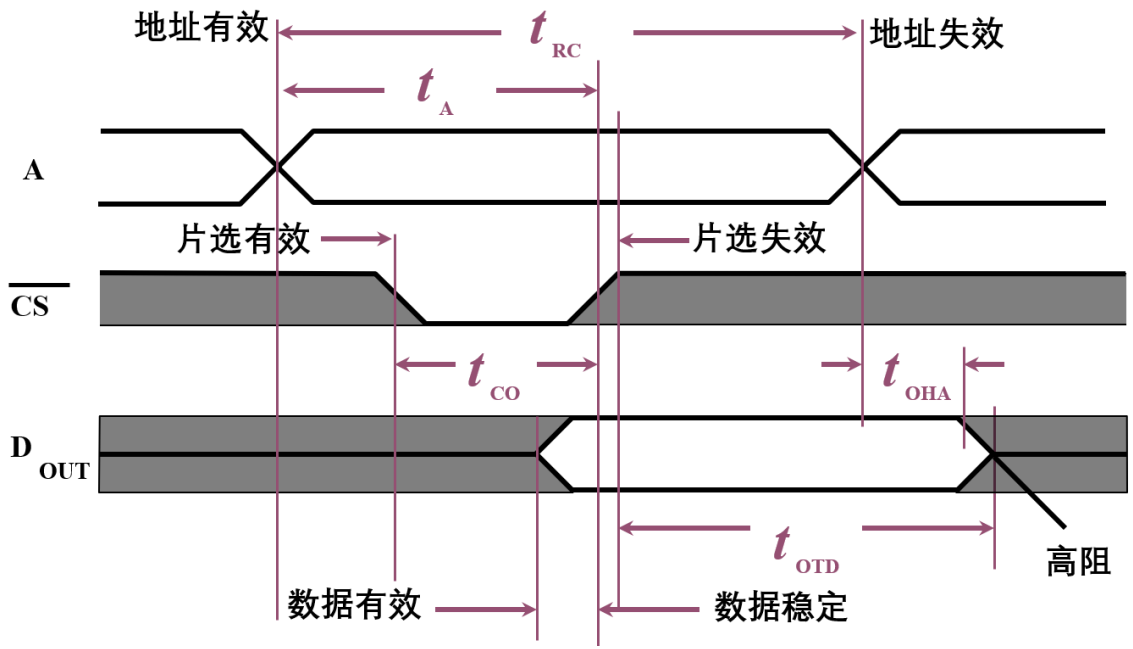
静态RAM（2114） 读 时序



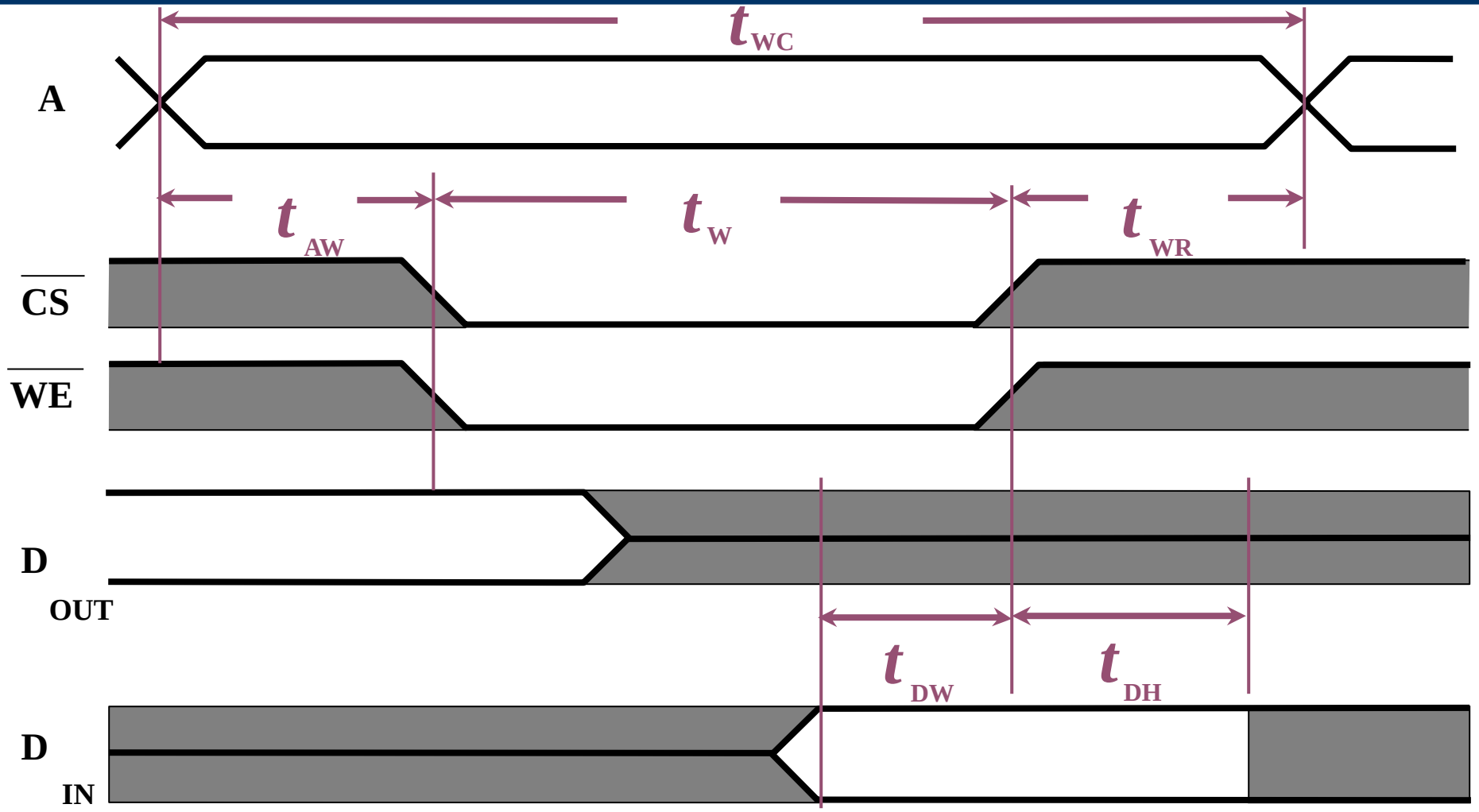
t_{OHA} 地址失效后的 数据维持时间

静态RAM（2114）读时序总结

- 当地址有效后经过t_A时间（读时间），且当片选有效后经过t_{co}时间，输出数据才能稳定。因此，我们必须在输入地址有效后的$(t_A - t_{co})$时间内给出片选有效信号，否则会导致无法正确地从存储芯片中读出数据。
- 当片选地址失效后，从片选失效到输出数据变为高阻态（输出数据失效），之间需要经过t_{OTD}的时间。这样当地址失效后，数据仍能在数据线上维持t_{OHA}的时间，这样做的目的是为了保证读取数据的可靠性，避免输入地址失效的瞬间，读取的数据就失效。



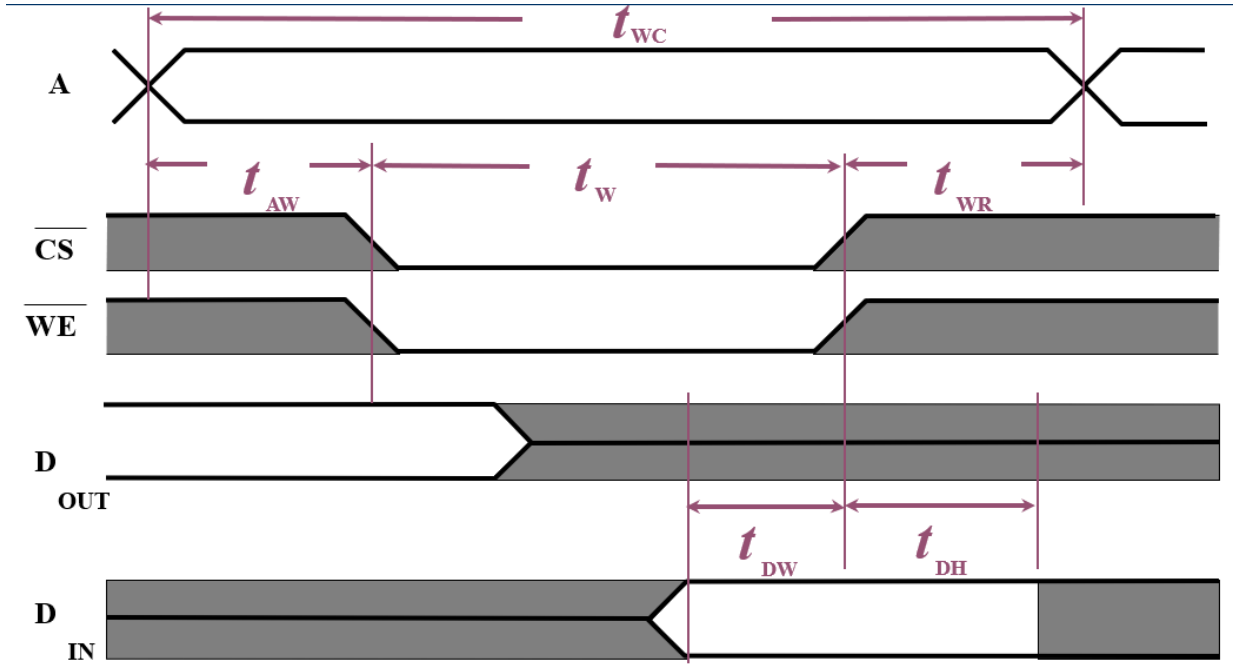
静态 RAM（2114） 写 时序



t_{DH} $\overline{WE}$ 失效后的数据维持时间

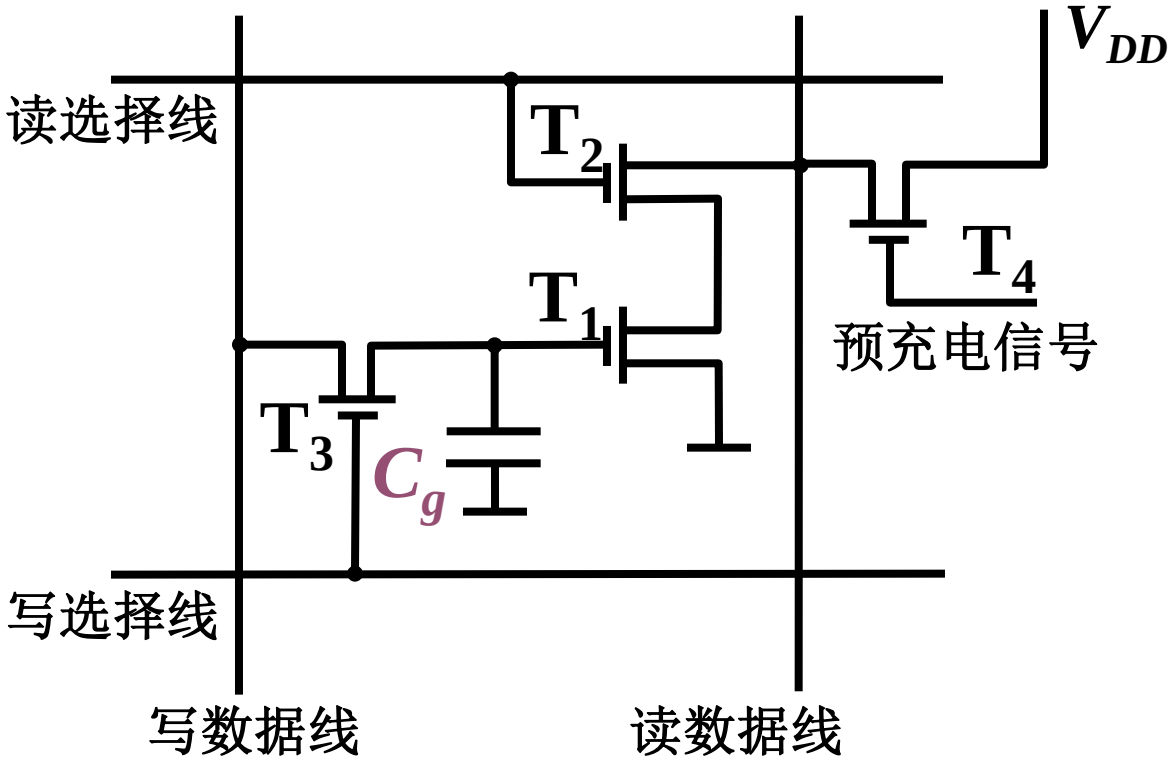
静态 RAM（2114）写时序总结

- 当输入地址有效后，需要滞后一段时间t_{AW}，片选信号$\overline{CS}$和写入使能信号$\overline{WE}$才有效。这是因为在需要写入的数据有效前，RAM的输入输出数据线上可能存在之前读取的数据。这样做可以避免写入错误的数据。
- 在写入使能信号$\overline{WE}$失效后，输入地址还必须保持一段时间t_{WR}，称为写恢复时间，这是为了保证数据能够可靠地写入。
- 需要写入的数据D_{IN}，必须在片选信号和写入使能信号失效前的t_{DW}时间内，在数据线上稳定，并且在片选信号和写入使能信号失效后仍要维持一段时间t_{DH},这样做也是为了保证数据可靠地写入。



2. 动态 RAM (DRAM)

(1) 动态 RAM 三管式基本单元电路

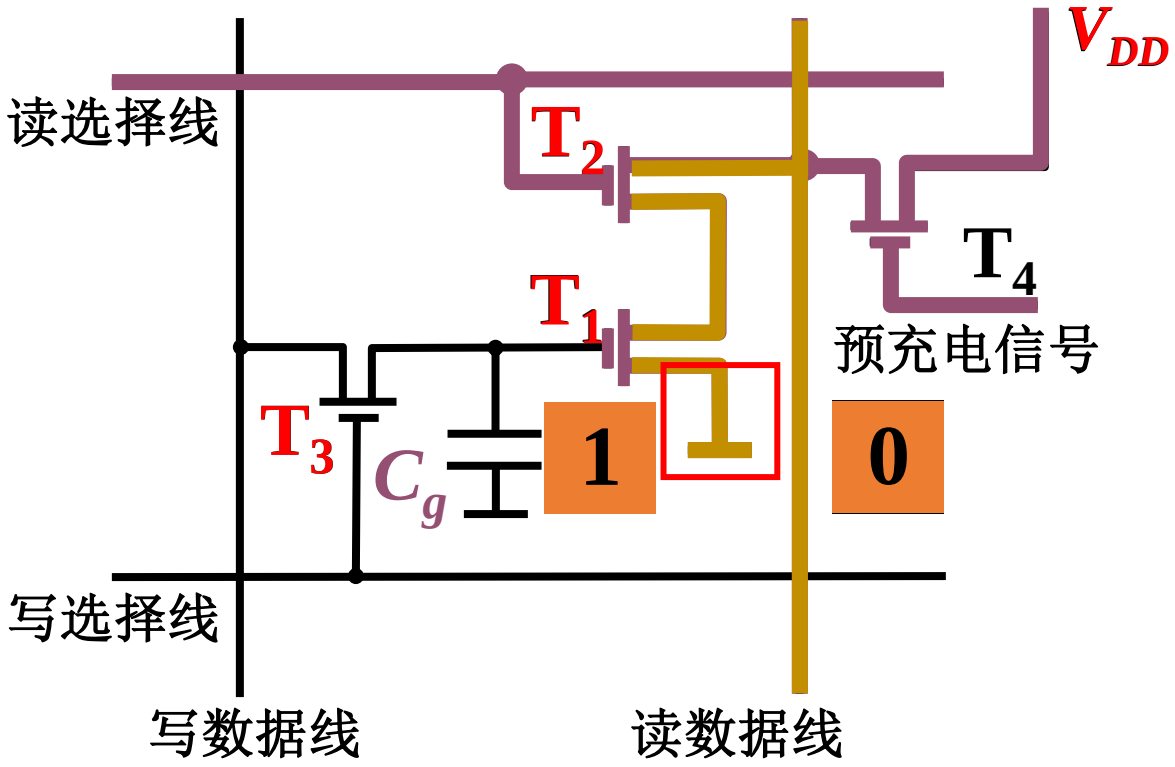


- 和静态RAM相比，动态RAM主要是靠电容存储电荷的原理来寄存信息（C_g）。如果电容C_g中存储有足够多的电荷，则表示存储信息“1”，如果电容中没有存储电荷则表示存储信息“0”。
- 三管式动态RAM基本单元电路中的三个MOS管T1、T2和T3主要用于控制基本单元电路读取或写入数据的过程。
- T4 MOS管不属于基本单元电路，共用于一列基本单元电路。

2. 动态 RAM (DRAM)

(1) 动态 RAM 三管式

基本单元电路读取



- 给T4 MOS管一个预充电信号，使得读数据线达到高电平V_{DD}（此时读数据线上的信息为“1”）。
- 读选择线选中该基本单元电路，打开T2开关
- 此时，如果电容C_g没有存储电荷，则T1不会被导通，读数据线上的高电平不变，读出“1”信息。
- 此时，如果电容C_g存储有足够多的电荷，则T1被导通。因为此时T2和T1均导通，使得读数据线接地（图中红色框），这样读数据线上的高电平就降为低电平，读出“0”信息。

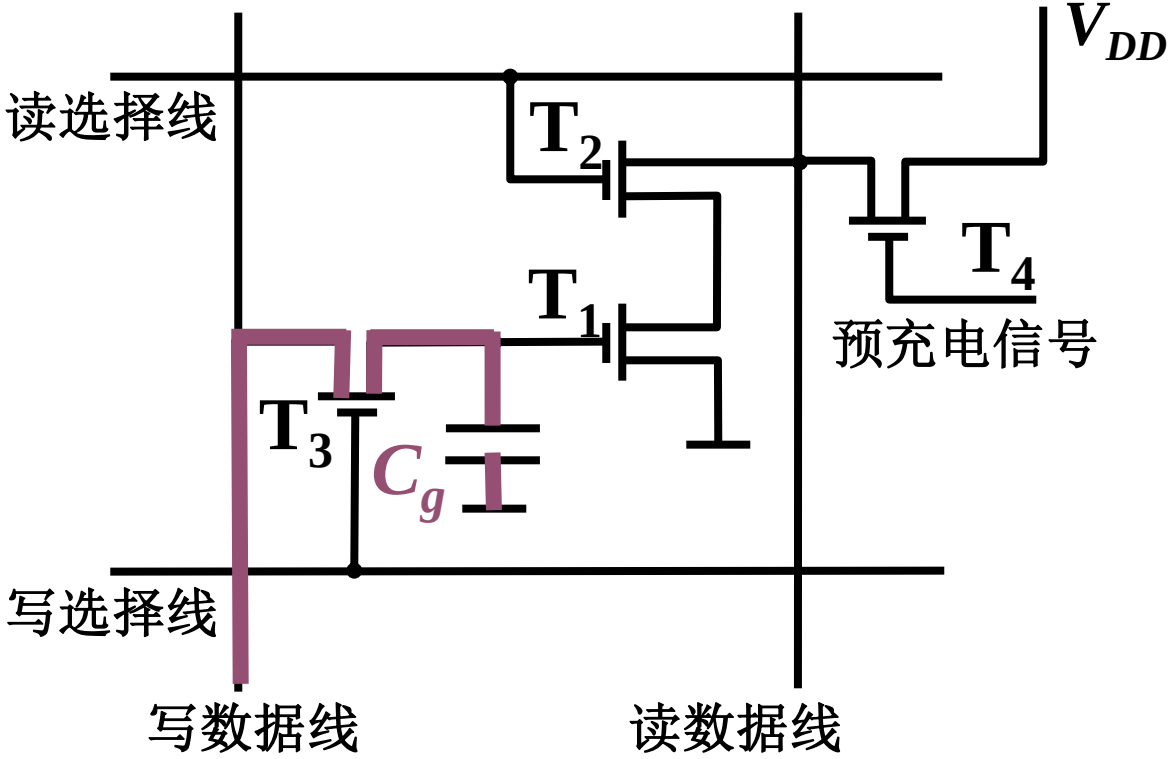
总结：

读出与原存信息相反（破坏性读）

2. 动态 RAM (DRAM)

(1) 动态 RAM三管式

基本单元电路写入



- 首先将需要写入的数据放到写数据线上。
- 写选择线选中该基本单元电路，打开T3。
- 若写数据线上的数据为“1”，则电容C_g充电，存储信息为“1”；若写数据线上的数据为“0”，则电容C_g放电，存储信息为“0”。

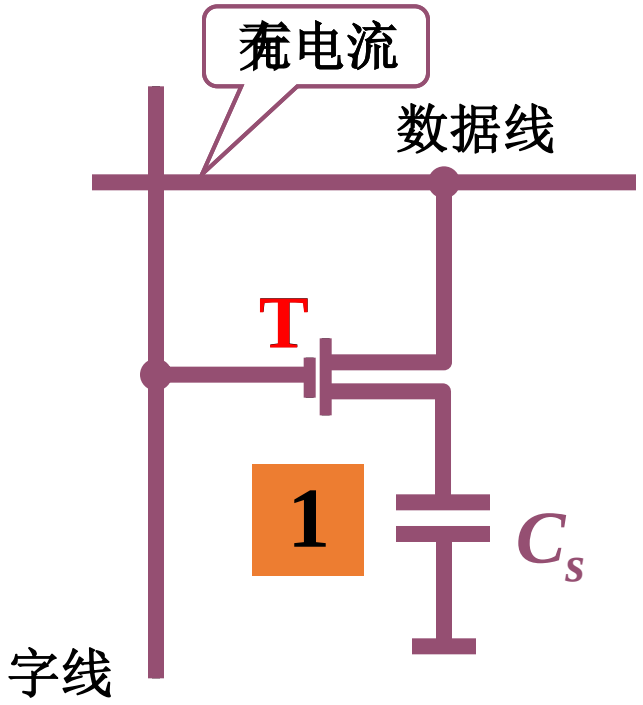
总结：

写入与输入信息相同

2. 动态 RAM (DRAM)

(1) 动态 RAM 单管式

基本单元电路读取



- 读出数据时，字线选中该基本单元电路，将T导通。
- 此时如果C_s中没有存储电荷(原始存储的数据为0)，则不会放电，数据线上也不会产生电流，从而读出数据“0”。
- 如果C_s中存储有电荷(原始存储的数据为1)，则会放电，经T管，在数据线上产生电流，从而读出数据“1”。

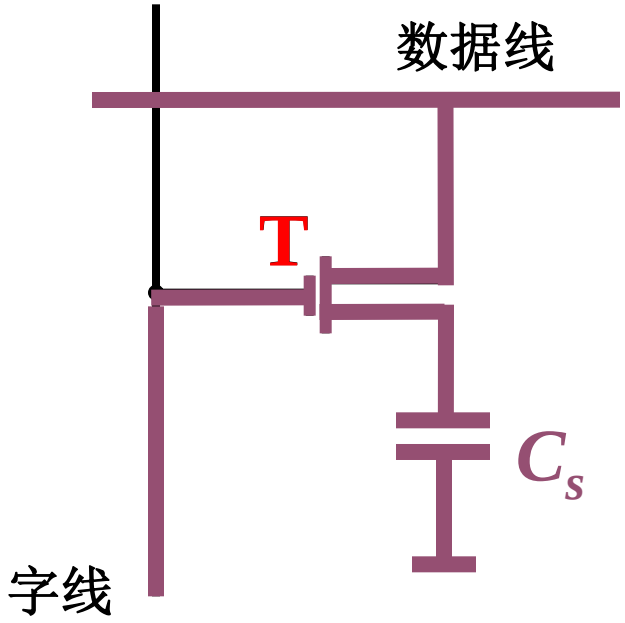
总结：

读出时，数据线有电流为“1”（破坏性读）

2. 动态 RAM (DRAM)

(1) 动态 RAM 单管式

基本单元电路写入

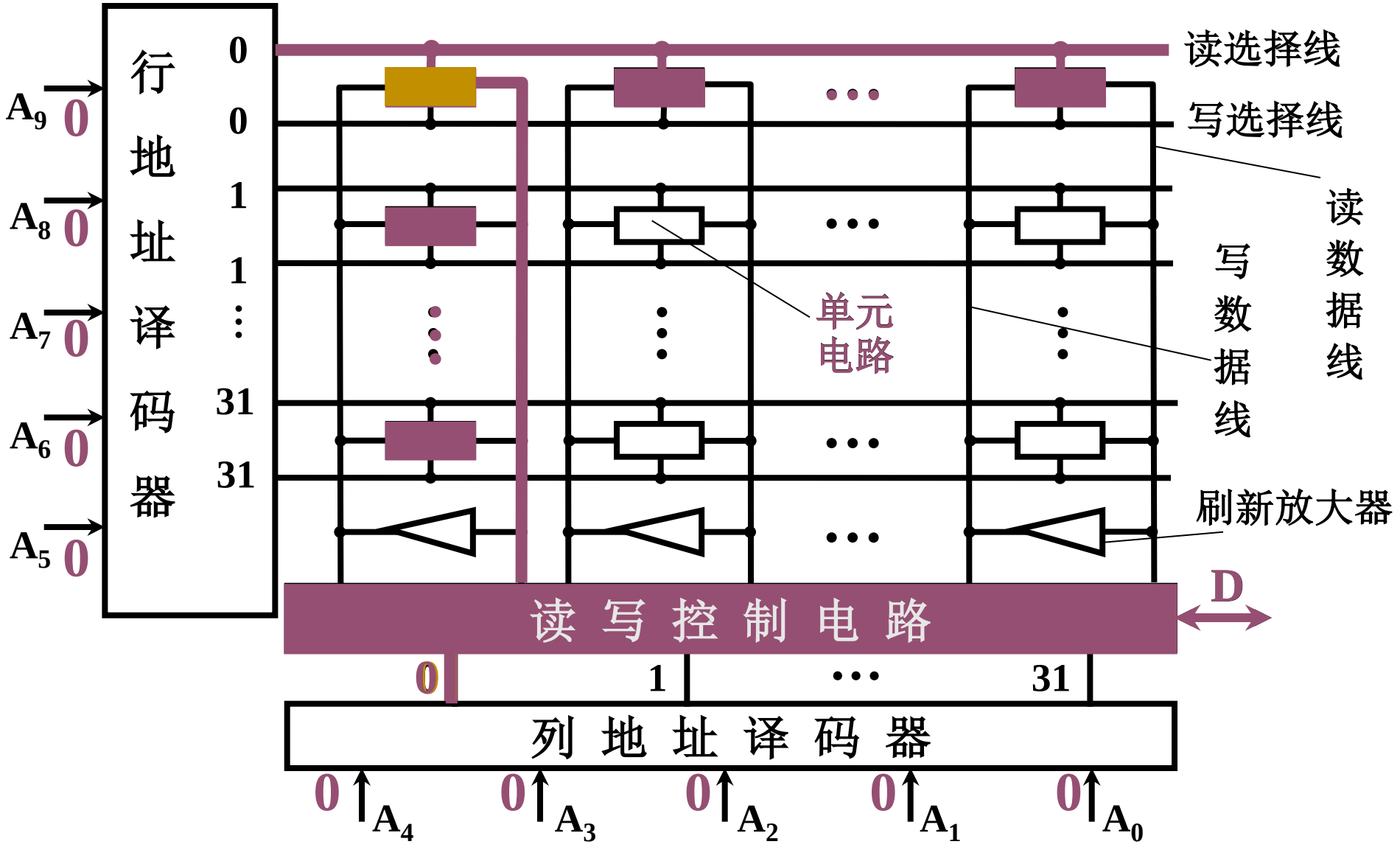


- 首先将需要写入的数据放到写数据线上。
- 字线选中该基本单元电路，将T导通。
- 若数据线上为高电平（数据为“1”），则经过T管对电容C_s充电，存储信息为“1”；若数据线上为低电平（数据为“0”），则电容C_s经T管放电，存储信息为“0。

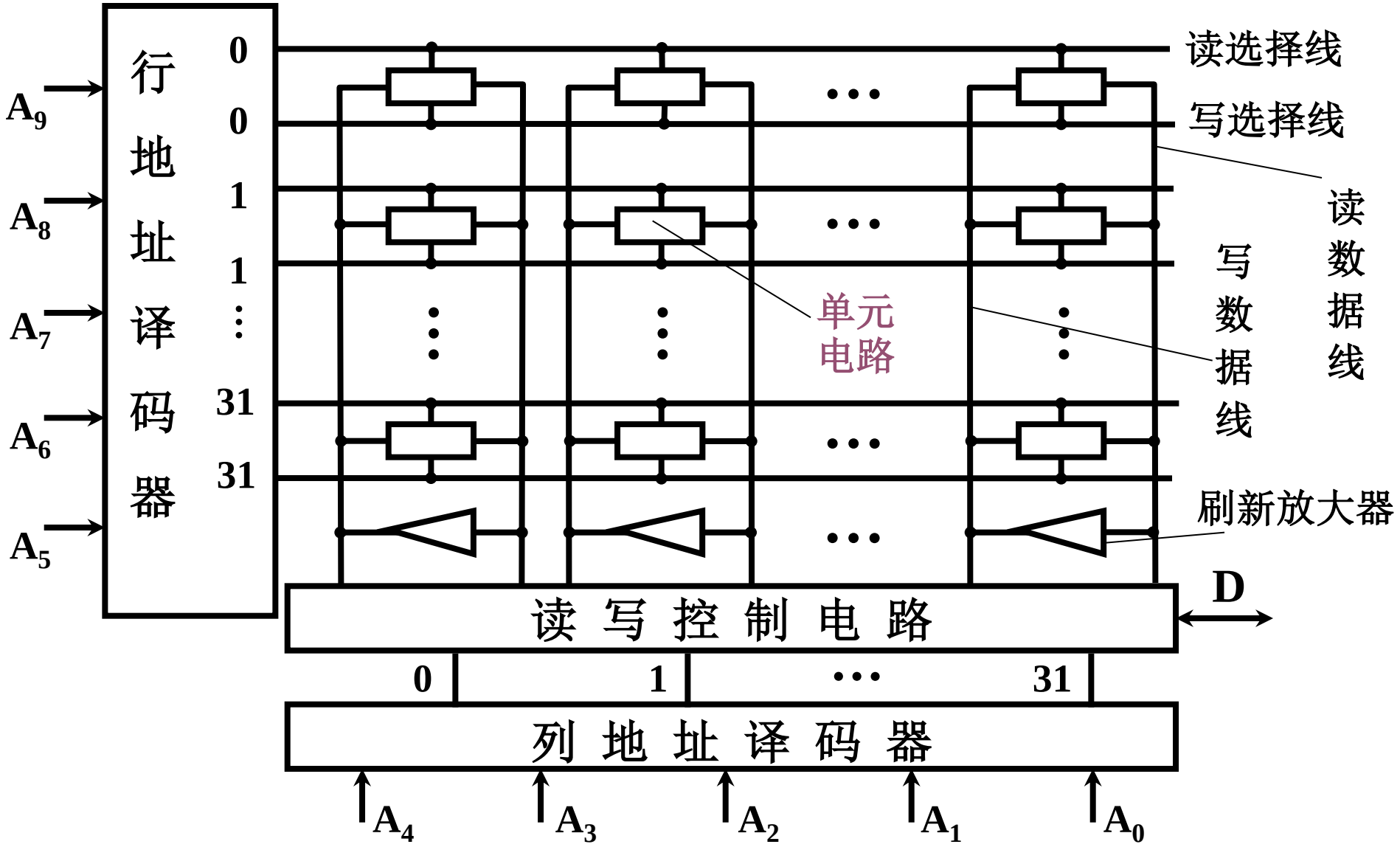
总结：

写入时 C_S 充电 为 “1” 放电 为 “0”

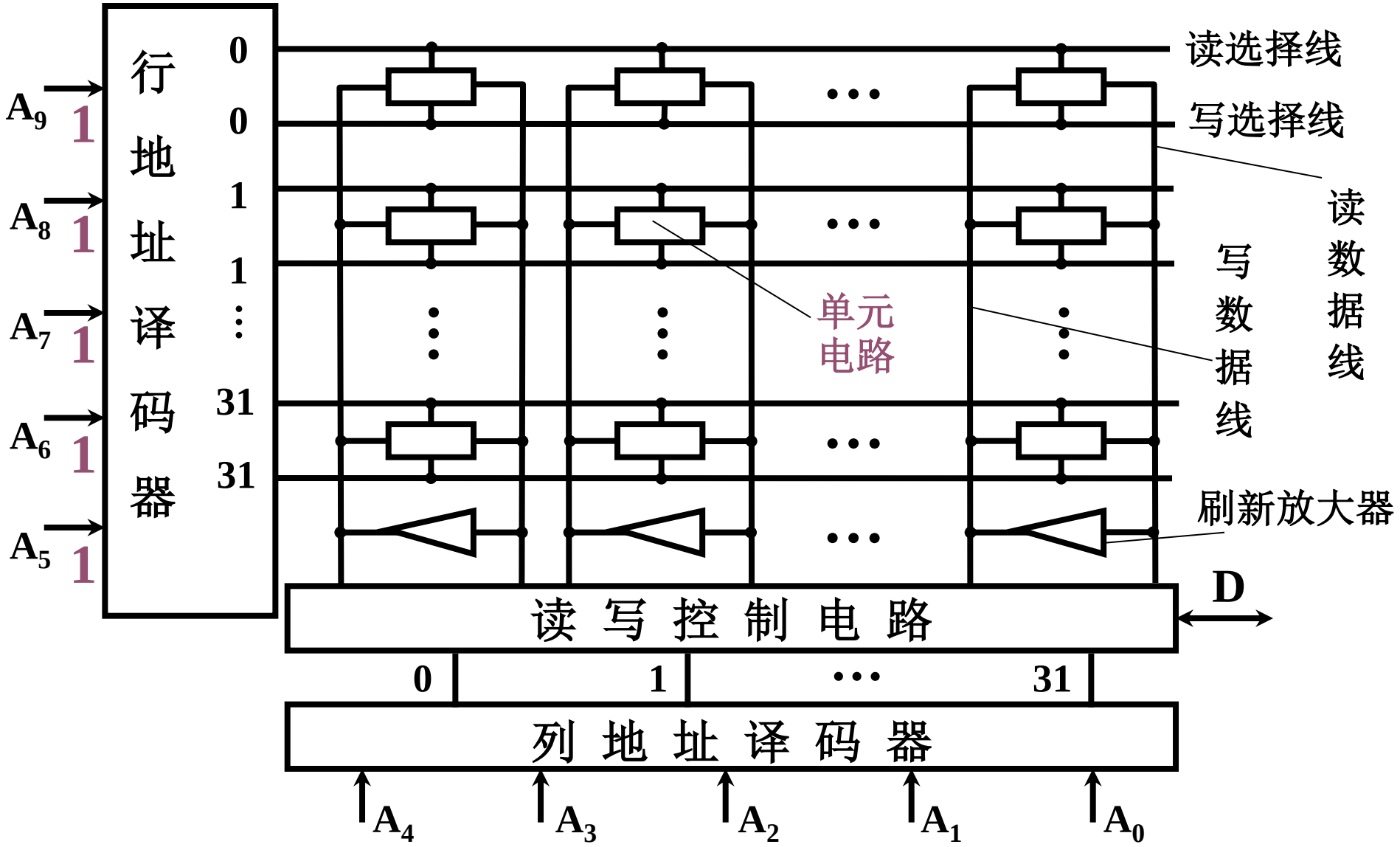
（2）动态RAM举例 ①三管动态RAM芯片 读



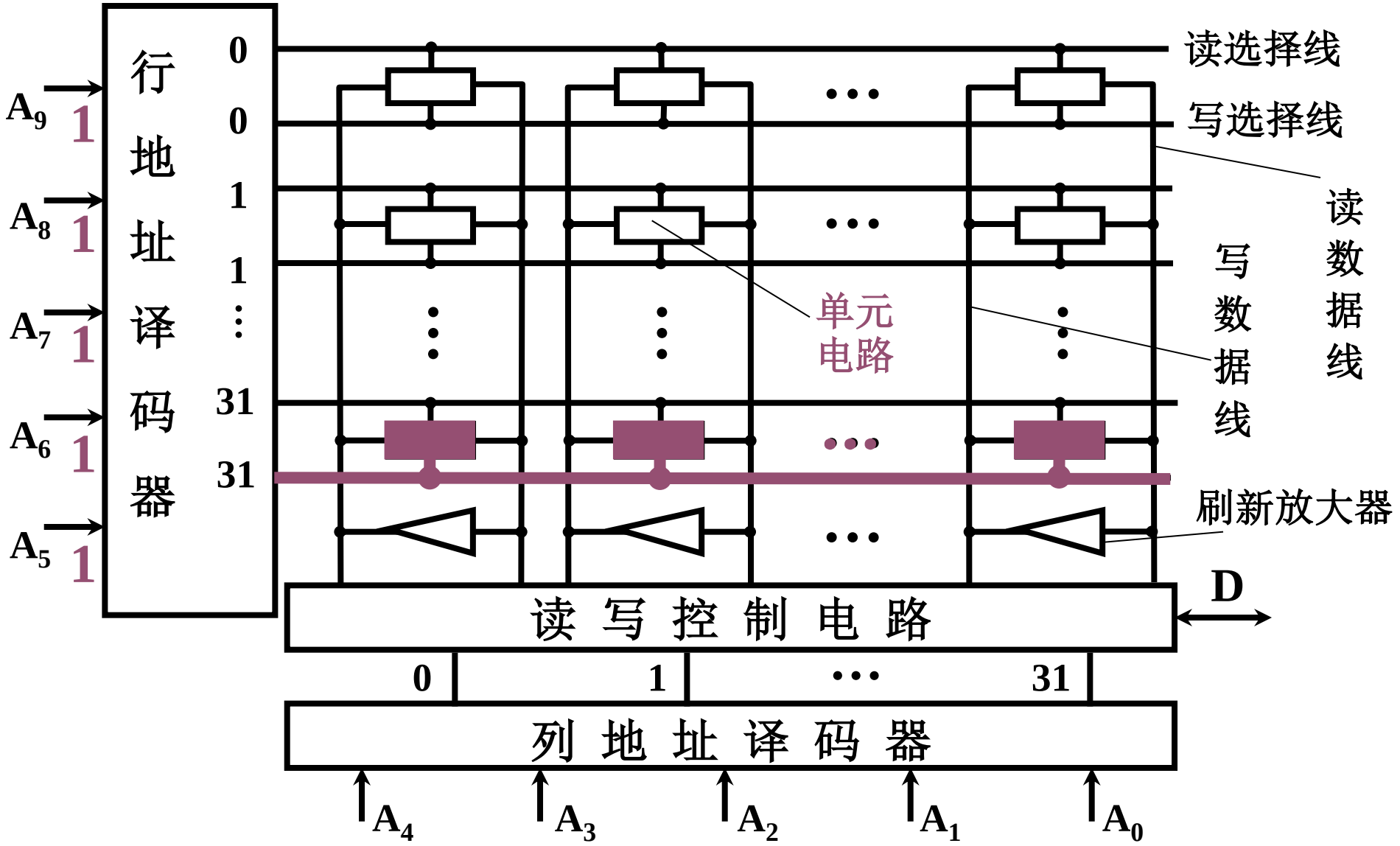
（2）动态RAM举例 ②三管动态RAM芯片 写



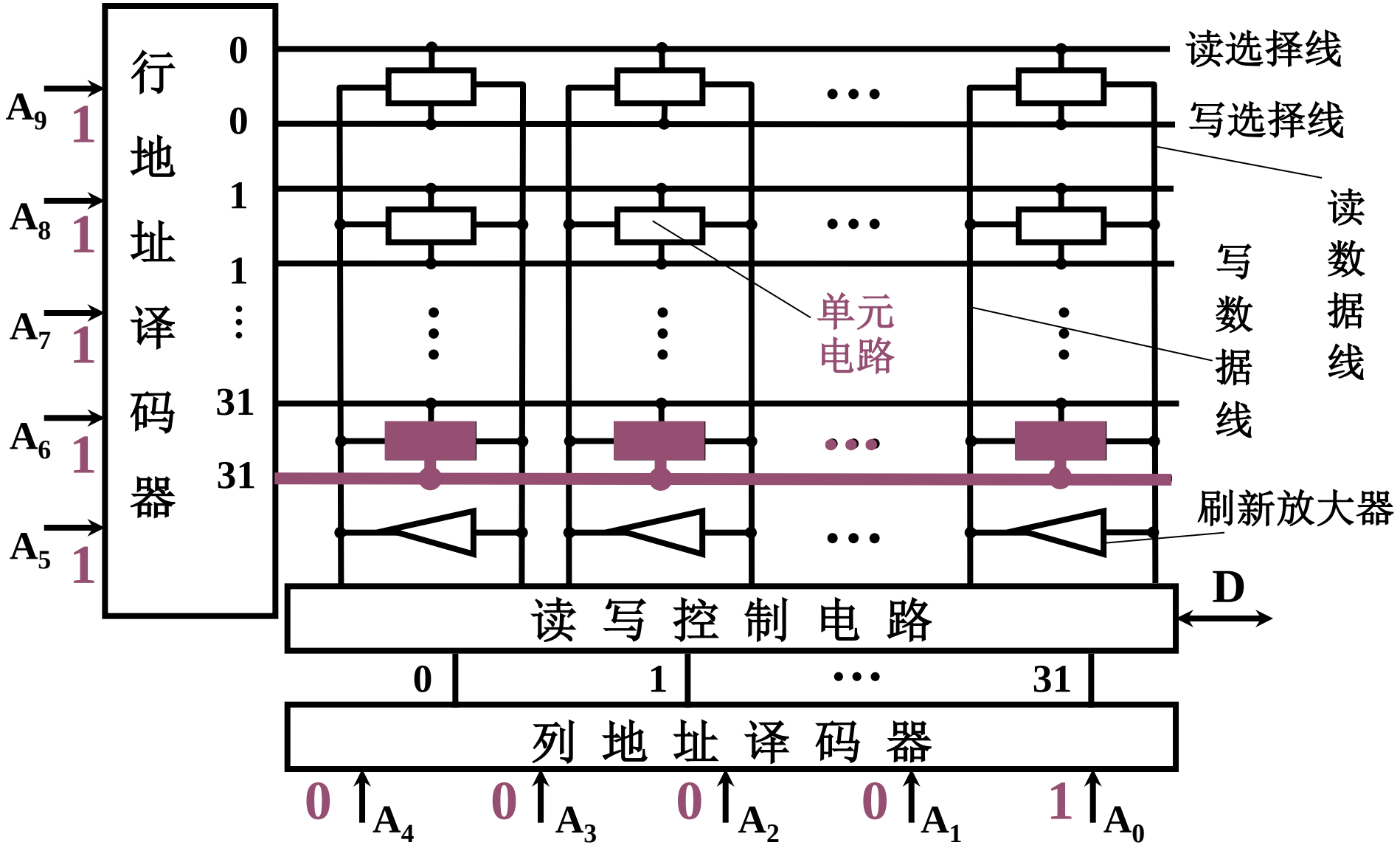
（2）动态RAM举例 ②三管动态RAM芯片 写



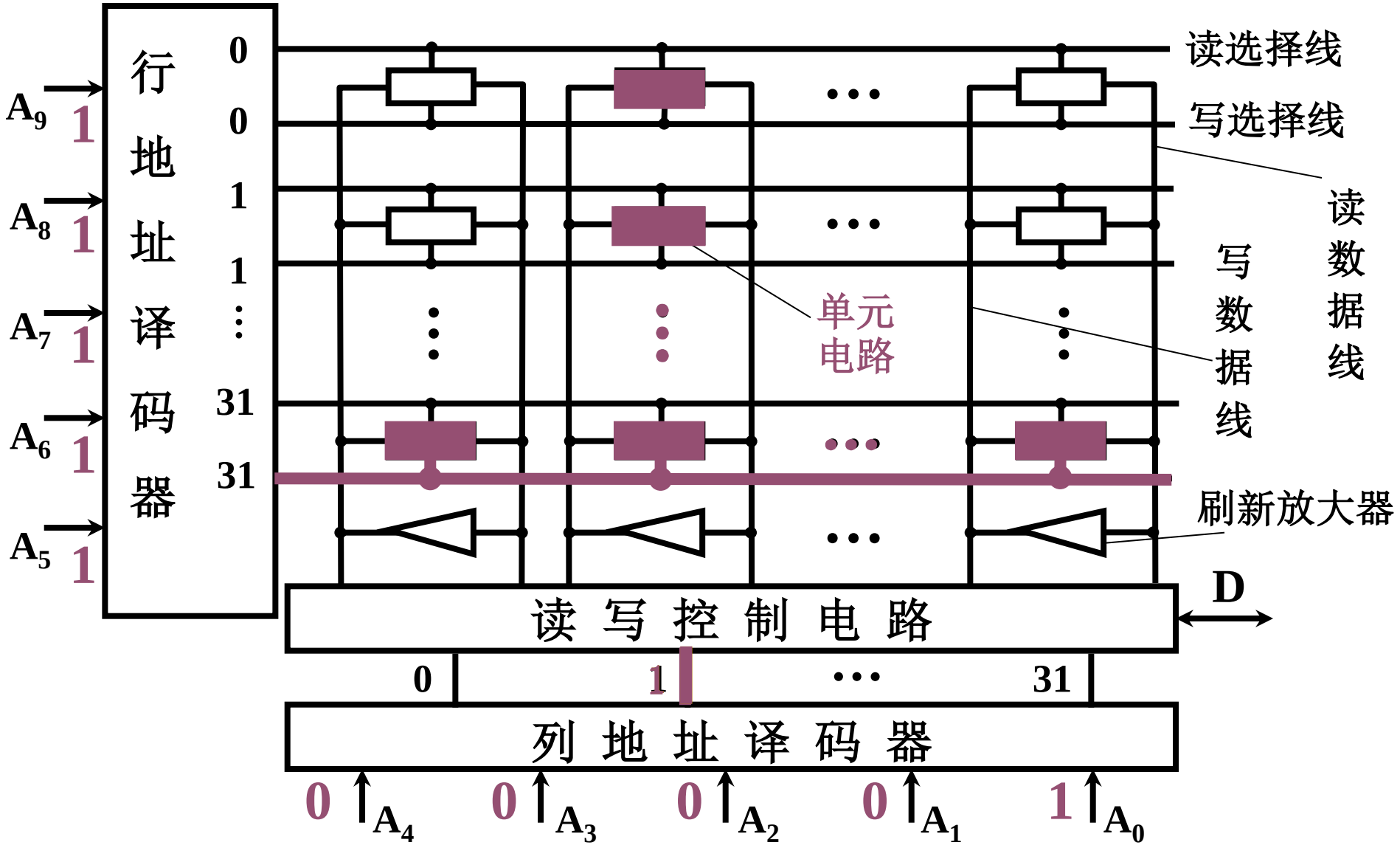
（2）动态RAM举例 ②三管动态RAM芯片 写



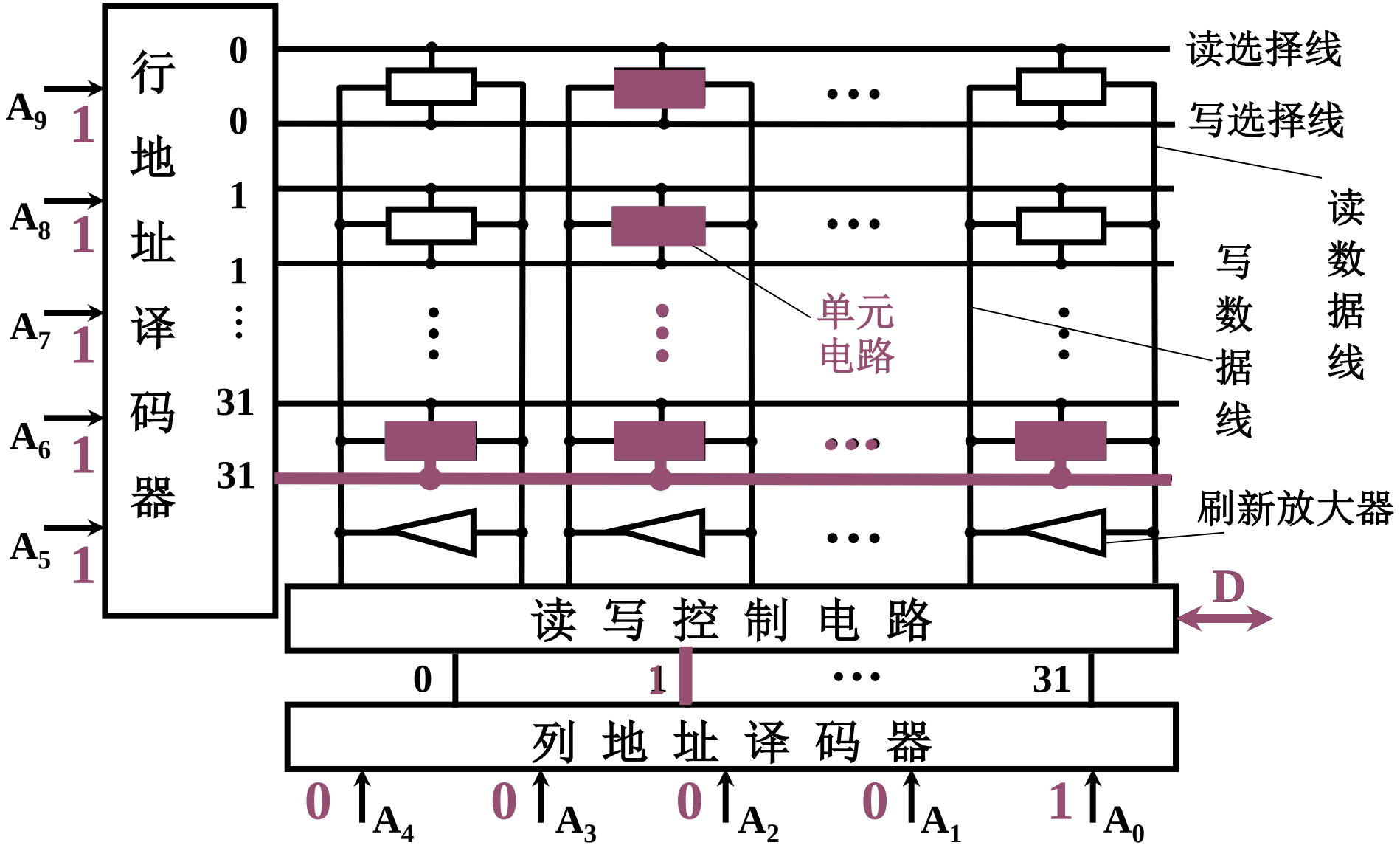
（2）动态RAM举例 ②三管动态RAM芯片 写



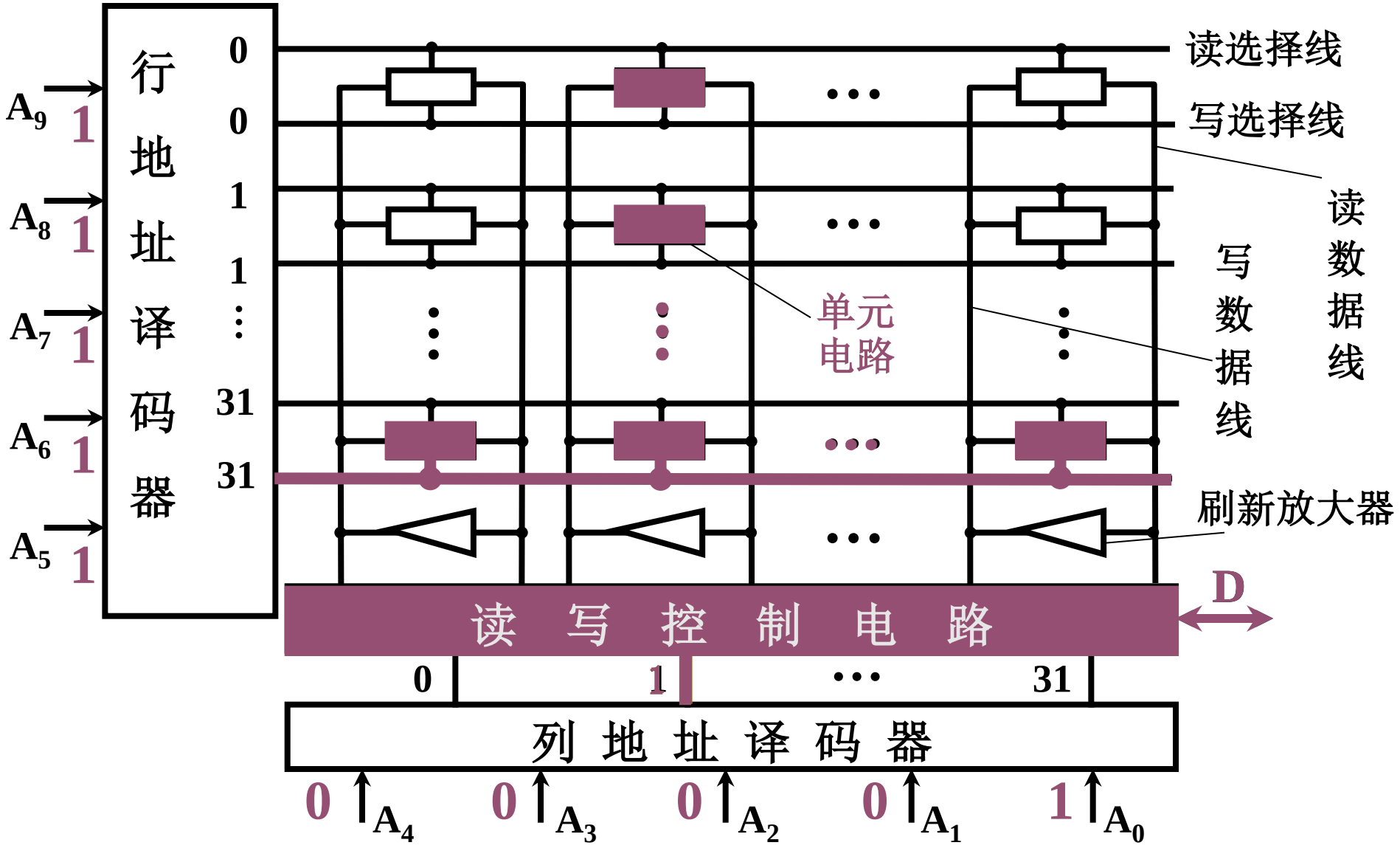
（2）动态RAM举例 ②三管动态RAM芯片 写



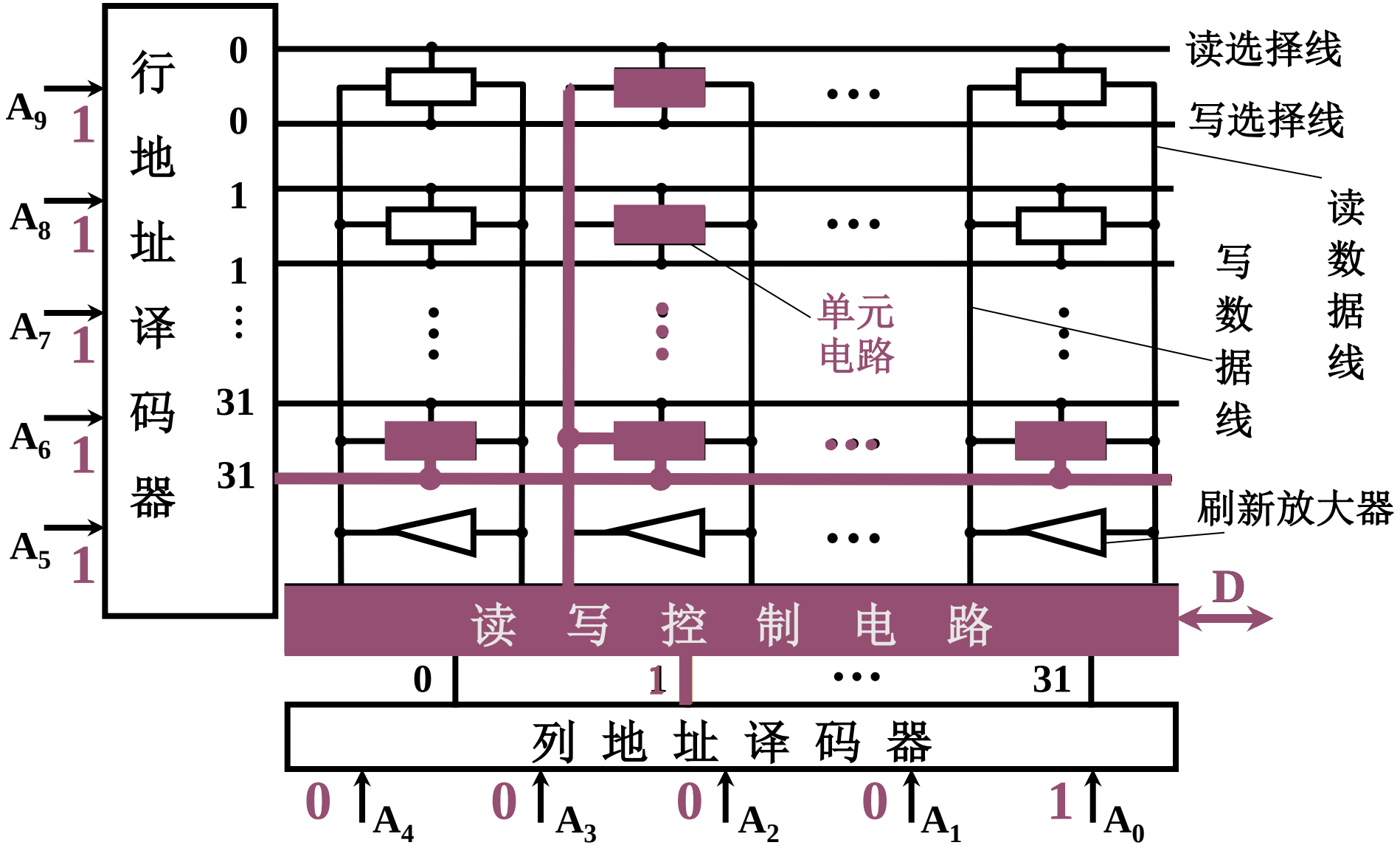
（2）动态RAM举例 ②三管动态RAM芯片 写



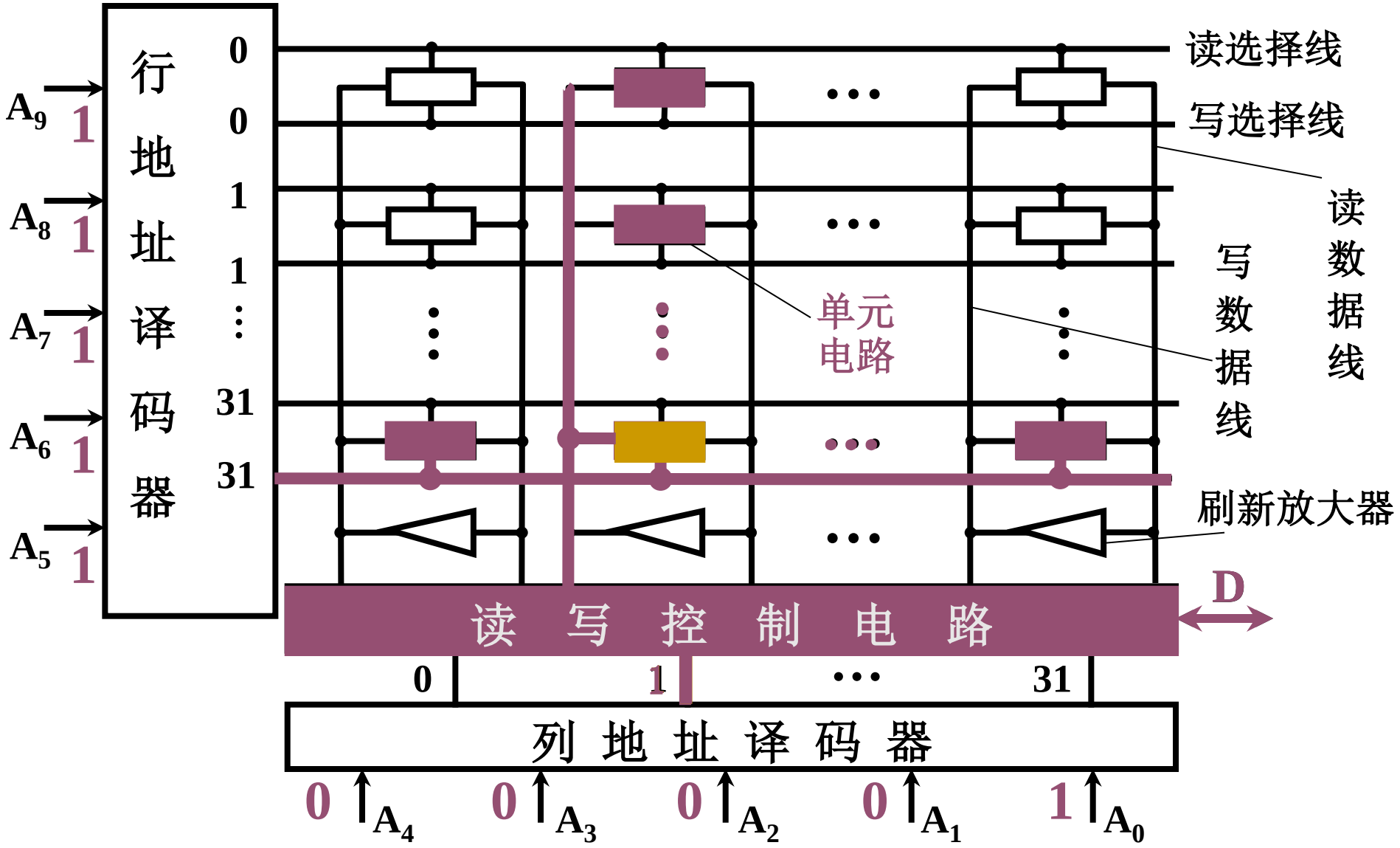
（2）动态RAM举例 ②三管动态RAM芯片 写



（2）动态RAM举例 ②三管动态RAM芯片 写



（2）动态RAM举例 ②三管动态RAM芯片 写



（3）动态RAM时序

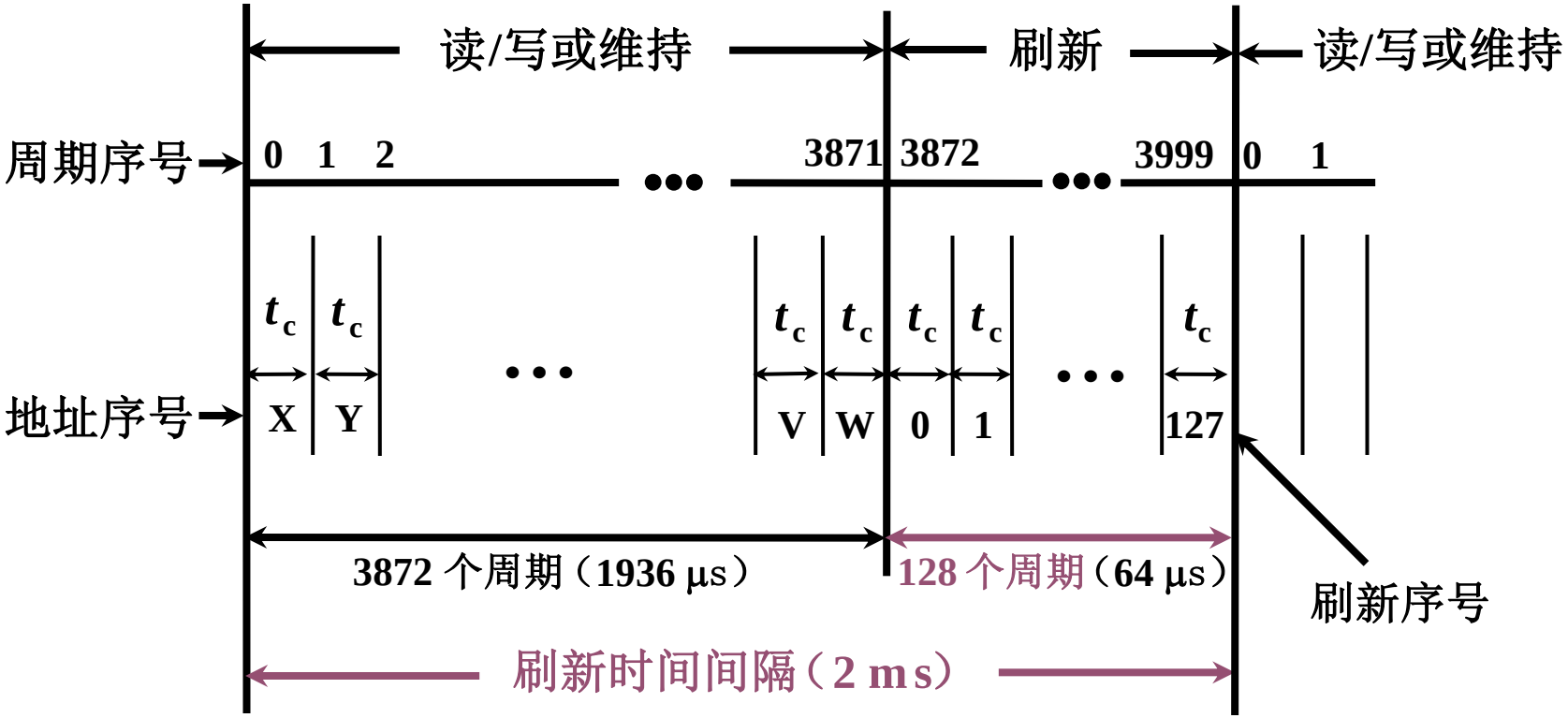
行、列地址分开传送

读时序	写时序
行地址 $\overline{\text{RAS}}$ 有效	行地址 $\overline{\text{RAS}}$ 有效
写允许 $\overline{\text{WE}}$ 有效(高)	写允许 $\overline{\text{WE}}$ 有效(低)
列地址 $\overline{\text{CAS}}$ 有效	数据 D_{IN} 有效
数据 D_{OUT} 有效	列地址 $\overline{\text{CAS}}$ 有效

（4）动态RAM刷新——以128×128矩阵为例

刷新与行地址有关

① 集中刷新 （**刷新周期一般为2ms，存取周期为0.5 μs**）

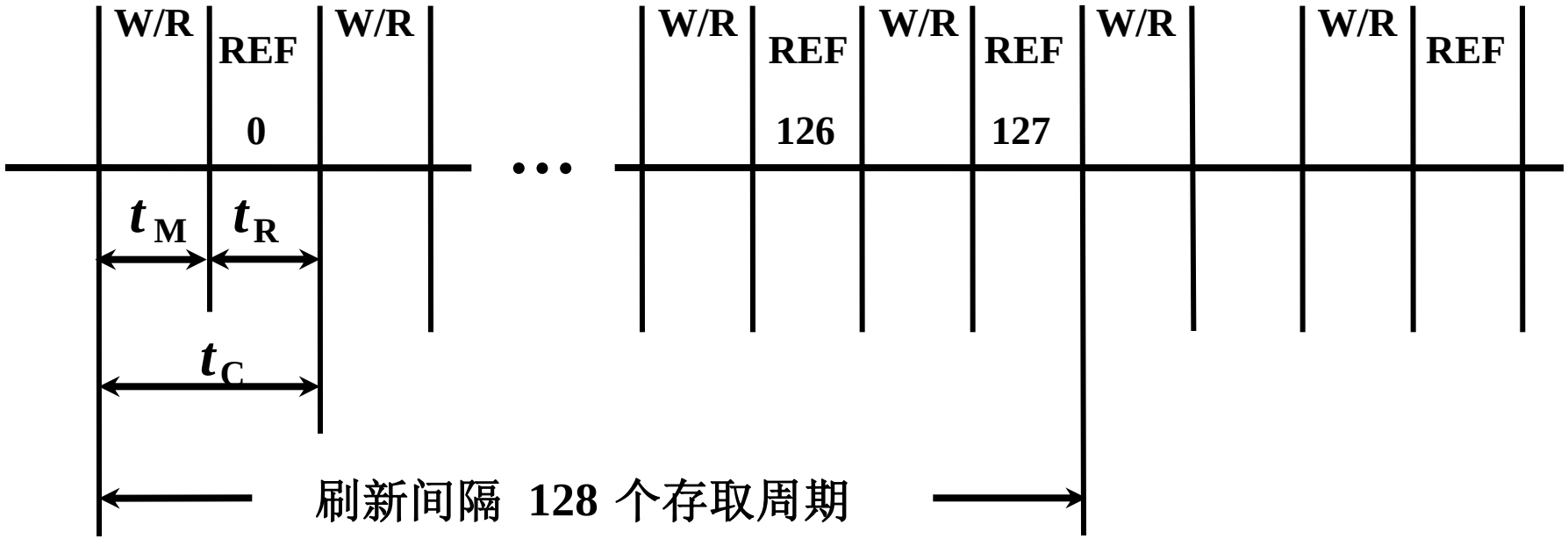


“死区”为 0.5 μs ×128 = 64 μs

“死时间率”为 128/4 000 ×100% = 3.2%

（4）动态RAM刷新——以128×128矩阵为例

② 分散刷新



$t_C = t_M + t_R$　　无“死区”

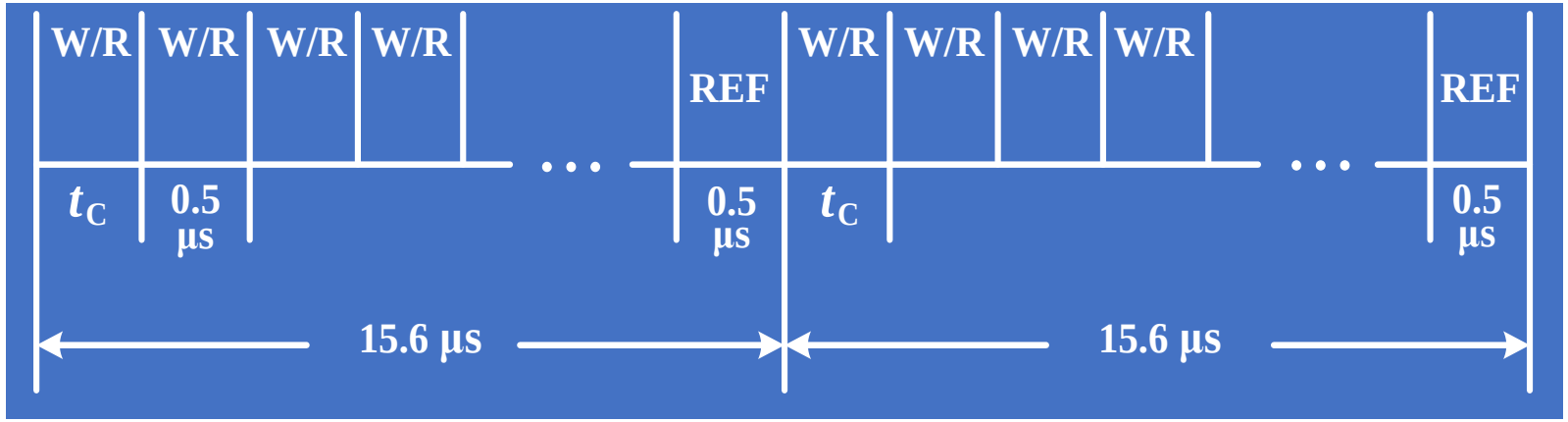
读写　刷新　　(存取周期为 0.5 μs + 0.5 μs)

（4）动态RAM刷新——以128×128矩阵为例

③ 分散刷新与集中刷新相结合（异步刷新）：充分利用了最大刷新间隔为2ms

（存取周期为 0.5 μs ）

若每隔 15.6 μs 刷新一行



每行每隔 2 ms 刷新一次　　“死区”为 0.5 μs

将刷新安排在指令译码阶段，不会出现“死区”

动态RAM和静态RAM的比较

主存

缓存

	DRAM	SRAM
存储原理	电容	触发器
集成度	高	低
芯片引脚	少	多
功耗	小	大
价格	低	高
速度	慢	快
刷新	有	无

6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

六、存储器的校验

七、提高访存速度的措施

6.2主存储器

四、只读存储器（Read Only Memory）

1. 掩模 ROM (MROM)

行列选择线交叉处有 MOS 管为“1”

行列选择线交叉处无 MOS 管为“0”

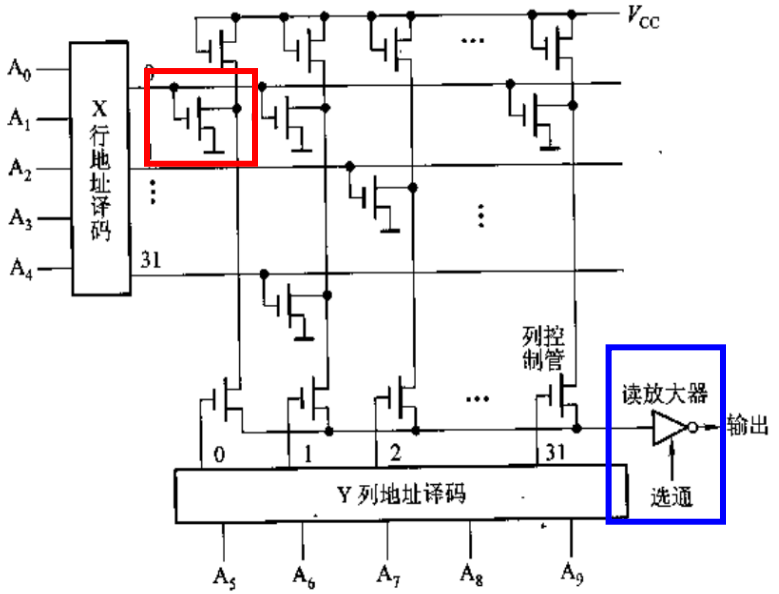
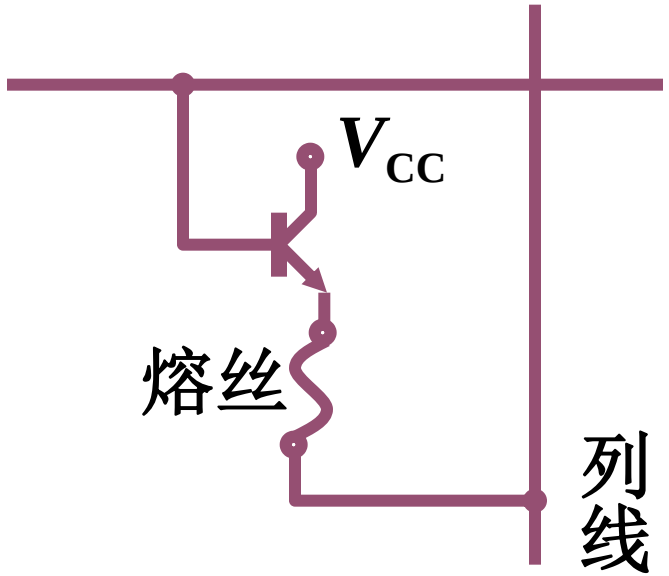


图 4.27　1 K×1 位的 MOS 管掩模 ROM

2. PROM (一次性编程)



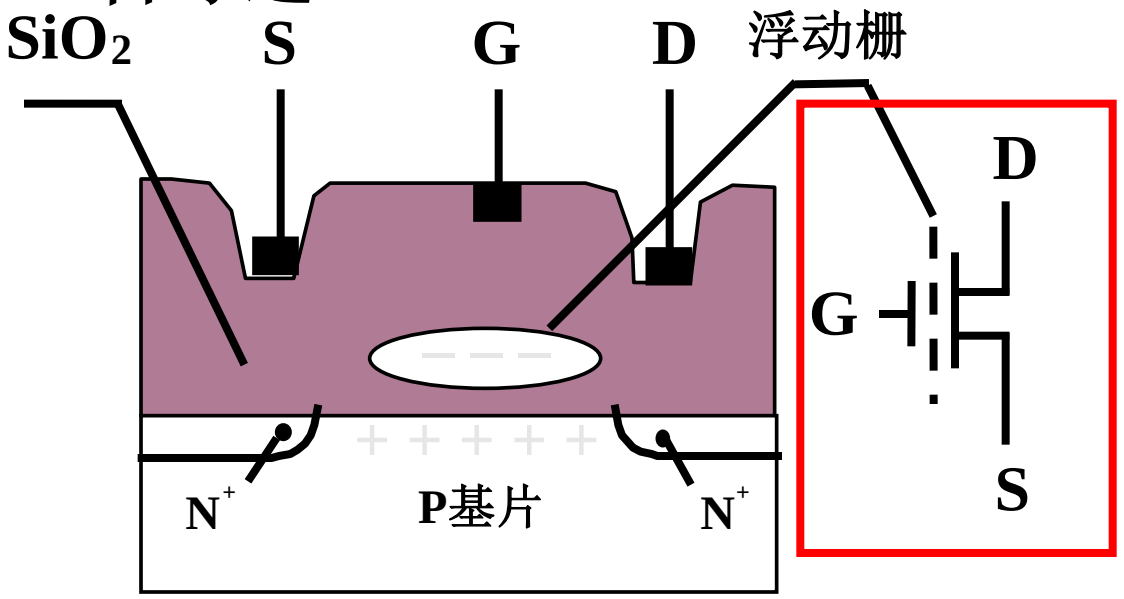
熔丝断　　为“0”

熔丝未断　为“1”

6.2主存储器

3. EPROM (多次性编程)

例子：N型沟道浮动栅 MOS 电路，存储信息的方式是基于三极管是否导通



G 栅极

S 源

D 漏

紫外线可驱散浮动栅

D 端加正电压　　形成浮动栅　　S 与 D 不导通为 “0”

D 端不加正电压　　不形成浮动栅　　S 与 D 导通为 “1”

6.2主存储器

4. EEPROM (多次性编程)

电可擦写　　基于电气方法擦除存储的内容，相比于紫外线照射的方法速度更快。

全部/局部擦写　同时支持局部擦写和全局擦写，可以对个别需要改写的基本单元进行单独擦除。

5. Flash Memory (闪速型存储器)

EPROM　　价格便宜 集成度高

EEPROM　　电可擦出重写

比 **EEPROM**快　具备 **RAM** 功能

6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

六、存储器的校验

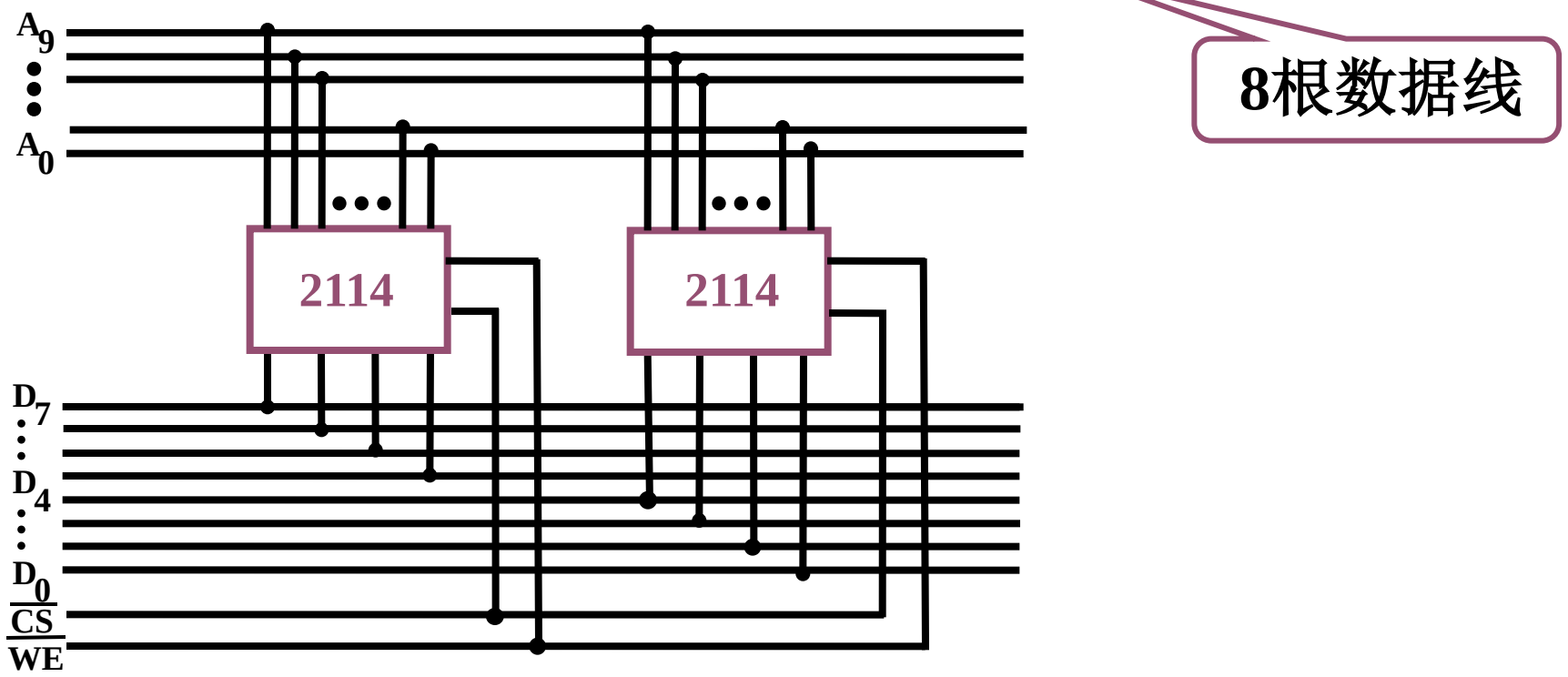
七、提高访存速度的措施

五、存储器与 CPU 的连接

1. 存储器容量的扩展

(1) 位扩展（增加存储字长）

用 2 片 1K×4位 存储芯片组成 1K×8位 的存储器

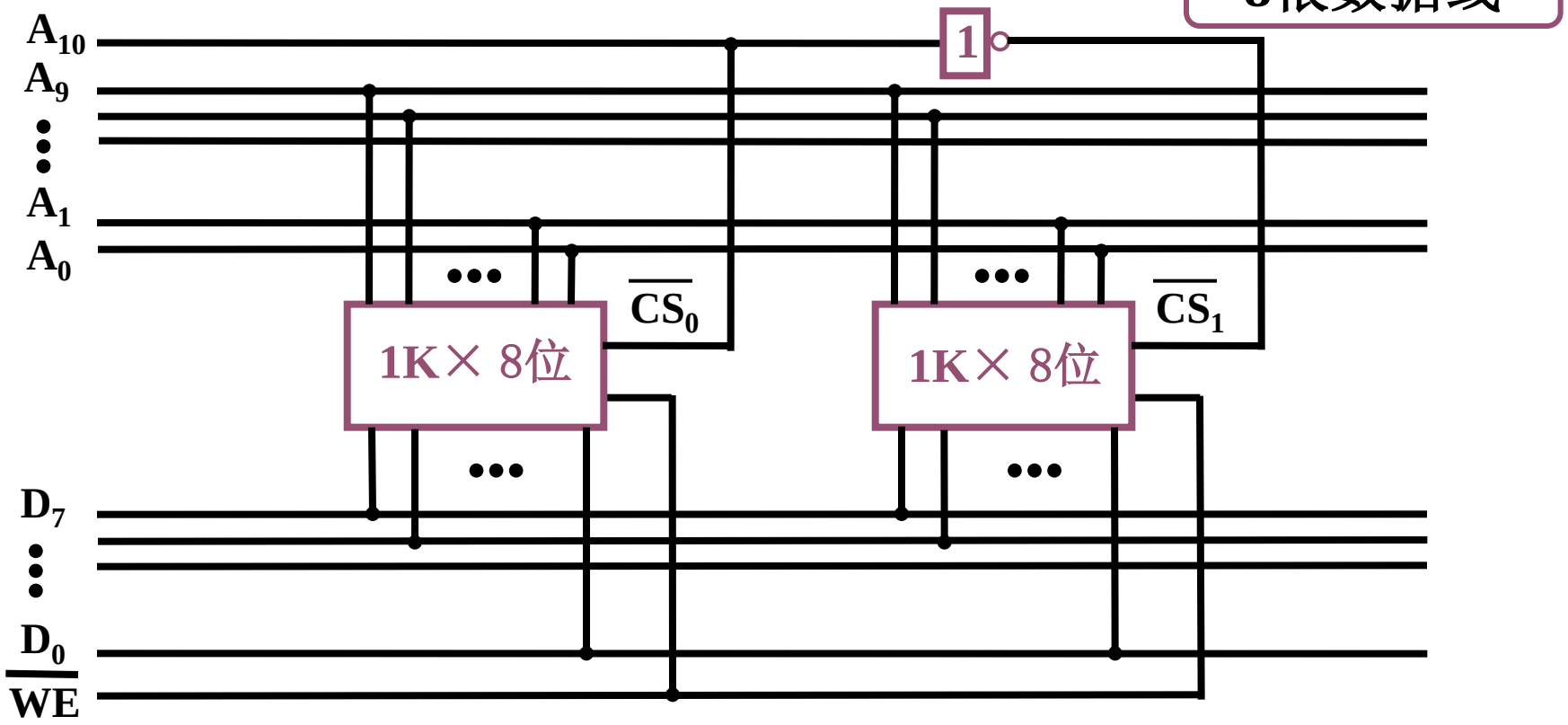


(2) 字扩展（增加存储字的数量）

11根地址线

用 2片 1K×8位 存储芯片组成 2K×8位 的存储器

8根数据线

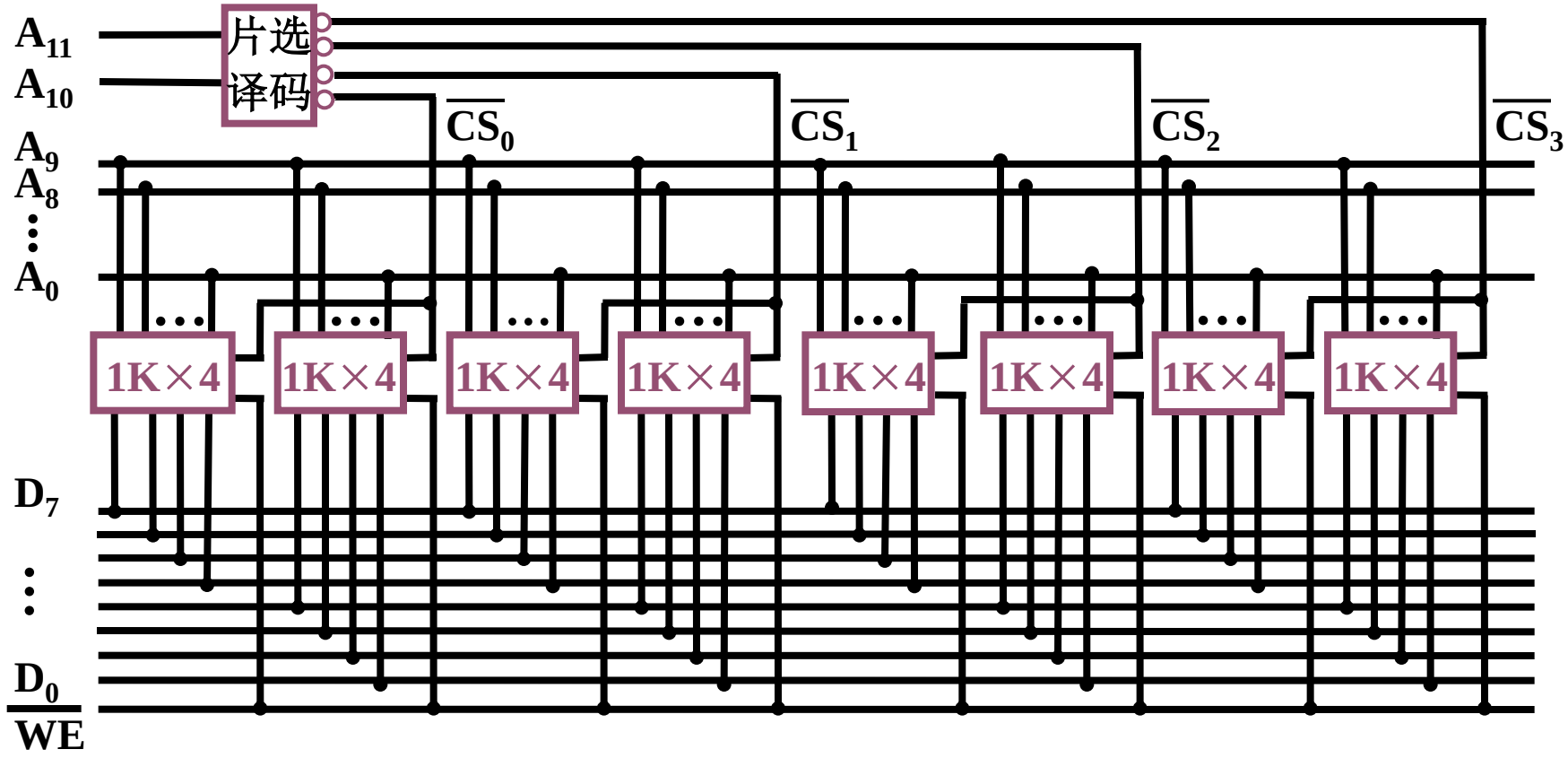


(3) 字、位扩展

用 8片 1K × 4位 存储芯片组成 4K × 8位 的存储器

12根地址线

8根数据线



6.2主存储器

2. 存储器与 CPU 连接的要点

(1) 地址线的连接

- CPU的地址线数往往比一个存储芯片的地址线数要多。通常总是将CPU地址线的低位与存储芯片的地址线相连，CPU地址线的高位一般用于生成片选信号。
- 如：8片1K × 4位的芯片扩展成4K × 8位的存储器，CPU地址线为12位，而存储芯片的地址线仅有10位。

(2) 数据线的连接

- CPU的数据线数与存储芯片的数据线数也不一定相等。需要对存储芯片进行位扩展，使其与CPU的数据线数相等。
- 如：8片1K × 4位的芯片扩展成4K × 8位的存储器，CPU数据线为8位，而存储芯片的数据线仅有4位。

6.2主存储器

2. 存储器与 CPU 连接的要点

(3) 片选线的连接

- 片选线的连接是CPU与存储芯片正确工作的关键。哪一片存储芯片被选中主要取决于存储芯片的**片选控制端$\overline{CS}$**能否接收到来自CPU的片选有效信号（低电平有效）。
- 片选有效信号与CPU访存控制信号$\overline{MREQ}$ 有关。只有当CPU要求访存时，才需要选择存储芯片。
- 片选有效信号还与地址有关。
- 在实际应用中，我们会使用访存控制信号$\overline{MREQ}$ 和高位地址共同产生片选有效信号。

6.2主存储器

2. 存储器与 CPU 连接的要点

(4) 读/写命令线的连接

- CPU的读/写命令线（ $\overline{WE}$ ）一般可以直接与存储芯片的读/写控制端相连，一般为高电平读，低电平写。

(5) 合理选择存储芯片

- 合理选择存储芯片主要包括选择芯片类型（RAM或者ROM），以及芯片的数量。
- 通常选用ROM存放系统程序、标准子程序和常数。
- 选用RAM存放用户程序。

2. 存储器与 CPU 连接的要点

(6)其他：时序、负载等

- CPU时序和存储器的时序要能够相互配合，才能够正确读出或写入数据。
- 负载考虑的是CPU最多可以带动多少片存储芯片。

例6.1

设 CPU 有 16 根地址线，8 根数据线，

$\overline{\text{MREQ}}$ 访存控制信号（低电平有效），

$\overline{\text{WR}}$ 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：

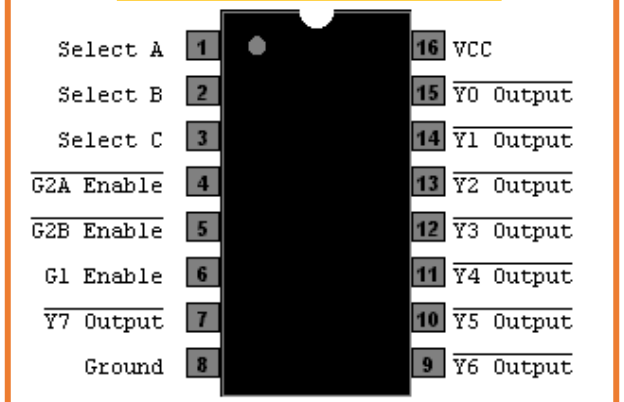
6000H～67FFH 为系统程序区；

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。

74LS138 译码器



CBA	输出(Y7-Y0)
000	00000001
001	00000010
010	00000100
011	00001000
100	00010000
101	00100000
110	01000000
111	10000000

6000H～67FFH :系统程序区

6800H～6BFFH :用户程序区

例6.1 解析

(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	1片 2K×8位
⋮												
0	1	1	0	0	1	1	1	1	1	1	1	ROM 2K×8位
0	1	1	0	1	0	0	0	0	0	0	0	1K×8位
⋮												
0	1	1	0	1	1	1	1	1	1	1	1	RAM

2片1K×4位

(2) 确定芯片的数量及类型

可选的：RAM： 1K×4位；4K×8位；8K×8 位

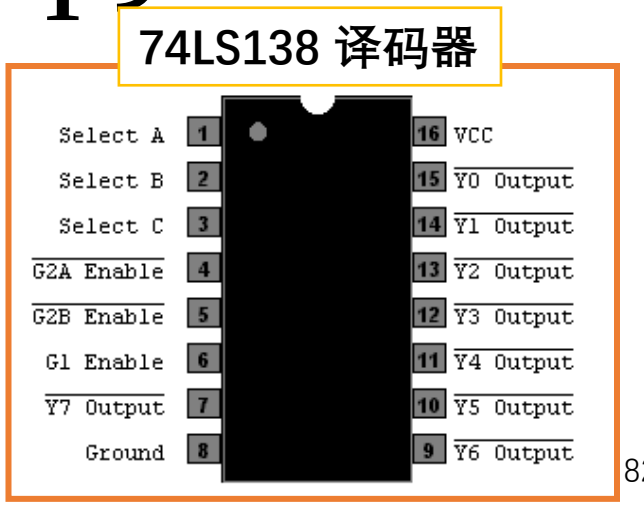
ROM： 2K×8位；4K×8位；8K×8 位

(3) 分配地址线

A_{15}	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	1片 ROM 2K×8位
⋮											
0	1	1	0	0	1	1	1	1	1	1	
0	1	1	0	1	0	0	0	0	0	0	2片RAM 1K×4位
⋮											
0	1	1	0	1	0	1	1	1	1	1	
		C	B	A							

A_{10}~A_0 接 2K×8位 ROM 的地址线

A_9 ~A_0 接 1K×4位 RAM 的地址线



(4) 确定片选信号

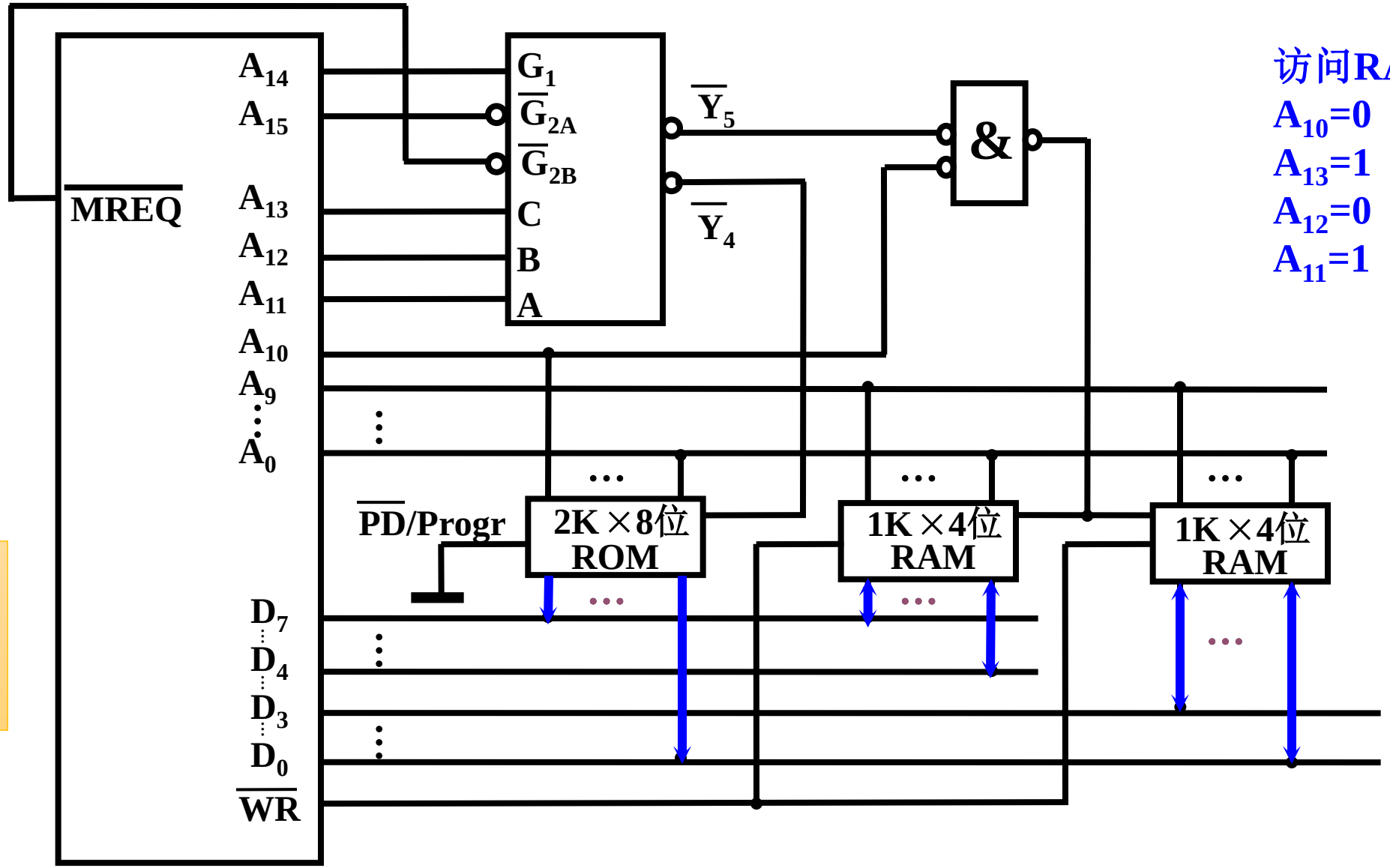
例 6.1 CPU 与存储器的连接图

A_{15}=0

A_{14}=1

访问ROM时
A_{13}=1（C）
A_{12}=0（B）
A_{11}=0（A）

ROM只能读不能写，所以ROM的编程端PD/progr接地



访问RAM时，
A_{10}=0
A_{13}=1（C）
A_{12}=0（B）
A_{11}=1（A）

例6.2 假设 CPU 有 16 根地址线，8 根数据线，

$\overline{\text{MREQ}}$ 访存控制信号（低电平有效），

$\overline{\text{WR}}$ 读/写控制信号（高电平为读，低电平为写）

要求最小 4K为系统程序区，相邻 8K为用户程序区。

(1) 写出对应的二进制地址码

(2) 确定芯片的数量及类型

(3) 分配地址线

RAM：1K×4位；4K×8位；8K×8 位

ROM：2K×8位；4K×8位；8K×8 位

(4) 确定片选信号

(5) 确定片选逻辑

例6.2 解析

4K为系统程序区

相邻8K为用户程序区

- ✓ 写出对应的二进制地址码
- ✓ 确定芯片数量和类型

$A_{15}A_{14}A_{13}$ $A_{11}A_{10}$ … A_7 … A_4 A_3 … A_0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

⋮

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1

ROM 4K×8位

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

⋮

0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1

第一个相邻4K

4K×8位 RAM 1片

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

⋮

0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1

第二个相邻4K

4K×8位 RAM 1片

例6.2 解析

4K为系统

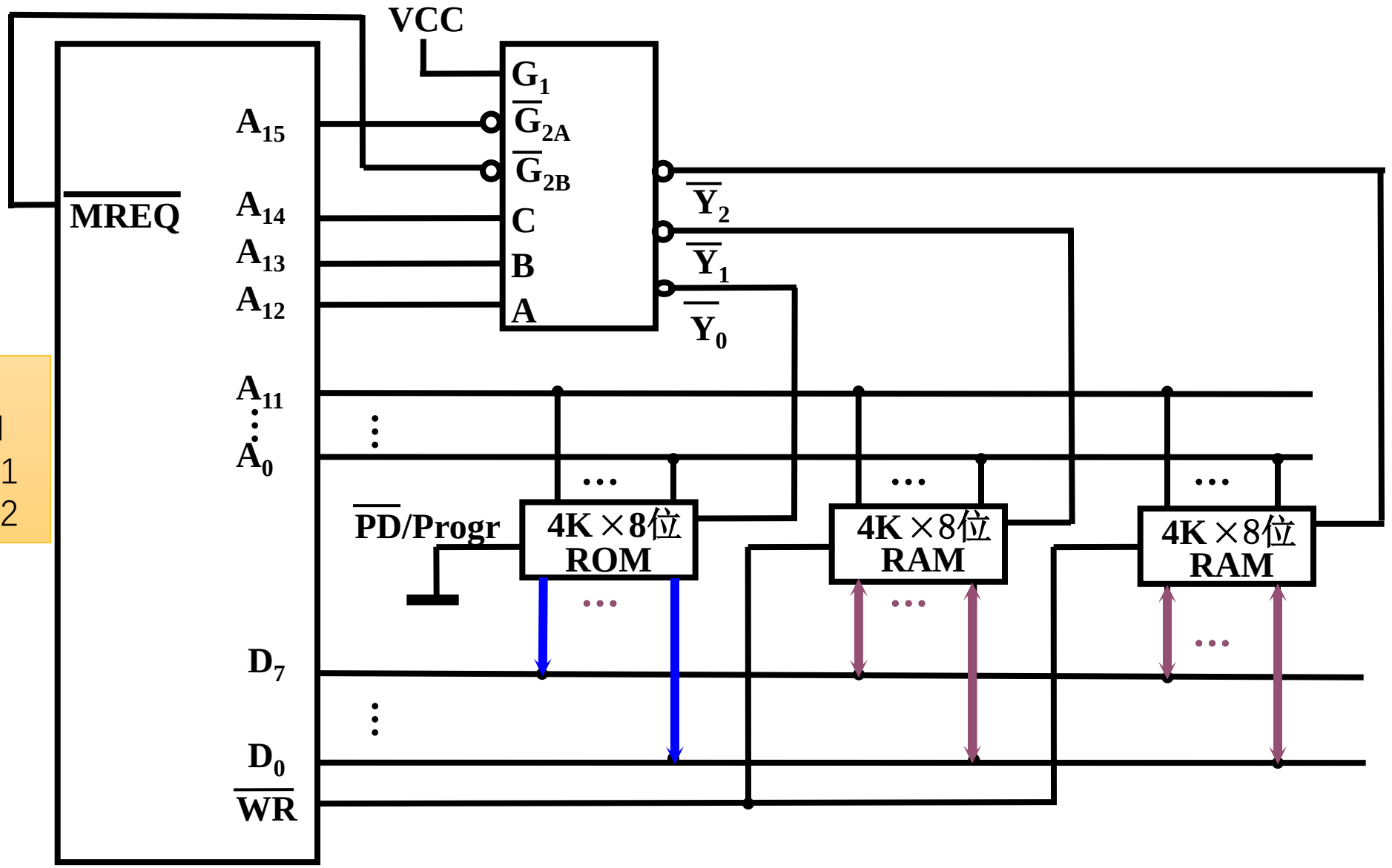
相邻8K为用户程序区

- ✓ 分配地址线
- ✓ 确定片选信号

A_{11}~ A_0 接 ROM 和 RAM 的地址线

A_{15}	A_{14}	A_{13}	A_{11}	A_{10}	…	A_7	…	A_4	A_3	…	A_0	
0	0	0	0	0	0	0	0	0	0	0	0	ROM 4K×8位
⋮												
0	0	0	0	1	1	1	1	1	1	1	1	
0	0	0	1	0	0	0	0	0	0	0	0	4K×8位 RAM 1片
第一个相邻4K ⋮												
0	0	0	1	1	1	1	1	1	1	1	1	
0	0	1	0	0	0	0	0	0	0	0	0	4K×8位 RAM 1片
第二个相邻4K ⋮												
0	0	1	0	1	1	1	1	1	1	1	1	
		C	B	A								

例 6.2 CPU 与存储器的连接图



C、B、A
000时访问ROM
001时访问RAM1
010是访问RAM2

6.2 主存储器

一、概述

二、半导体存储芯片简介

三、随机存取存储器（RAM）

四、只读存储器（ROM）

五、存储器与CPU的连接

六、存储器的校验

七、提高访存速度的措施

六、存储器的校验

- 为什么要对存储器的信息进行校验？
- 为了校验信息，如何进行编码？
- 纠错和检错能力与什么因素有关？
- 校验出信息出错后是如何进行纠错？

以RAM为例，信息保存在触发器（静态RAM）或电容（动态RAM）中，如果RAM所处的电磁环境比较复杂，则可能造成电容错误地进行充/放电或者造成触发器的反转，存放在RAM中的信息就可能出错，因此需要对存储器的信息进行校验。

六、存储器的校验

1、{000, 001, 010, 011, 100, 101, 110, 111}　检0位错、纠0位错

2、{000, 011, 101, 110}

100　　检1位错，纠0位错

3、{000, 111}　　检1位错，纠1位错

100　110

4、{0000, 1111}　检2位错，纠1位错

1000　1100

5、{00000, 11111}　检2位错，纠2位错

11000　11100

编码的检测能力和纠错能力和什么有关呢？

任意两组合法代码之间二进制位的最小差异数

六、存储器的校验

1 . 编码的最小距离

任意两组合法代码之间 二进制位数 的 最少差异

编码的纠错 、检错能力与编码的最小距离有关

$$L - 1 = D + C \ (\ D \geq C\ )$$

L —— 编码的最小距离

D —— 检测错误的位数

C —— 纠正错误的位数

$L = 3$

具有 一位 纠错能力

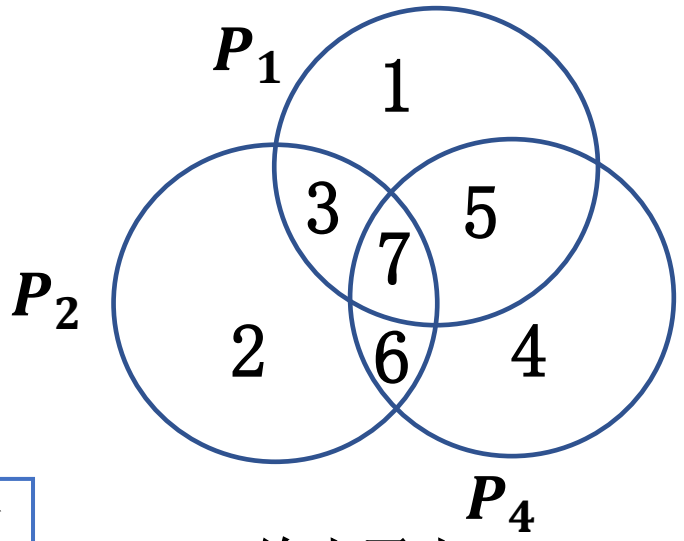
汉明码——一种具有一位纠错能力的编码

6.2主存储器

- 组成汉明码的三要素

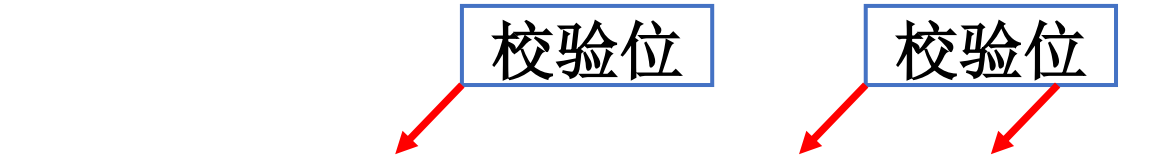
  汉明码采用奇偶校验

  汉明码采用分组校验



校验位应放哪里呢？

1, 2, 4, 8, …
位置放校验码



给出了出错的位置

$P_4P_2P_1$	无差错
0 0 0	
0 0 1	1
1 0 1	5
1 1 0	6
1 1 1	7

汉明码的分组是一种非划分方式

1	2	3	4	5	6	7

如何分组的呢？

第1组 XXXX1
第2组 XXX1X
第3组 XX1XX
第4组 X1XXX
第5组 1XXXX

分成3组，每组有1位校验位，共包括4位数据位

6.2主存储器

- 组成汉明码的三要素

汉明码的组成需增添 ？ 位检测位

$$2^k \geq n + k + 1$$

检测位的位置 ？

$$2^i \ (i = 0，\ 1，\ 2，\ 3，\ \ldots)$$

检测位的取值 ？

与该位所在的检测“小组”中承担的奇偶校验任务有关

各检测位 C_i 所承担的检测小组

- C_1 检测 g_1 小组包含位 1,3,5,7,9,11,... (XXXX**1**)
- C_2 检测 g_2 小组包含位 2,3,6,7,10,11,... (XXX**1**X)
- C_4 检测 g_3 小组包含位 4,5,6,7,12,13,... (XX**1**XX)
- C_8 检测 g_4 小组包含位 8,9,10,11,12,13,14,15,24,... (X**1**XXX)

- g_i 小组独占第 2^{i-1} 位
- g_i 和 g_j 小组共同占第 $2^{i-1}+2^{j-1}$ 位
- g_i、g_j 和 g_l 小组共同占第 $2^{i-1}+2^{j-1}+2^{l-1}$ 位

例6.3 求 0101 按“偶校验”配置的汉明码

解：∵ $n = 4$

根据 $2^k \geq n + k + 1$

得 $k = 3$

- C_1 检测 g_1 小组包含位 1,3,5,7,9,11,... (XXXX1)
- C_2 检测 g_2 小组包含位 2,3,6,7,10,11,... (XXX1X)
- C_4 检测 g_3 小组包含位 4,5,6,7,12,13,... (XX1XX)
- C_8 检测 g_4 小组包含位 8,9,10,11,12,13,14,15,24,... (X1XXX)

汉明码排序如下：

二进制序号	1	2	3	4	5	6	7
名称	C_1	C_2	0	C_4	1	0	1
	0	1		0			

∴ 0101 的汉明码为 0100101

主观题 10分

设置

练习1　按配偶原则配置 0011 的汉明码

正常使用主观题需2.0以上版本雨课堂

作答

练习1　按配偶原则配置 0011 的汉明码

解：　∵ $n = 4$　　根据 $2^k \geq n + k + 1$

取 $k = 3$

二进制序号	1	2	3	4	5	6	7
名称	C_1	C_2	0	C_4	0	1	1
	1	0		0			

$C_1 = 3 \oplus 5 \oplus 7 = 1$

$C_2 = 3 \oplus 6 \oplus 7 = 0$

$C_4 = 5 \oplus 6 \oplus 7 = 0$

∴ 0011 的汉明码为 **1000011**

3. 汉明码的纠错过程

形成新的检测位 P_i，其位数与增添的检测位有关，

如增添 3 位（$k = 3$），新的检测位为 $P_4 P_2 P_1$。

以 $k = 3$ 为例，P_i 的取值为

$$P_1 = \overset{C_1}{1} \oplus 3 \oplus 5 \oplus 7$$

$$P_2 = \overset{C_2}{2} \oplus 3 \oplus 6 \oplus 7$$

$$P_4 = \overset{C_4}{4} \oplus 5 \oplus 6 \oplus 7$$

对于按“偶校验”配置的汉明码

不出错时 $P_1 = 0$，$P_2 = 0$，$P_4 = 0$

例6.4 已知接收到的汉明码为 **0100111**

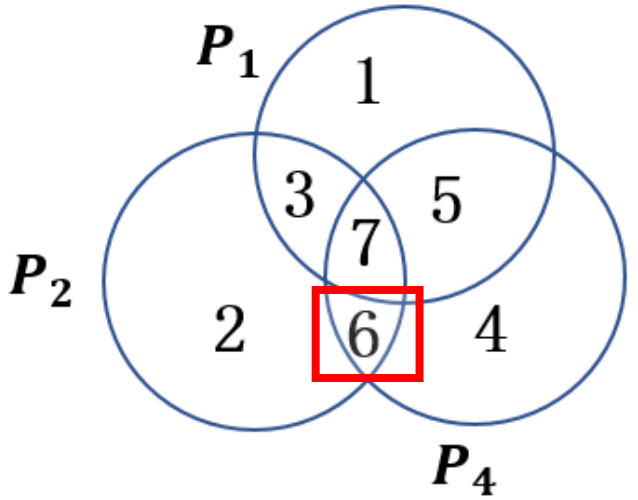
（按配偶原则配置）试问要求传送的信息是什么？

解：纠错过程如下

P_1= 位1⊕位3⊕位5⊕位7 = 0 无错

P_2= 位2⊕位3⊕位6⊕位7 = 1 有错

P_4= 位4⊕位5⊕位6⊕位7 = 1 有错



∴ $P_4P_2P_1 = 110$

第 6 位出错，可纠正为 0100101，

故要求传送的信息为 0101。

主观题 10分

设置

练习2 写出按偶校验配置的汉明码

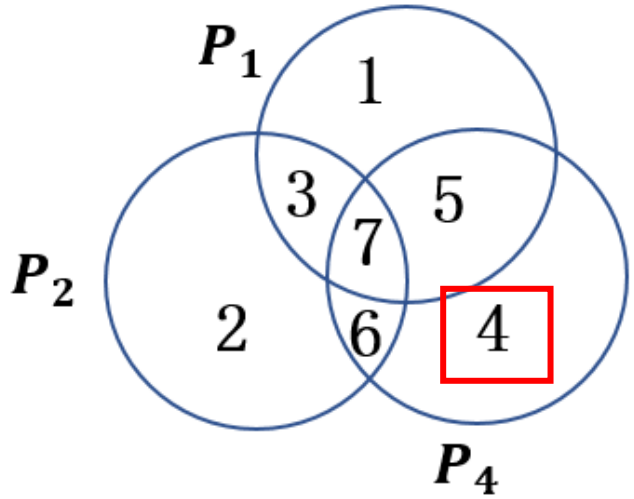
0101101 的纠错过程

正常使用主观题需2.0以上版本雨课堂

作答

练习2 写出按偶校验配置的汉明码

0101101 的纠错过程



解：纠错过程如下

$$P_4 = 位4 \oplus 位5 \oplus 位6 \oplus 位7 = 1$$

$$P_2 = 位2 \oplus 位3 \oplus 位6 \oplus 位7 = 0$$

$$P_1 = 位1 \oplus 位3 \oplus 位5 \oplus 位7 = 0$$

$\therefore P_4P_2P_1 = 100$ 第 4 位错，可不纠

真实数据为0101

主观题 10分

设置

练习3 按配奇校验配置 0011 的汉明码

正常使用主观题需2.0以上版本雨课堂

作答

练习3 按配奇校验配置 0011 的汉明码

解： ∵ $n = 4$ 根据 $2^k \geq n + k + 1$

取 $k = 3$

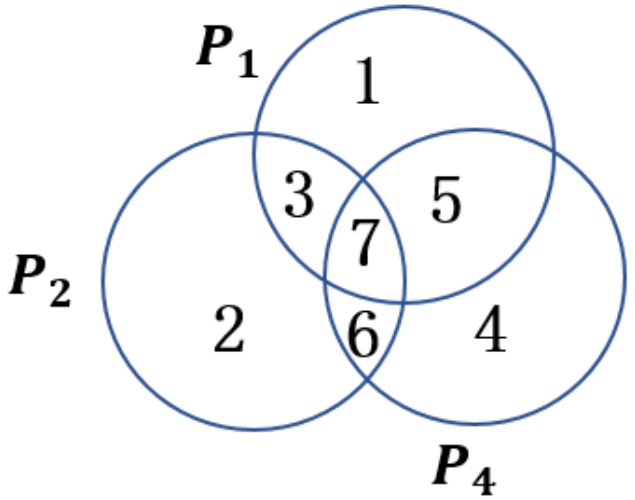
二进制序号	1	2	3	4	5	6	7
名称	C_1	C_2	0	C_4	0	1	1
	0	1		1			

$C_1 = \overline{\text{位3} \oplus \text{位5} \oplus \text{位7}} = 0$

$C_2 = \overline{\text{位3} \oplus \text{位6} \oplus \text{位7}} = 1$

$C_4 = \overline{\text{位5} \oplus \text{位6} \oplus \text{位7}} = 1$

配奇的汉明码为 0101011



提高访问主存储器速度的方法

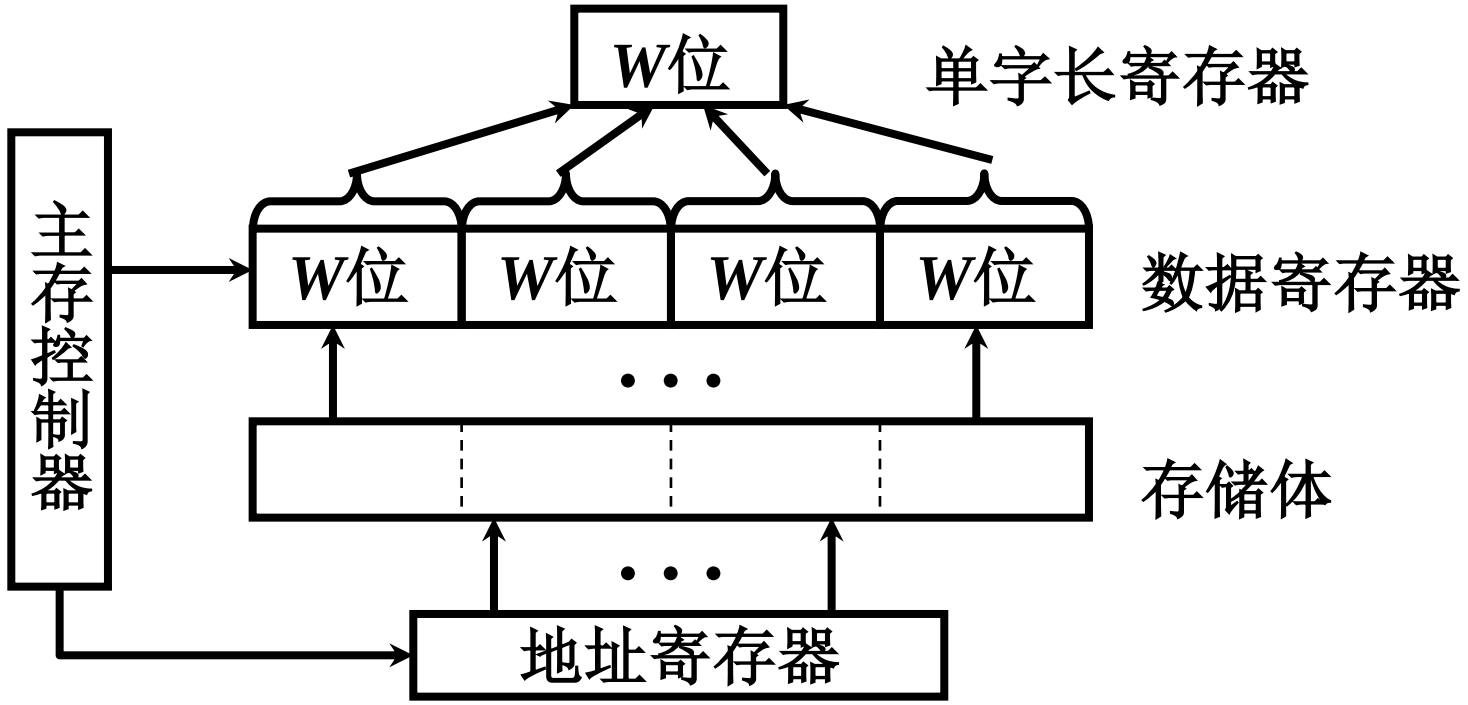
- 采用高性能存储芯片
  - SDRAM (同步 DRAM):在系统时钟的控制下进行读出和写入, CPU 无须等待
  - RDRAM（Rambus DRAM）: 主要解决 存储器带宽 问题
  - 带 Cache 的 DRAM: 在 DRAM 的芯片内集成了一个由 SRAM 组成的 Cache, 有利于 猝发式读取
- 采用层次结构 Cache –主存
- 优化主存结构

调整主存结构，提高访存速度

1. 单体多字系统

思想：增加存储器带宽，提升单体存储器工作速度。

如：在一个存取周期内，从同一地址取出4条指令，然后再逐条送到CPU执行，即每隔1/4存取周期，主存向CPU送一条指令，带宽相比原来提升了4倍。



缺陷：对于存储体，每次必须要写入/读出4W位

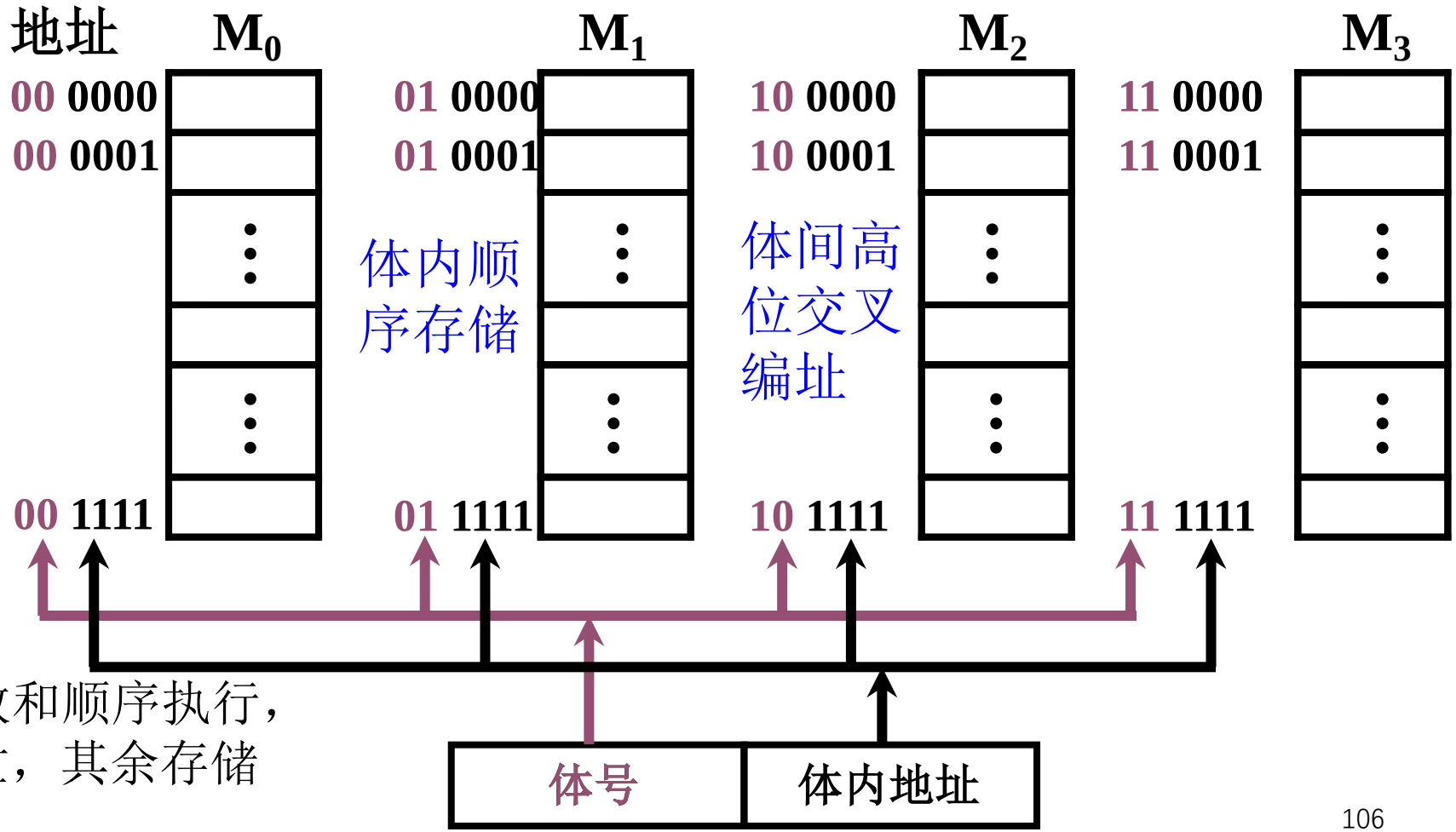
- 若一条指令为1W，指令读出时可能只有其中一条指令是有用的，如第一条指令是跳转指令
- 数据写入时，当只写入W位数据时，其他3W位数据会被篡改

2. 多体并行系统—(1)高位交叉 顺序编址

多体并行系统：采用多体模块组成的存储器。每个模块有相同的容量和存取速度。

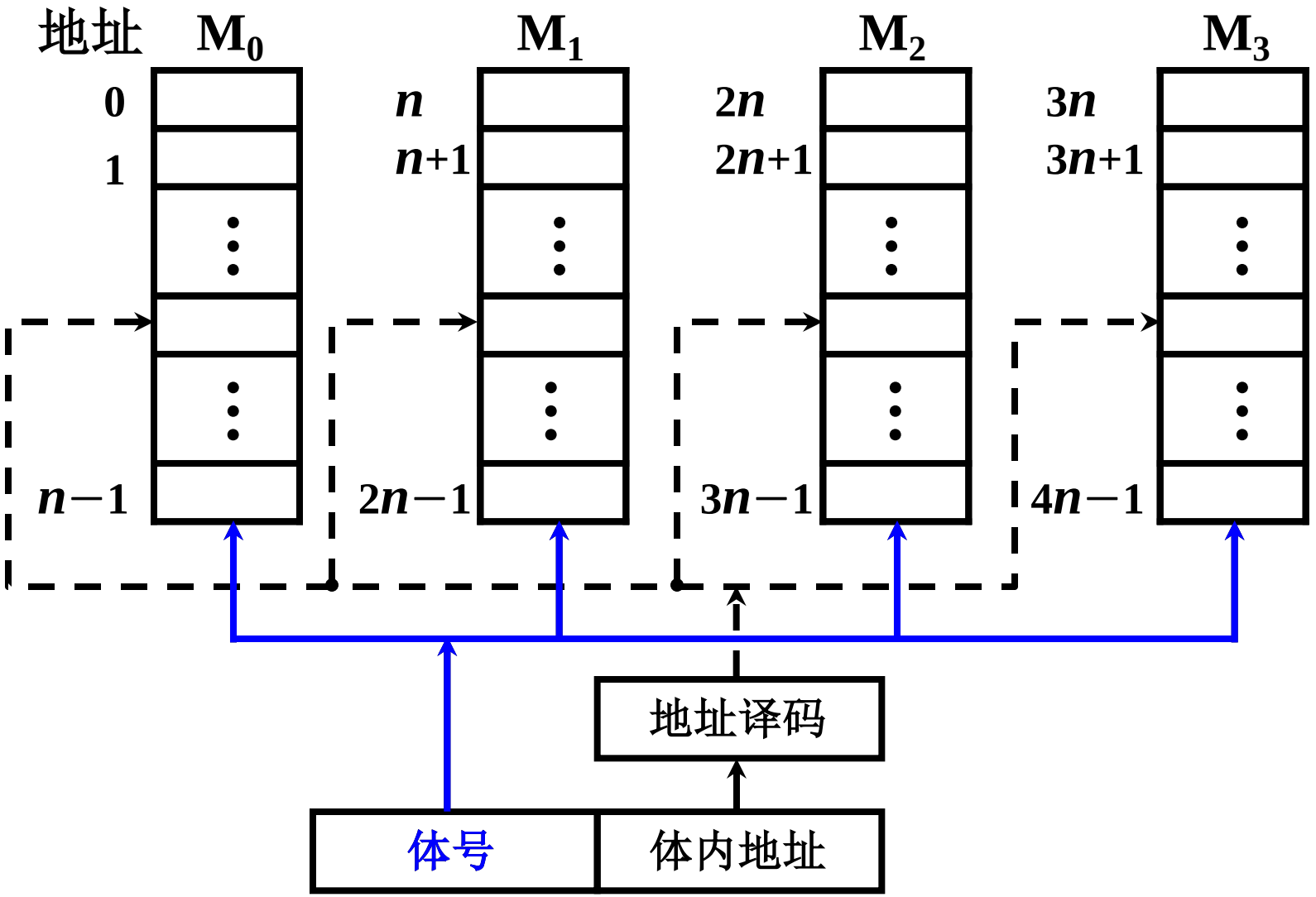
并行工作：同时访问N个模块，同时启动，同时读出，完全并行工作。

优点：①让不同的请求源同时访问不同的体，可实现并行工作。
如：一个体与CPU交换信息时，另一个体可同时与外部设备进行访存
②体内连续编址，利于存储器扩充



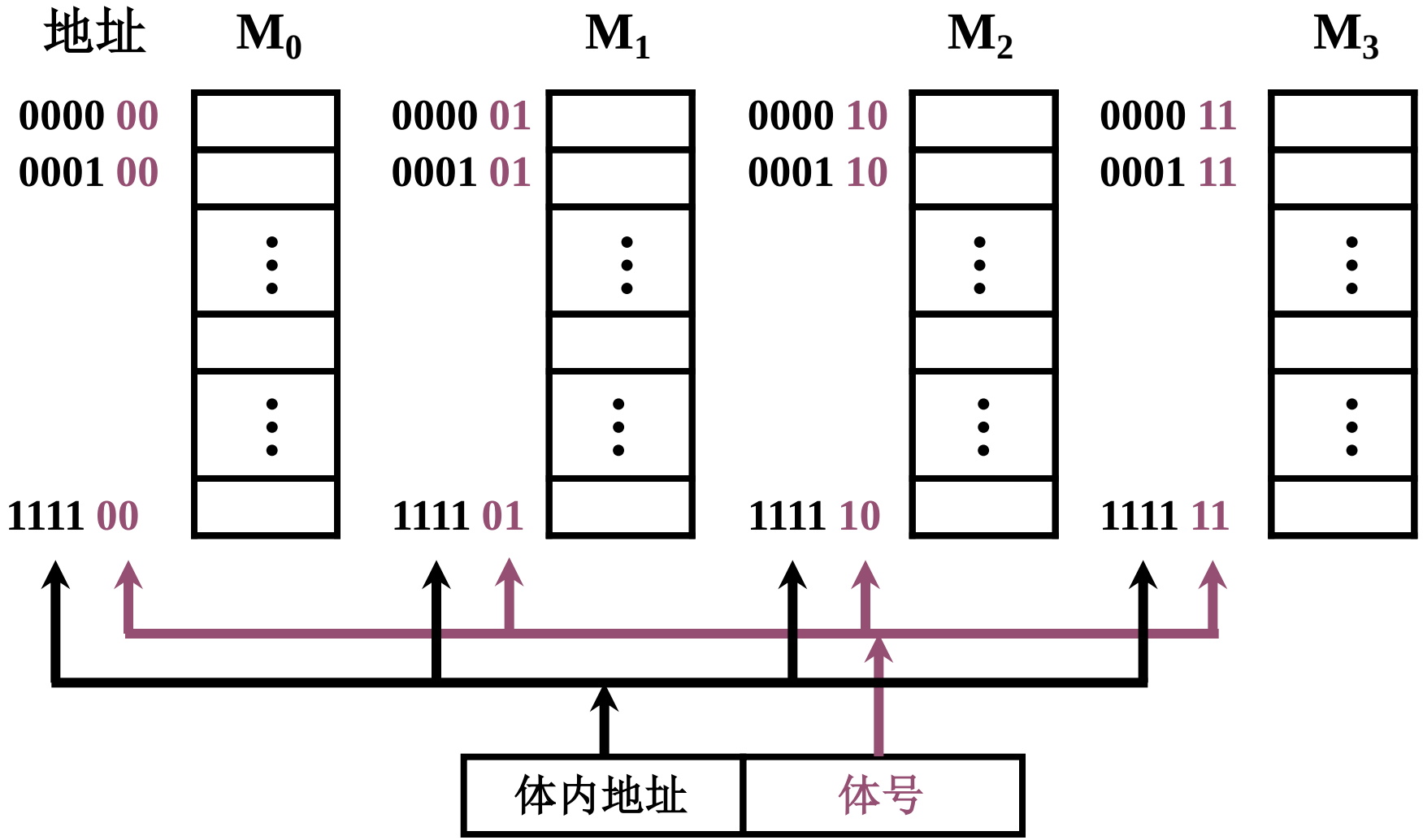
缺点：体内指令的连续存放和顺序执行，会造成某个存储体特别繁忙，其余存储体特别空闲

(1) 高位交叉——各个体并行工作

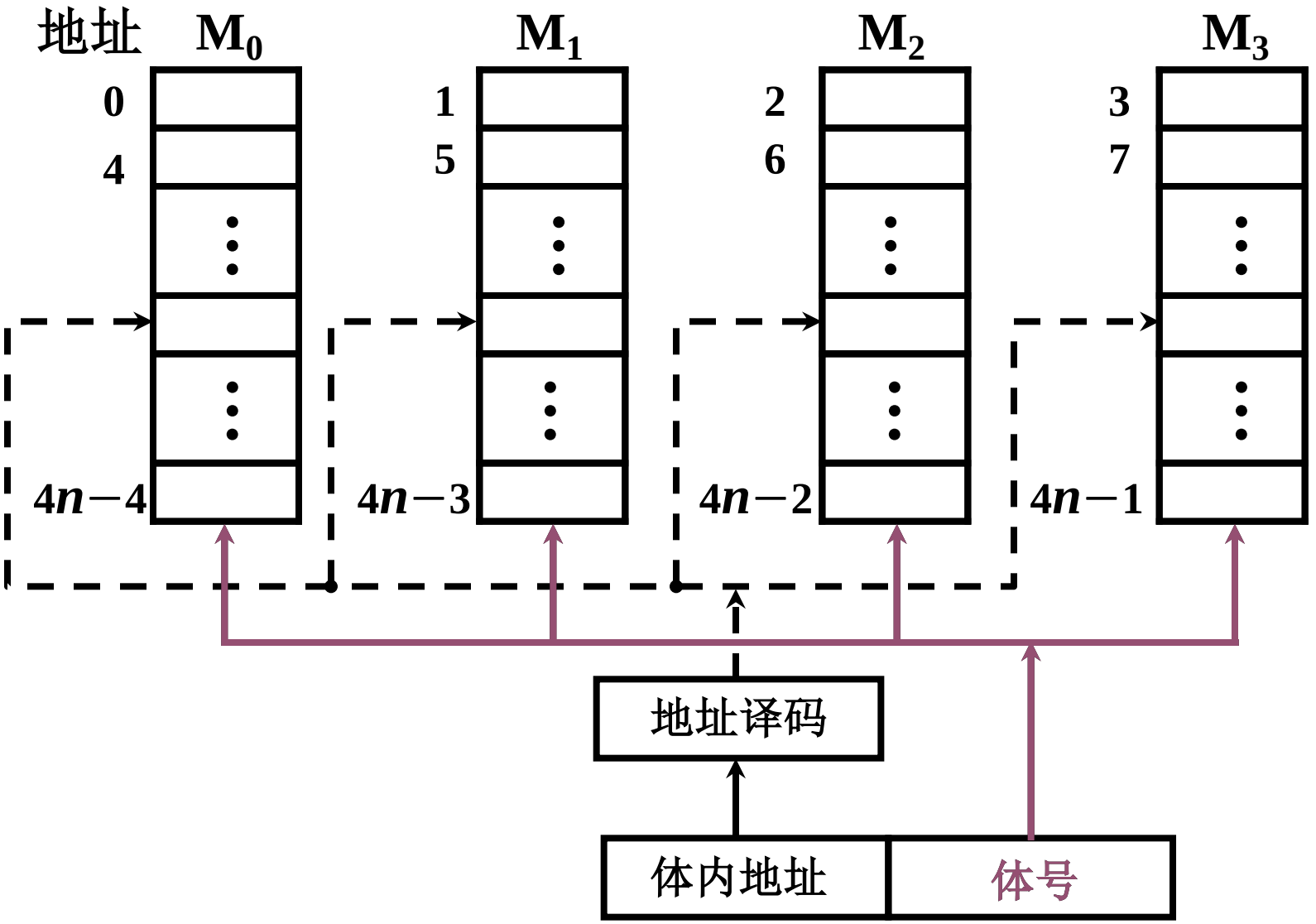


用高位作为选择信号选择存储体，用低位作为存储体内部的地址输入到存储体当中，寻找指定的存储单页，实际上类似存储器容量的扩展。

(2) 低位交叉——各个体轮流编址

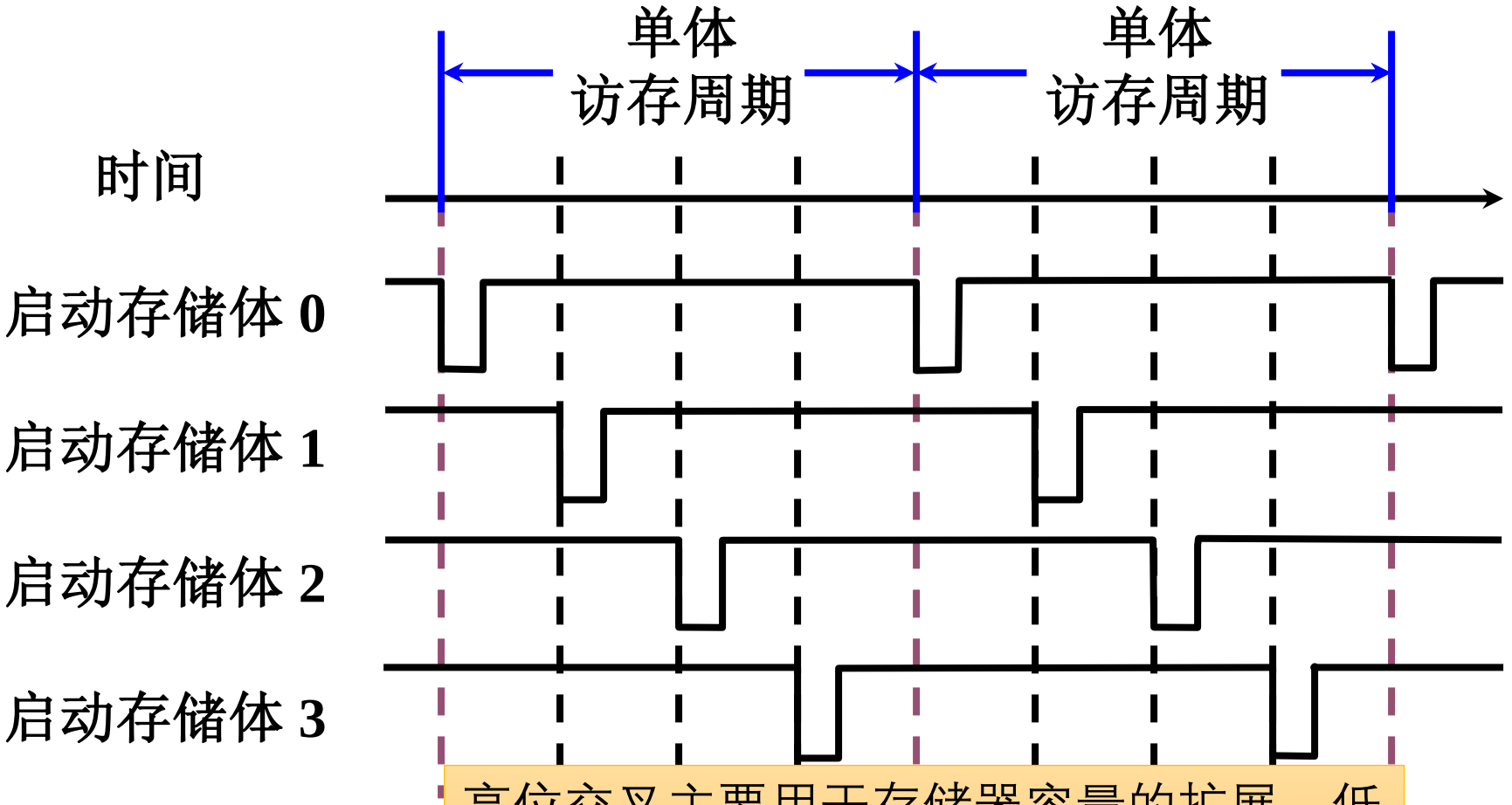


(2) 低位交叉——各个体轮流编址



低位交叉的特点

在不改变存取周期的前提下，增加存储器的带宽

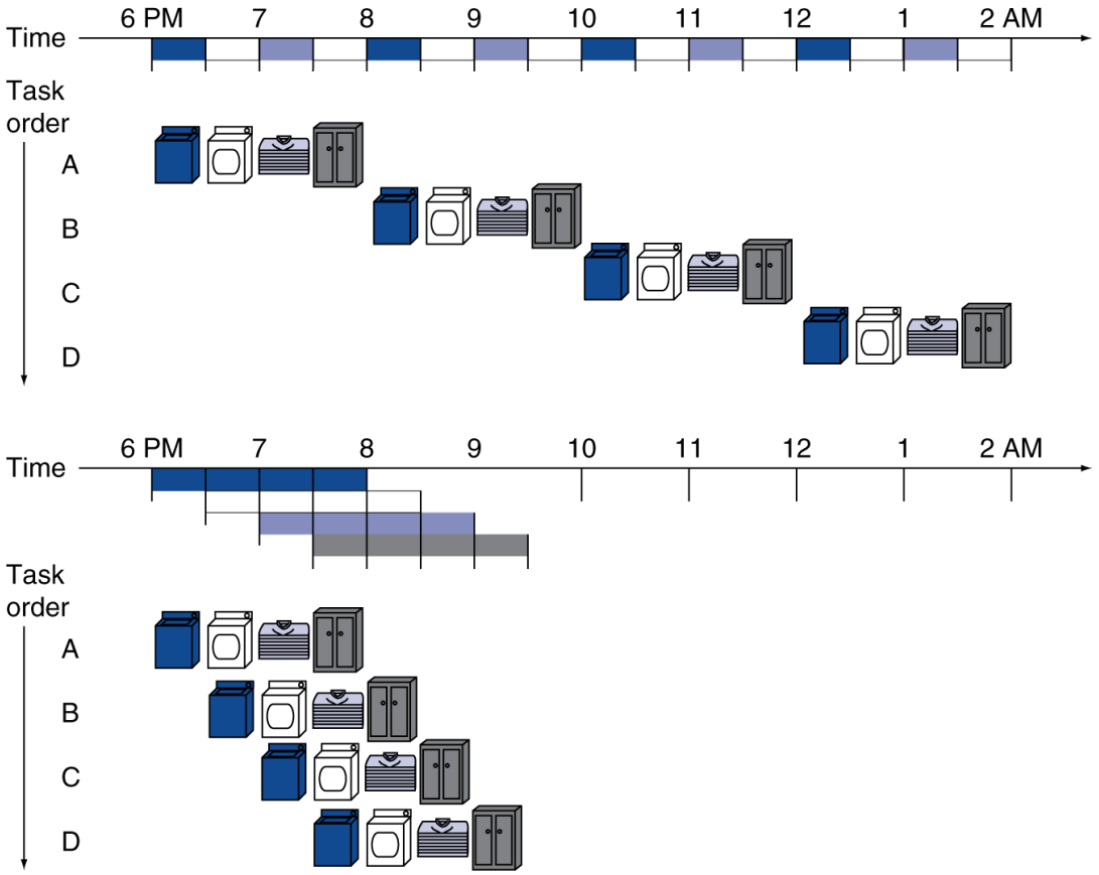


- 负脉冲为各体工作启动信号
- **带宽增加：** CPU交叉访问各体，各体读/写过程重叠执行，一个存储周期，存储器实际向CPU提供了4个存储字

高位交叉主要用于存储器容量的扩展，低位交叉用于存储器带宽和访问速度的提高。

流水线的概念介绍：一个比喻

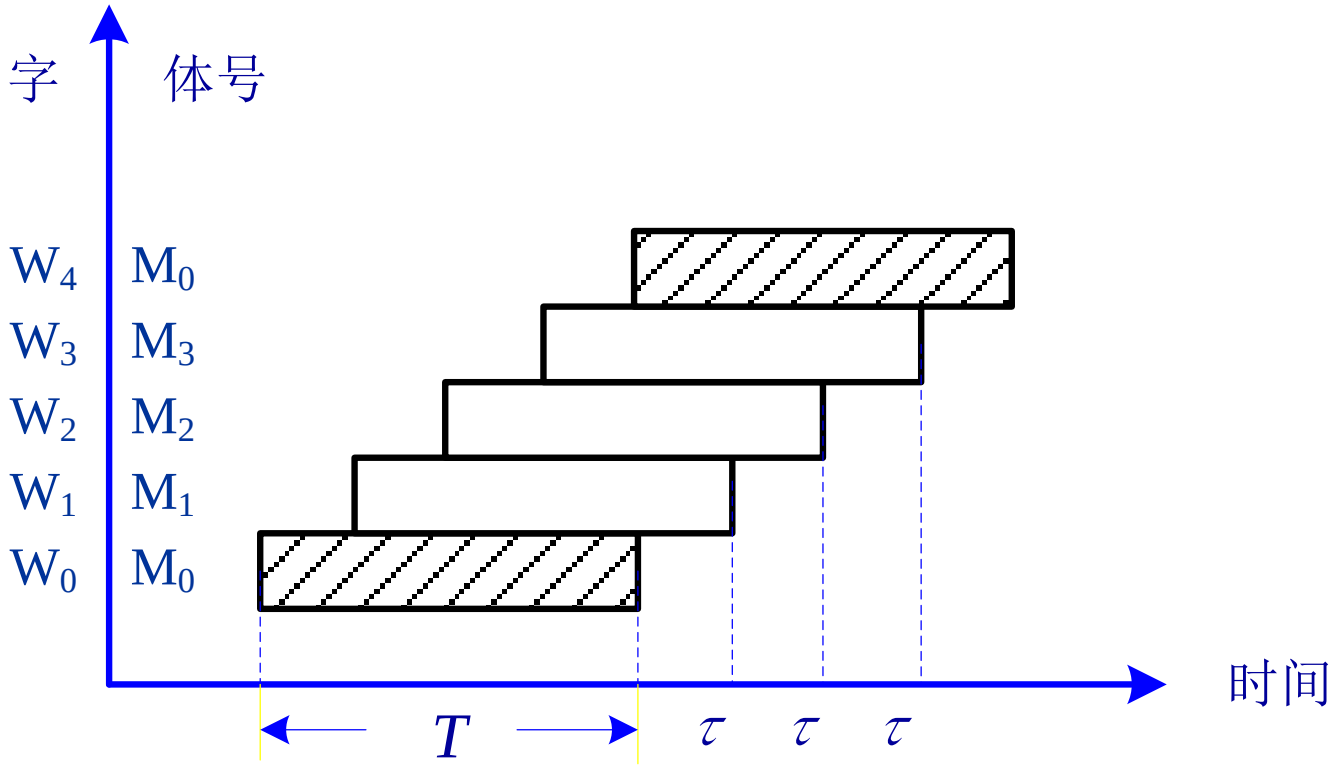
- 以洗衣服为例类比流水线:重叠执行
  - 假设洗衣包括四个步骤：洗衣机中洗衣、烘干机中烘干、叠衣服、收纳到柜子中



- 负载为4:
  - 加速比
    = 8/3.5 = 2.3
- 负载足够多:
  - 加速比
    = 2n/(0.5n+1.5) ≈ 4
    = 流水线中步骤的数目

流水线方式存取

- 设四体低位交叉存储器，存取周期为T，总线传输周期为τ，为实现流水线方式存取，应满足 $T = 4\tau$。



- 连续读取 4 个字所需的时间为 $T+(4-1)\tau$
- 连续读取 n 个字所需的时间为 $T+(n-1)\tau$
- 非低位交叉存储器连续读取 n 个字所需的时间为 n T
- 加速比为 $nT/(T+(n-1)\tau)$ $= 4n\tau/((n+3)\tau) = 4$ (n足够大时)

第六章 存储器

6.1 存储器概述

6.2 主存储器

6.3 高速缓冲存储器

6.4 虚拟存储器

6.5 辅助存储器

6.3 高速缓冲存储器

一、概述

1. 为什么用Cache
2. Cache的工作原理
   - 主存与缓存的编址
   - 命中与未命中
   - Cache的命中率
   - Cache-主存系统的效率
3. Cache的基本结构
4. Cache的读写操作
5. Cache的改进

二、Cache-主存的地址映射

三、缓存替换算法

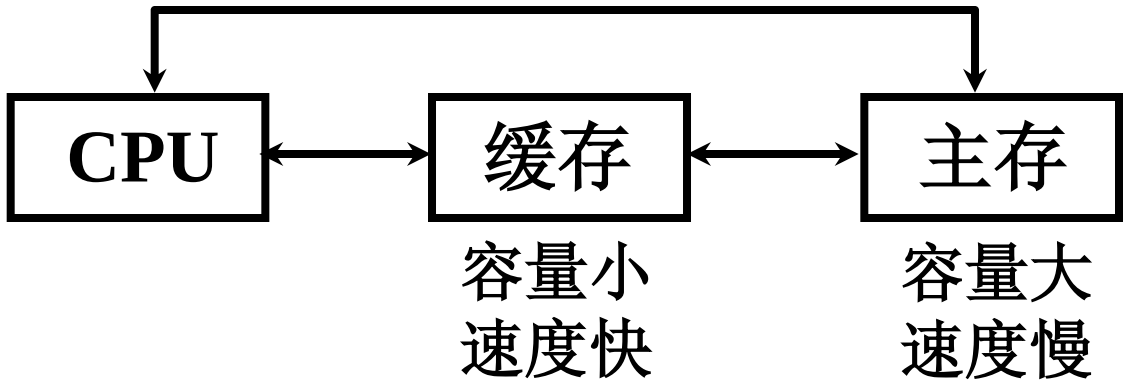
6.3 高速缓冲存储器

一、概述

1. 问题：为什么用Cache？

避免 CPU “空等” 现象

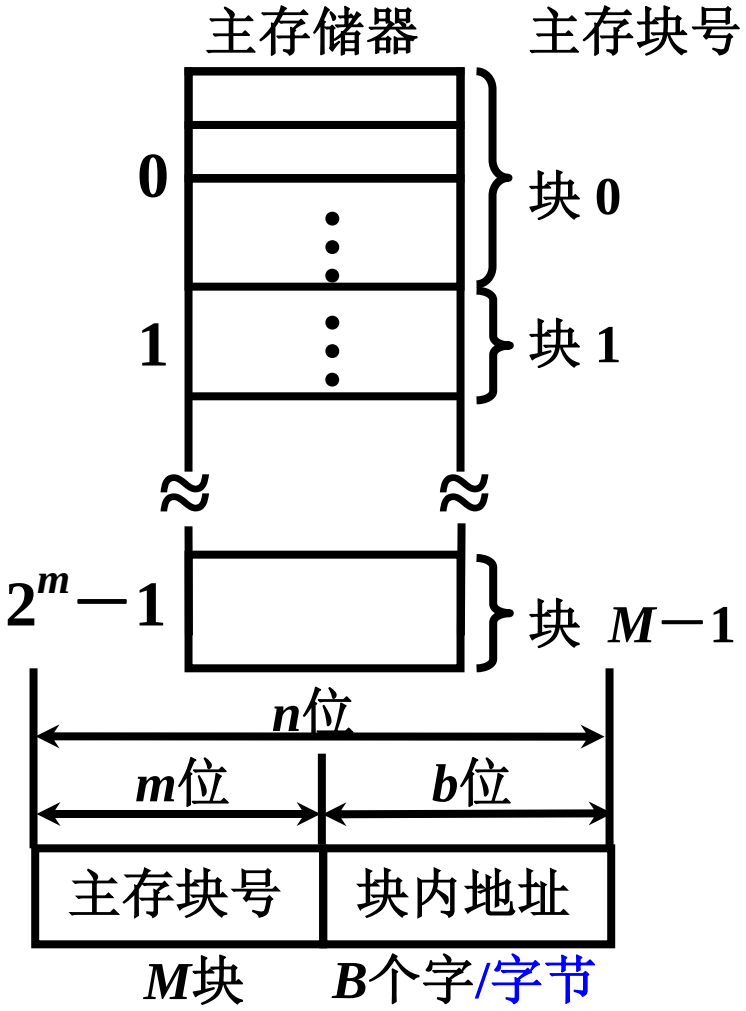
CPU 和主存（DRAM）的速度差异

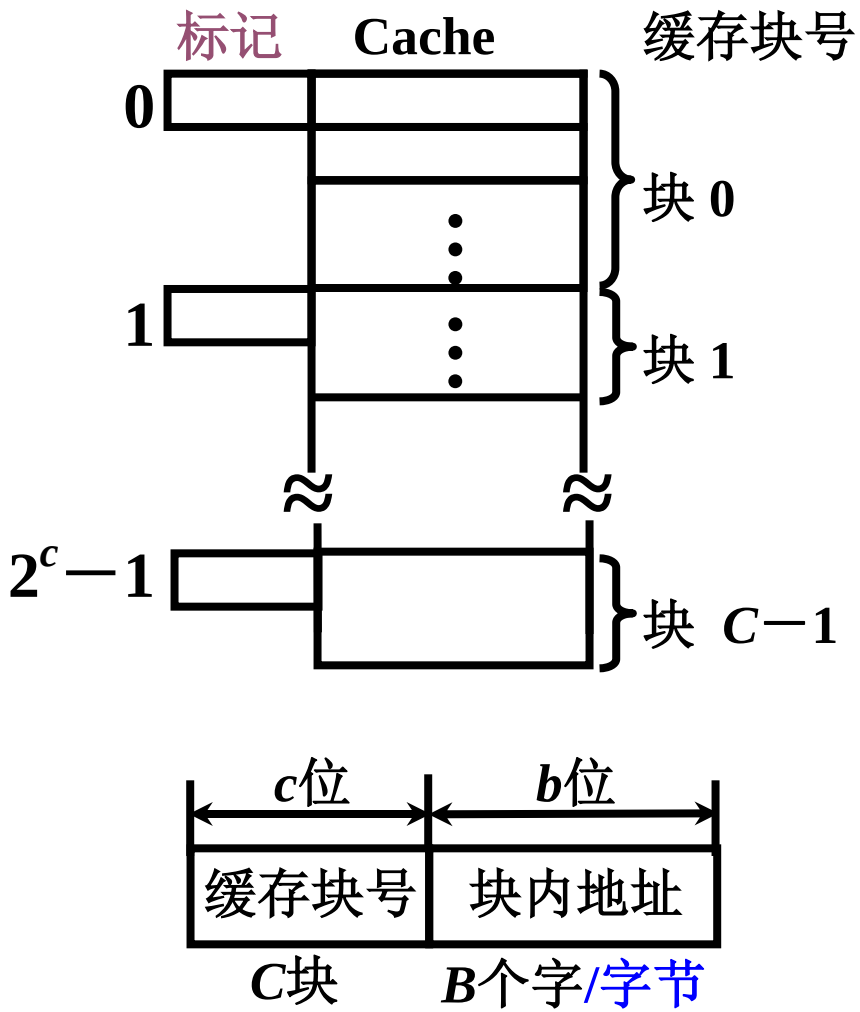


程序访问的局部性原理（时间局部性、空间局部性）

2. Cache 的工作原理

(1) 主存和缓存的编址





- **Cache的标记**

表示当前位置存放的是哪一个主存块。标记中存储的内容一般为主存字块的m位编号。

主存和缓存按块存储　　块的大小相同　　*B* 为块长

(2) 命中与未命中

缓存共有 C 块

主存共有 M 块　$M >> C$

命中　主存块 调入 缓存

主存块与缓存块 建立 了对应关系

用 标记 记录与某缓存块建立了对应关系的 主存块号

未命中　主存块 未调入 缓存

主存块与缓存块 未建立 对应关系

(3) Cache 的命中率

CPU 欲访问的信息在 Cache 中的 比率

命中率 与 Cache 的 容量 与 块长 有关

一般每块可取 4 ~ 8 个字

块长：一个存取周期内从主存获取的信息长度

CRAY_1	16体交叉	块长取 16 个存储字
IBM 370/168	4体交叉	块长取 4 个存储字

（64位×4 = 256位）

(4) Cache –主存系统的效率

效率 e 与 命中率 有关

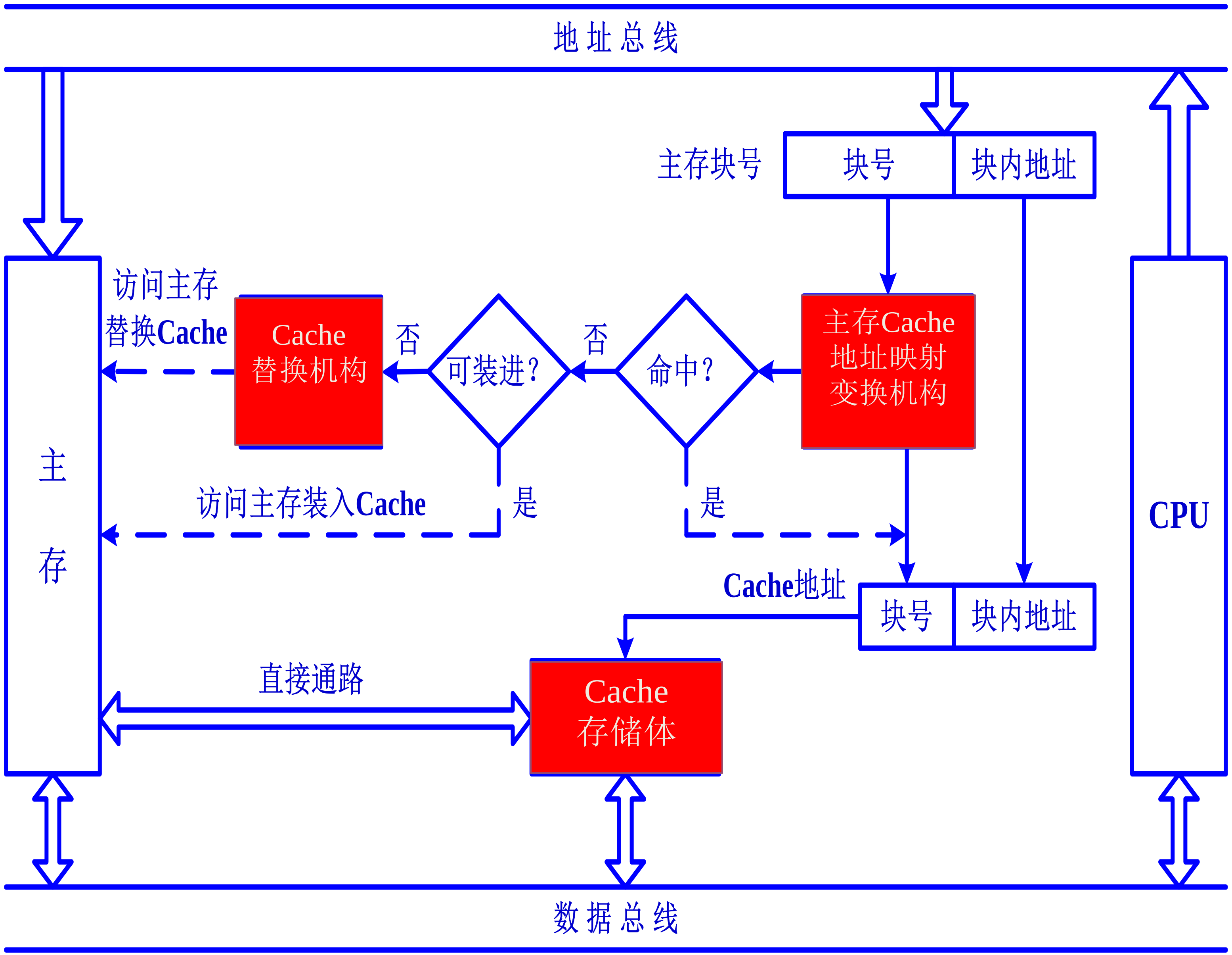
$$e = \frac{\text{访问 Cache 的时间}}{\text{平均访问时间}} \times 100\%$$

设 Cache 命中率 为 h，访问 Cache 的时间为 t_c，
访问 主存 的时间为 t_m

$$\text{则} \quad e = \frac{t_c}{h \times t_c + (1-h) \times t_m} \times 100\%$$

假设访问Cache和访问主存是同时进行的

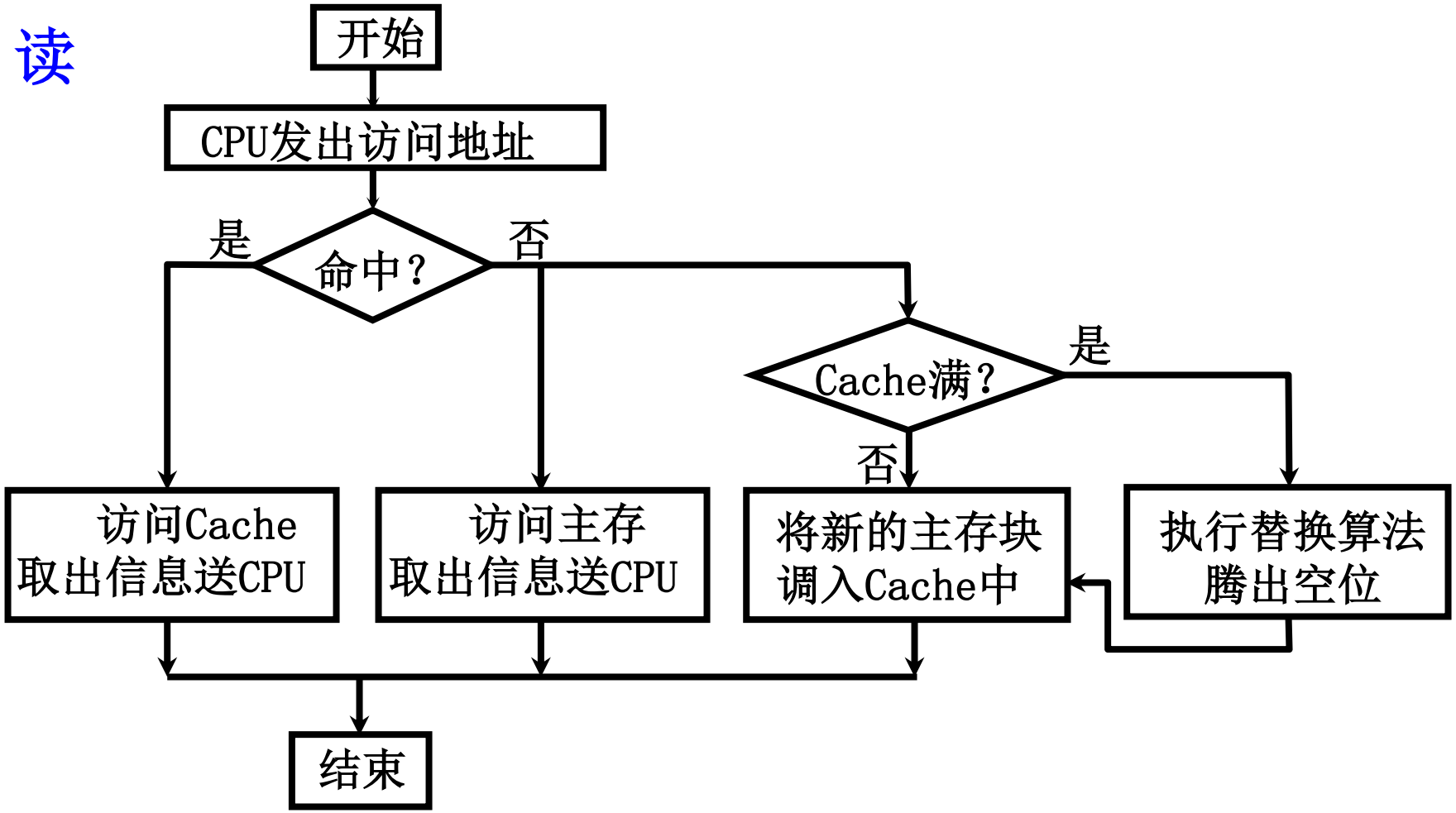
3. Cache 的基本结构



- [illegible]地址映射变换机构 [illegible] 需要由Cache内的替换机构按照 [illegible] 过程称为地址映射。

6.3 高速缓冲存储器

- Cache 的 读写 操作
  - 读



4. Cache 的 读写 操作

写　Cache 和主存的一致性

- **写直达法（Write – through）**

  写操作时数据既写入Cache又写入主存

  写操作时间就是访问主存的时间，读操作时不涉及对主存的写操作，更新策略比较容易实现

- **写回法（Write – back）**

  写操作时只把数据写入 Cache 而不写入主存,当 Cache 数据被替换出去时才写回主存

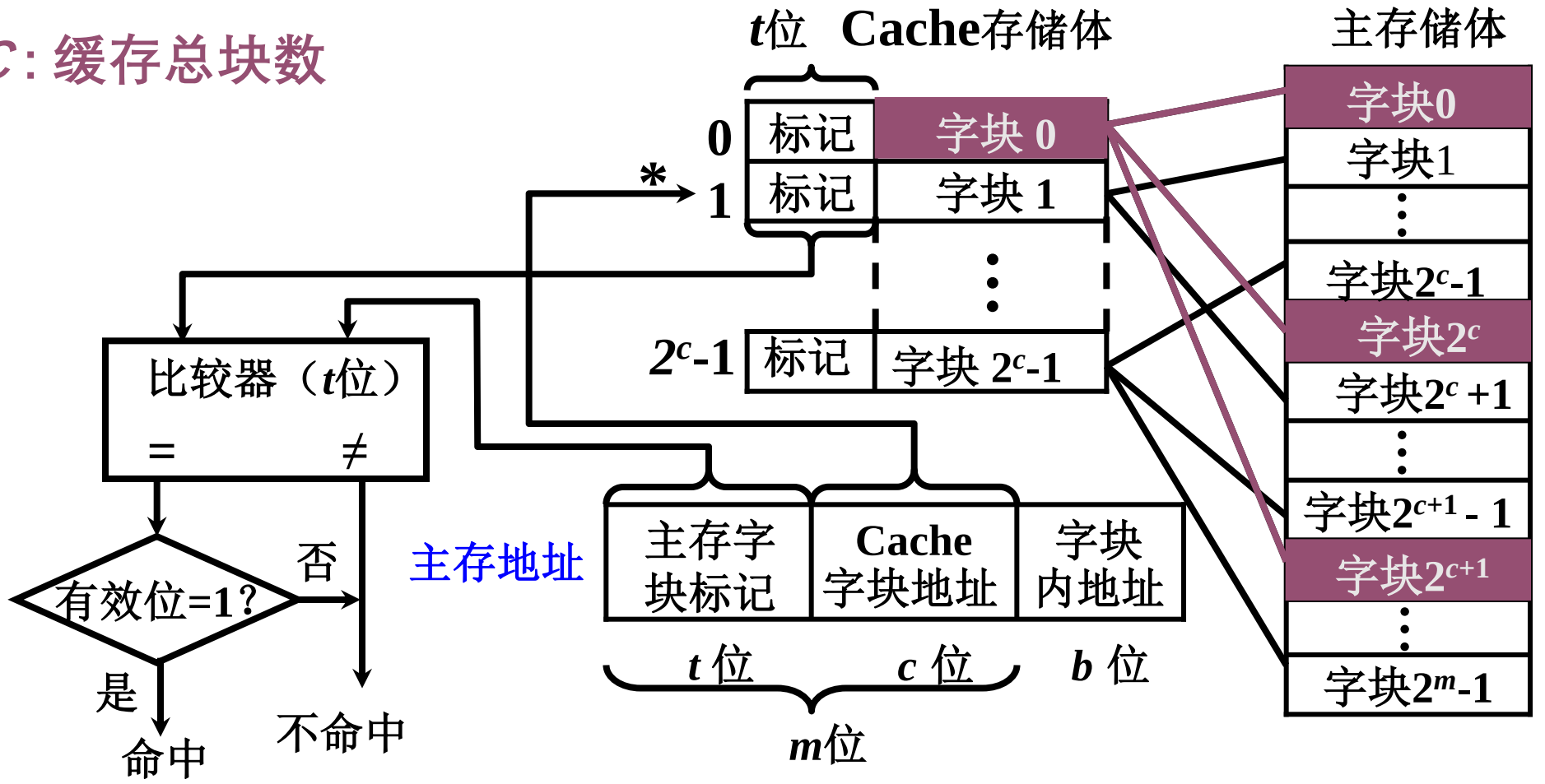
  写操作时间就是访问 Cache 的时间，读操作 Cache 失效发生数据替换时，被替换的块需写回主存，增加了 Cache 的复杂性

6.3 高速缓冲存储器

- 二、Cache – 主存的地址映射
  - 直接映射
  - 全相联映射
  - 组相联映射

1.直接映射

C: 缓存总块数



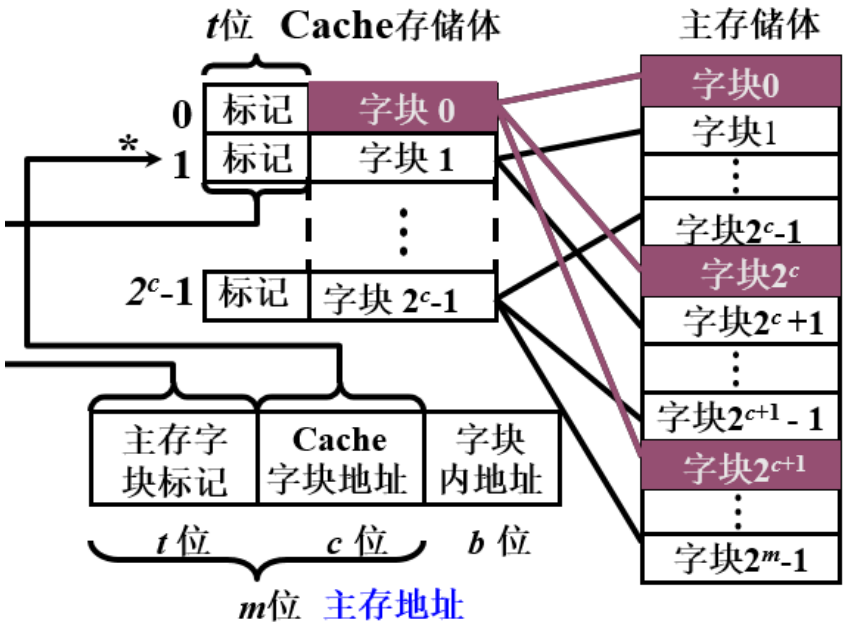
每个缓存块 *i* 可以和 若干 个 主存块 对应

每个主存块 *j* 只能和 一 个 缓存块 对应

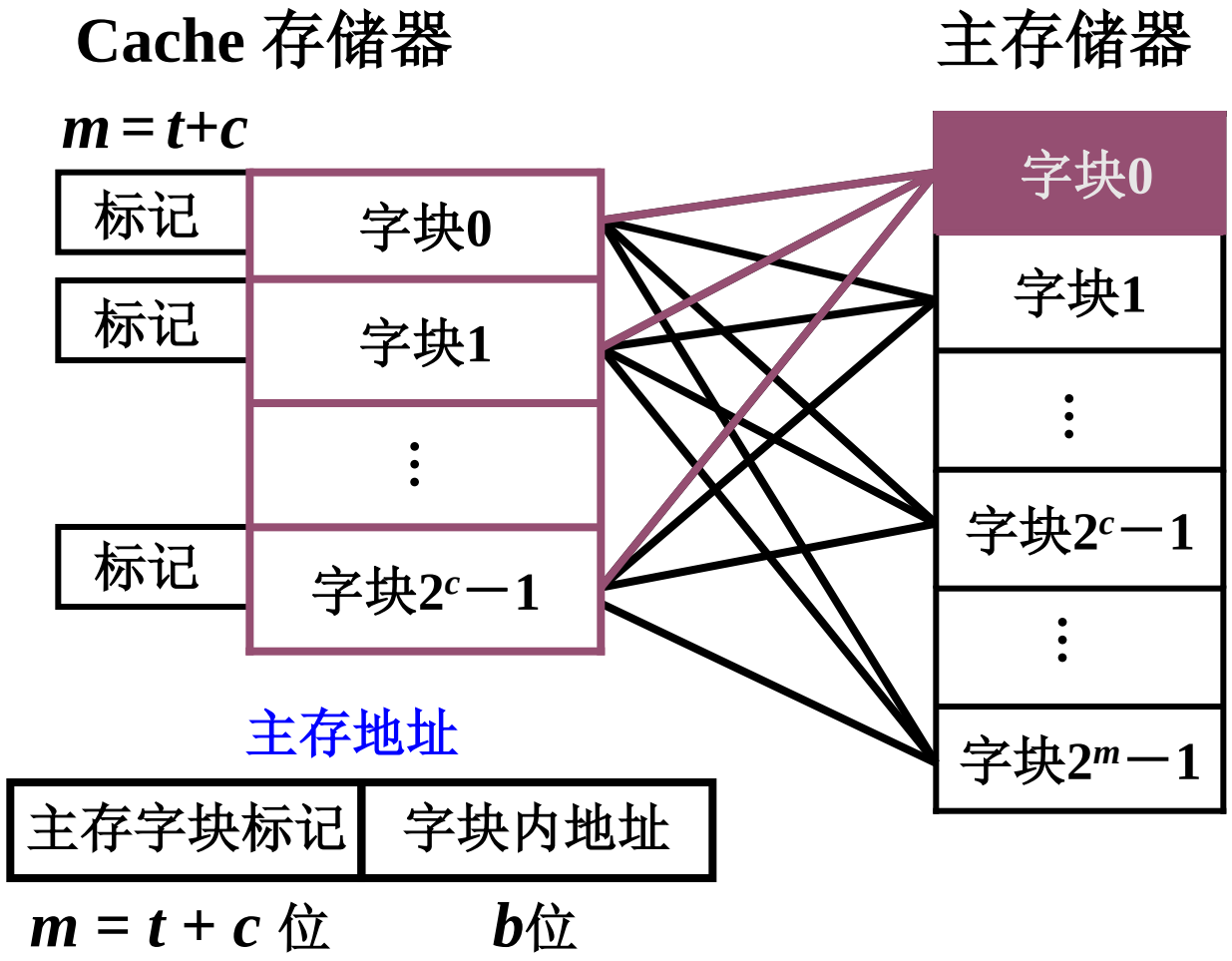
$i = j \bmod C$

1.直接映射

- 优点：直接映射实现/使用简单。
- 缺点：不够灵活
  - 每个主存块都映射到固定的缓存块，因此即使缓存中还空闲很多空间也不能占用，使缓存的存储空间得不到充分的利用。
  - 例：如果程序恰好重复访问对应于同一缓存块的不同主存块，例如，程序依次访问第0、2^c、2^{c+1}、2^{c+2}…个主存块，那么缓存每次都未命中，需要不停地将主存块替换到第0个缓存块所在位置。



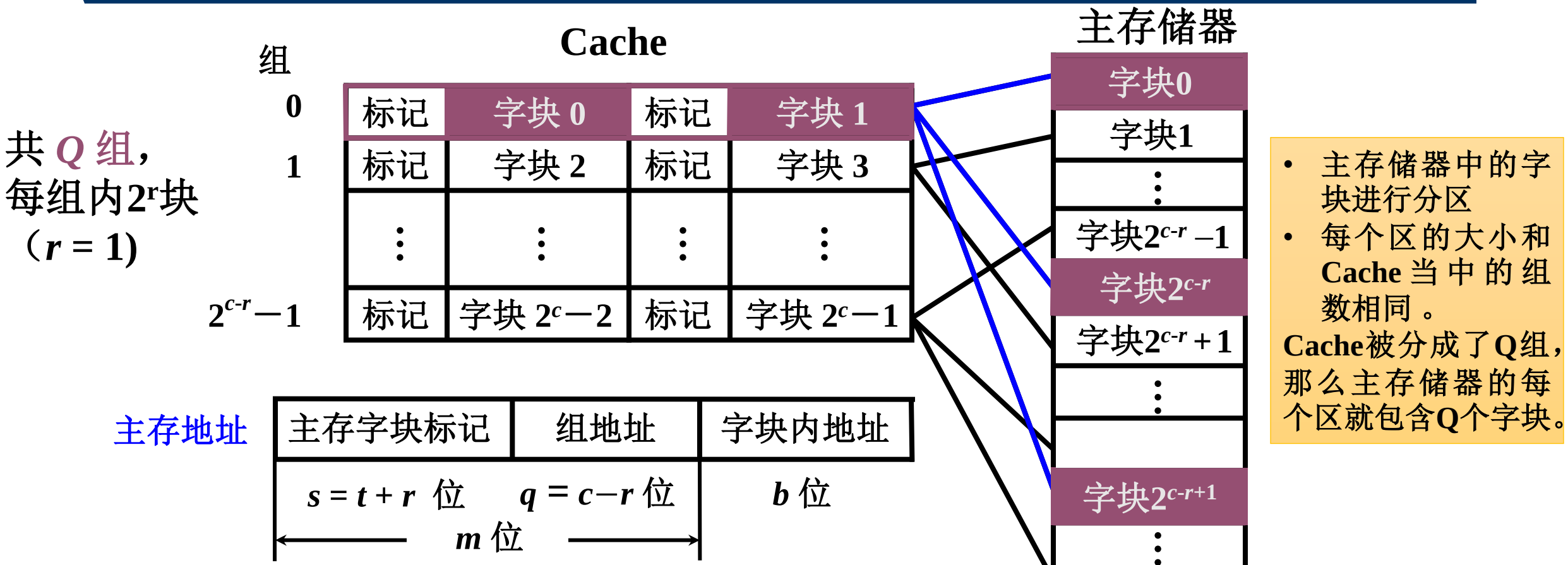
2. 全相联映射



- **优点：**全相联映射相比直接映射更加灵活，命中率更高。
- **缺点：实现复杂，成本较高。**
  - 实现全相联映射的Cache需要大量逻辑电路，成本高。
  - 当CPU给出一个n位访存地址后，需要依次比较所有缓存块的t+c位标记与当前访存地址中的t+c位标记是否一致。在实际应用时，需要实现各种措施来尽可能减少标记比较次数，因此实现较为复杂。

主存 中的 任一块 可以映射到 缓存 中的 任一块

3. 组相联映射



- 主存储器中的字块进行分区
- 每个区的大小和Cache当中的组数相同。

Cache被分成了Q组，那么主存储器的每个区就包含Q个字块。

例6.8

假设主存容量为512KB，Cache容量为4KB，每个块为16个字，**每个字32位**。

1) Cache 地址有多少位？可容纳多少块？

Cache 容量 4KB—>地址 **12** 位。 4KB/(4B*16) =**64** 块

2) 主存地址有多少位？可容纳多少块？

512KB—>主存地址**19**位。 512KB/(4B*16) = **8192** 块。

3) 在直接映射方式下，主存的第几块映射到 Cache 中的第5 块（设起始字块为第 1 块）？

主存的第 5，64 + **5**，2×64 + **5**， …， 8192 – 64 + **5** 块

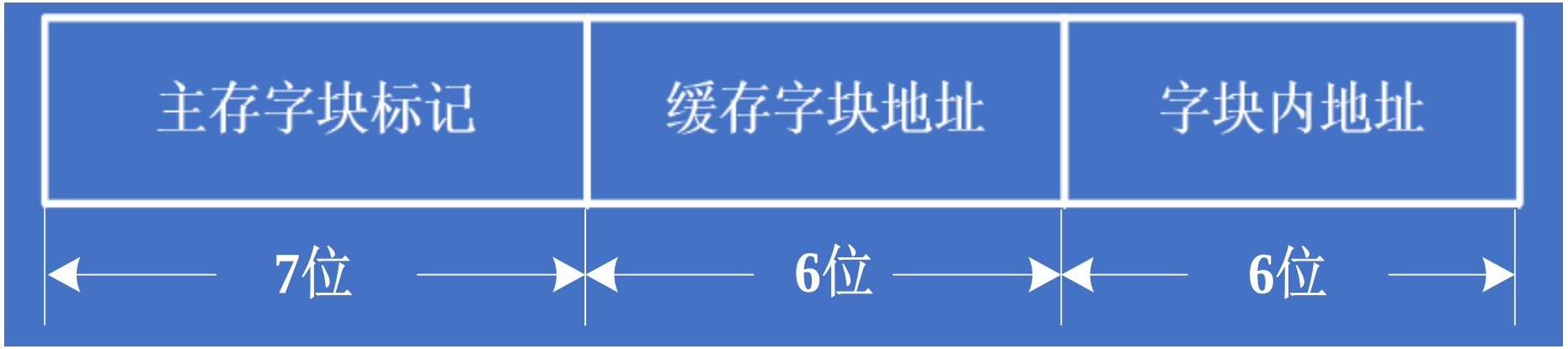
例6.8——续

(主存512KB，Cache 4KB，每个字块16个字，每个字32位)

4）画出直接映射方式下主存地址字段中各段的位数

(主存字块标记位数*t*+缓存字块地址位数*c*+字块内地址位数*b*)

- ✓ 字块内地址位数 ***b*= 6** 位（字块大小为：4B*16=64B）
- ✓ 缓存共 64 块，故缓存字块地址位数 ***c* = 6** 位
- ✓ 主存字块标记为主存地址与Cache地址长度之差***t*=*m*-*c*=19-12=7**位



例子6.9

- 假设主存容量 512K×16 位，Cache 容量 4096×16 位，每个块为4个16位的字，访存地址为字地址。

1）在直接映射方式下，设计主存的地址格式。

✓ 根据 Cache 容量为 4K字，得 **Cache 地址为 12 位。**

✓ 根据块长为 4，**按字访问，得字块内地址 2 位，**

Cache 块数= 4K/4 = **1024 块（10位）。**

✓ 根据主存容量为 512K字，得**主存字地址 19 位。**

例6.9 续

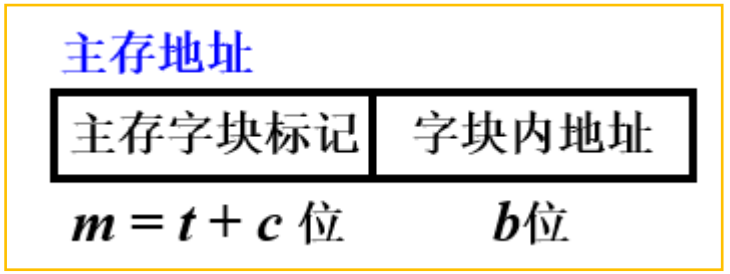
- 假设主存容量 512K×16 位，Cache 容量 4096×16 位，每个块为4个16位的字，访存地址为字地址。

2）在全相联映射方式下，设计主存的地址格式。

字块内地址 **2** 位，其余 **17** 位为主存字块标记

3）在二路组相联映射方式下，设计主存的地址格式。

一组内有 **2** 块，Cache有**1024块** ——>512组，组地址 **9** 位



填空题 4分

设置

假设主存容量 512K×**32** 位，Cache 容量 4096×16 位，每个块为4个**16位的字**，访存地址为**字地址**。

在四路组相联映射方式下，

1. 主存地址为[填空1] 位，
2. 字块内地址为 [填空2] 位，
3. 组地址为 [填空3] 位，
4. 主存字块标记为[填空4] 位。

作答

正常使用填空题需3.0以上版本雨课堂

练习解析

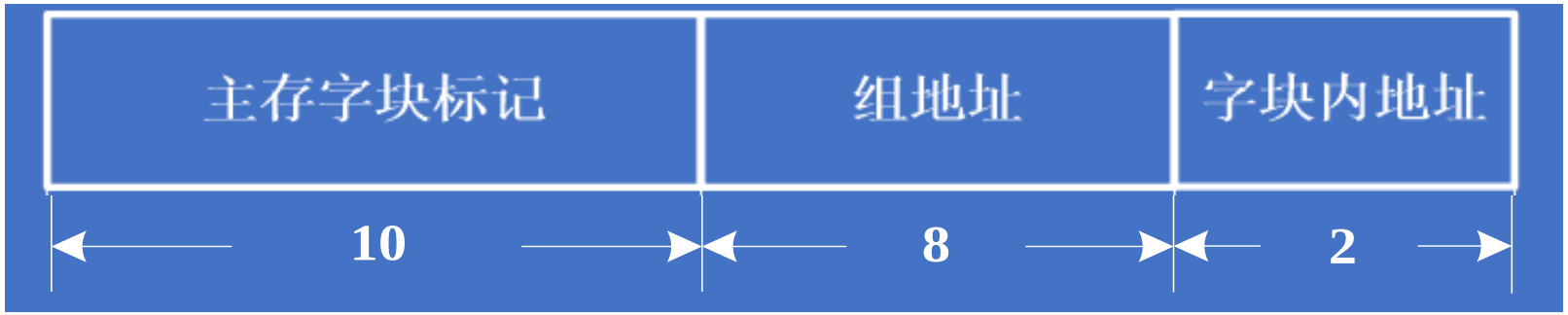
- 假设主存容量 512K×**32** 位，Cache 容量 4096×16 位，每个块为4个16位的字，访存地址为**字地址**。

在四路组相联映射方式下，设计主存的地址格式。

解：每块4个字—>**字块内地址为2位**

主存容量 512K×32 位—>1M×16 位 —>主存地址 20 位

四路组相联，一组内4块，Cache共有1024/4 = 256 组—>8位



6.3 高速缓冲存储器——缓存替换算法

- 先进先出法（FIFO：First In First Out）
- 近期最少使用法（LRU：Least Recently Used）
- 最不经常使用法（LFU：Least Frequently Used）
- 随机替换法

6.3高速缓冲存储器——缓存替换算法

- **先进先出法（FIFO：First In First Out）**
  - 当Cache占满时，它总是选择最早调入Cache的字块进行替换。
  - 优点：先进先出法容易实现、开销小，不需要记录Cache中各字块的使用情况。
  - 缺陷：没有基于访存的局部性原理，不能提高Cache的命中率。

6.3高速缓冲存储器——缓存替换算法

- **近期最少使用法（LRU：Least Recently Used）**
  - 当Cache占满时，选择近期用得最少的字块进行替换。
  - 近期最少使用法需要随时**记录**Cache中各个缓存块的使用情况（记录从上一次使用到此刻的时间），并将间隔时间最长的缓存块替换掉，实现较为复杂。
  - 近期最少使用法的平均命中率比先进先出法要高。

6.3高速缓冲存储器——缓存替换算法

- **最不经常使用法（LFU：Least Frequently Used）**
  - 最不经常使用法的思路与近期最少使用法类似，当Cache占满时，选择最不经常使用的字块进行替换。
  - 最不经常使用法也需要随时记录Cache中各个缓存块的使用情况（记录一定时间内每个缓存块被访问的次数），并将访问次数最少的缓存块替换掉，**实现也较为复杂**。
  - 最不经常使用法的平均命中率比先进先出法要高。

6.3高速缓冲存储器——缓存替换算法

- 随机替换法
  - 当Cache占满时，随机选择一个缓存块替换。
  - 实现简单，但是也没有考虑程序访问的局部性原理，无法提高Cache的命中率。

6.3 高速缓冲存储器——小结

- **实际应用：** 块设备缓存、硬盘缓存、web cache...

不灵活

直接映射　　某一 主存块 只能固定 映射到 某一 缓存块

全相联映射　某一 主存块 能 映射到 任一 缓存块

组相联映射　某一 主存块只能映射某一缓存组中的任一块

成本高

第六章 存储器

6.1 存储器概述

6.2 主存储器

6.3 高速缓冲存储器

6.4 虚拟存储器

6.5 辅助存储器

虚拟存储器思想的提出

- **主存容量限制**带来的诸多不便
  - 用户编写程序必须考虑主存容量的限制
  - 多道程序设计的道数受到限制
- **用户编程行为分析**
  - 全面考虑各种情况，执行时有互斥性
  - 顺序性和循环性等空间局部性行为
  - 某一阶段执行的时间局部性行为
- **因此可以考虑部分调入进程内容**

虚拟存储器的基本思想

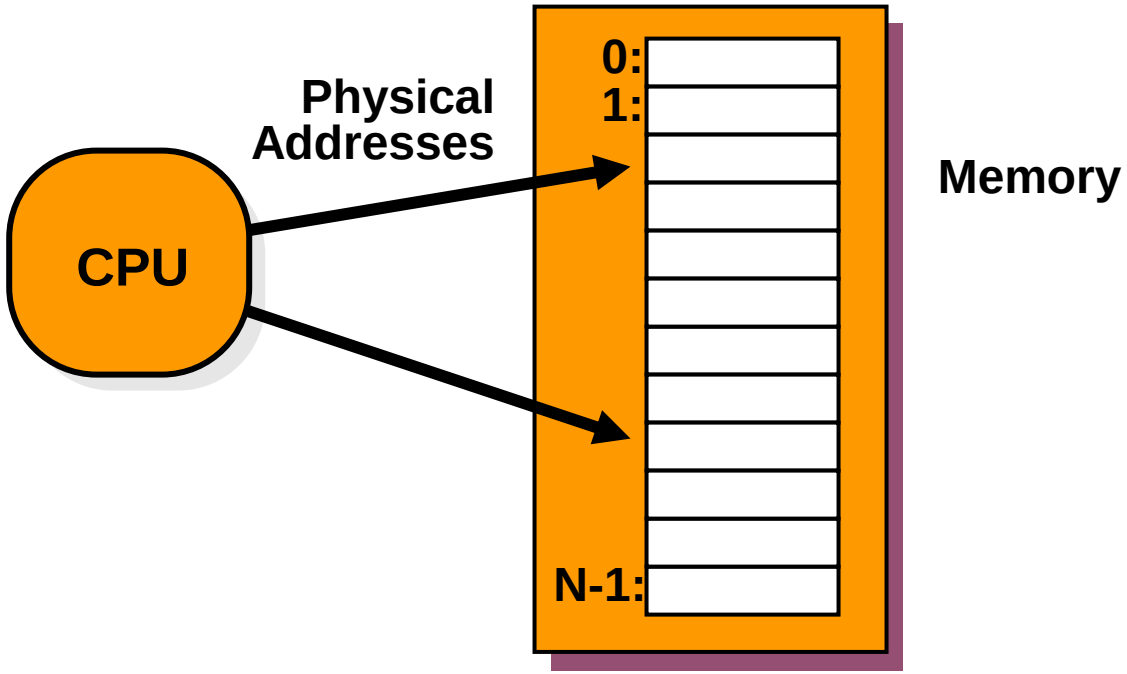
- 存储管理把进程全部信息放在辅存中，执行时先将其中一部分装入主存，以后根据执行行为**随用随调入**
- 如果主存中没有足够的空间，存储管理需要根据执行行为把主存中暂时不用的信息调出到辅存上去
- 虚拟存储采用与Cache类似原理，提高速度，降低成本
- 利用程序局部性，使存储系统的性能接近于高速存储器；价格接近于低速存储器，并扩充主存容量

虚拟存储器的实现思路

- 需要建立与自动管理两个地址空间
- （辅存）虚拟地址空间容纳进程装入
- （主存）实际地址空间承载进程执行
- 对于用户，计算机系统具有一个容量大得多得主存空间，即虚拟存储器，虚存空间是靠辅存（磁盘）来支持的
- 虚拟存储器是一种地址空间扩展技术，通常意义上对用户编程是透明的，除非用户需要进行高性能的程序设计。

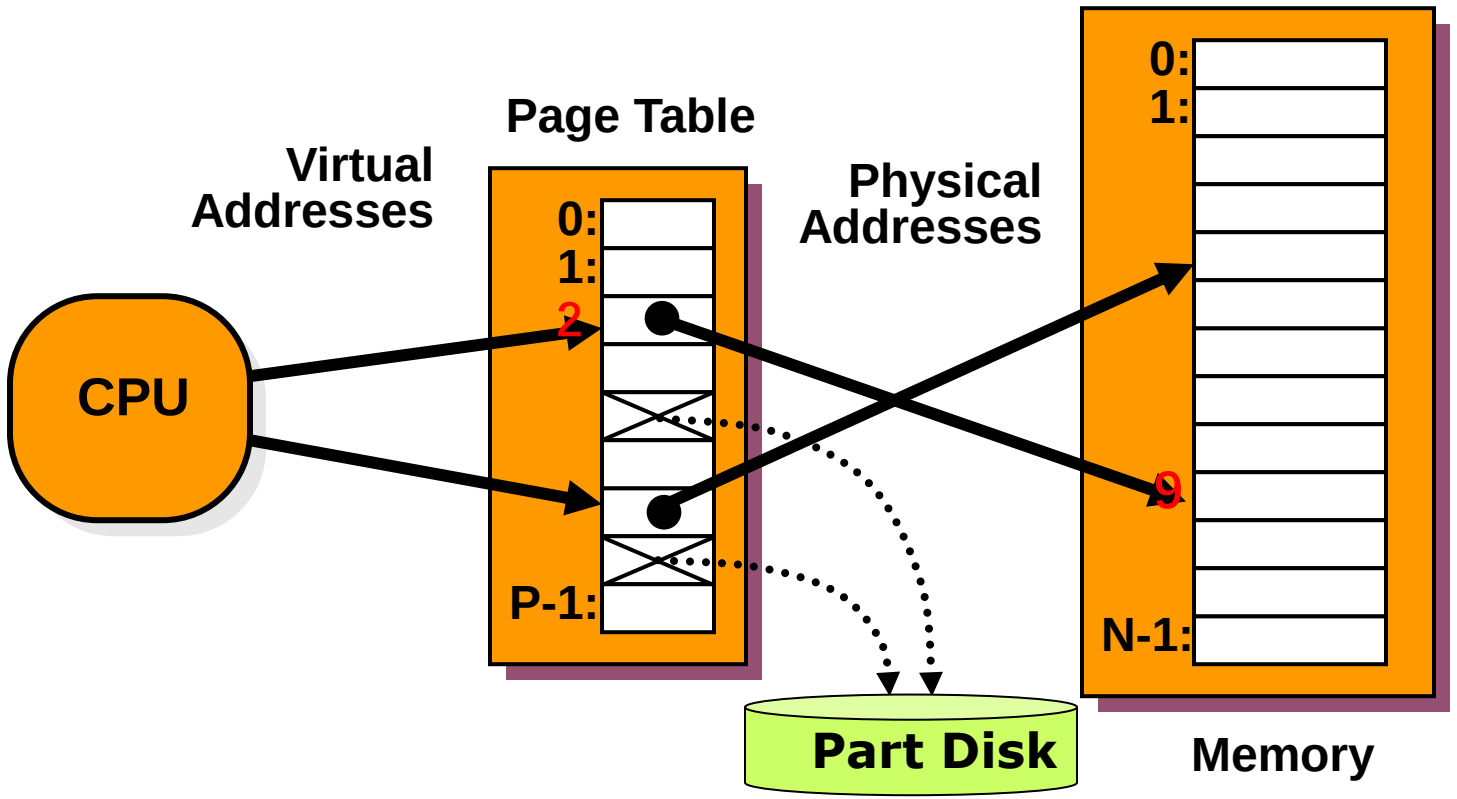
虚拟存储器

- 实地址计算机系统
  - CPU给出的地址直接访问物理内存
  - 大多数Cray计算机、早期PC、大多数嵌入式系统



虚拟存储器

- 虚地址计算机系统
  - CPU给出的地址需要虚实变换
    - 通过地址映射与转换机构以及操作系统维护的页表将虚拟地址转换为物理地址
  - 工作站、服务器、现代PC



页：虚拟存储器中的存储单元，默认大小为4KB，因此需要12位页内偏移地址。对于4GB的虚拟存储器，页表项有4GB/4KB=2^{20}项。

虚地址：

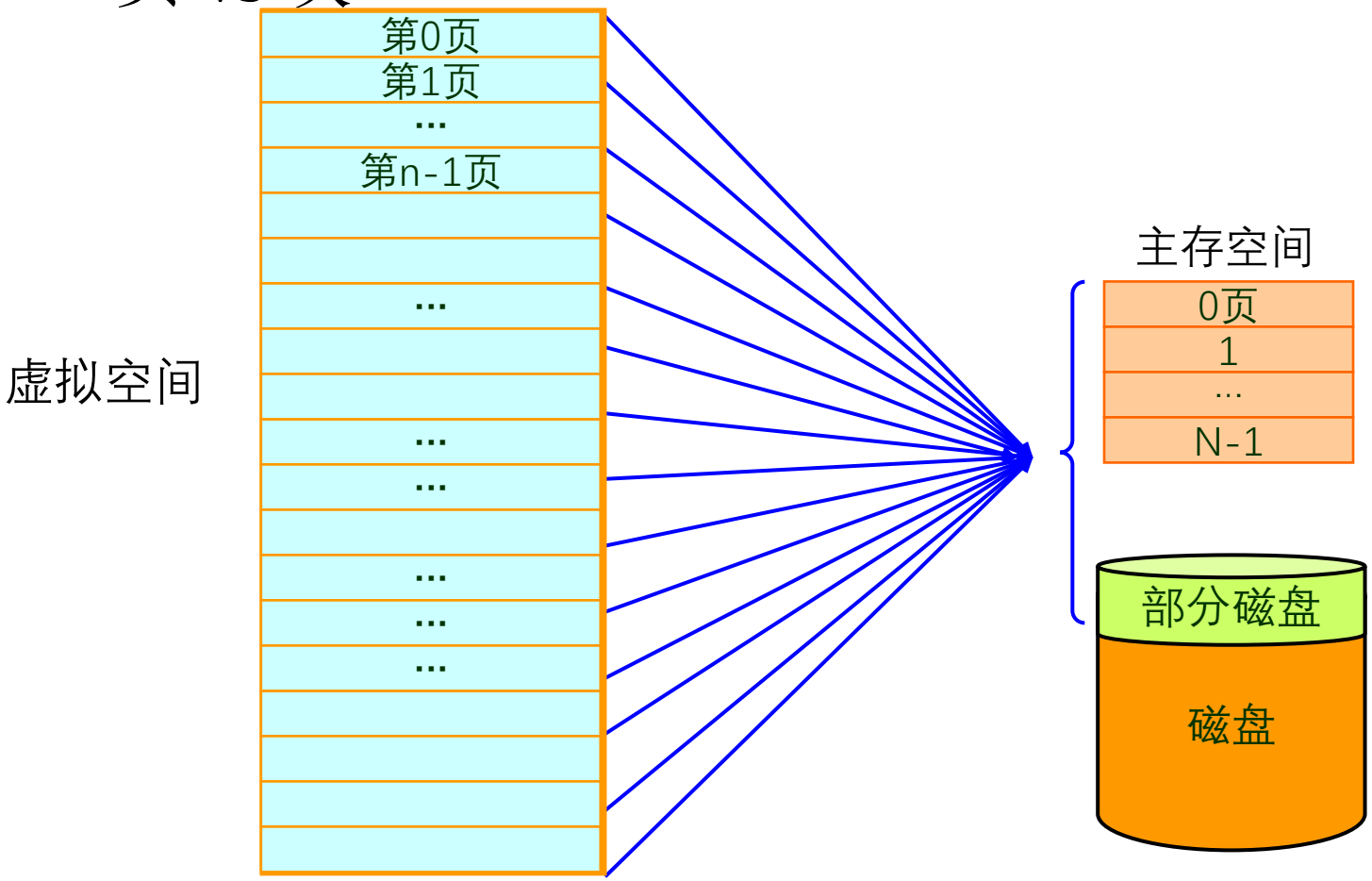
虚页号	页内偏移

实地址：

实页号	页内偏移

虚拟存储器

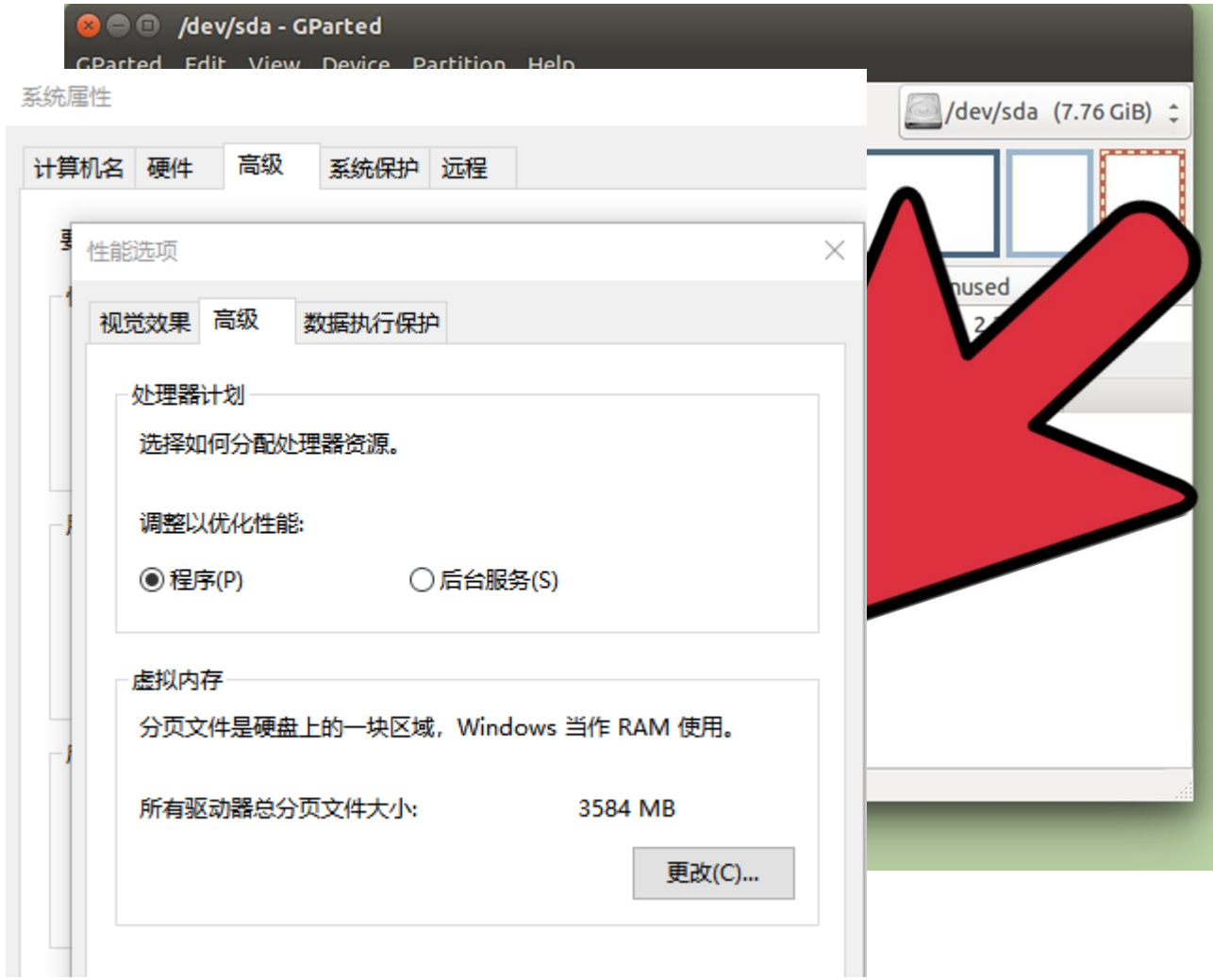
- 页式虚拟存储器：以页（page）为逻辑结构划分信息传送单位的虚拟存储器
- 页 vs 块



- 32位计算机，虚拟地址空间4GB，若内存空间1GB、页大小4KB。
- 虚拟空间总共有：4GB/4KB=2^{20}页。
- 主存空间总共有：1GB/4KB= 2^{18}页。
- **虚拟地址空间为辅存（磁盘）的部分空间，进程执行的代码和数据都放在辅存中，仅有需要执行的一部分被放入主存。**

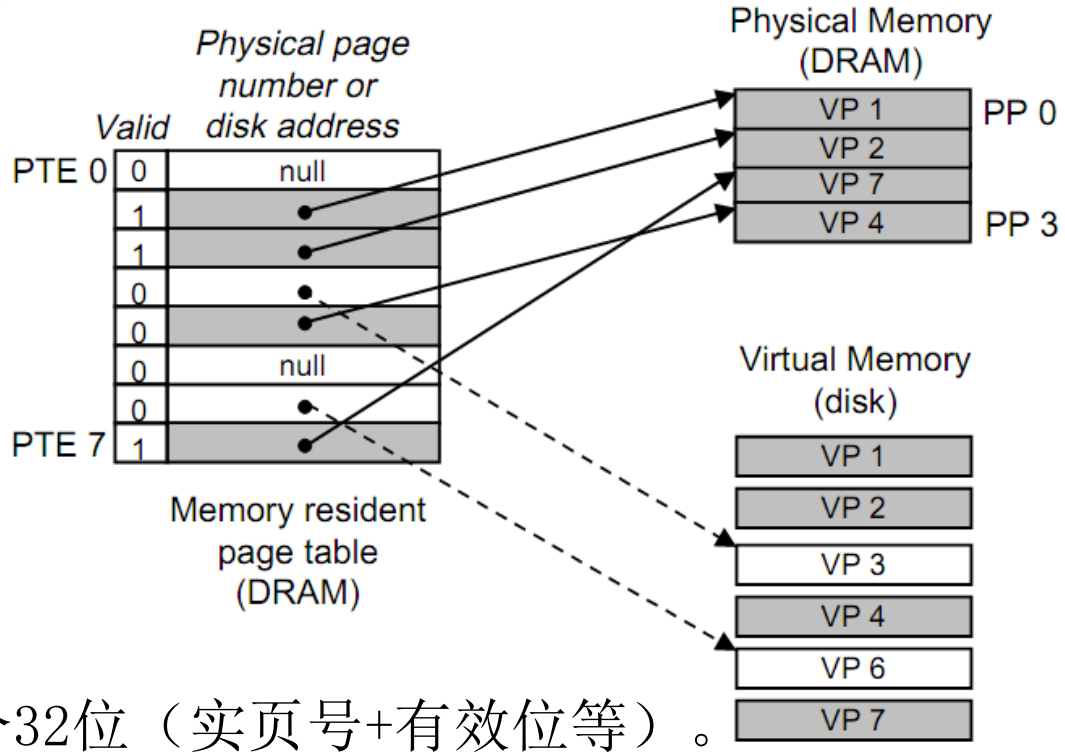
虚拟存储器

- Linux
  - 交换分区swap
- Windows
  - 页面文件



虚拟存储器：页表结构

- Page Table(页表，查找表)：一张保存虚拟页号和物理页号对应关系的查找表。常驻内存
- 虚拟地址→物理地址（页命中/未命中(缺页)）
- **VA**：Virtual Address　**PA**：Physical Address
- MMU：Memory Mangement Unit
- Page Table Entry：PTE 页表项
- Virtual Page Number：VPN，虚拟页号
- Physical Page Number：PPN，物理页号

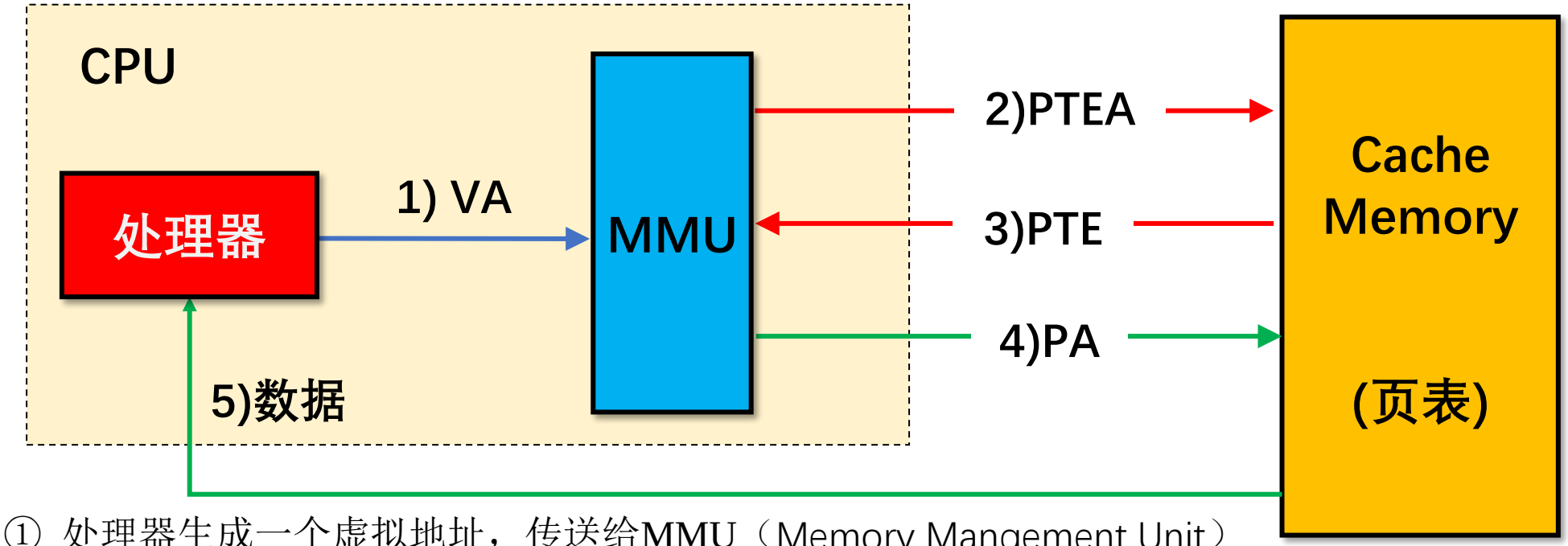


以32位计算机，虚拟地址空间4GB，
内存空间1GB，页大小4KB为例：

- 页表大小？ 4GB/4KB=2^{20}个页表项，每一个32位（实页号+有效位等）。
- 页表存放哪里？主存中，占用主存$2^{20}*2^{5}$=32Mb=4MB的空间。
- 页表有多少个？1个或多个（多级页表），每个进程都有一张完整的页表。

虚实地址转换中存在的问题

- 页式虚拟存储器的访问流程（页面命中）



需要访问主存

① 处理器生成一个虚拟地址，传送给MMU（Memory Mangement Unit）

② MMU利用页表基址寄存器PTBR和虚页号生成页表项地址PTEA，访问存放在Cache/主存中的页表，请求与虚拟页号对应的页表项PTE

③ Cache/主存向MMU返回页表项PTE，以构成访问信息的物理地址PA

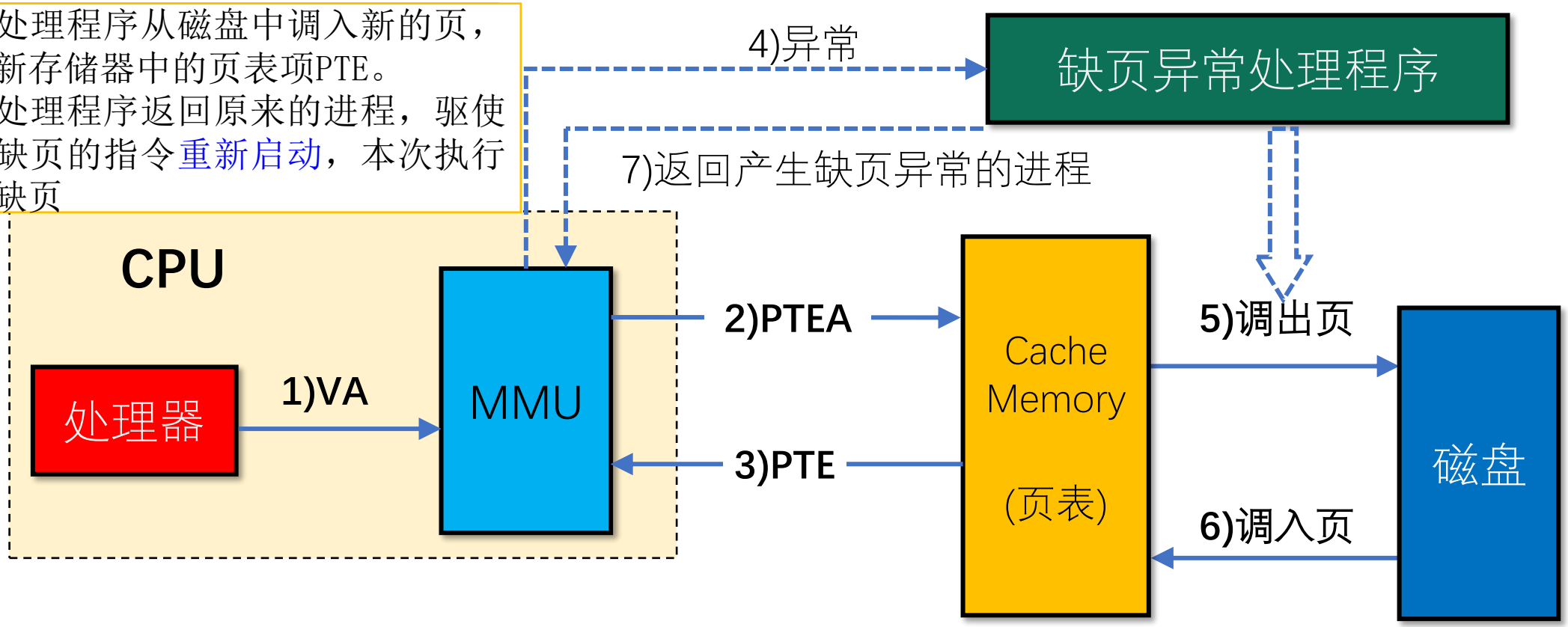
- 若返回的PTE中有效位为1，则MMU利用返回的PTE构造物理地址PA，并利用构造出的物理地址PA访问Cache/主存

④ Cache/主存返回所请求的数据给CPU

虚实地址转换中存在的问题

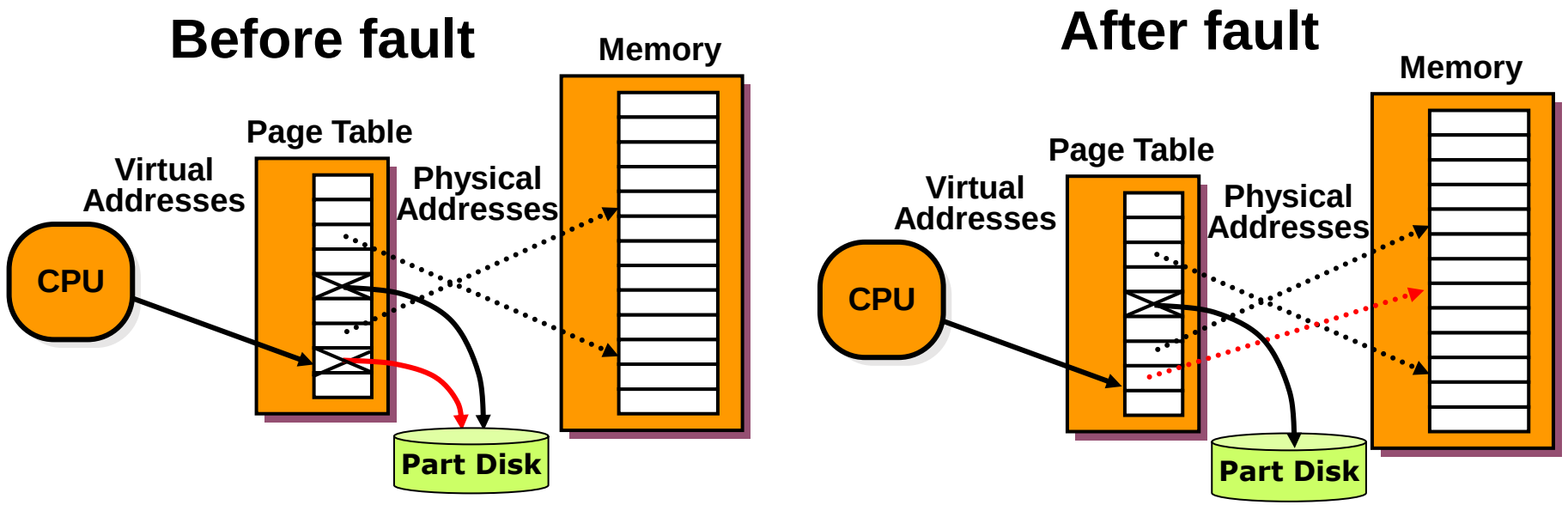
- 页式虚拟存储器的访问流程（**页面缺失**）
  - 虚实地址转换访问主存
  - 从磁盘调入缺失页访问主存
  - 缺页异常处理后再次进行虚实地址转换将再次访问主存

- 缺页处理程序从磁盘中调入新的页，并更新存储器中的页表项PTE。
- 缺页处理程序返回原来的进程，驱使引起缺页的指令重新启动，本次执行不再缺页



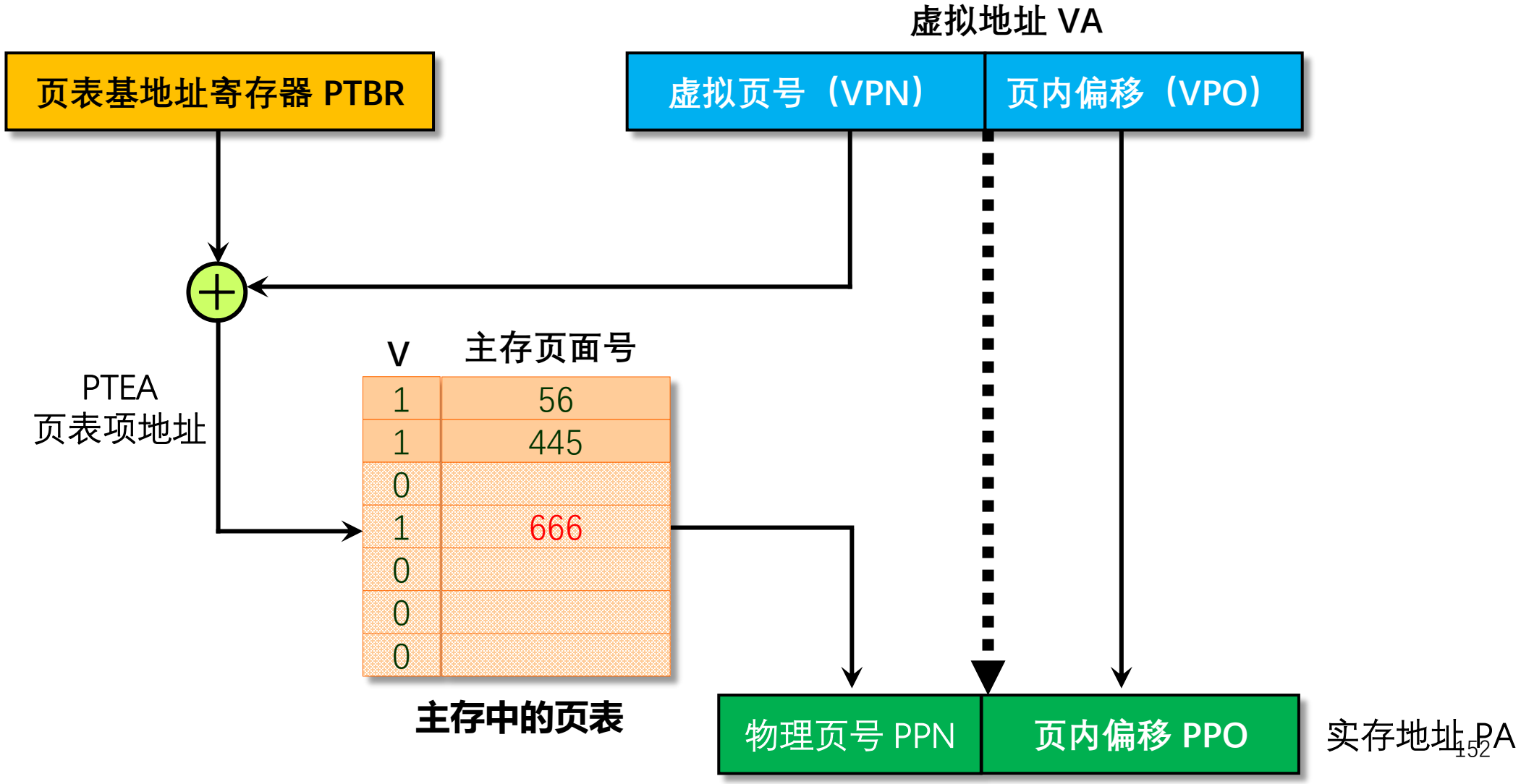
虚拟存储器

- 缺页 Page Faults
  - 页表指示虚拟地址不在内存中
    - 操作系统负责将数据从磁盘迁移到内存中
    - 当前进程挂起
    - 操作系统负责所有的替换策略
    - 唤醒挂起进程



虚拟存储器

- 页式虚拟存储器结构



例子

- 在下表所示的页式虚拟存储页表中，假定页面大小为1024B，求对应于虚拟地址2050和3080的主存地址(10进制)。（最大物理空间为64KB）

页表

0	1	000010
1	1	000110
2	1	000111
3	0	000100
..	.	.....

$(2050)_{10} = 2048 + 2 = (10\ \ 0000000010)_2$

虚页号为 2 ,查页表可得到物理页号为 000111,则对应的物理地址为:

000111 0000000010

$(3080)_{10} = 2048 + 1024 + 8 = (11\ \ 0000001000)_2$

虚页号为3,查页表可知物理页号 缺失，虚页号3对应的页无效

页式管理

- 信息交换的单位是定长的页
- 主存与外存均划分成等长的页面
- 便于维护，便于生成页表，类似于cache的块表
- 不容易产生碎块，存储空间浪费小
- 页不是独立的实体，处理、保护和共享不方便

段式管理

- 段的分界与程序的自然分界相对应，段长不固定，因程序而异
- 段的独立性—易于编译、管理、修改和维护，也便于多道程序共享
- 各段的长度不同，给主存的分配带来麻烦
- 段式管理容易产生碎块，浪费主存空间

段页式管理

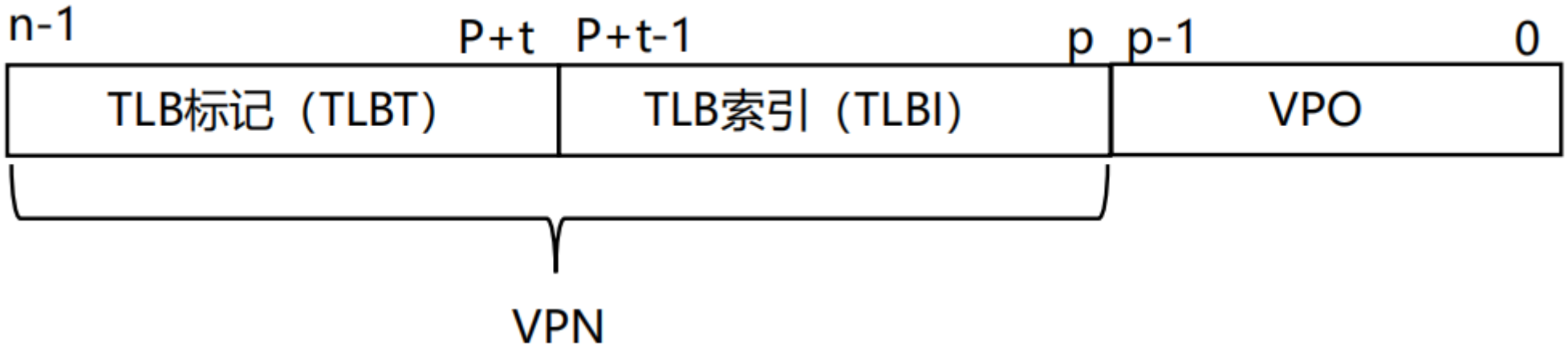
- 程序按模块分段
- 段再分成长度固定的页
- 程序调入调出按页面来进行
- 程序共享保护按段进行
- 兼备段式，页式管理的优点
- 在地址映像中需要多次查表

TLB——快表提高页式管理速度

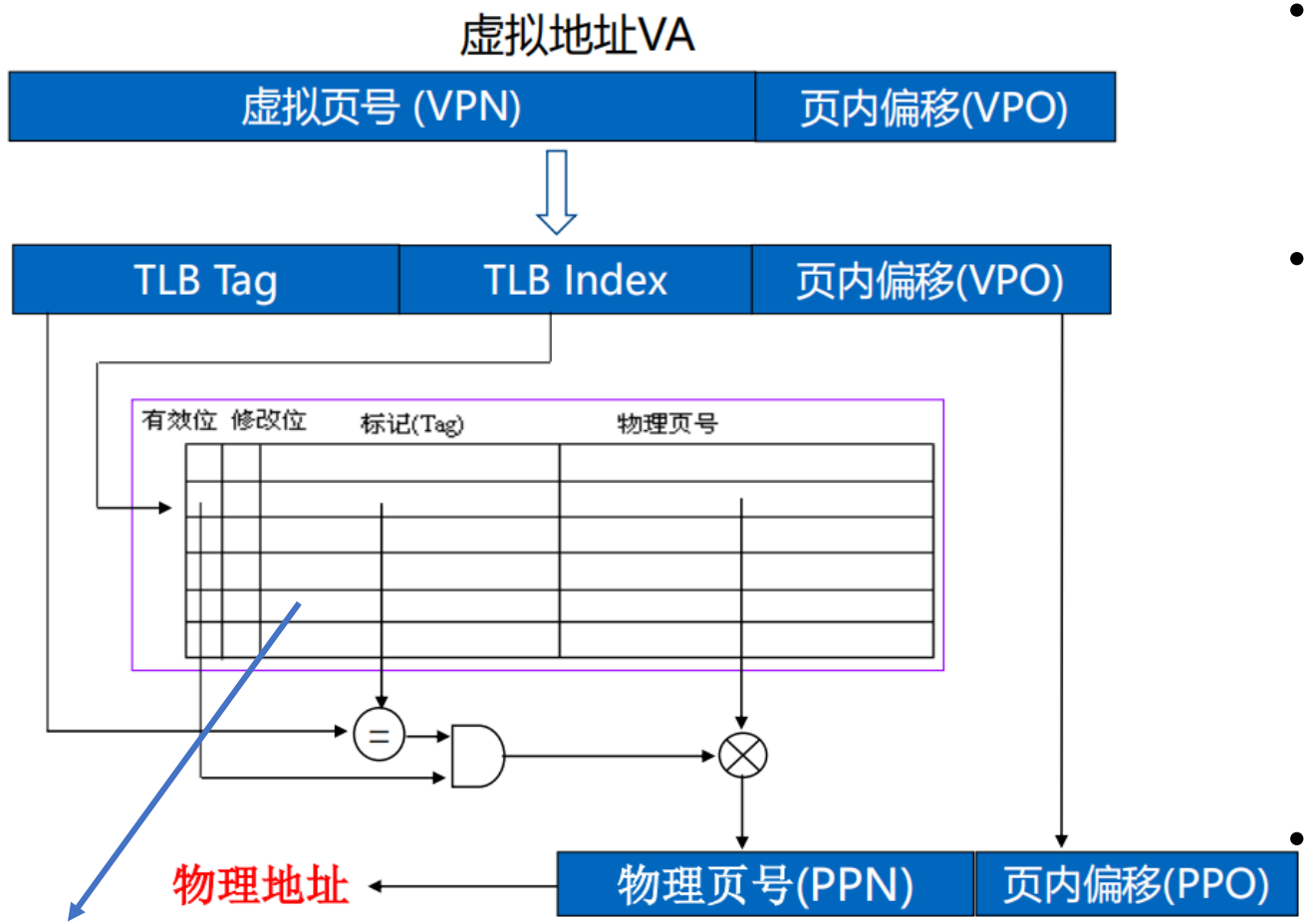
- 页表存放在主存，虚存空间即使命中也须先查页表，再访问主存，两次访存才能取出数据，速度慢
- 为提高页表查找速度，采用Cache的方法，引入一个体积小的快表，存放页表中经常被访问的表项
- **TLB(Translation Lookaside Buffer，地址转换后备缓冲器)：快表，页表缓冲**
- CPU寻址时，使用虚拟地址，优先在TLB中寻址，TLB负责将虚拟内存地址翻译成实际的物理内存地址，从而可**加速主存和辅存间访问速度**

TLB

- 为实现对TLB的快速访问，类似于Cache中的映射方法，对来自于CPU的虚页号进行逻辑划分，得到相应的标识和数据。标识中存放的是虚地址的一部分，而数据部分中存放物理页号、存储保护信息以及其他一些辅助信息。
- CPU寻址时，使用虚拟地址,优先在TLB中进行寻址，TLB负责将虚拟内存地址翻译成实际的物理内存地址。



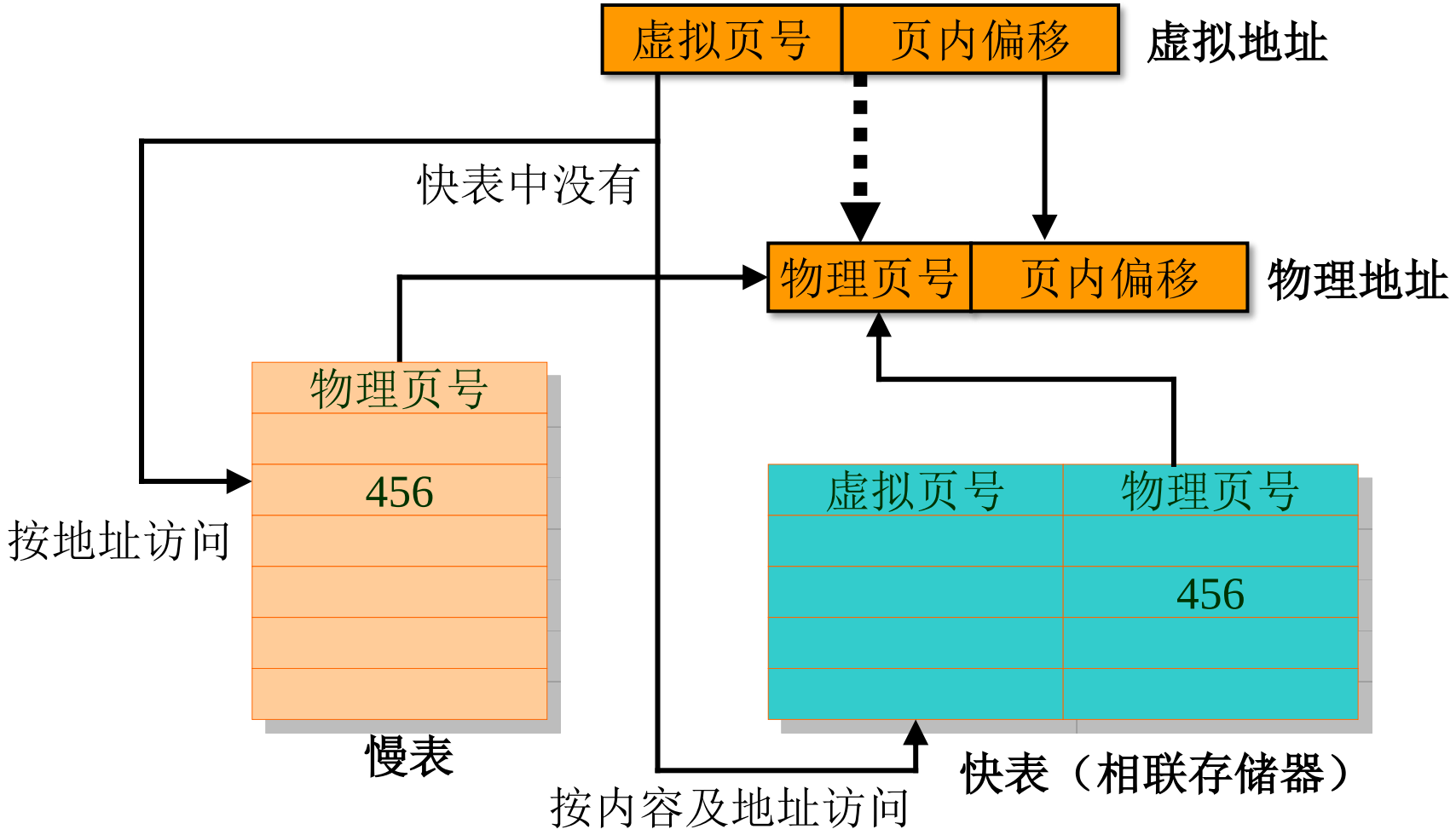
TLB虚实地址转换



TLB中的页表项和主存中的页表项的差异在于多出了一个Tag标记，用于区分不同的虚拟页。

- 采用TLB后，对虚拟地址进行二次划分，主要是把VPN划分为TLB Tag和TLB index，页内偏移地址不变。
- 当CPU传送一个新的虚拟地址后，从虚拟地址中划分出TLB Tag和TLB index，使用TLB index找到快表中对应的页表项，比较当前虚拟地址的TLB Tag和页表项中存储的TLB Tag是否相同。**如果相同且有效位为1，则快表命中，直接基于页表项中存储的物理页号生成物理地址；否则，快表未命中，需要访问主存中的页表。**
- TLB中存放的是页表的子集，取决于TLB Index的位数。因此，不同的虚拟页可能会有相同的TLB Index，需要通过TLB Tag来区分。这类似于Cache的直接相连映射。

TLB——经快慢表实现内部地址转换

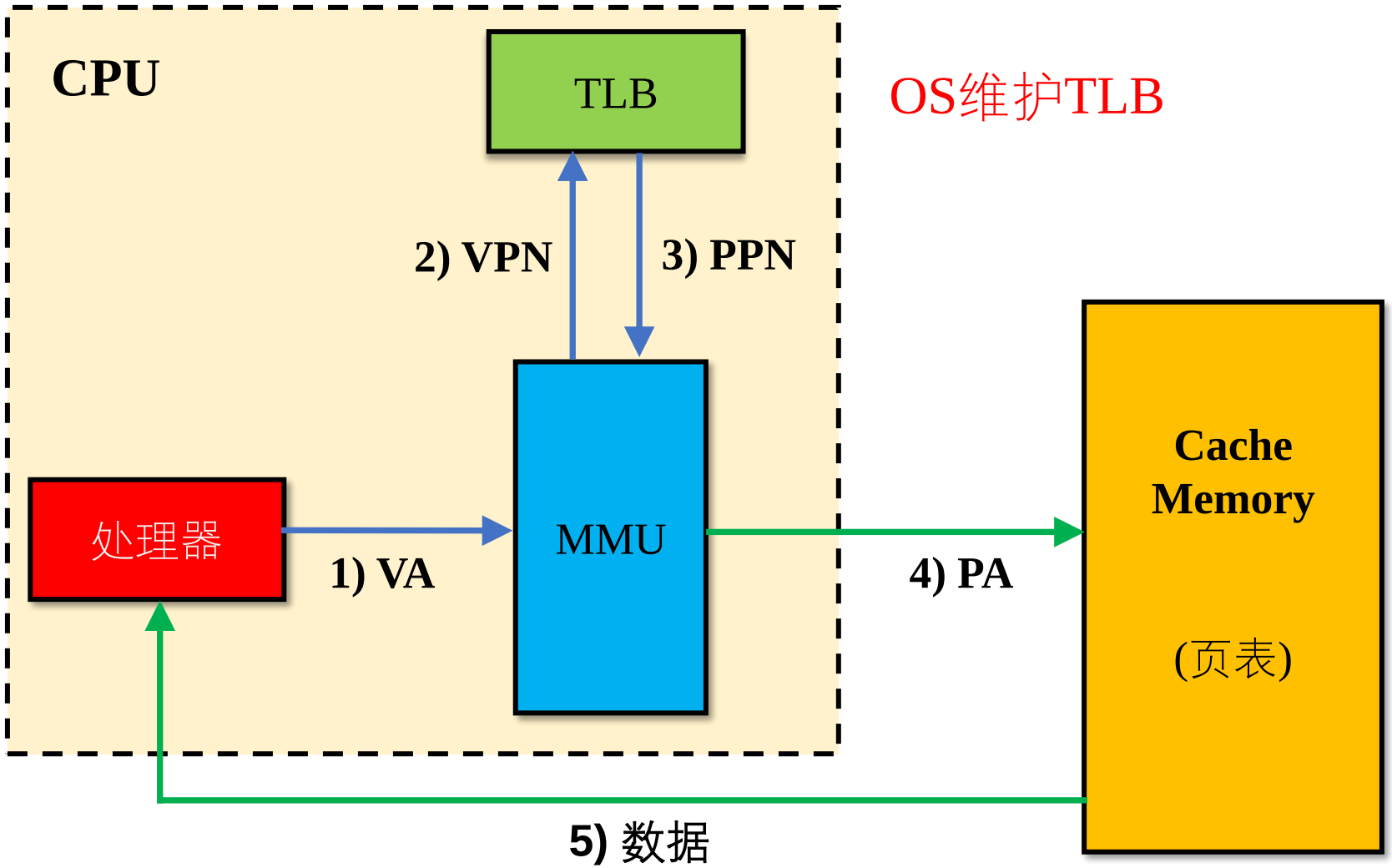


TLB命中

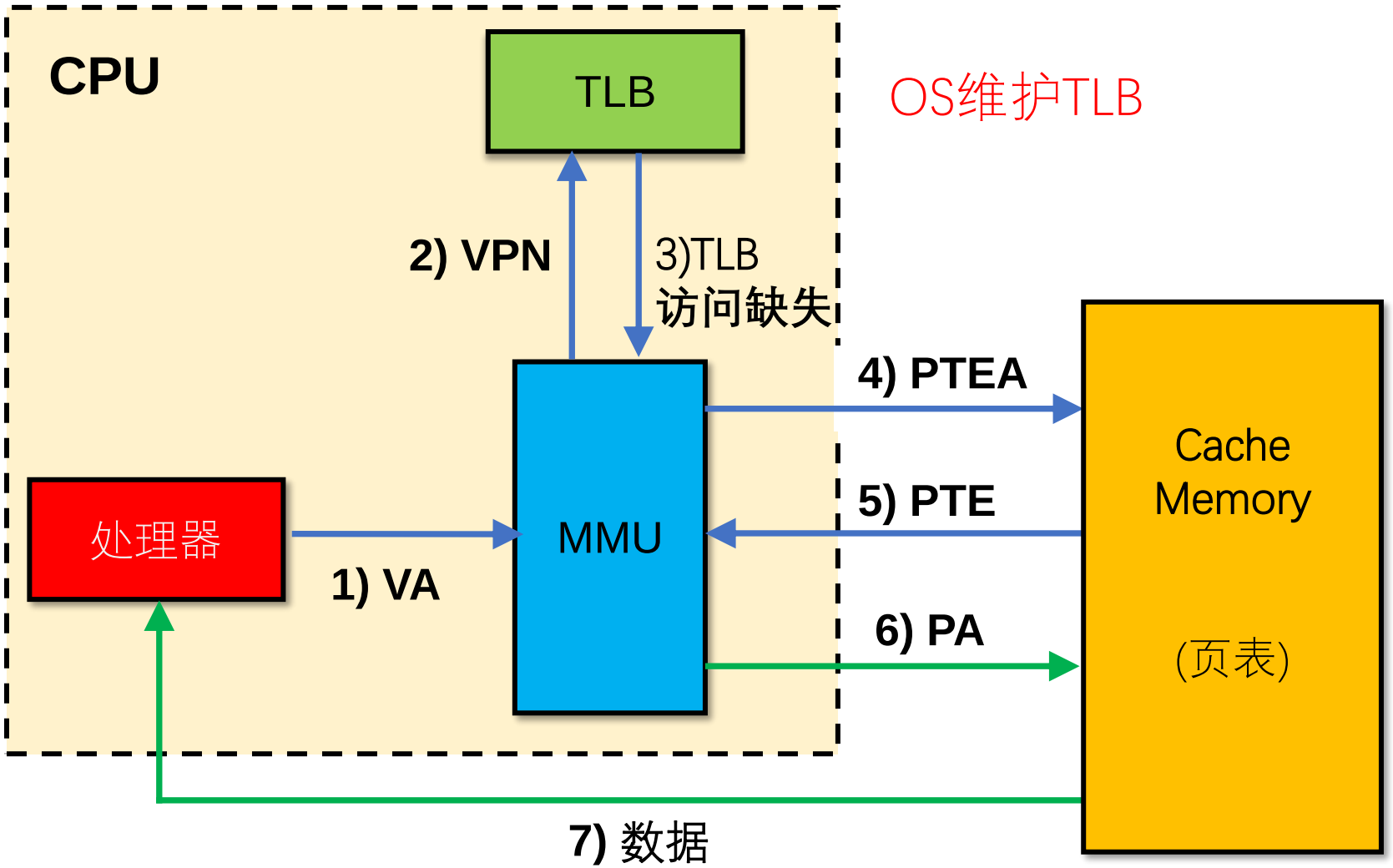
- CPU生成虚拟地址，传送给MMU
- MMU利用虚页号VPN查询TLB

如果TLB访问命中，即页表项在TLB中且相应的PTE中有效位为1，则TLB向MMU返回与VPN对应的物理页号PPN

- MMU利用返回的PPN构造物理地址PA，利用PA访问Cache/主存
- Cache/主存返回CPU请求的数据给CPU

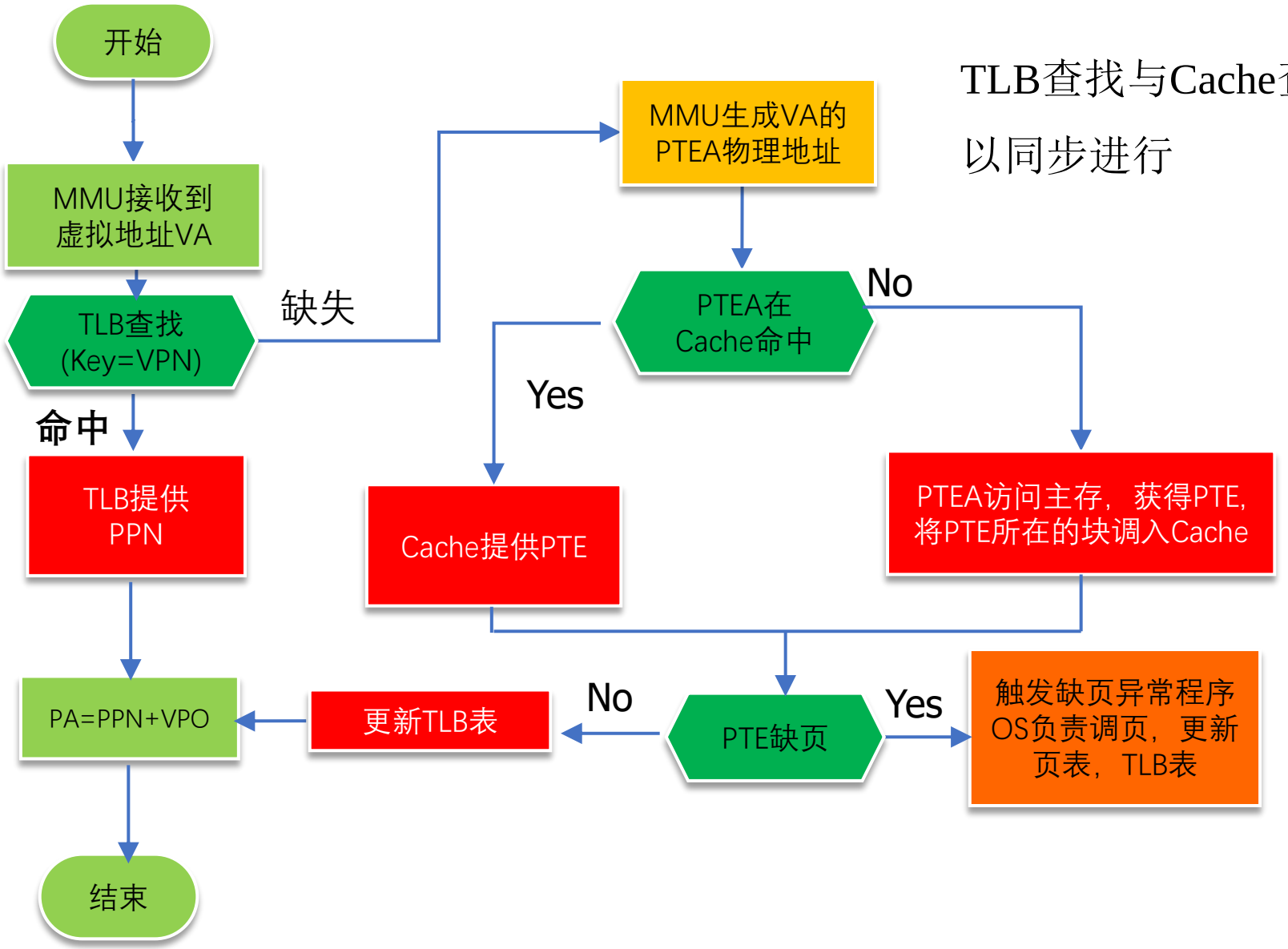


TLB缺失



- MMU利用虚页号VPN查询TLB 如果TLB访问未命中，则MMU返回TLB访问缺失信息
- MMU利用页表基址寄存器PTBR和虚页号生成页表地址PTEA，访问存放在Cache/主存中的页表，请求与虚拟页号对应的页表项内容。
- Cache/主存向MMU返回页表项PTE，构成所访问信息的物理地址PA
- 若返回的PTE中有效位为1，则需要更新TLB表，同时利用返回的PTE构造PA，并利用构造出的PA访问Cache/主存
- Cache/主存返回CPU请求的数据给CPU

虚拟地址→物理地址流程



TLB查找与Cache查找可以同步进行

虚拟存储器与Cache的相似之处

- 将程序中常用的部分驻留在高速存储器
  - 程序载入按需载入（不是全部载入）
  - 高速存储器分块或者分页（粒度问题）
  - 主存空间满，将不常用程序或数据淘汰或交换到辅存中
- 数据的换入换出由硬件或操作系统完成，对用户透明
- 利用程序局部性，使存储系统的性能接近于高速存储器；价格接近于低速存储器，并扩充主存容量

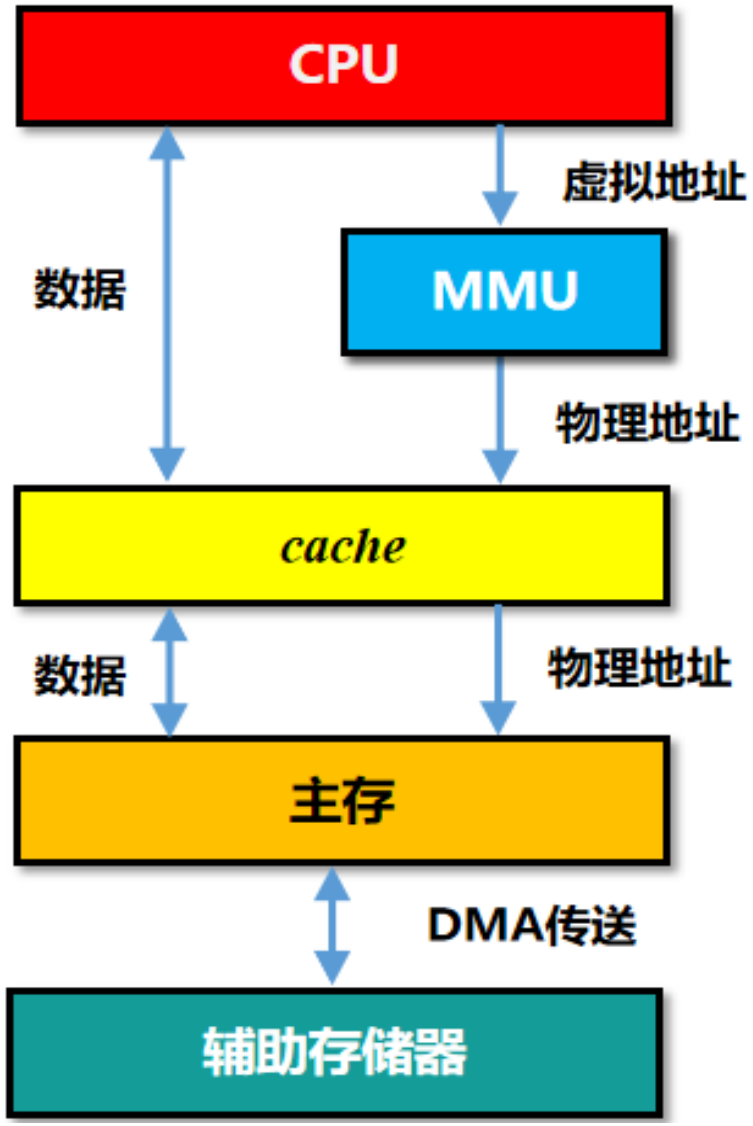
虚拟存储器与Cache的差异

- 虚存用于扩大主存容量，Cache用于加速主存性能
- 虚存中未命中性能损失远大于Cache系统
  - 全相联提升命中率
  - 更大的交换单位页
  - 近似LRU算法 (CLOCK算法)
- 虚存由硬件和OS联合管理，Cache由硬件管理

虚拟存储器小结

- 程序员在比实际主存大得多的逻辑地址空间中编程
- 程序执行时，按需载入代码和数据，无需全部载入
- 硬件将逻辑地址（虚拟地址）转化为物理地址（实地址）
- 缺页时，由操作系统进行主存和磁盘之间的信息交换
- 虚存机制由硬件与操作系统协作实现
  - 进程及进程上下文切换、存储器分配
  - 虚拟地址空间管理、缺页处理
  - 存储器保护

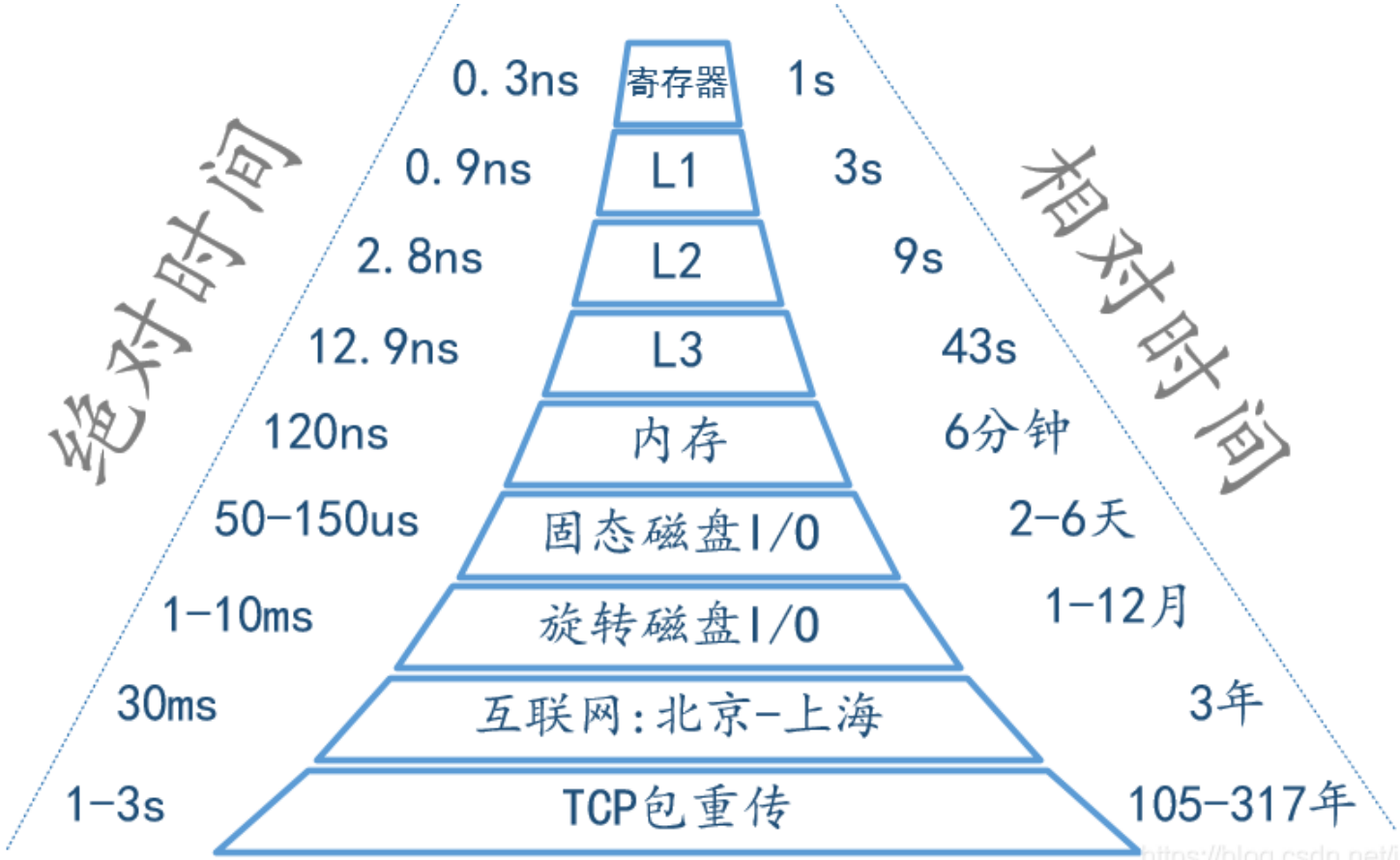
存储层次结构



计算机各种延时示意图

洞悉系统、企业与云计算

https://blog.csdn.net/jizhu4873/article/details/84341884



单选题 1分

设置

下列关于缺页处理的叙述中，错误的是（）

提交

A 缺页是在地址转换时CPU检测到的一种异常

B 缺页处理由操作系统提供的缺页处理程序完成

C 缺页处理程序根据页故障地址从外存读入所缺失的页

D 缺页处理完成后回到发生缺页的指令的下一条指令执行

单选题 1分

设置

Cache由SRAM组成，TLB通常由相联存储器组成，也可由SRAM组成

下列关于TLB和Cache的叙述中，错误的是（）

A 命中率都与程序局部性有关

B 缺失后都需要去访问主存

C 缺失处理都可以由硬件实现

D 都由DRAM存储器组成

提交

第六章 存储器

6.1 存储器概述

6.2 主存储器

6.3 高速缓冲存储器

6.4 虚拟存储器

6.5 辅助存储器

6.5辅助存储器的主要类型

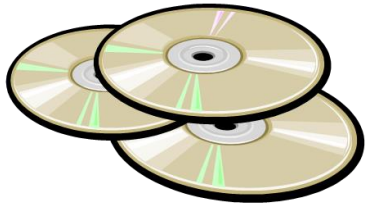
硬盘

U盘...

软盘

光盘

磁带



6.5辅助存储器——RAID

RAID (Redundant Array of Inexpensive Disks）

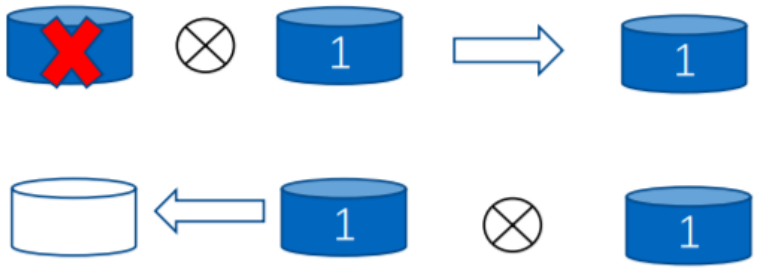
（Redundant Array of Independent Disks)

- **UC Berkeley David A. Patterson, 1988**
- **替代SLEDs（Single Large Expensive Disiks）**
- **磁盘访问速度过慢**
- **可靠性**
- **多磁盘管理不方便**

6.5辅助存储器——RAID的核心技术

- 将数据条带化后存放在不同磁盘上，通过多磁盘的并行操作提高磁盘系统的读写速率
- 使用基于异或运算为基础的校验恢复损坏的数据

$1 \oplus 0 = 1$	$1 = 0 \oplus 1$
$1 \oplus 1 = 0$	$1 = 1 \oplus 0$
$0 \oplus 1 = 1$	$0 = 1 \oplus 1$
$0 \oplus 0 = 0$	$0 = 0 \oplus 0$

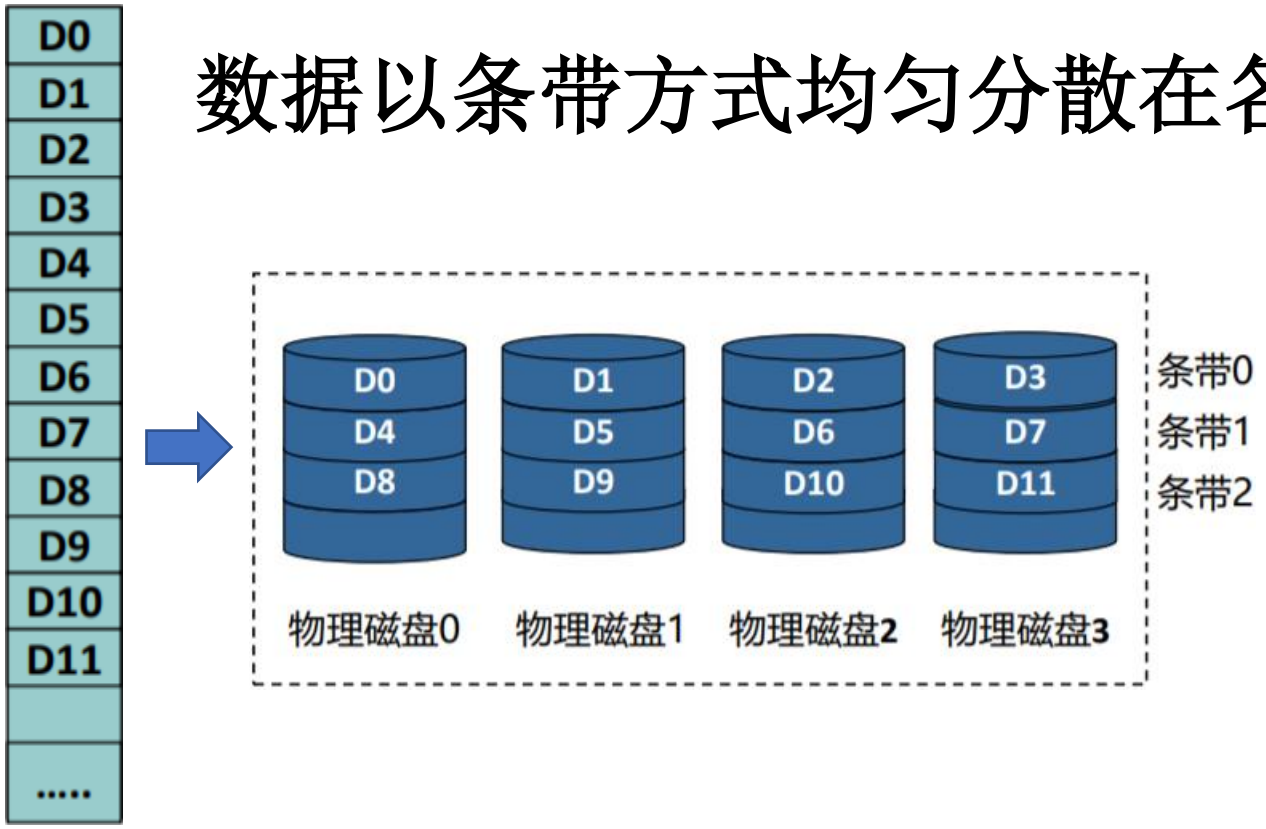


6.5辅助存储器——RAID 0

3.常见的几种RAID技术

- **RAID 0**

数据以条带方式均匀分散在各磁盘

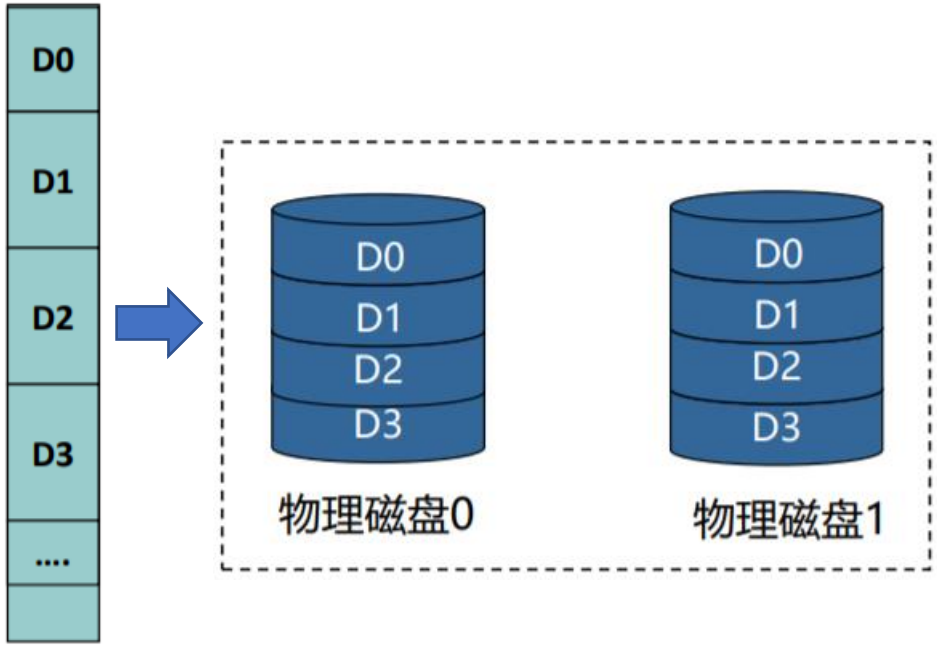


所需磁盘数	2个或更多
优点	磁盘读写效率高 无校验带来使用和配置方便
缺点	无冗余，数据安全性低
适用领域	视频、图像及需高传输带宽的应用

6.5辅助存储器——RAID 1

RAID 1 镜像磁盘阵列

数据采用镜像的冗余方式，同一数据有多份拷贝



所需磁盘数	至少2个
优点	100%数据冗余，数据安全性高 理论上可以实现2倍的读取效率
缺点	空间利用率只有50%
适用领域	财务、金融等高可用应用

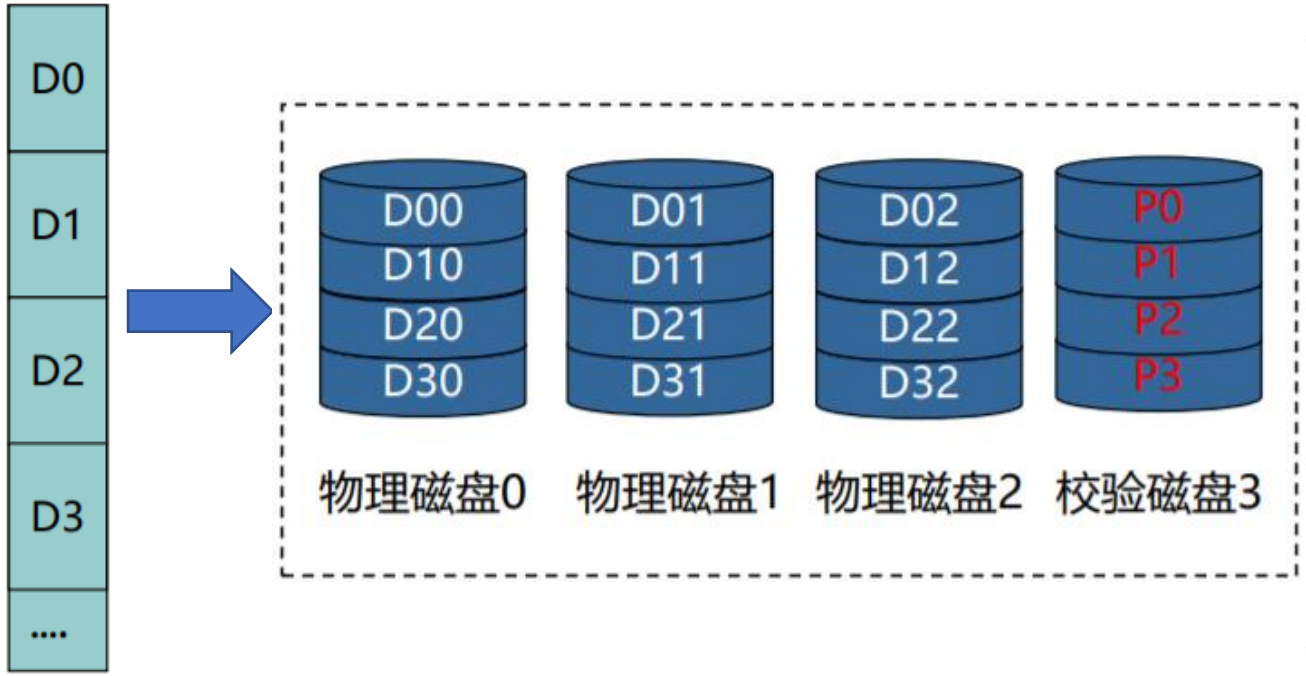
6.5辅助存储器——RAID 2

- **RAID 2，纠错海明码磁盘阵列。**
- **磁盘驱动器组中的第一个、第二个、第四个……第2^n个磁盘驱动器是专门的校验盘，用于校验和纠错，其余的用于存放数据。**
- **使用的磁盘驱动器越多，校验盘在其中占的百分比越少**
- **RAID 2对大数据量的输入输出有很高的性能，但少量数据的输入输出时性能不好。RAID 2很少实际使用。**

6.5辅助存储器——RAID 3/4

RAID 3/4　奇校验或偶校验的磁盘阵列。

数据按 位/条带 并行传输到多个磁盘上，同时奇偶校验数据存放到专用校验盘上。

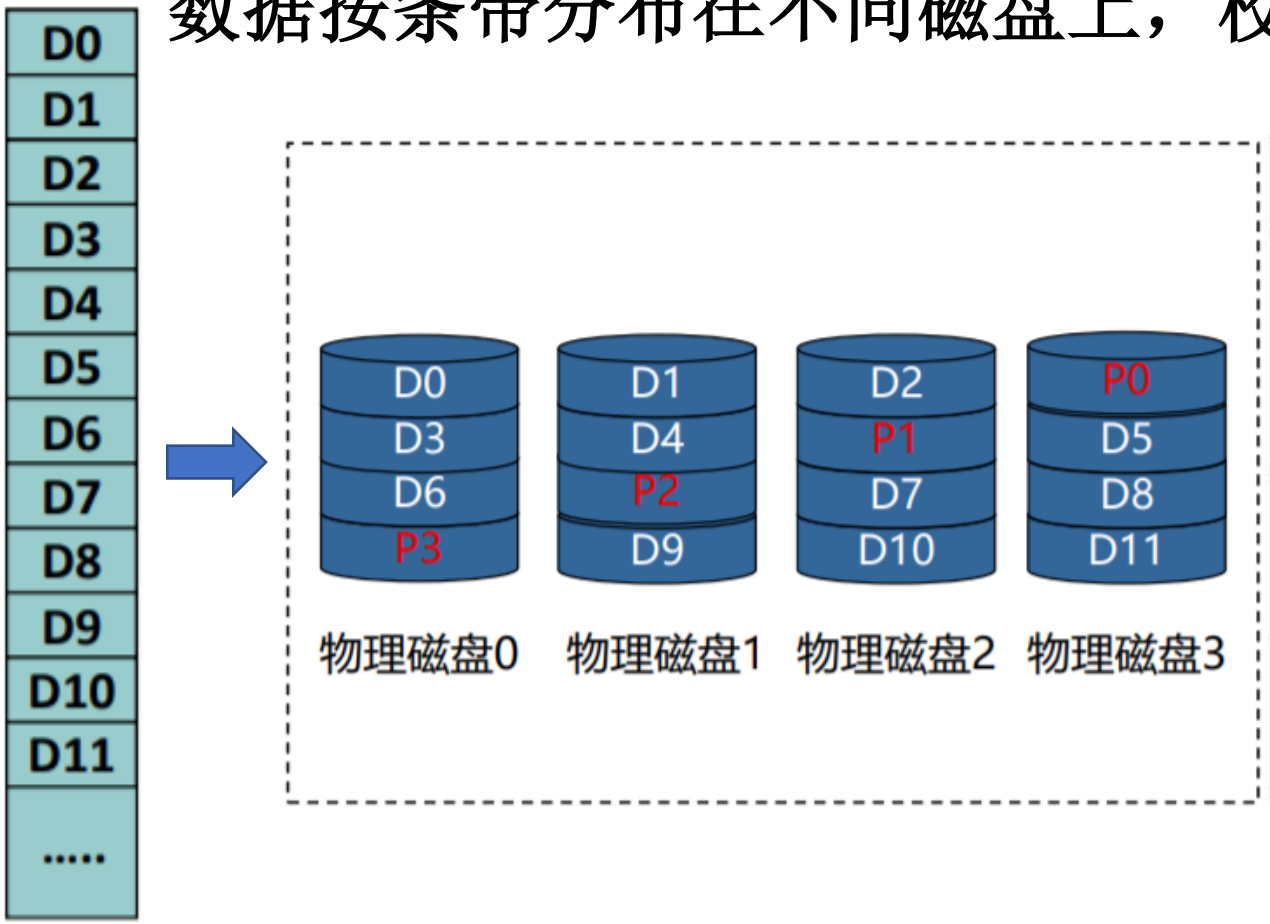


所需磁盘数	至少3块
优点	读写性能都比较好 磁盘利用率高， N-1/N
缺点	控制器设计复杂 校验磁盘的写性能有瓶颈
适用领域	视频生成和图像、视频编辑等 需要高吞吐量的应用环境

6.5辅助存储器——RAID 5

RAID 5 奇校验或偶校验的磁盘阵列

数据按条带分布在不同磁盘上，校验信息被均匀分散到各磁盘上

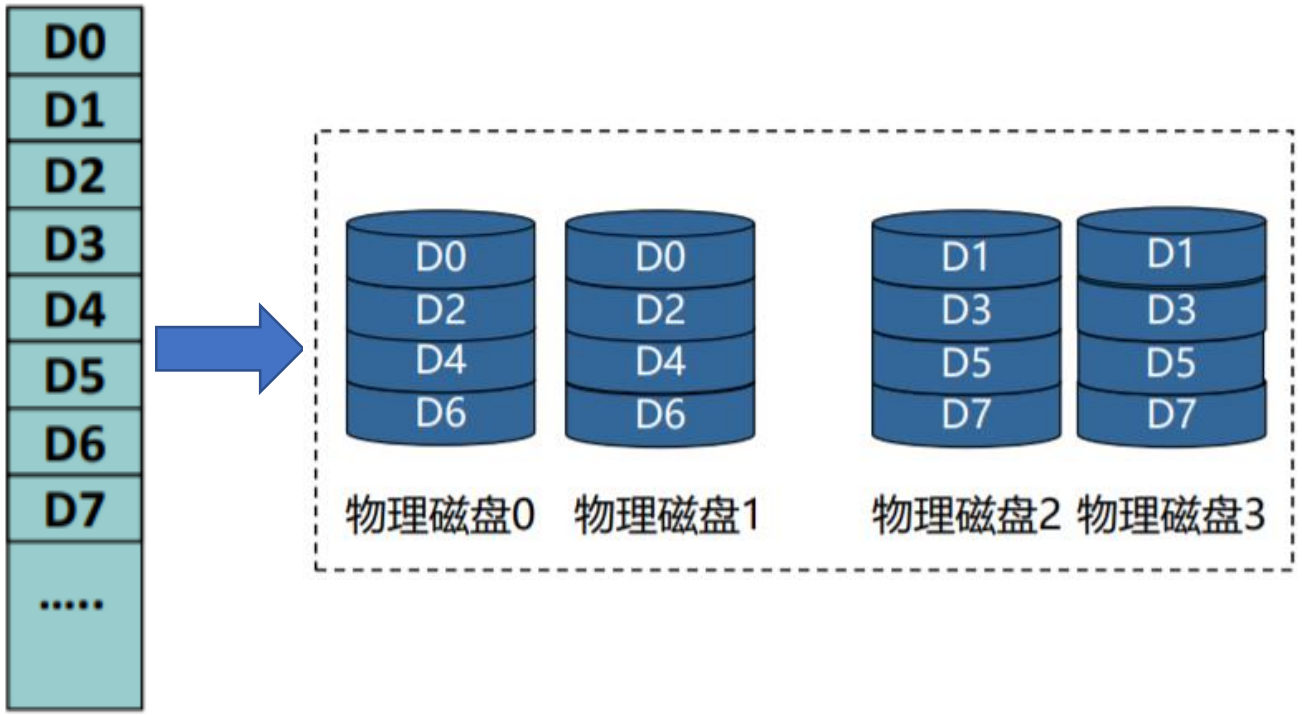


所需磁盘数	最低为3个
优点	读性能比较高 校验信息的分布式存取，避免出现写操作的瓶颈
缺点	控制器设计复杂 磁盘重建的过程比较复杂
适用领域	FTP、Email、Web、数据库

6.5辅助存储器——RAID 1 0

RAID 1 0

结合RAID 1和 RAID 0 ,先镜像，再条带化

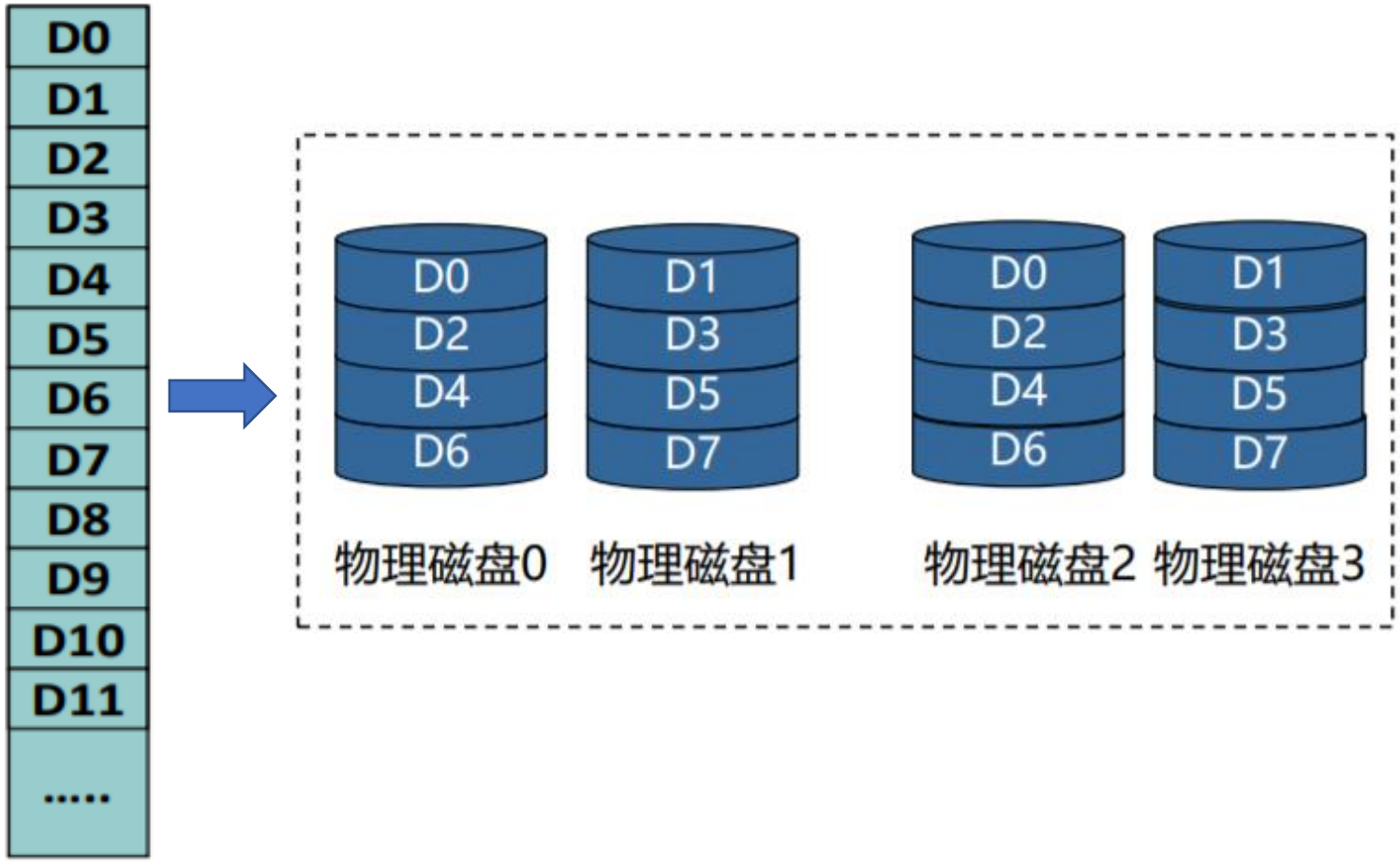


所需磁盘数	最低为4个
优点	读性能高 数据安全性好，允许同时有半数磁盘失效
缺点	空间利用率也只有50%
适用领域	多用于高可用性和高安全性的应用场合

6.5辅助存储器——RAID0 1

- **RAID 0 1**

  结合RAID 0和 RAID 1 ,先条带化，再镜像



6.5辅助存储器——RAID的实现方式

- 软件RAID

  依赖于主机CPU完成，没有第三方的控制处理器和I/O芯片

- 硬件RAID

  专用RAID控制处理器和I/O芯片处理RAID任务，不占CPU

RAID技术小结

RAID级别	RAID0	RAID1	RAID3	RAID5	RAID1 0
容错性	无	有	有	有	有
冗余类型	无	镜像	奇偶校验	奇偶校验	镜像
备盘	无	有	有	有	有
读性能	高	低	高	高	中间
随机写性能	高	低	最低	低	中间
连续写性能	高	低	低	低	中间
需要的磁盘数	2个或更多	2个或2N个	3个或更多	3个或更多	4个或2N (N>2)
可用容量	总的磁盘容量	磁盘容量的50%	磁盘容量的(N-1) /N	磁盘容量的(N-1) /N	磁盘容量的50%

第六章 存储器

6.1 存储器概述

6.2 主存储器

6.3 高速缓冲存储器

6.4 虚拟存储器

6.5 辅助存储器