计算机组成原理

第十三讲

刘松波

哈工大计算学部

模式识别与智能系统研究中心

第 4 章 存 储 器

4.1 概述

4.2 主存储器

4.3 高速缓冲存储器

4.4 辅助存储器

4.1 概 述

一、存储器分类

1. 按存储介质分类

(1) 半导体存储器　　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

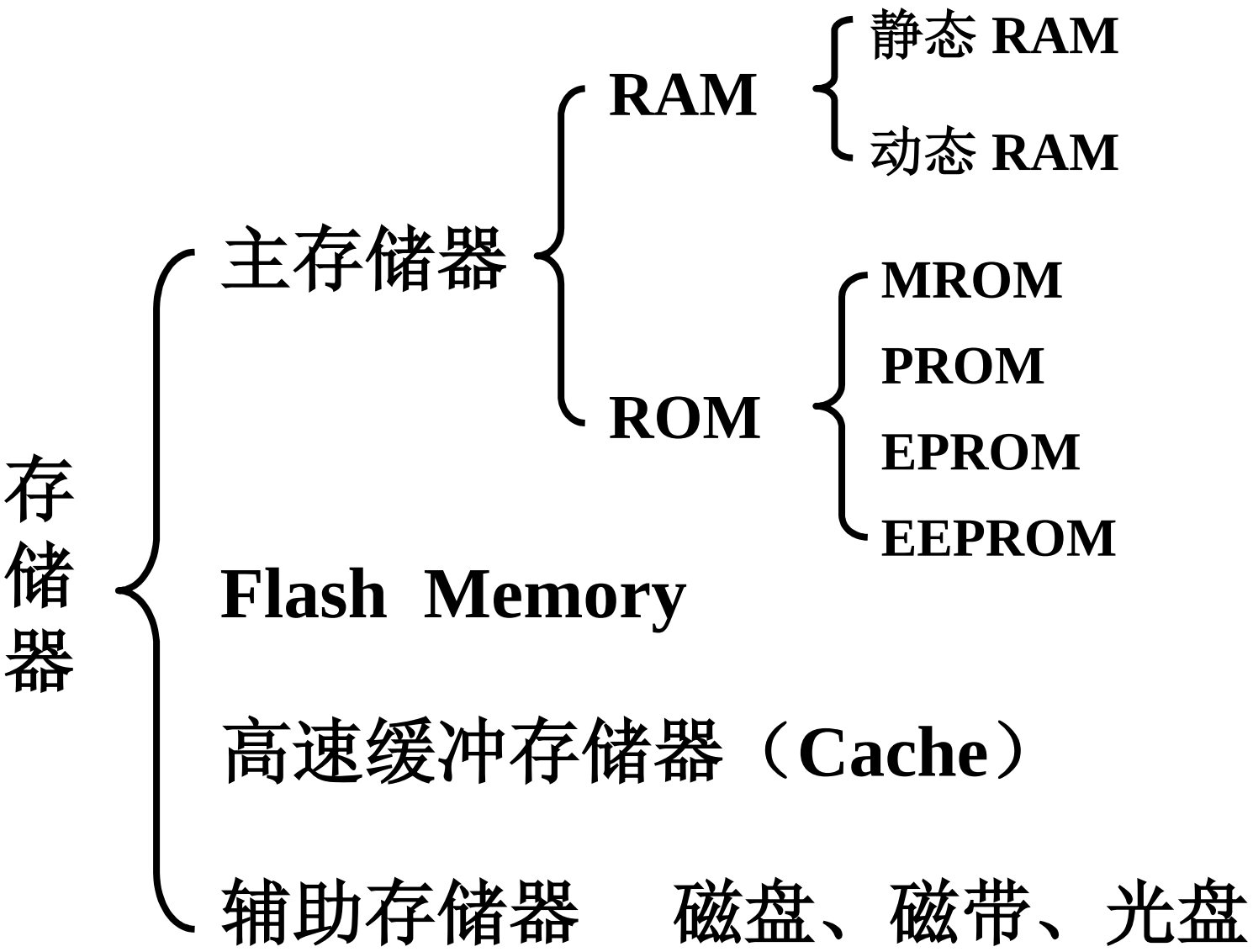
(1) 存取时间与物理地址无关（随机访问）

- 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

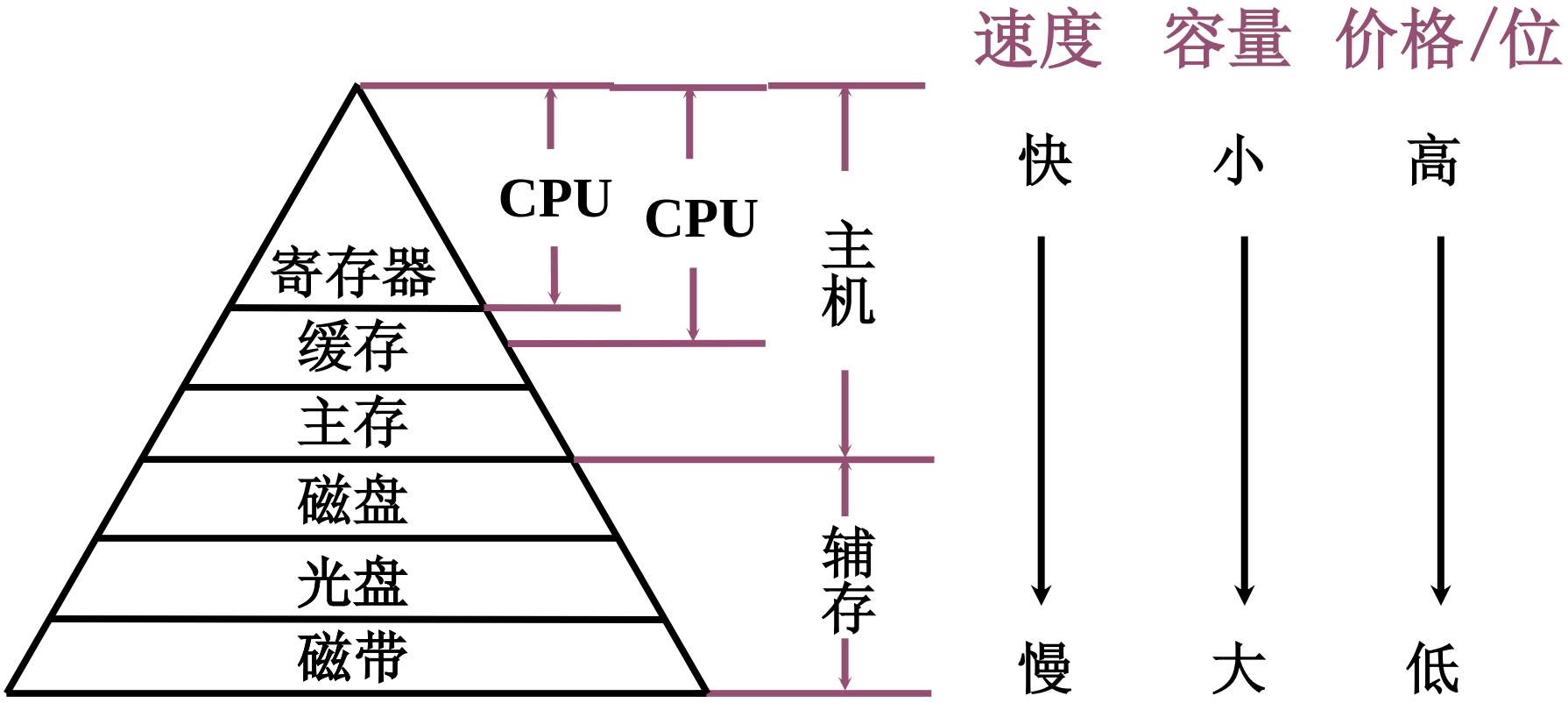
3. 按在计算机中的作用分类



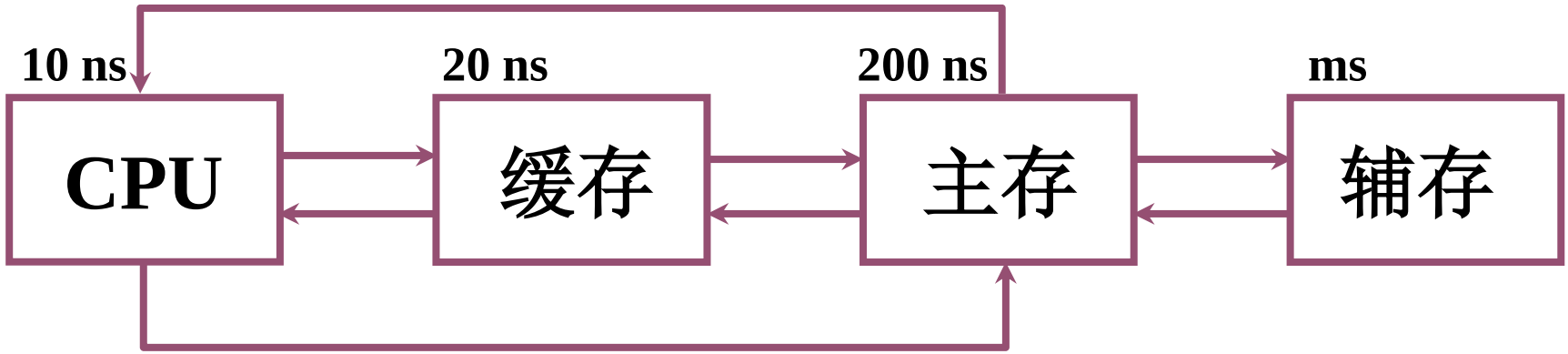
二、存储器的层次结构

4.1

1. 存储器三个主要特性的关系



2. 缓存—主存层次和主存－辅存层次 4.1



（速度）　　（容量）

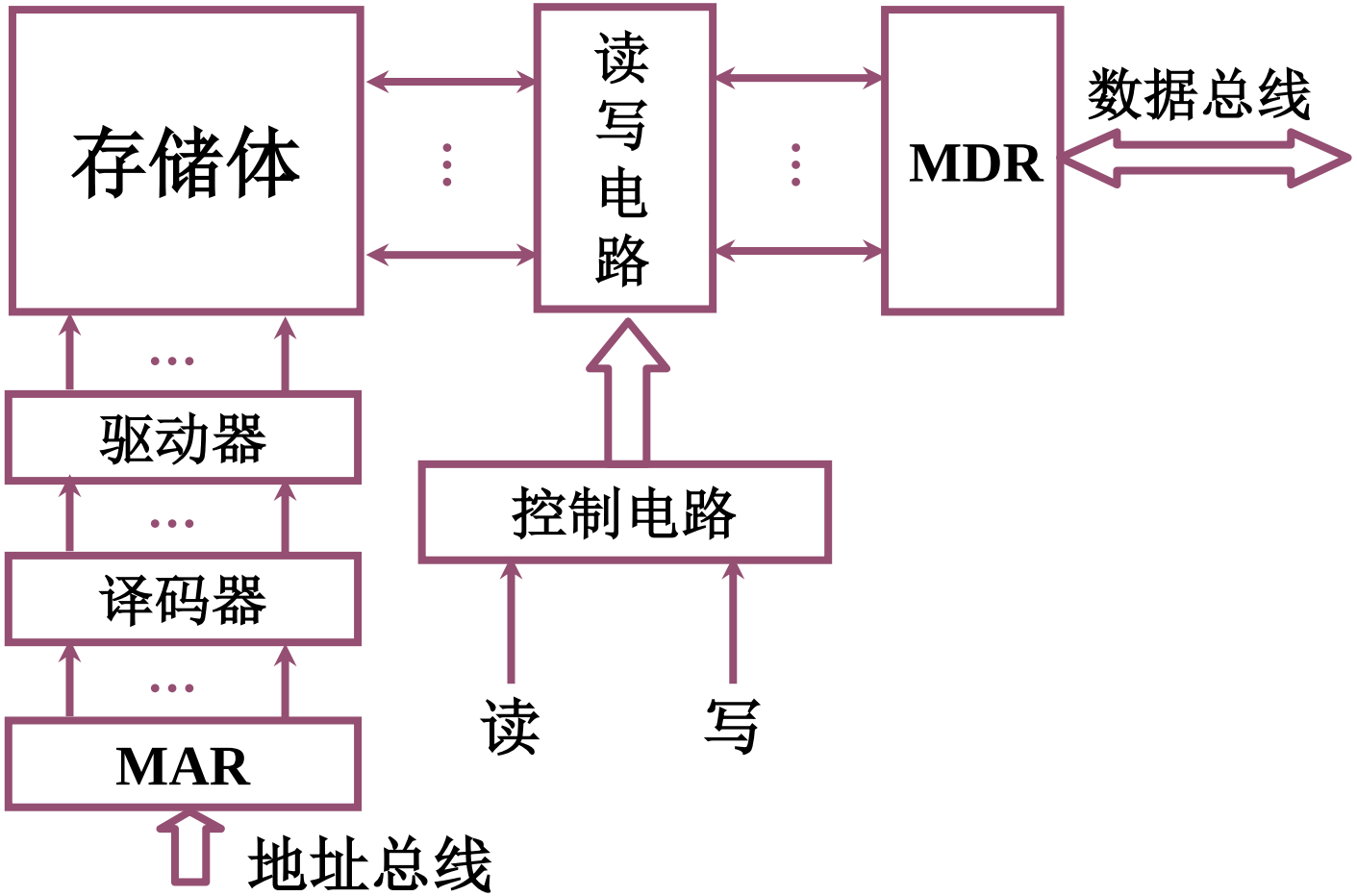
缓存－主存　　主存－辅存

主存储器	虚拟存储器
实地址	虚地址
物理地址	逻辑地址

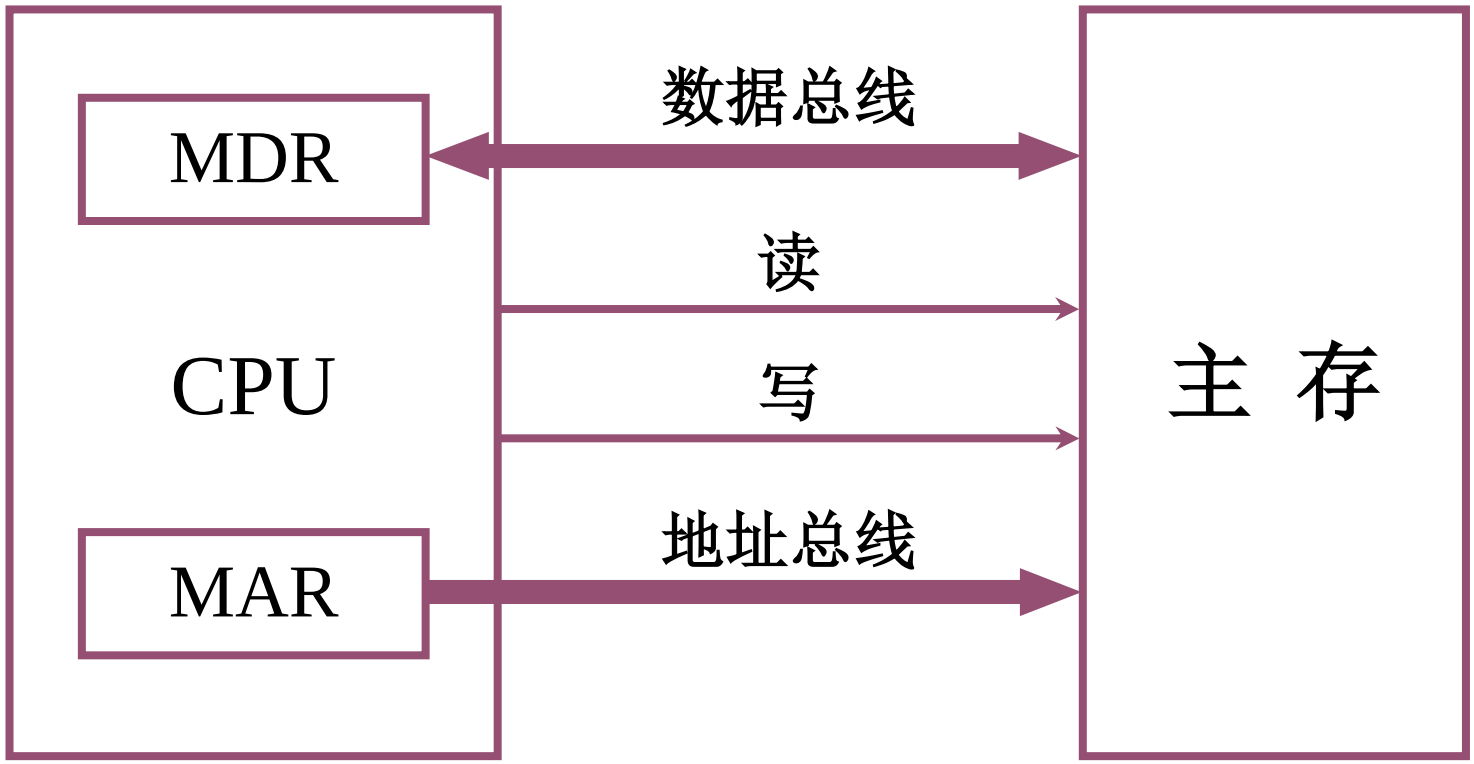
4.2 主存储器

一、概述

1. 主存的基本组成



2. 主存和 CPU 的联系



3. 主存中存储单元地址的分配

高位字节 地址为字地址

字地址	字节地址			
0	0	1	2	3
4	4	5	6	7
8	8	9	10	11

低位字节 地址为字地址

字地址	字节地址	
0	1	0
2	3	2
4	5	4

设地址线 24 根　　按 字节 寻址　2^{24} = 16 MB

若字长为 16 位　　按 字 寻址　8 MW

若字长为 32 位　　按 字 寻址　4 MW

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

(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

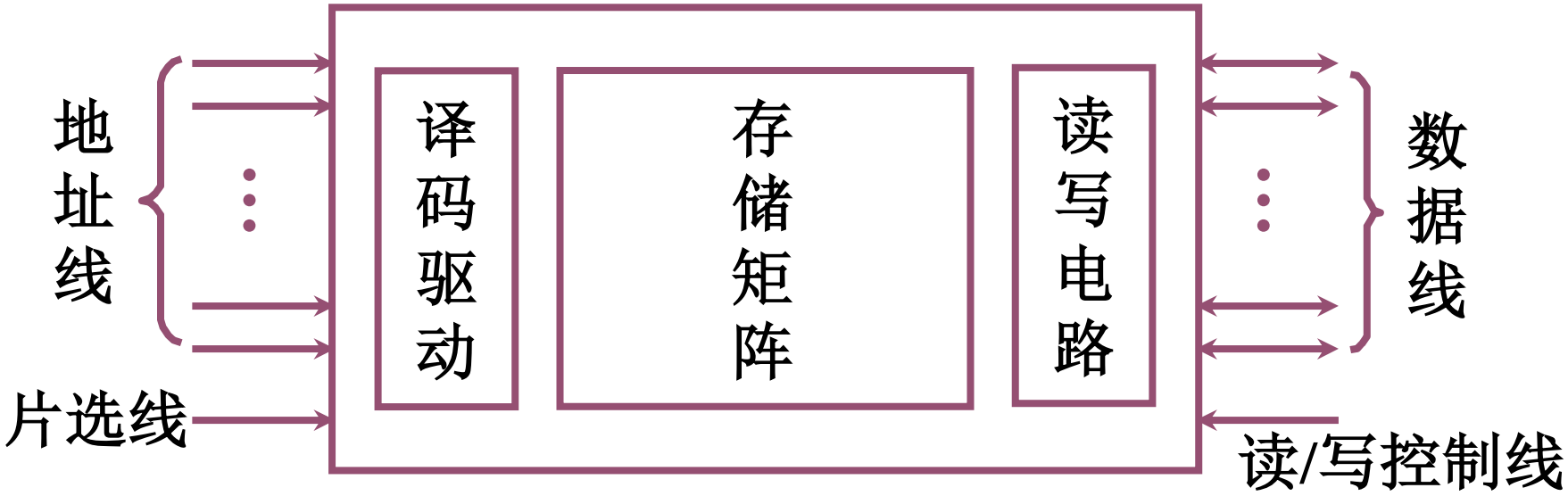
  读周期　写周期

(3) 存储器的带宽　位/秒

二、半导体存储芯片简介

4.2

1. 半导体存储芯片的基本结构

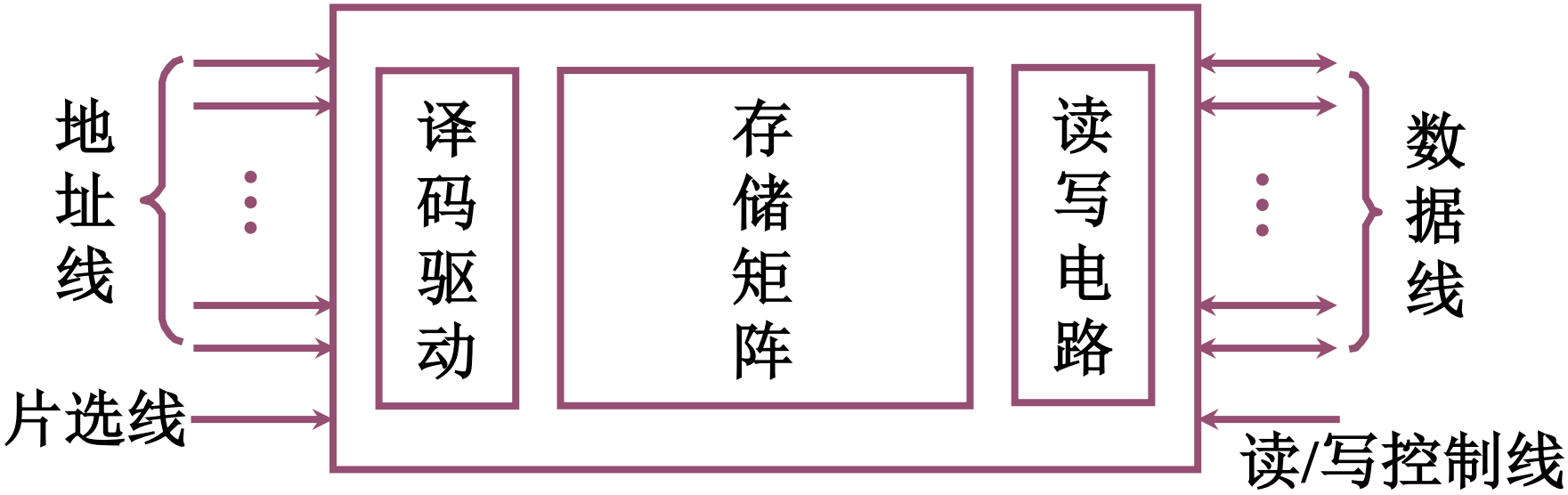


地址线（单向）	数据线（双向）	芯片容量
10	4	1K×4位
14	1	16K×1位
13	8	8K×8位

二、半导体存储芯片简介

4.2

1. 半导体存储芯片的基本结构



片选线 $\overline{CS}$ $\overline{CE}$

读/写控制线 $\overline{WE}$（低电平写 高电平读）

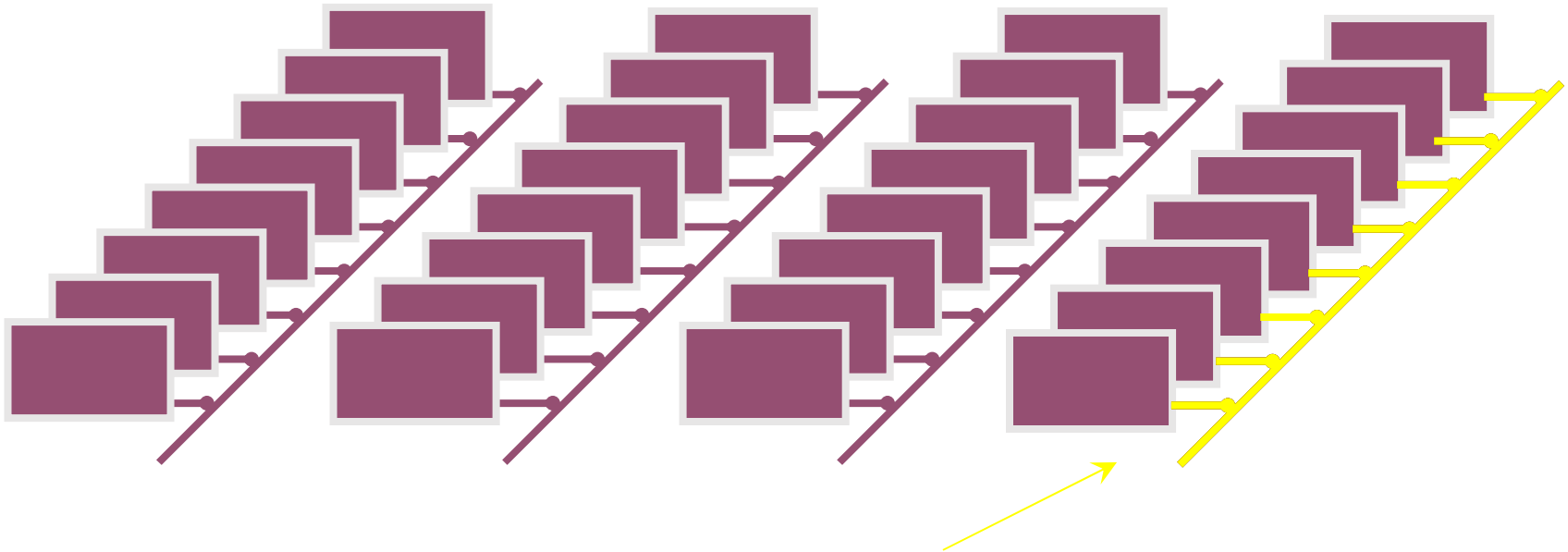
$\overline{OE}$（允许读）$\overline{WE}$（允许写）

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

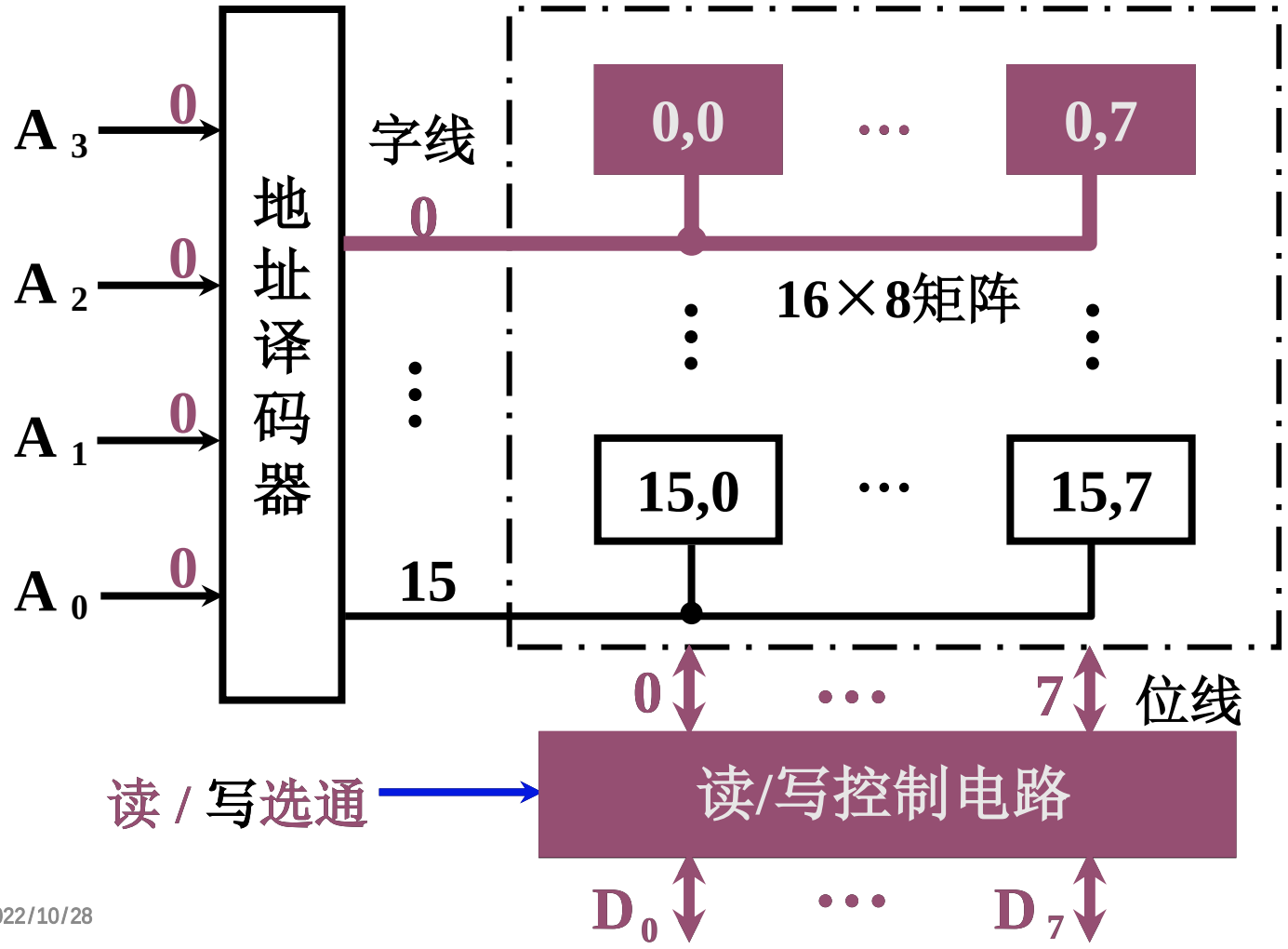
8片 16K × 1位　8片 16K × 1位　8片 16K × 1位　8片 16K × 1位



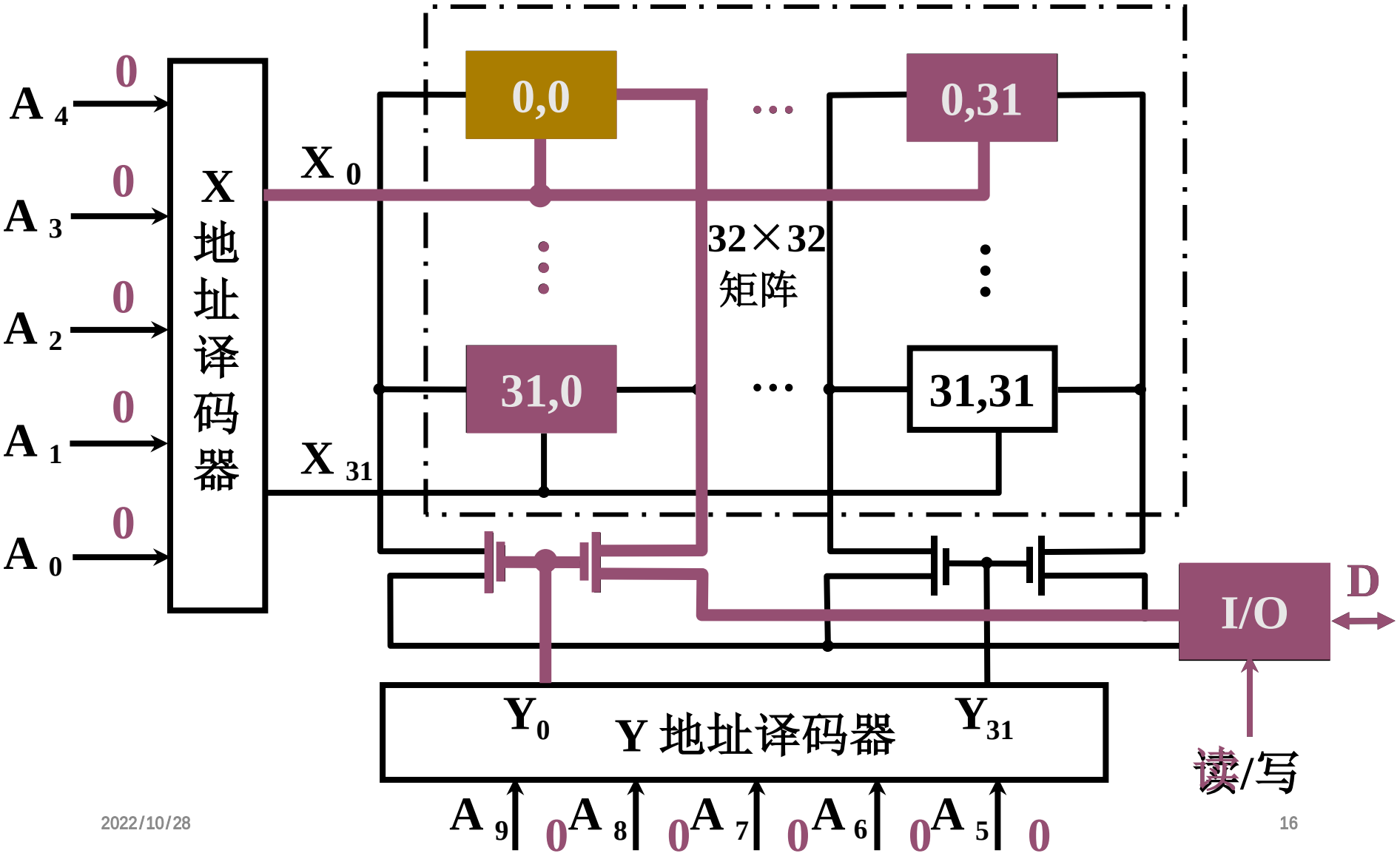
当地址为 65 535 时，此 8 片的片选有效

2. 半导体存储芯片的译码驱动方式 4.2

(1) 线选法



(2) 重合法

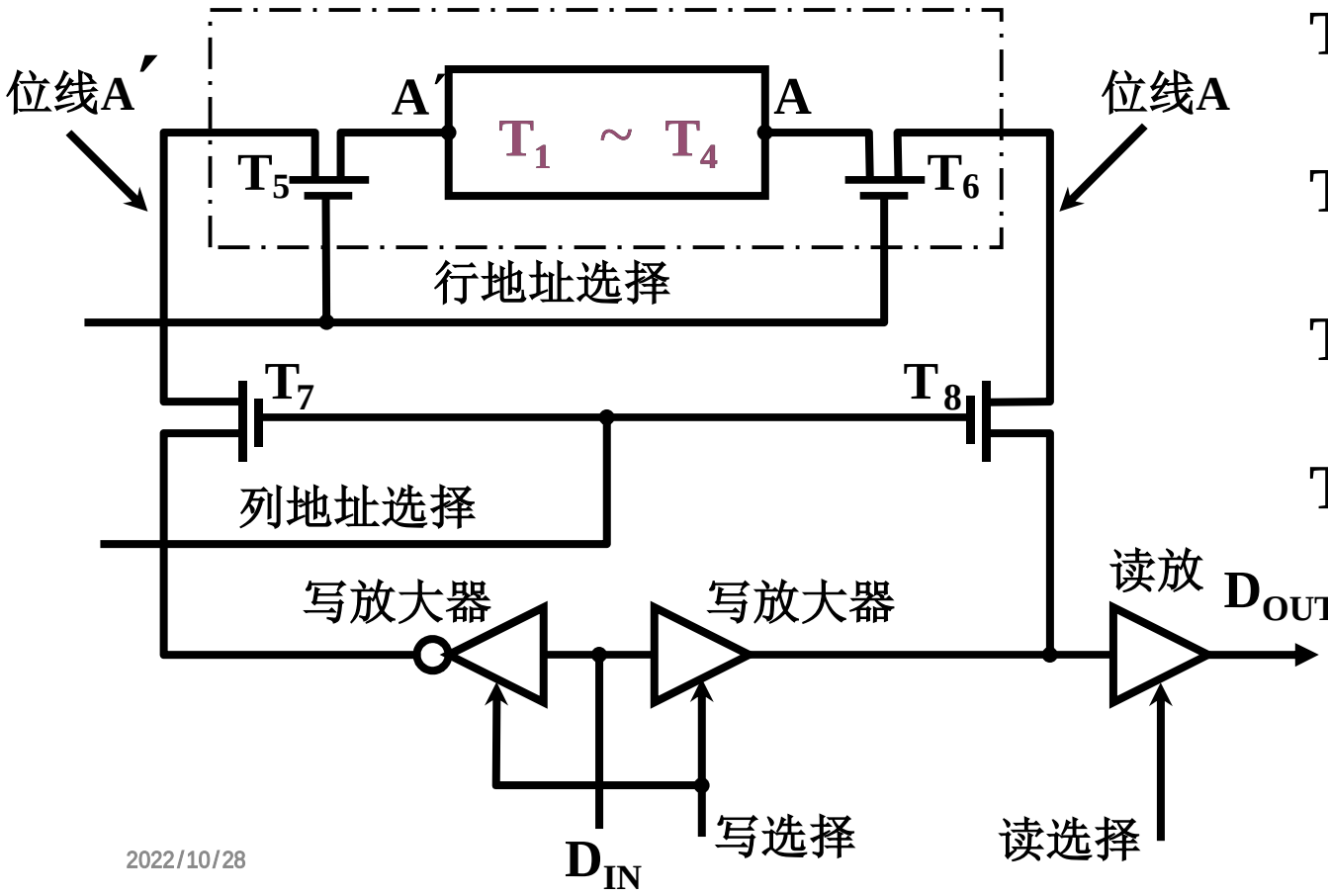

0
A 4
0
A 3
0
A 2
0
A 1
0
A 0
X
地
址
译
码
器
X 0
X 31
0,0
0,31
31,0
31,31
32×32
矩阵
I/O
D
Y 0
Y 地址译码器
Y 31
读/写
A 9
0
A 8
0
A 7
0
A 6
0
A 5
0

三、随机存取存储器 (RAM)

4.2

1. 静态 RAM (SRAM)

(1) 静态 RAM 基本电路



$T_1 \sim T_4$ 触发器

T_5 、T_6 行开关

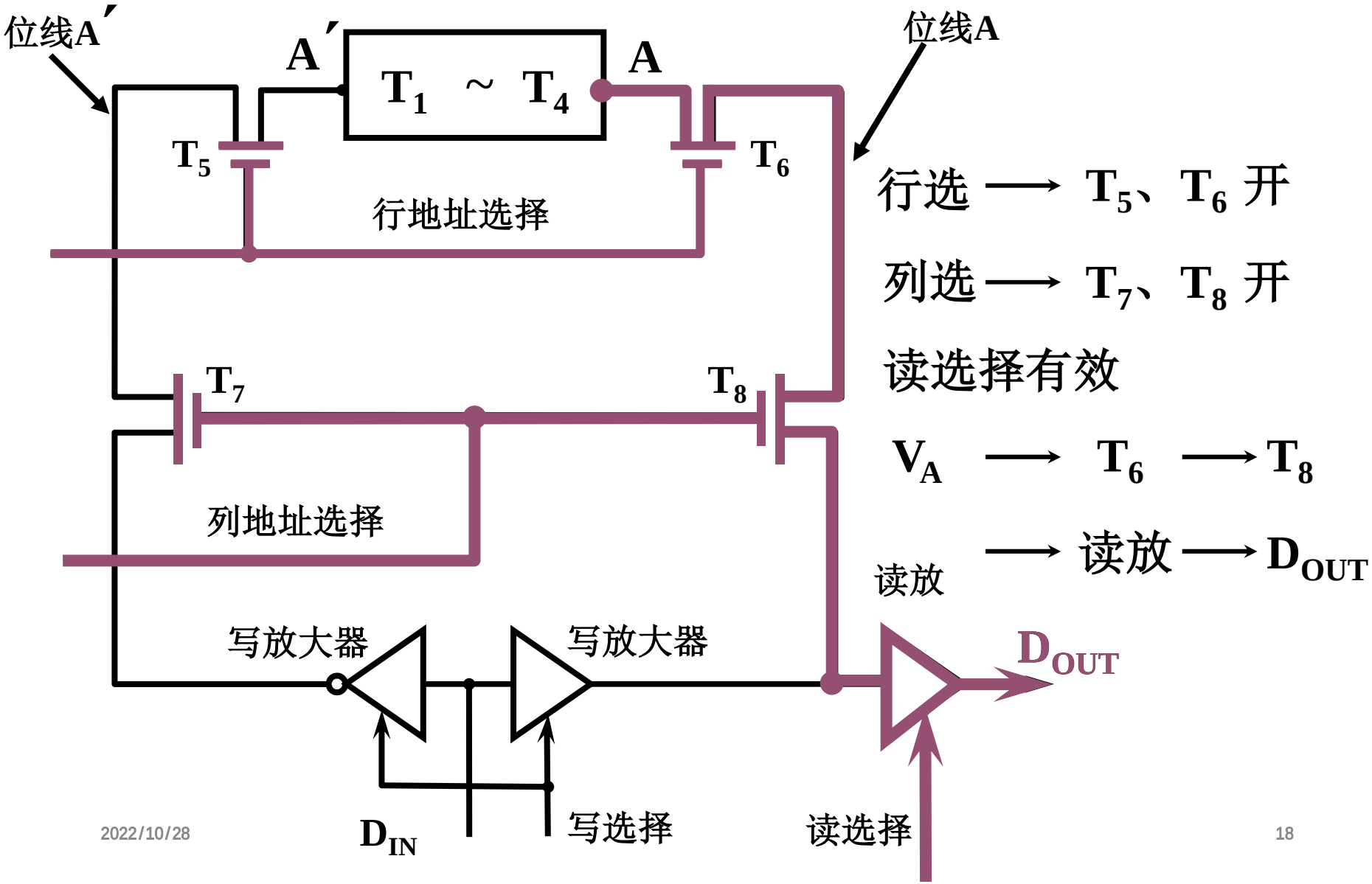
T_7 、T_8 列开关

T_7 、T_8 一列共用

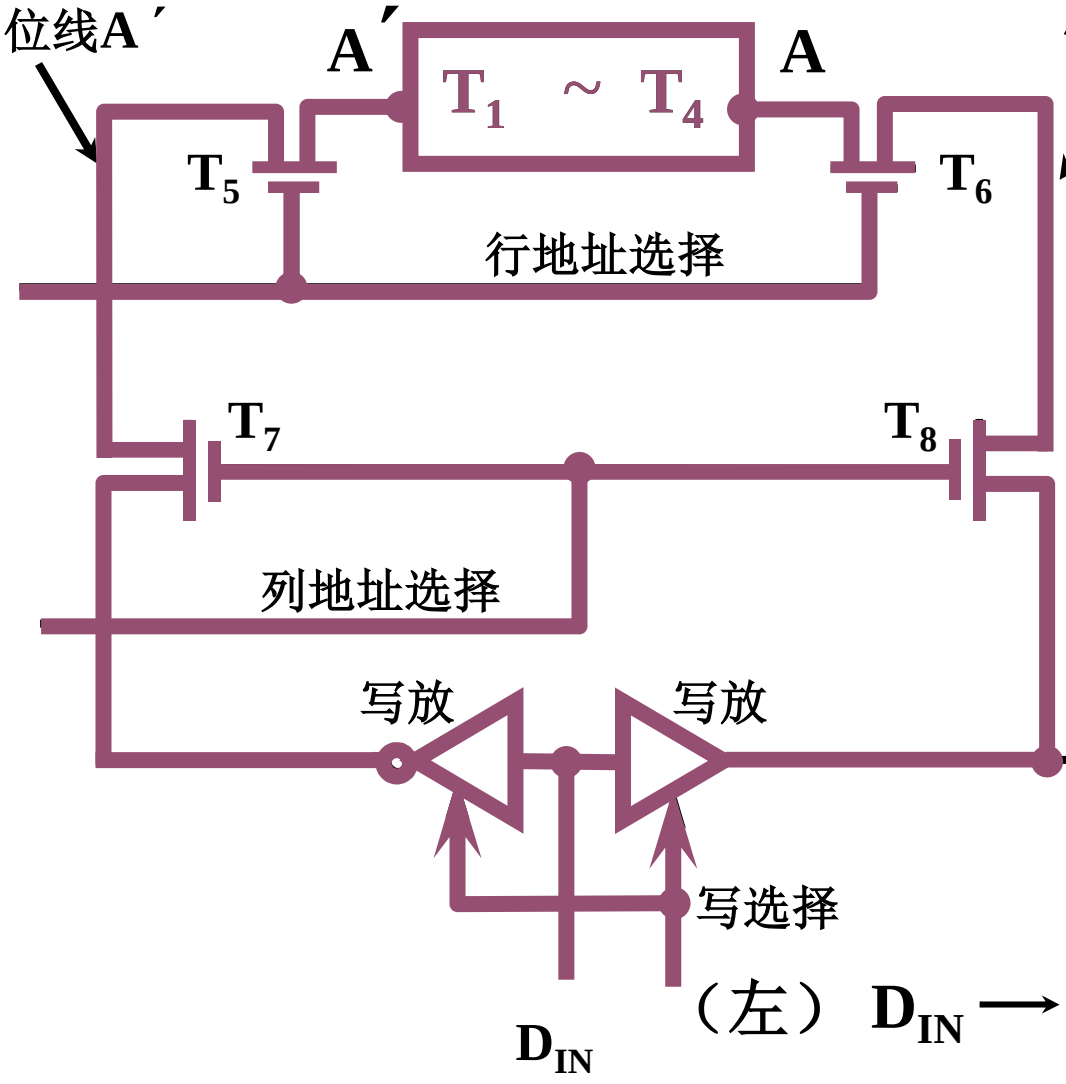
A 触发器原端

A′ 触发器非端

① 静态 RAM 基本电路的 读 操作



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

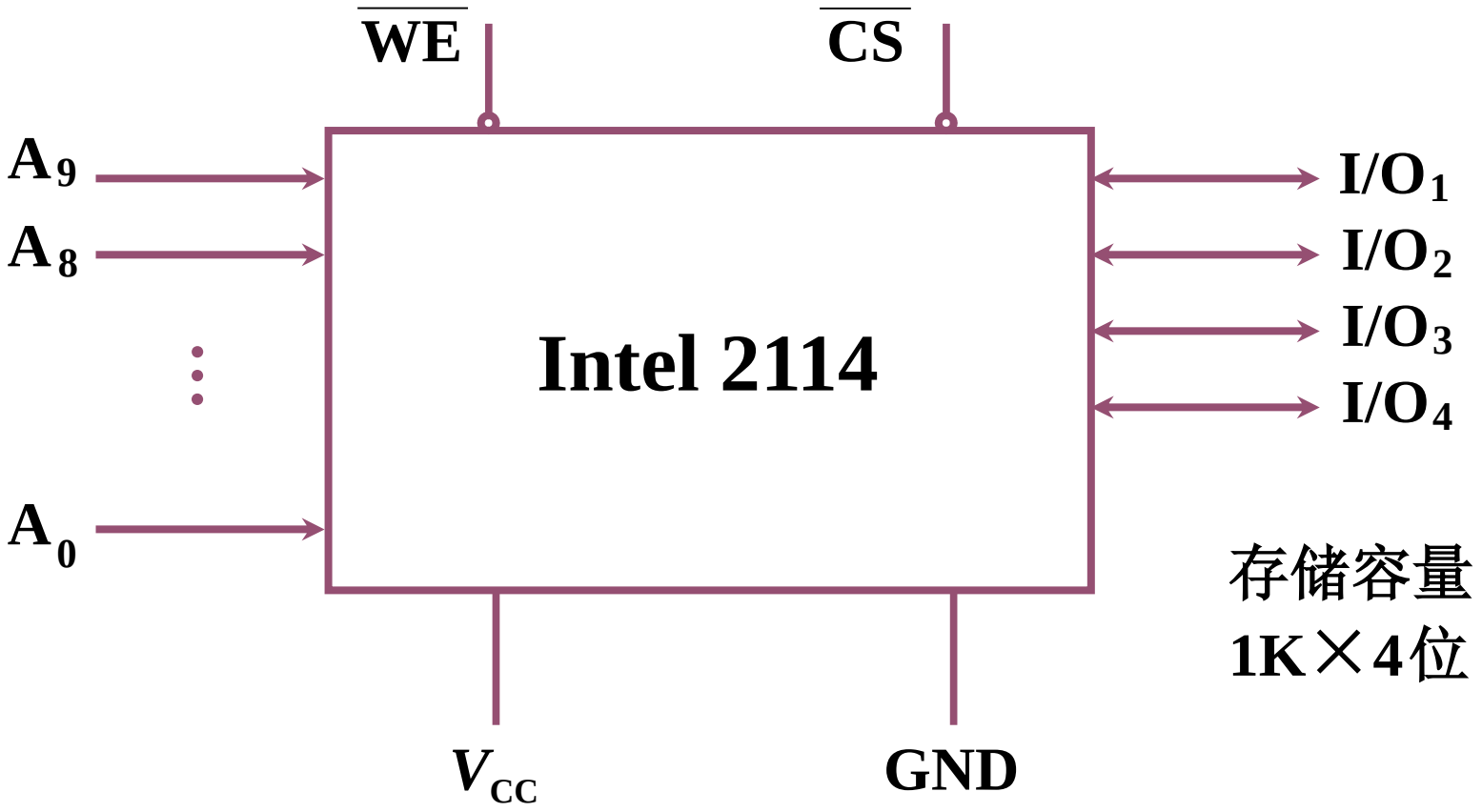
D_{IN} → 两个写放

（左） D_{IN} → 反相 → T_7 → T_5 → A′

（右） D_{IN} → T_8 → T_6 → A

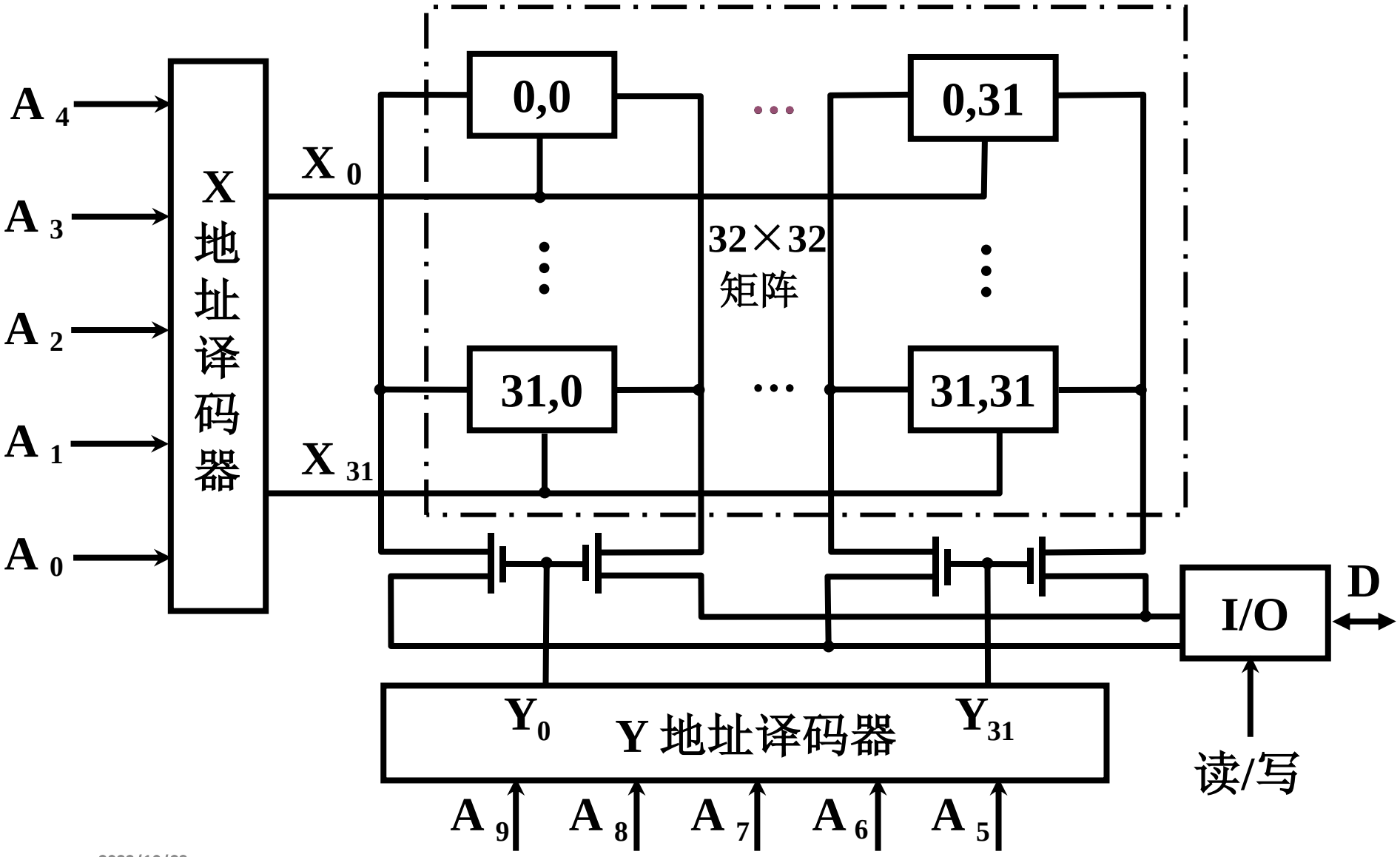
(2) 静态 RAM 芯片举例

① Intel 2114 外特性



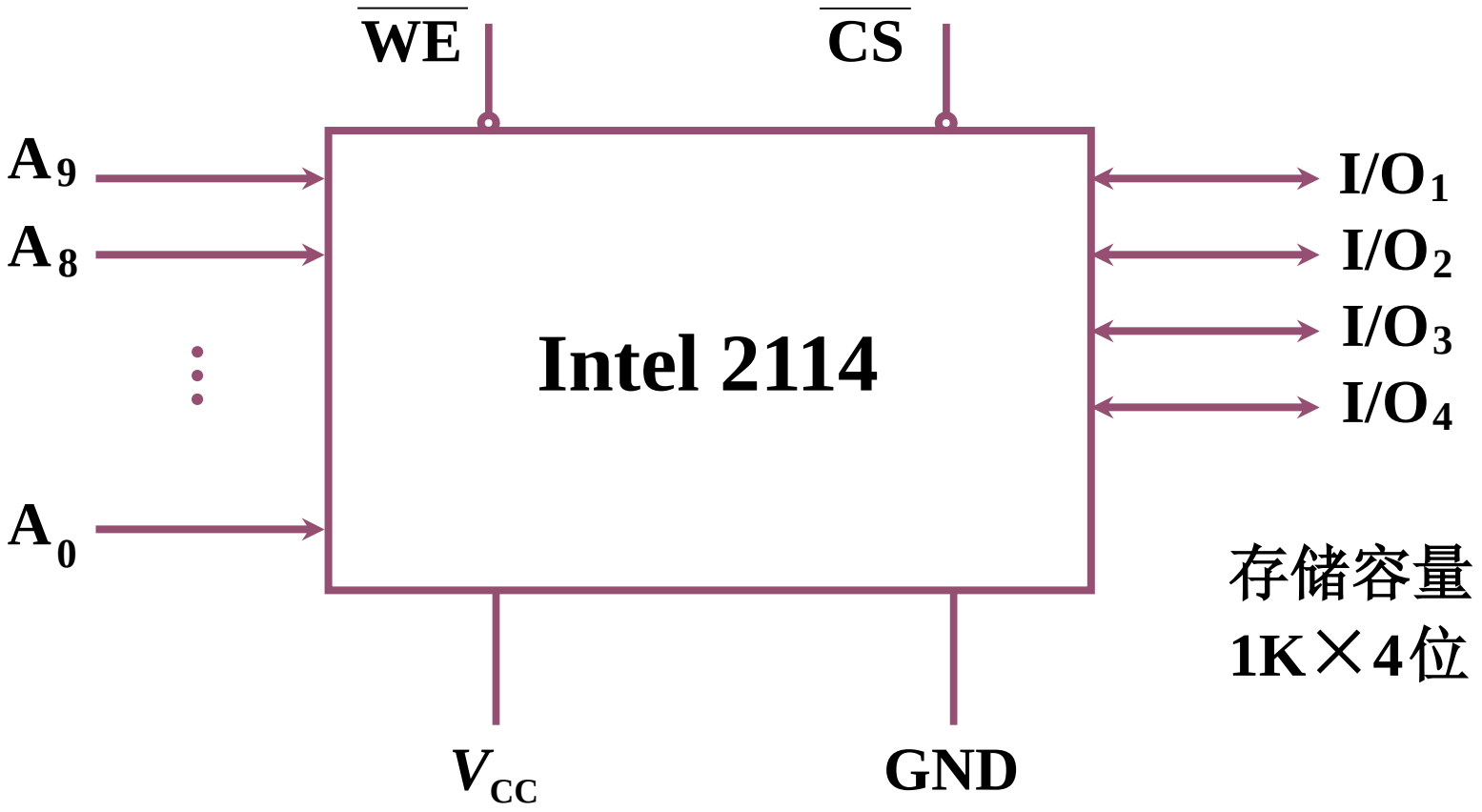
存储容量
1K×4位

曾经讲到过的重合法，怎么实现选一次四列？



(2) 静态 RAM 芯片举例

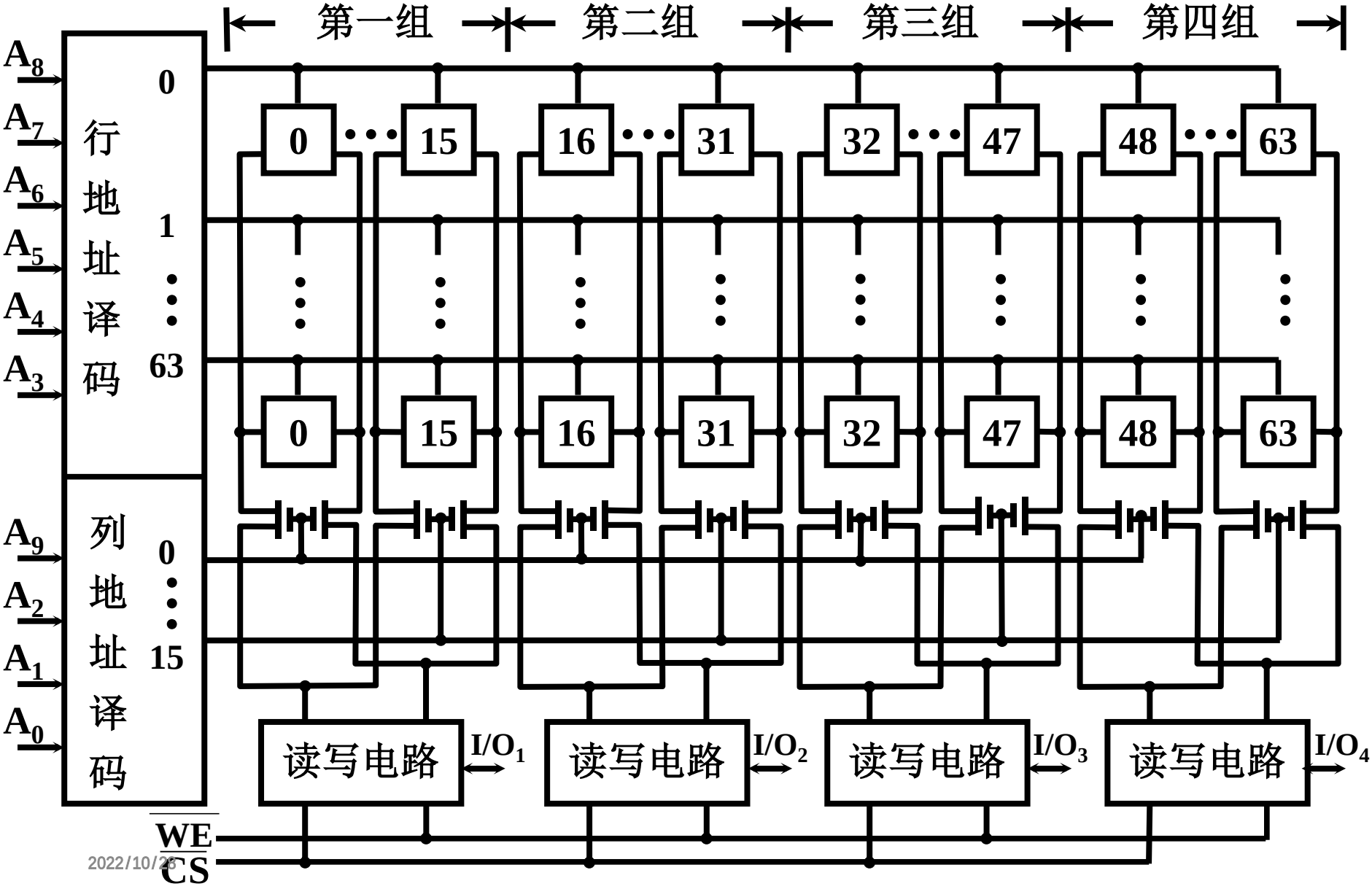
① Intel 2114 外特性



存储容量
1K×4位

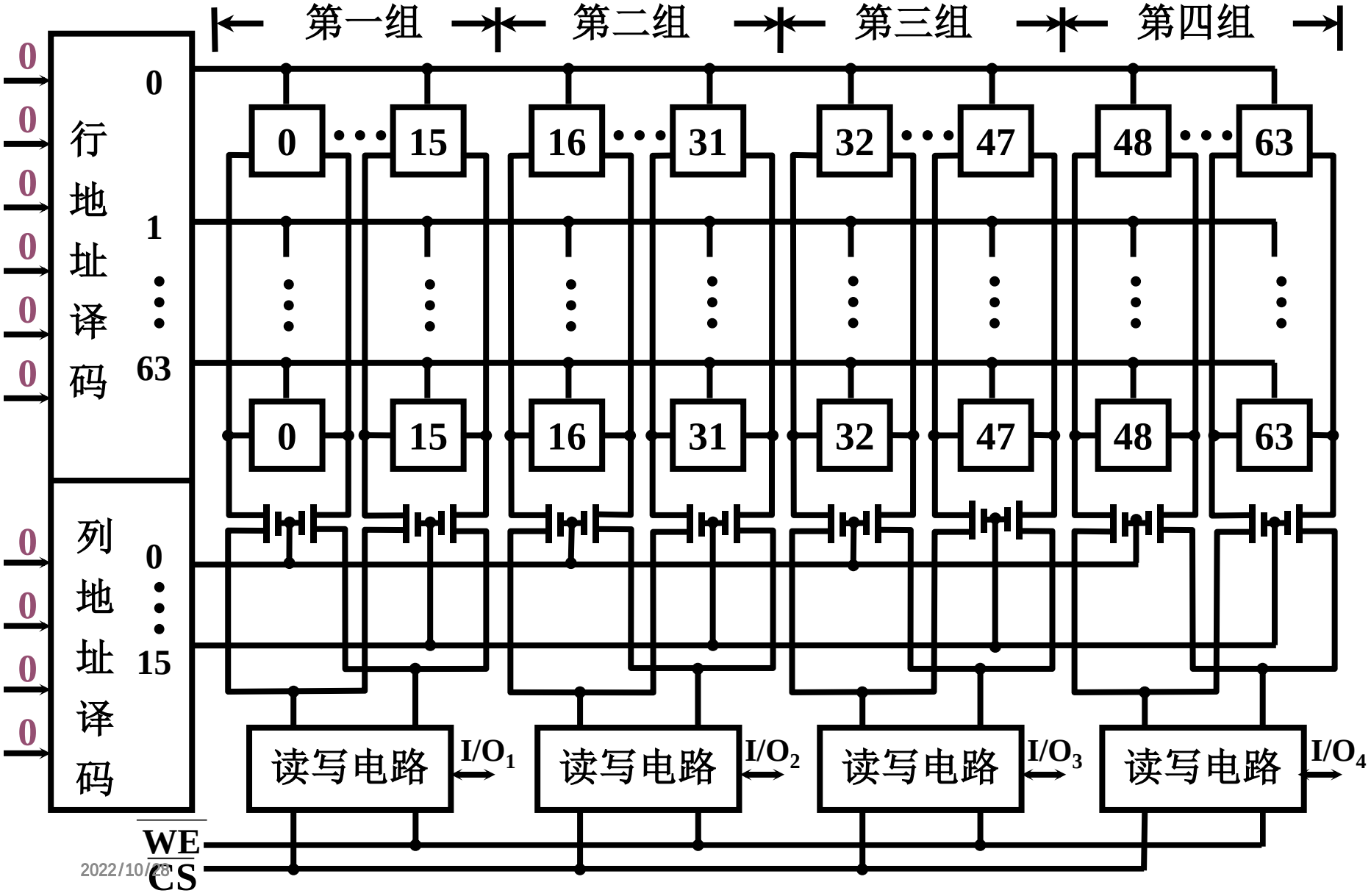
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2

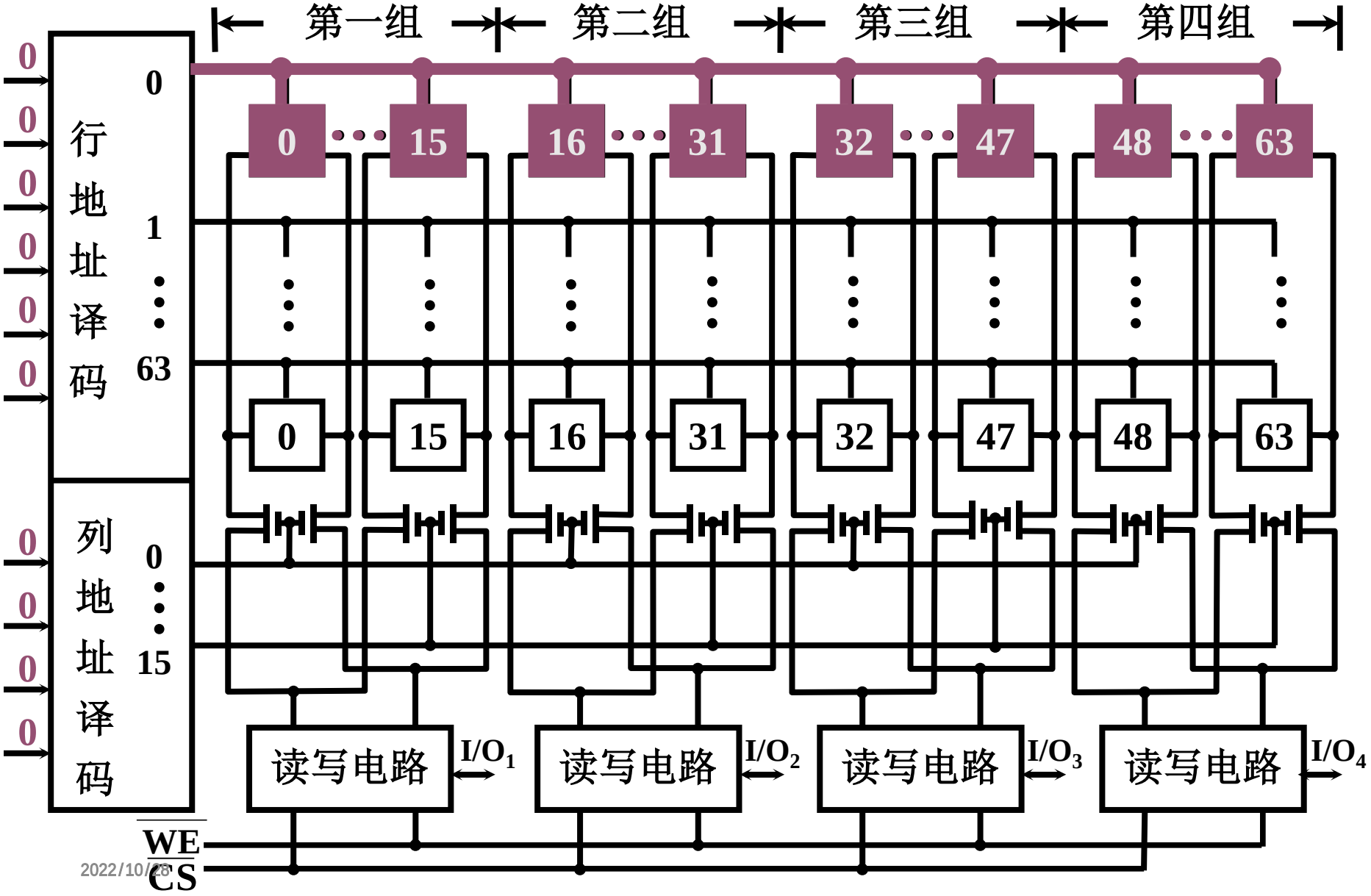


② Intel 2114 RAM 矩阵 (64 × 64) 读

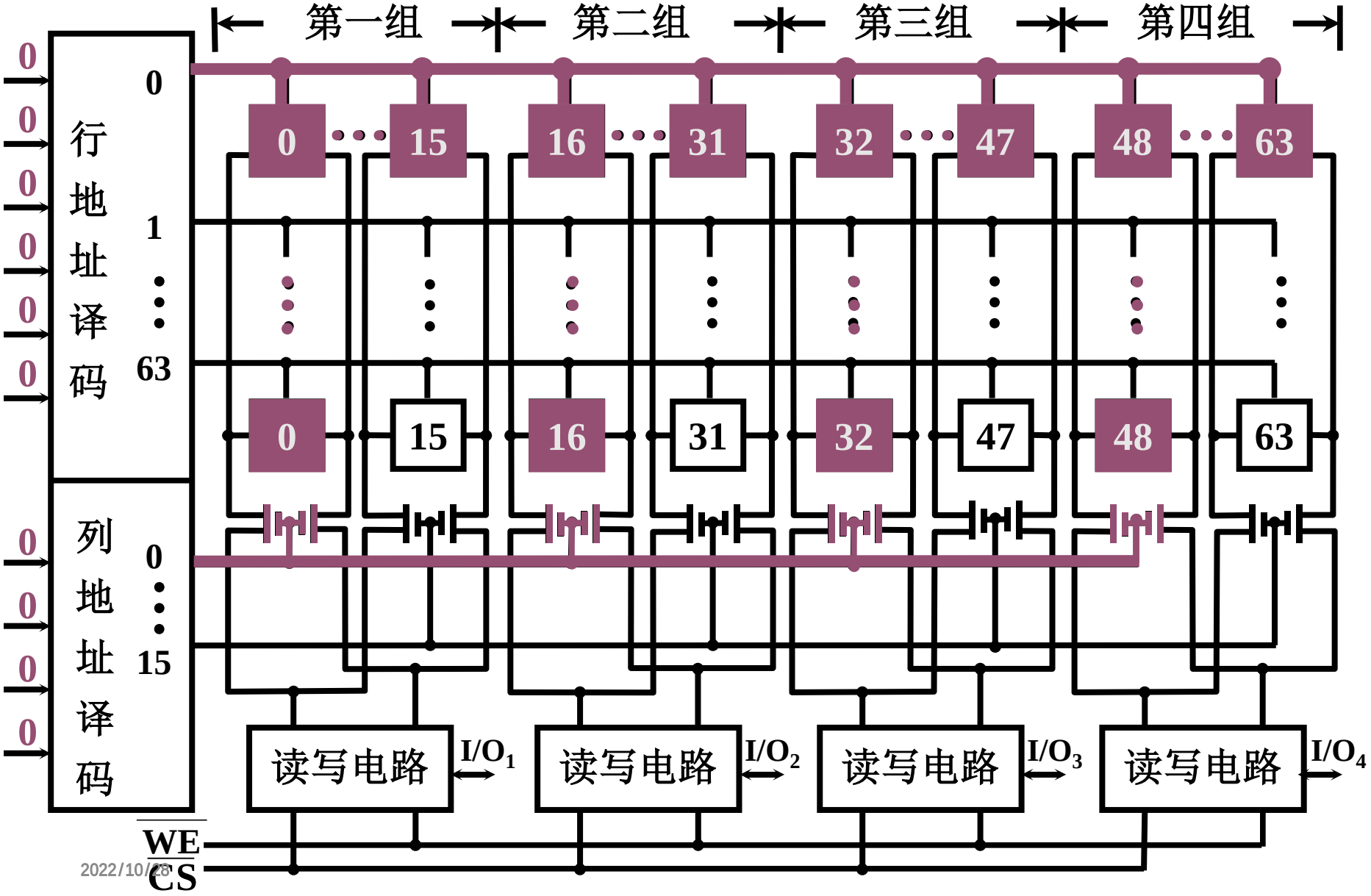
4.2



② Intel 2114 RAM 矩阵 (64 × 64) 读

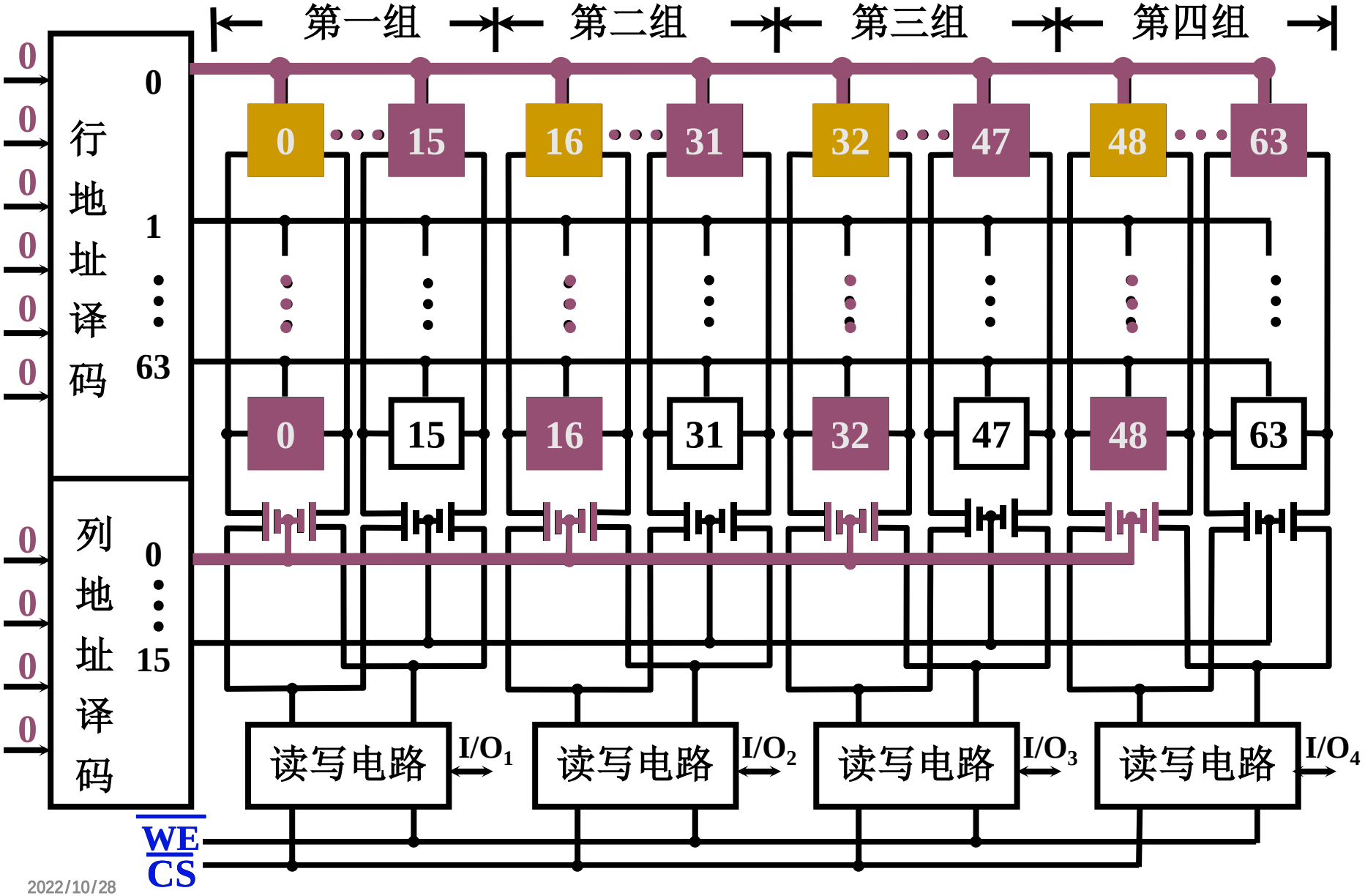


② Intel 2114 RAM 矩阵 (64 × 64) 读



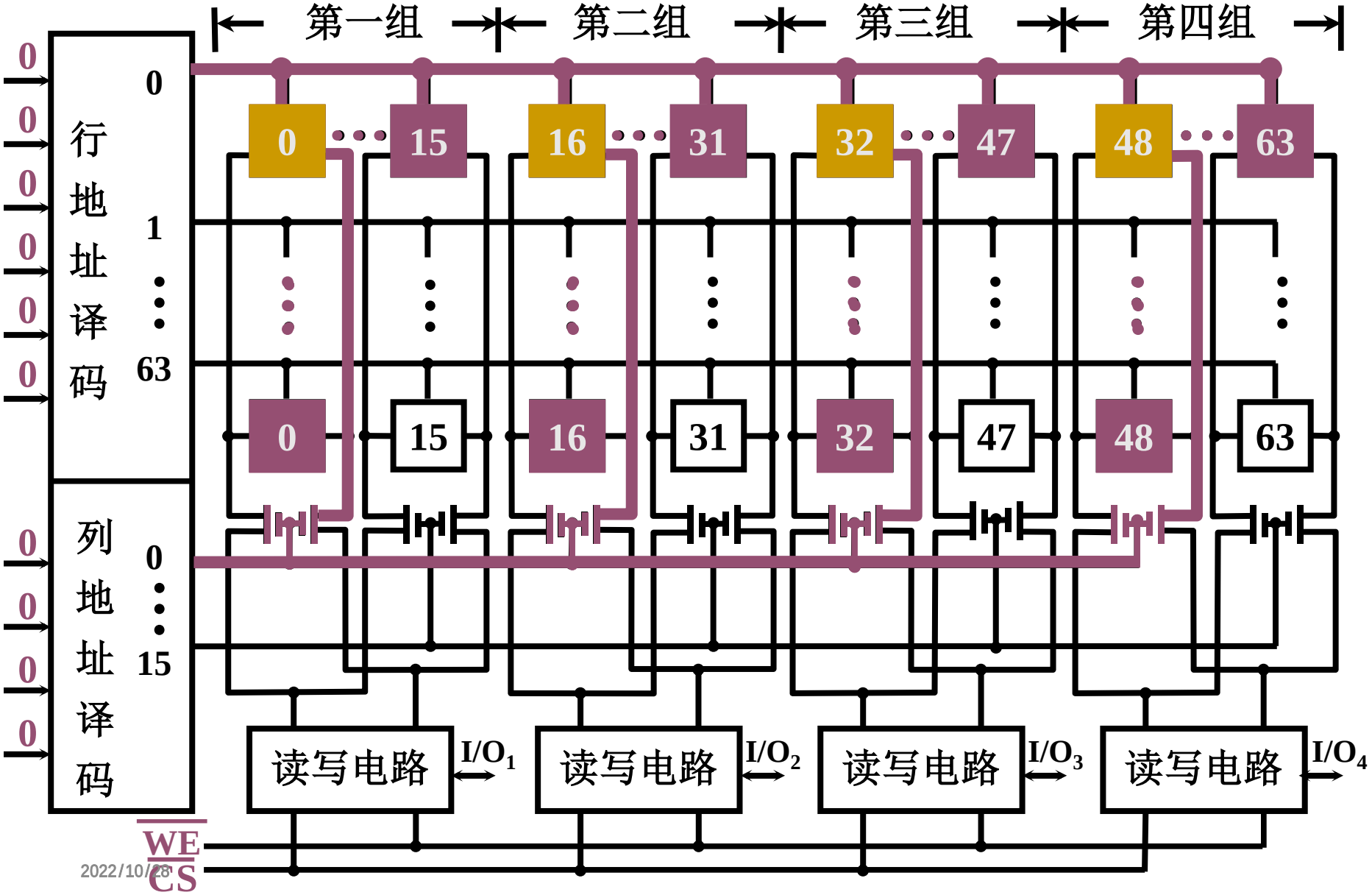
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2



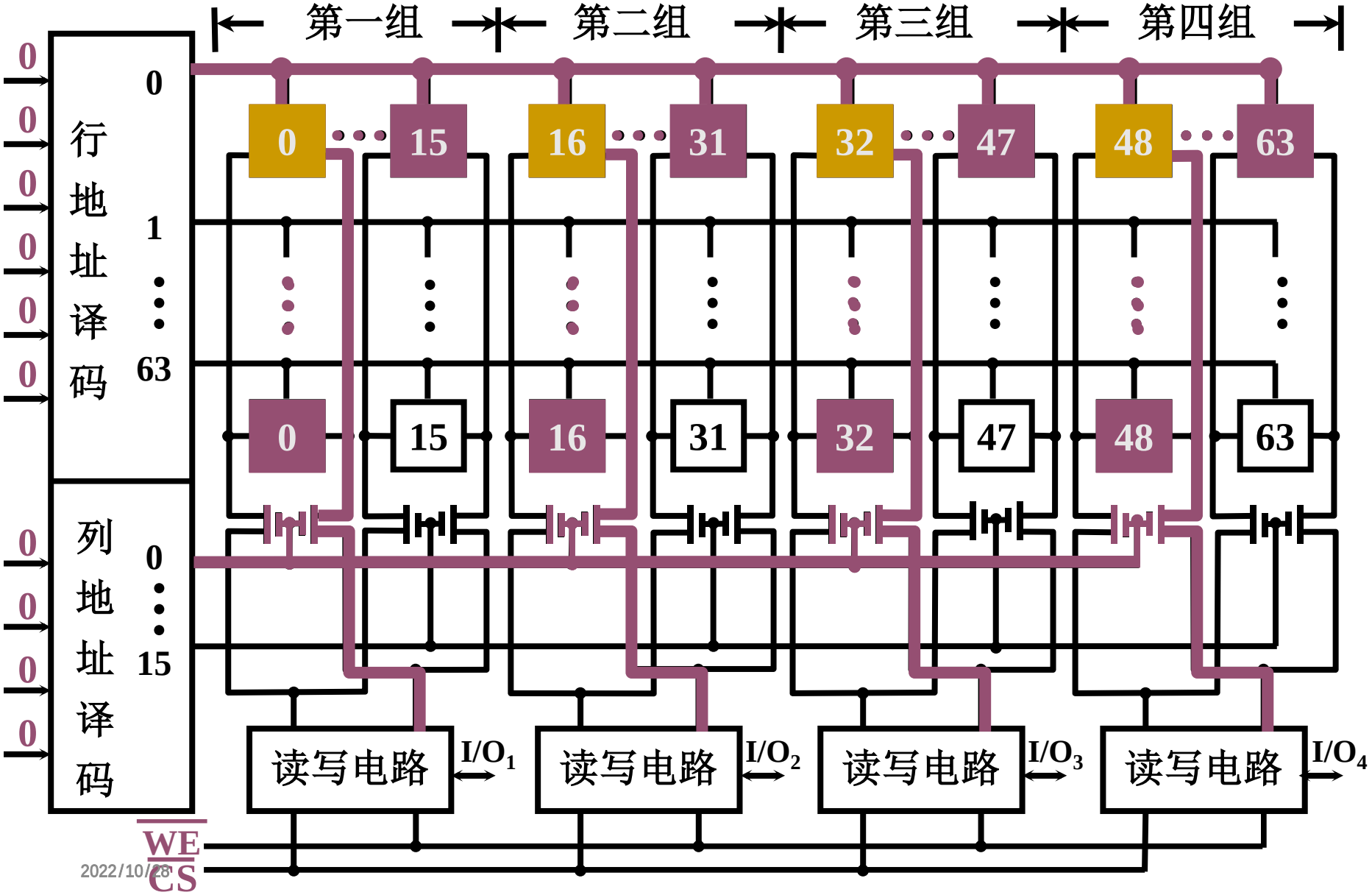
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2



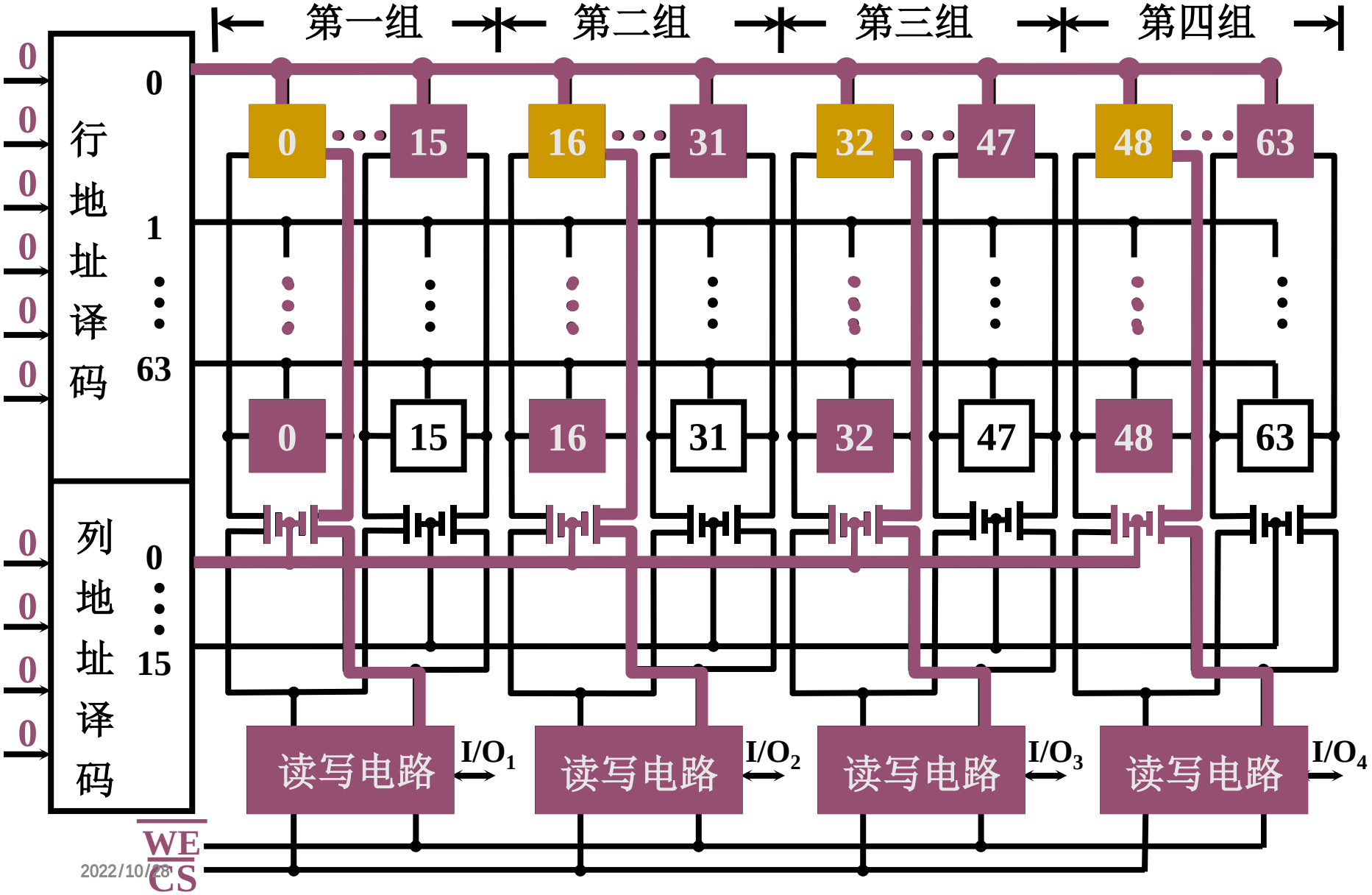
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2



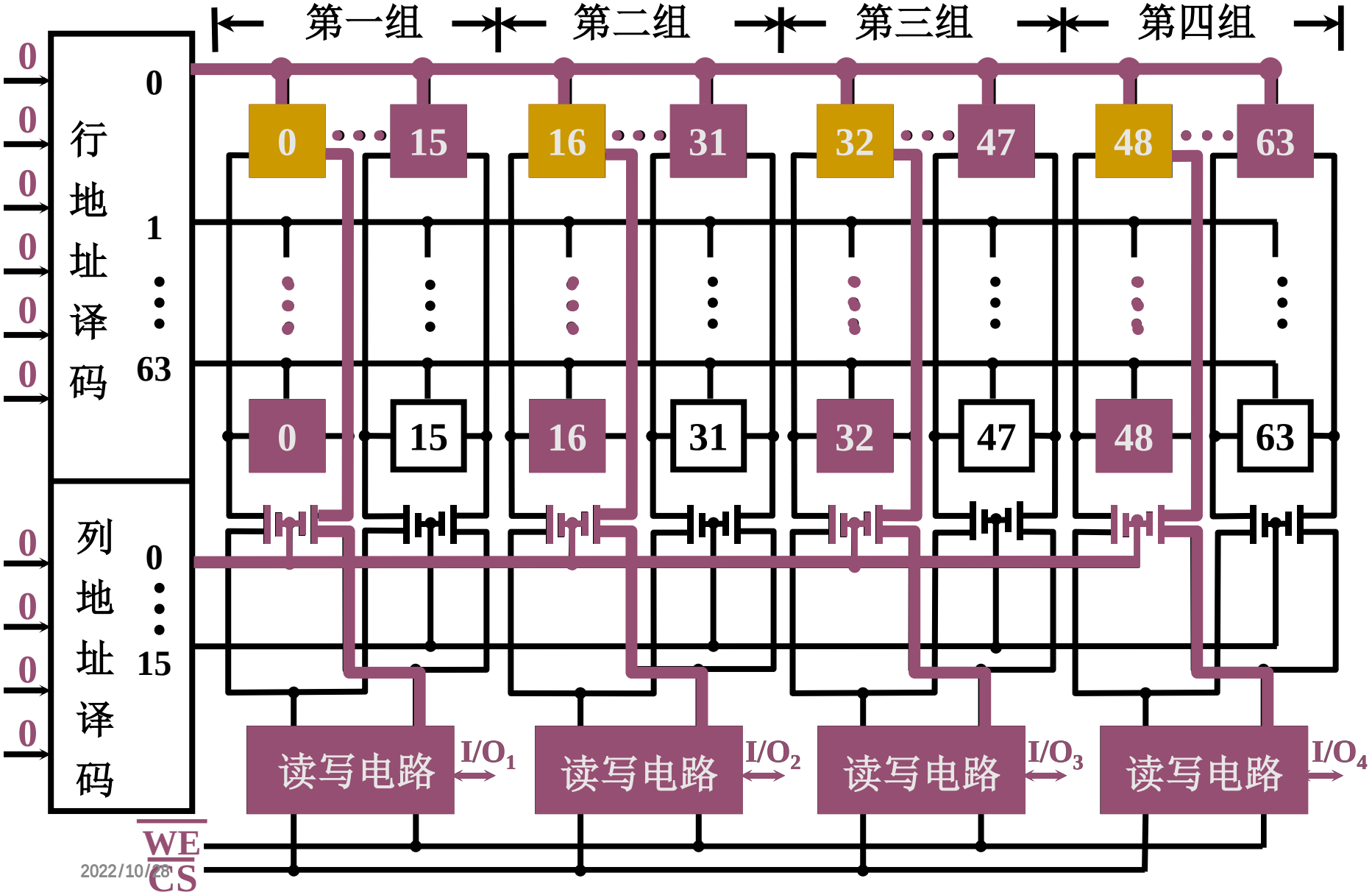
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2



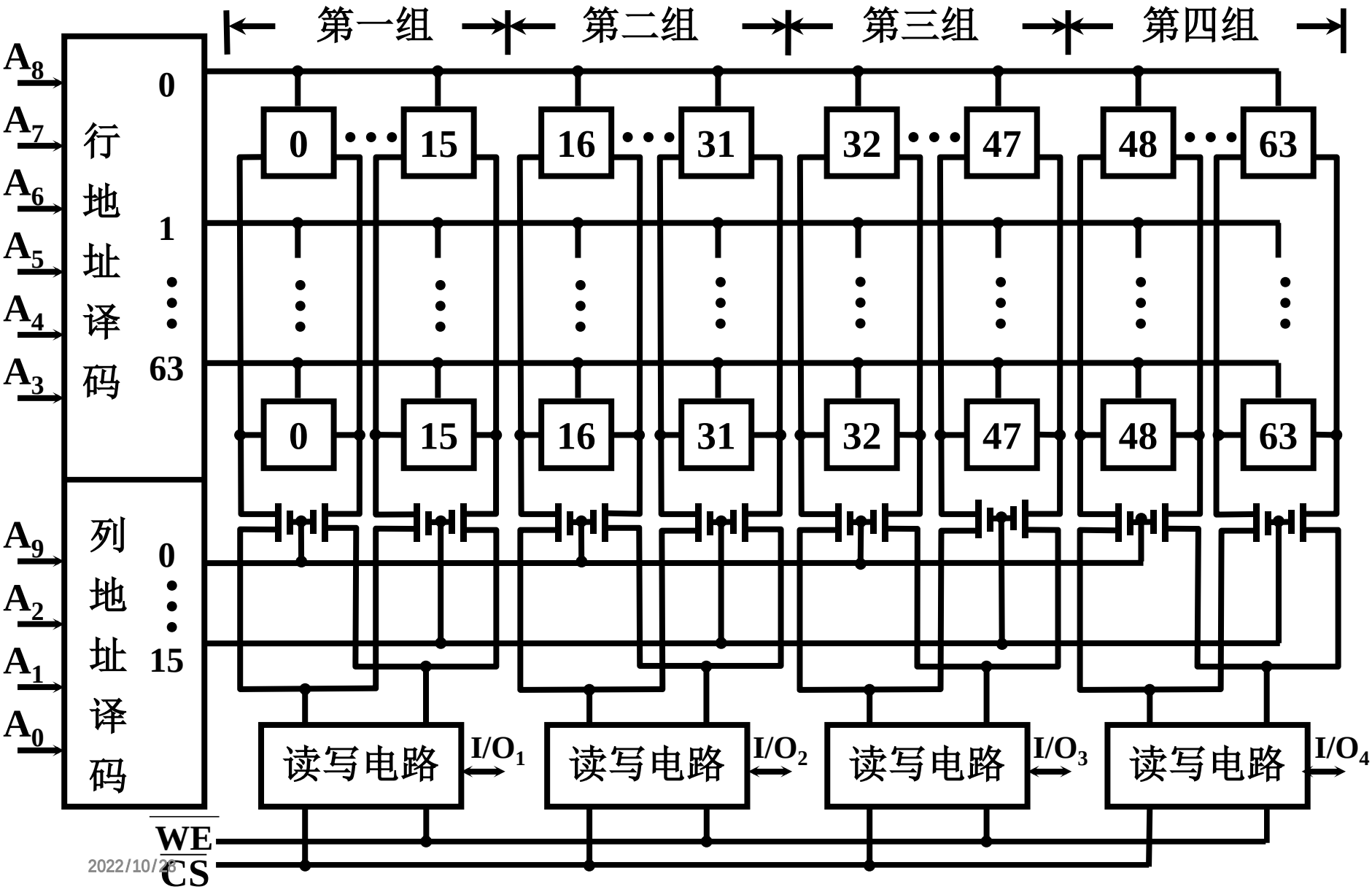
② Intel 2114 RAM 矩阵 (64 × 64) 读

4.2



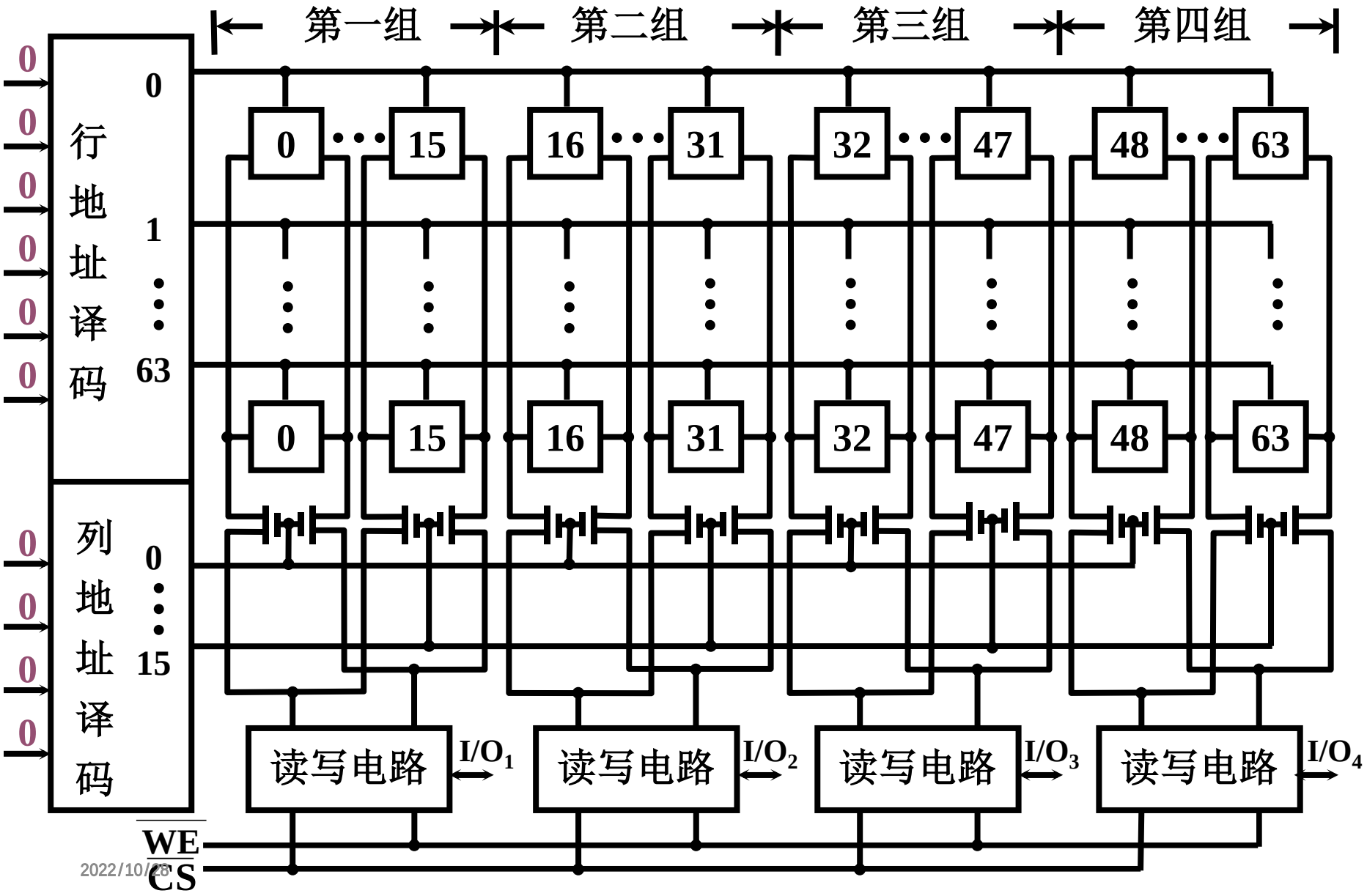
③ Intel 2114 RAM 矩阵 (64 × 64) 写

4.2

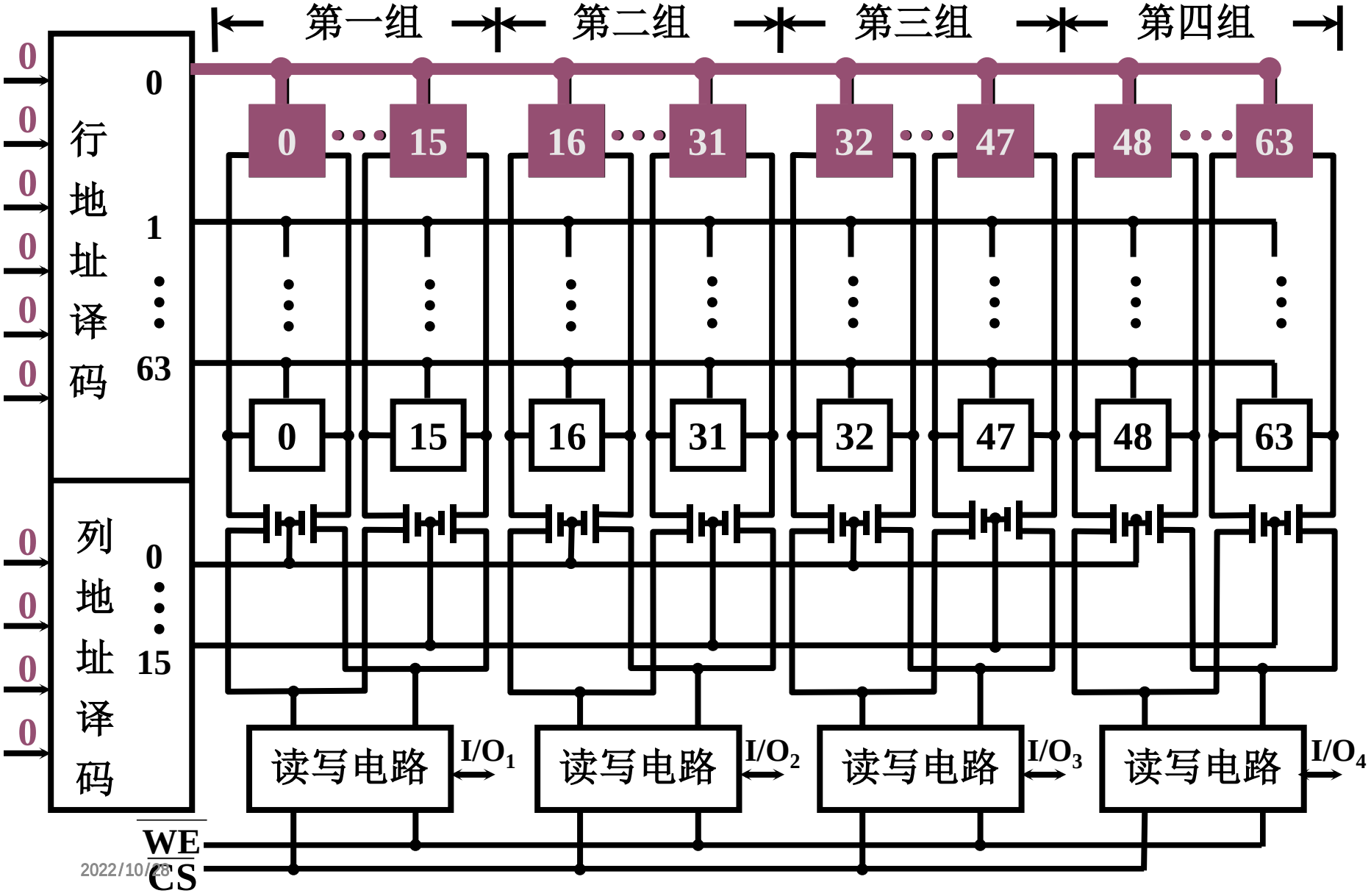


③ Intel 2114 RAM 矩阵 (64 × 64) 写

4.2

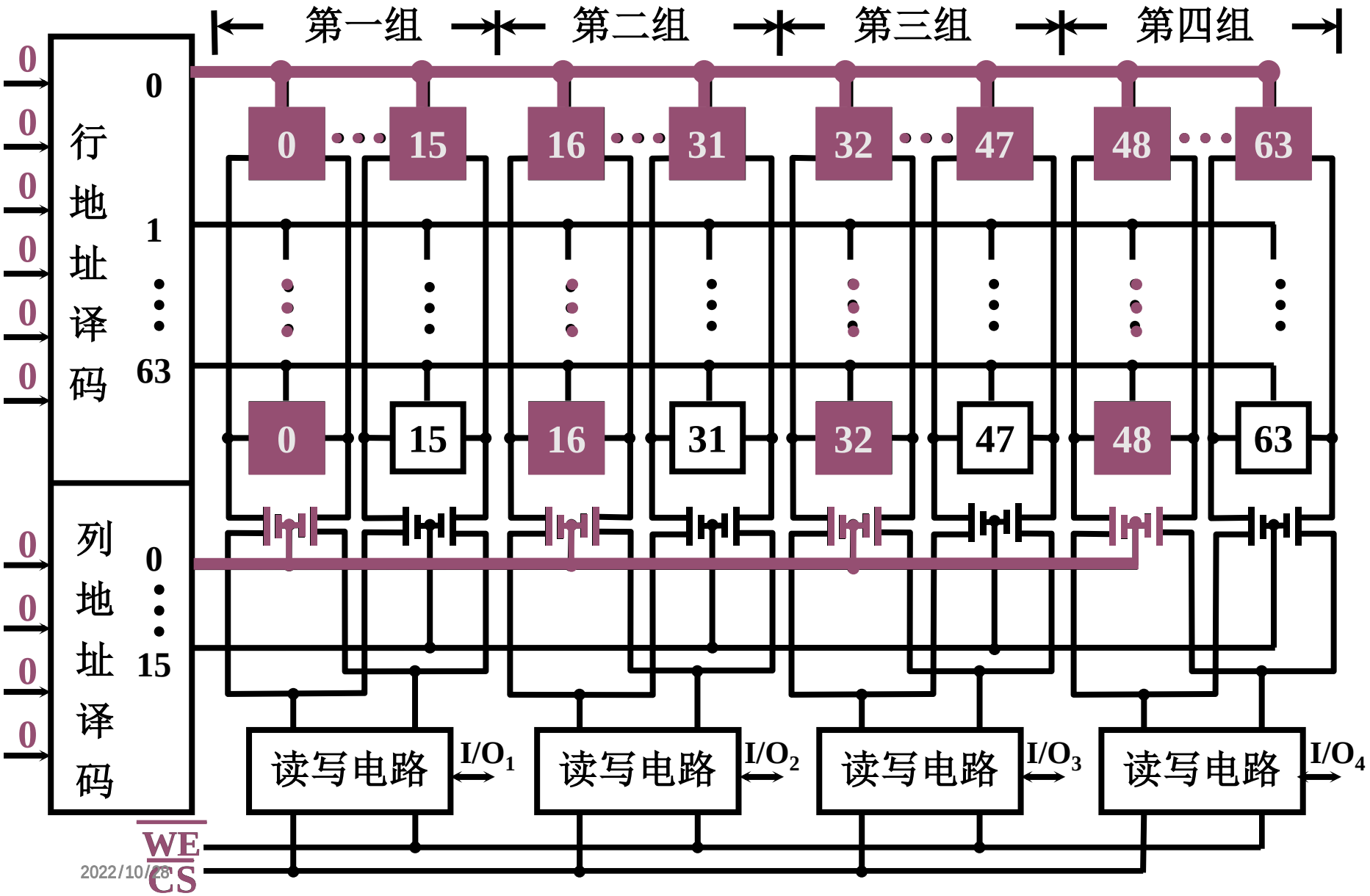


③ Intel 2114 RAM 矩阵 (64 × 64) 写

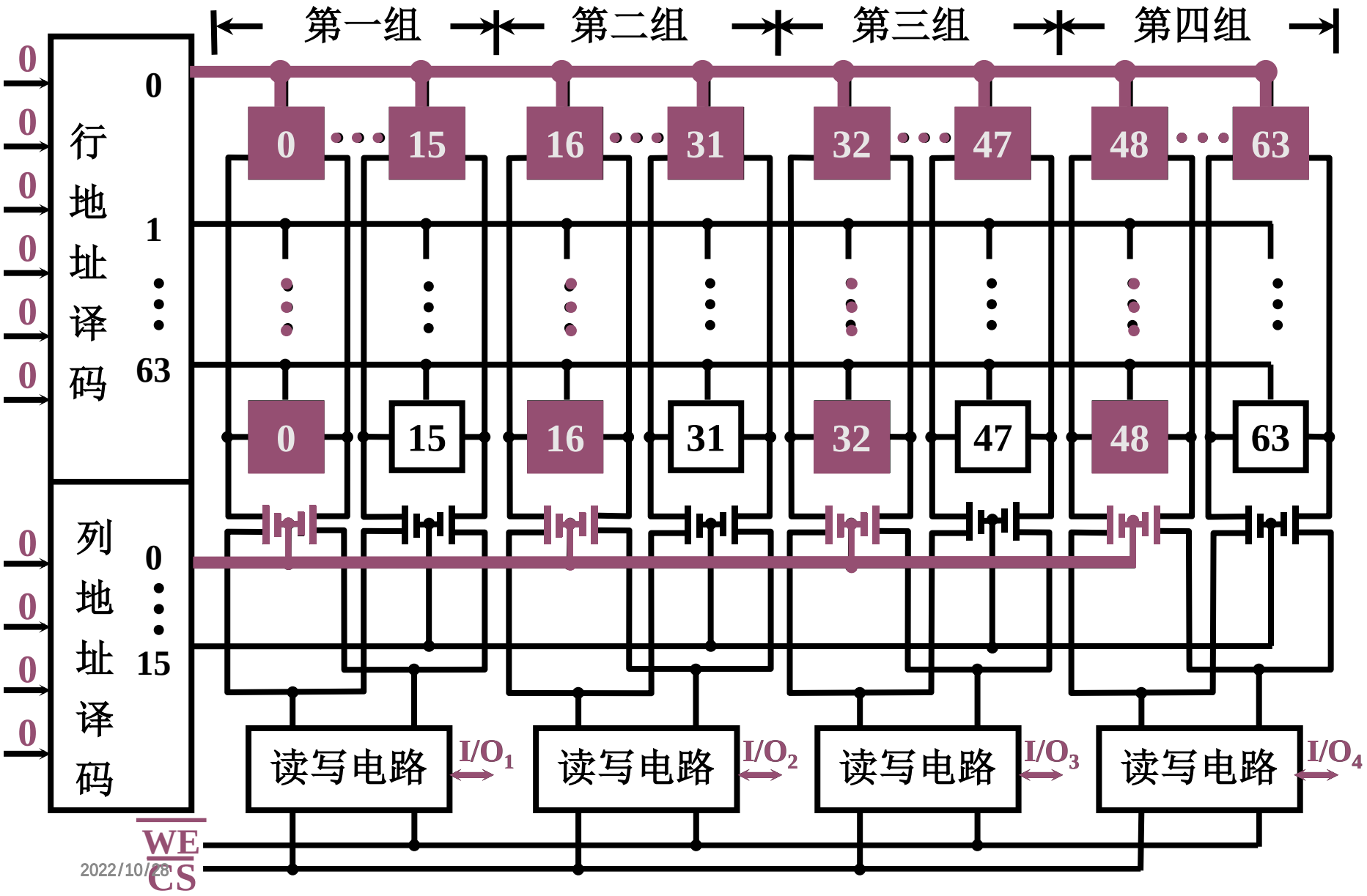


③ Intel 2114 RAM 矩阵 (64 × 64) 写

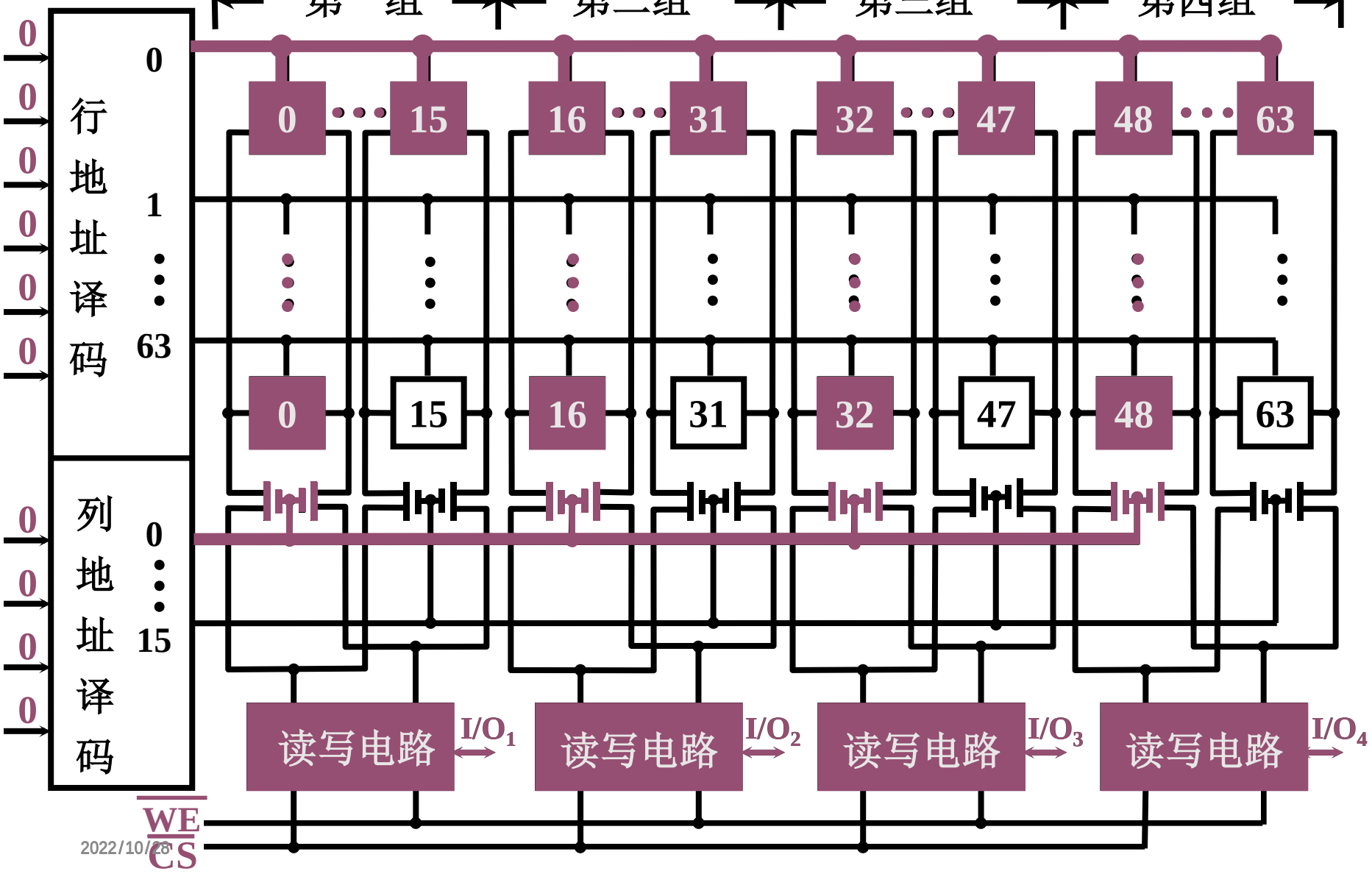
4.2



③ Intel 2114 RAM 矩阵 (64 × 64) 写

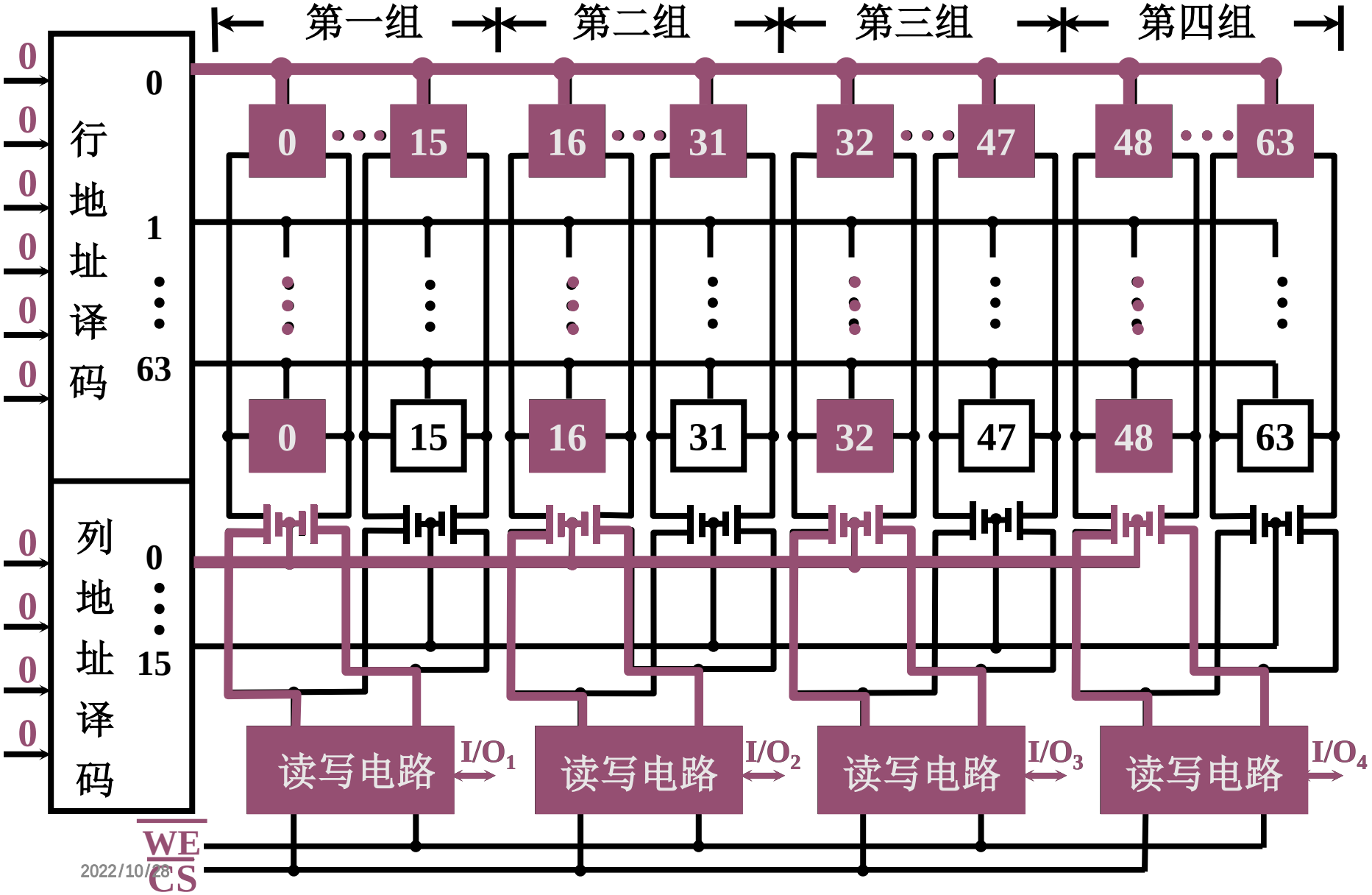


③ Intel 2114 RAM 矩阵 (64 × 64) 写



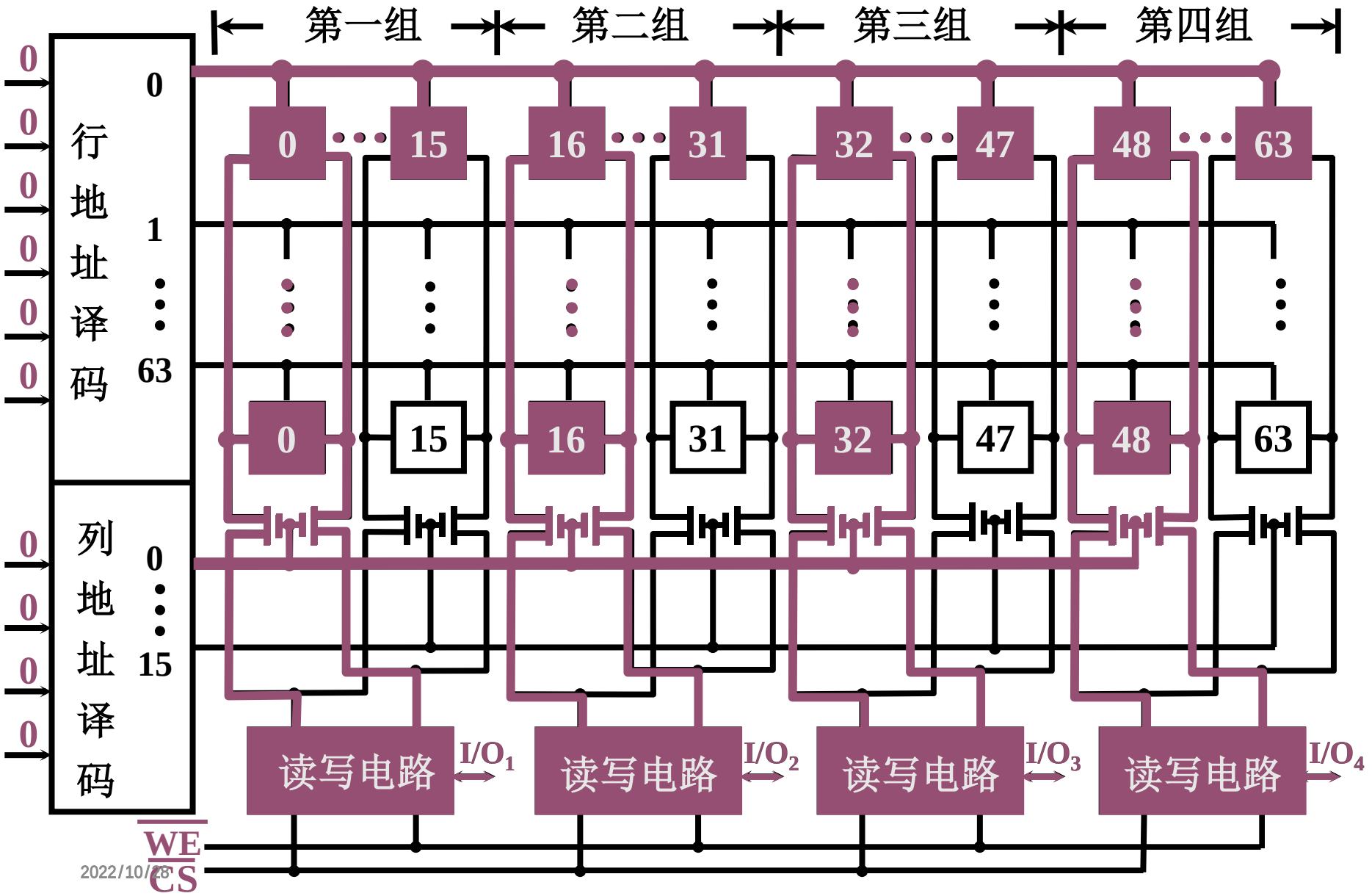
③ Intel 2114 RAM 矩阵 (64 × 64) 写

4.2



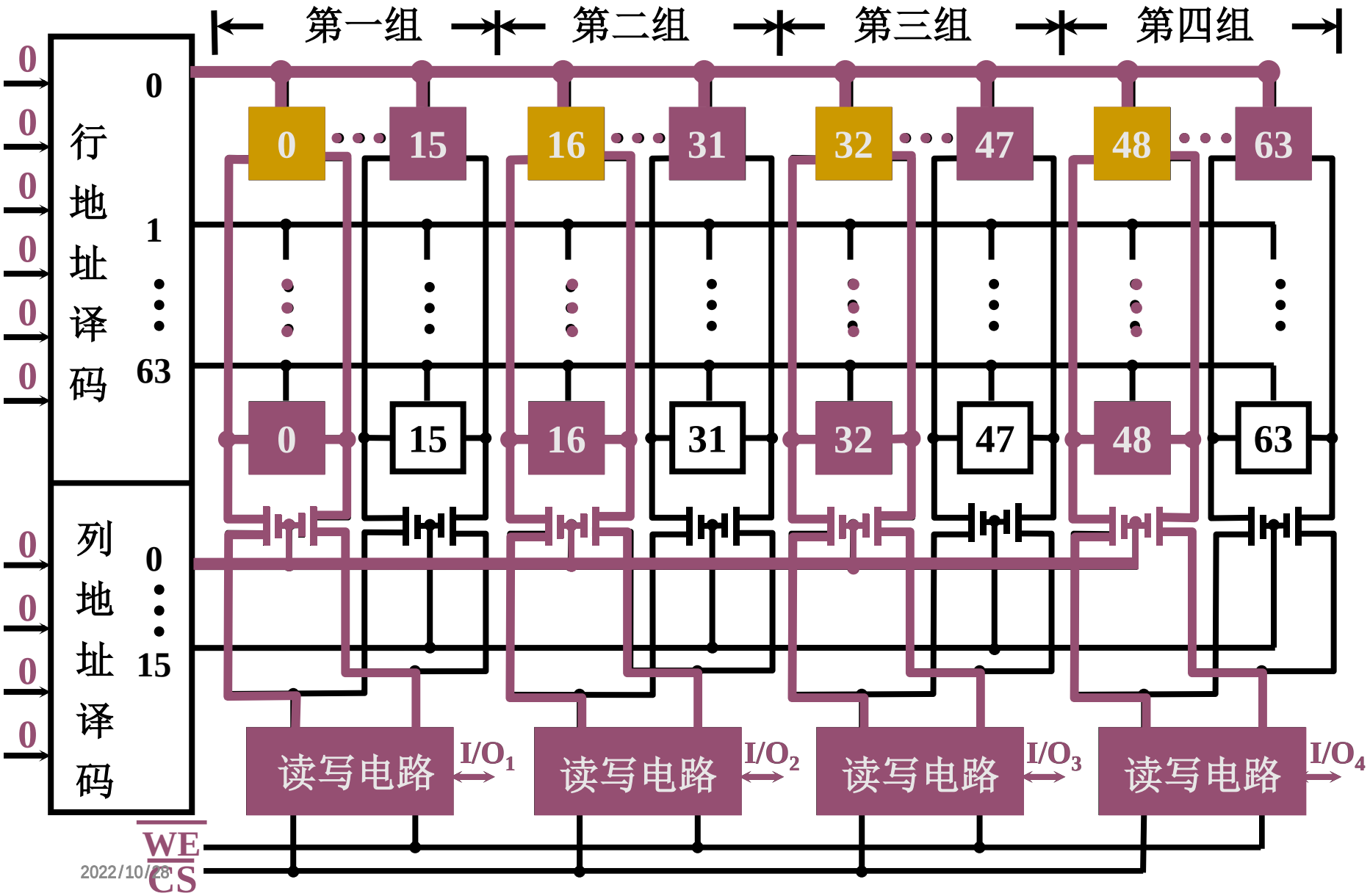
③ Intel 2114 RAM 矩阵 (64 × 64) 写

4.2

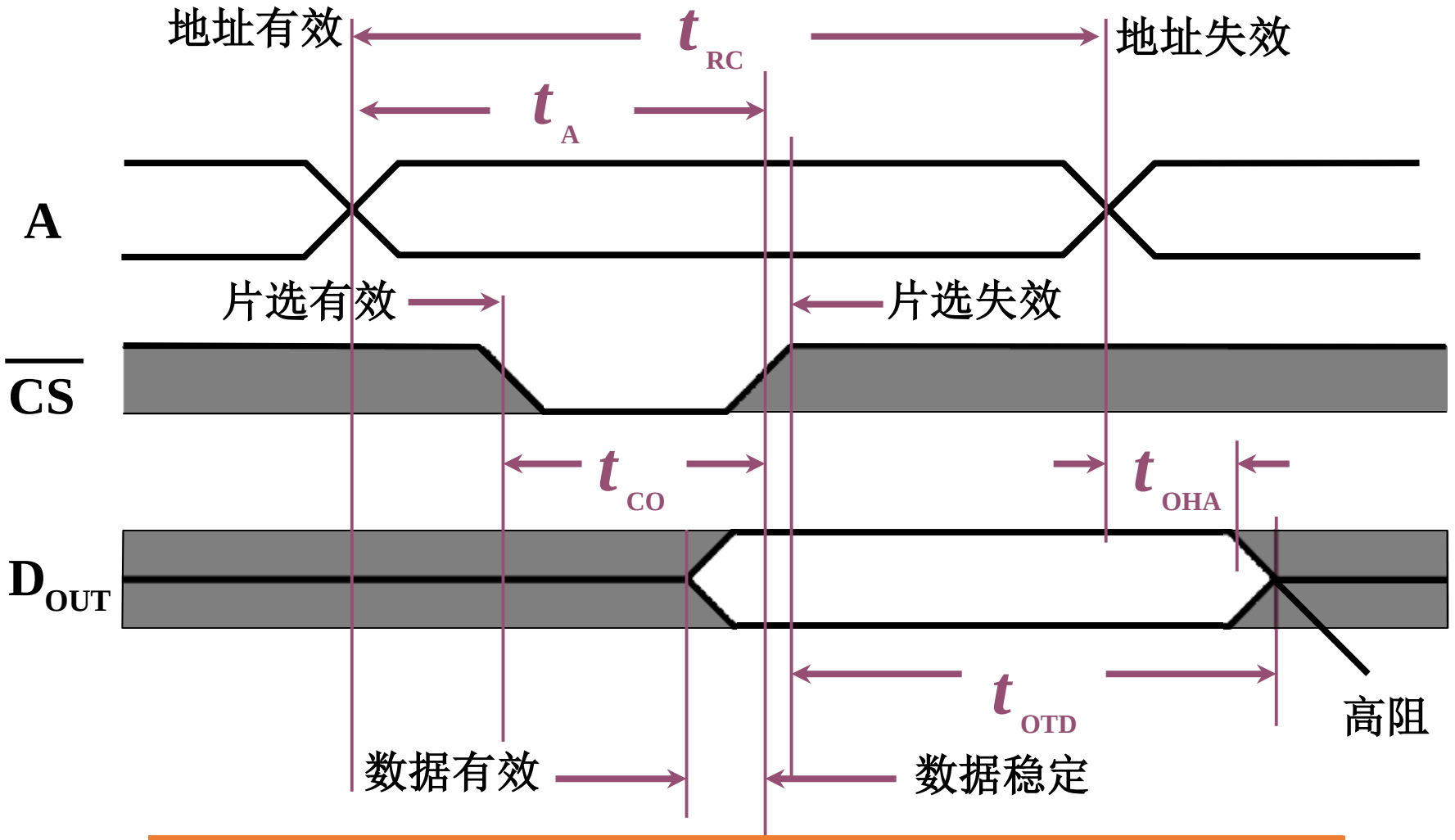


③ Intel 2114 RAM 矩阵 (64 × 64) 写

4.2

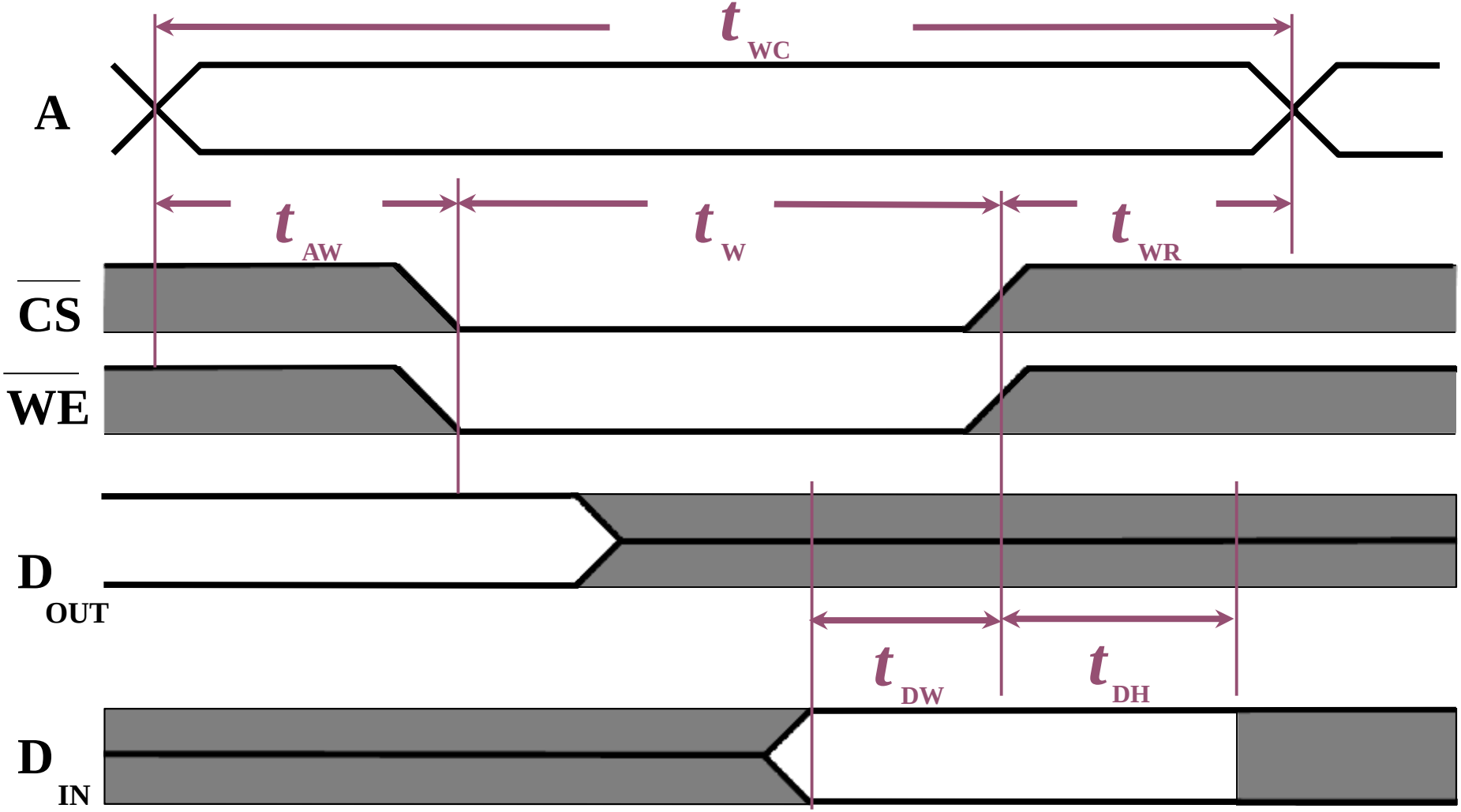


(3) 静态 RAM 读 时序



t_{OHA} 地址失效后的 数据维持时间

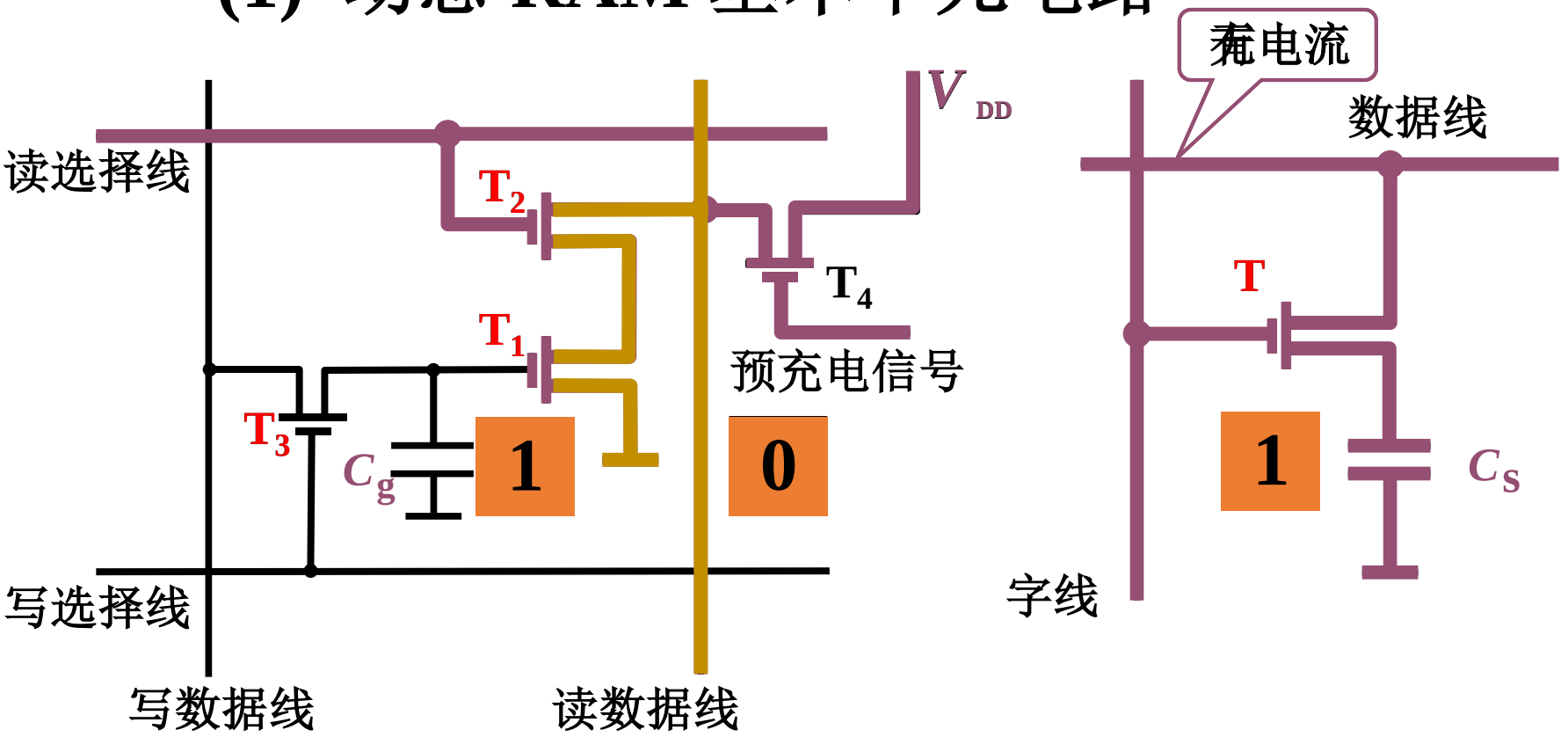
(4) 静态 RAM (2114) 写 时序



t_{DH} $\overline{WE}$ 失效后的数据维持时间

2. 动态 RAM (DRAM)

(1) 动态 RAM 基本单元电路



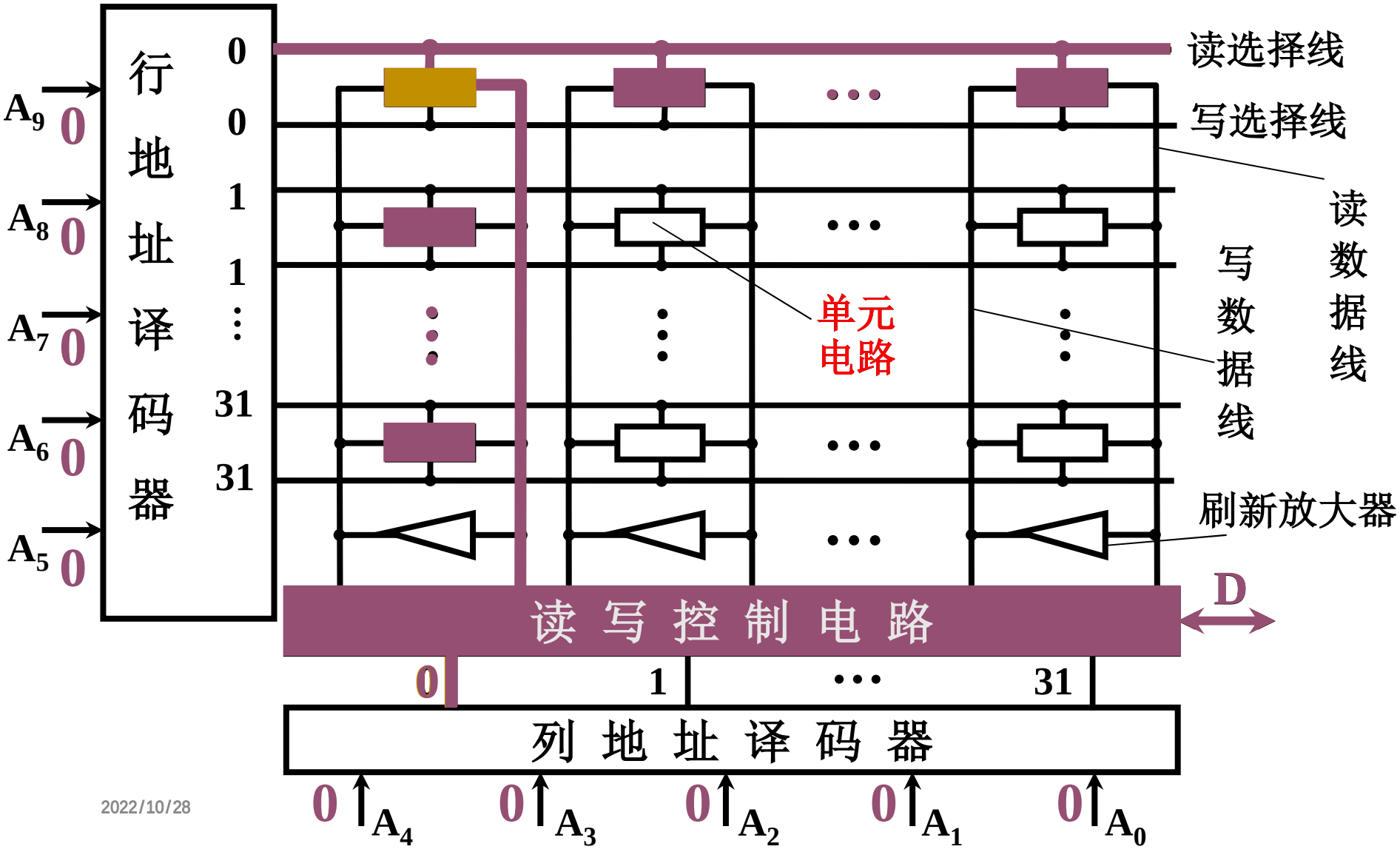
读出与原存信息相反　　读出时数据线有电流 为 “1”

写入与输入信息相同　　写入时 C_S 充电为 “1” 放电为 “0”

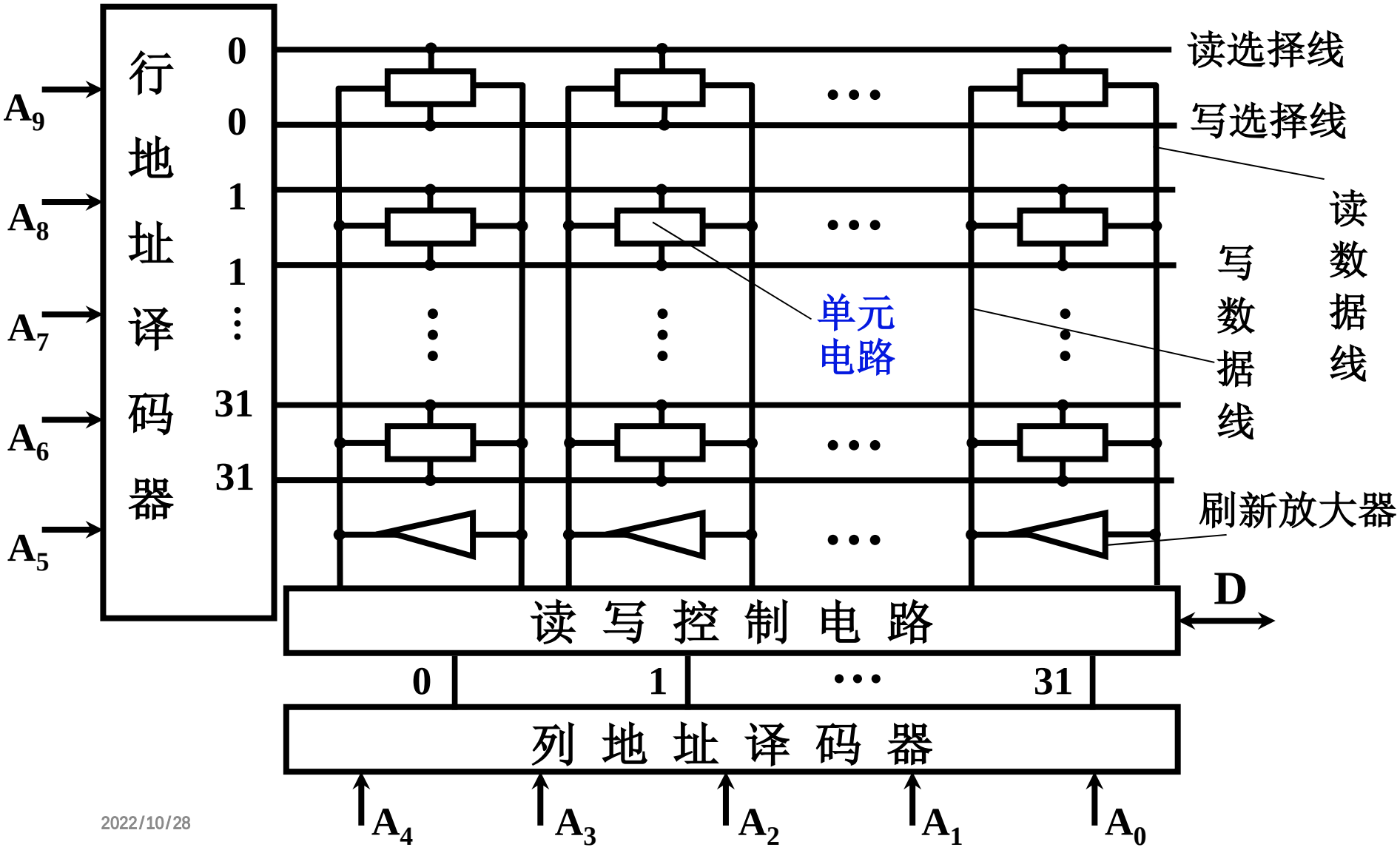
(2) 动态 RAM 芯片举例

4.2

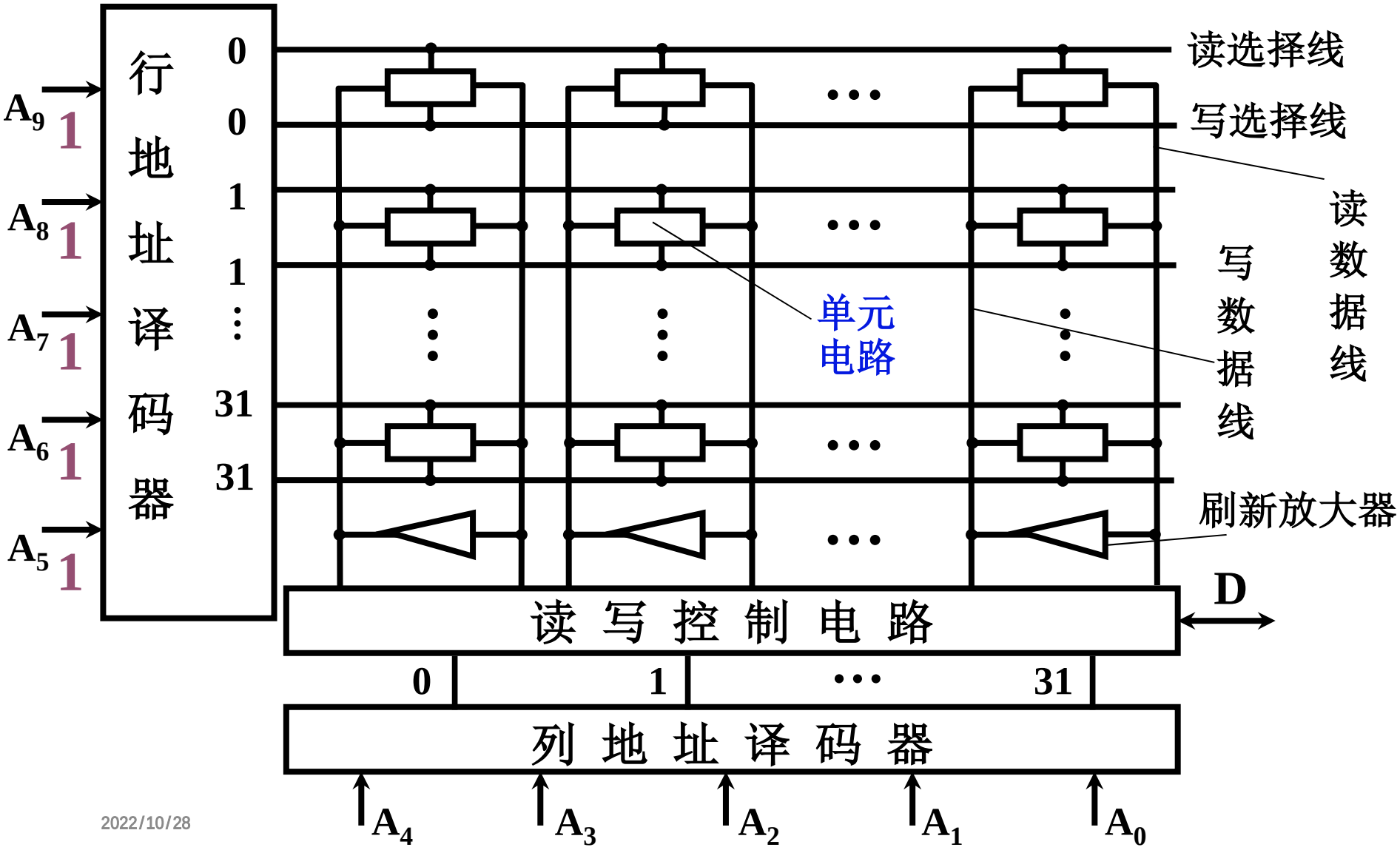
① 三管动态 RAM 芯片 (Intel 1103) 读



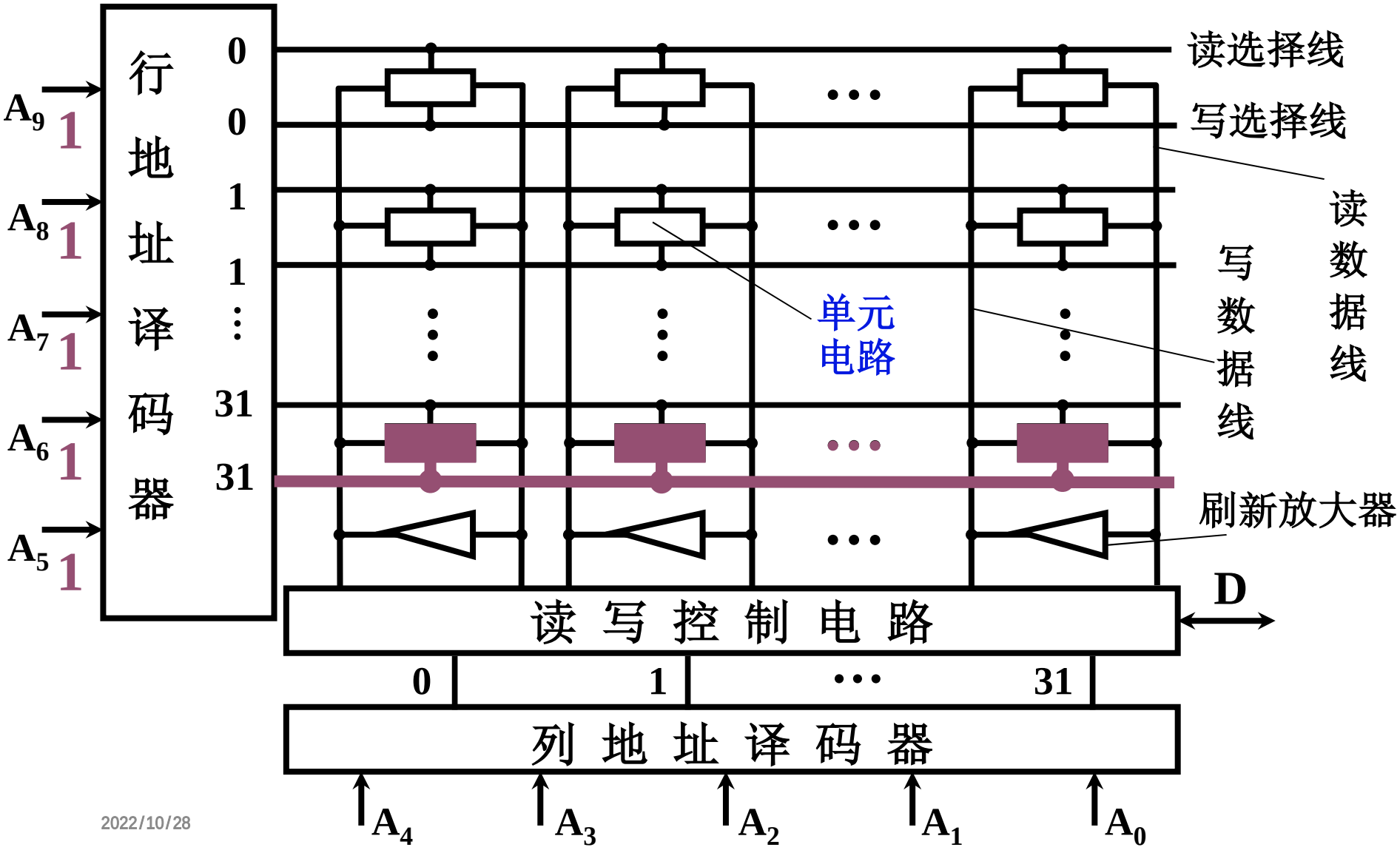
② 三管动态 RAM 芯片 (Intel 1103) 写



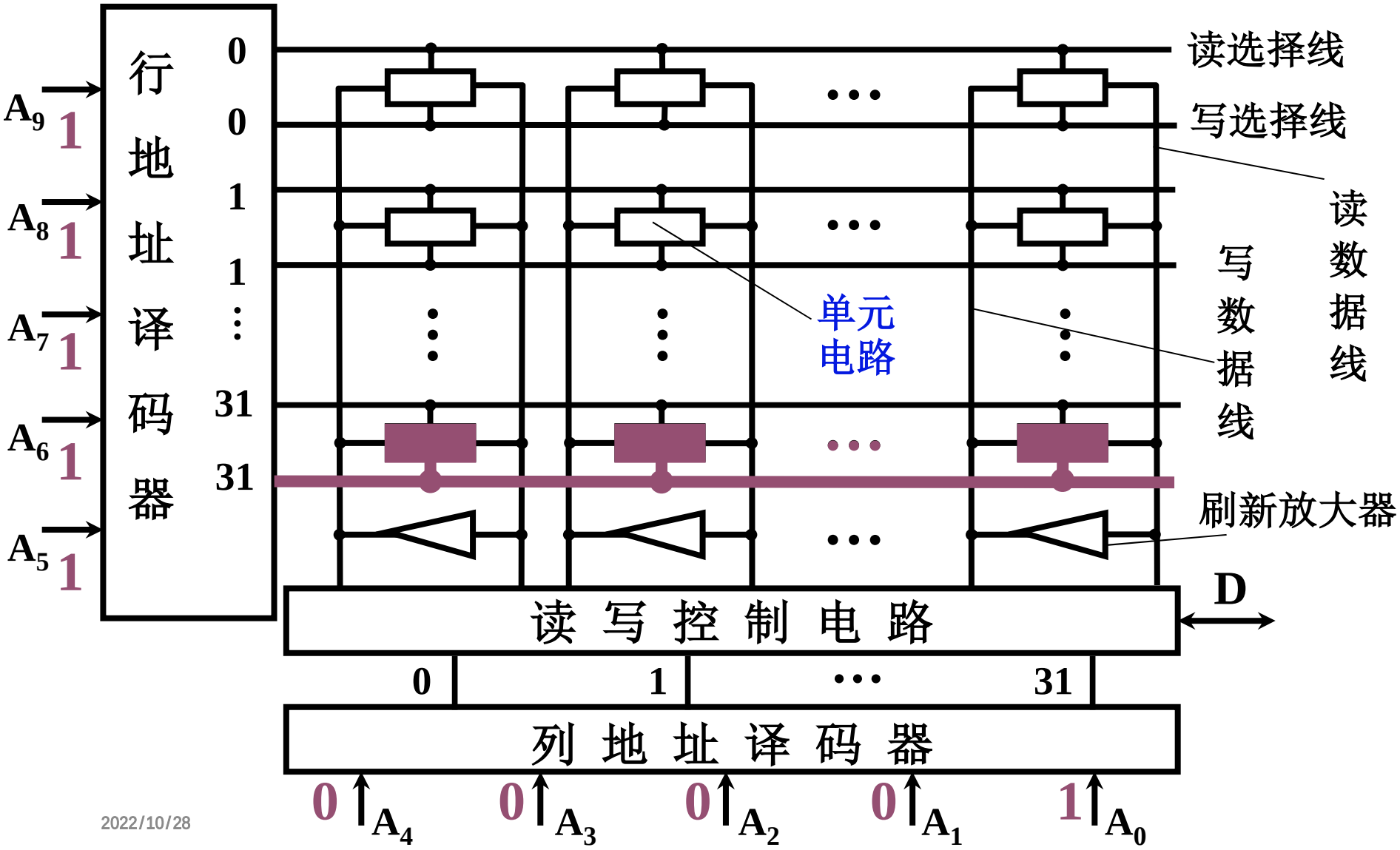
② 三管动态 RAM 芯片 (Intel 1103) 写



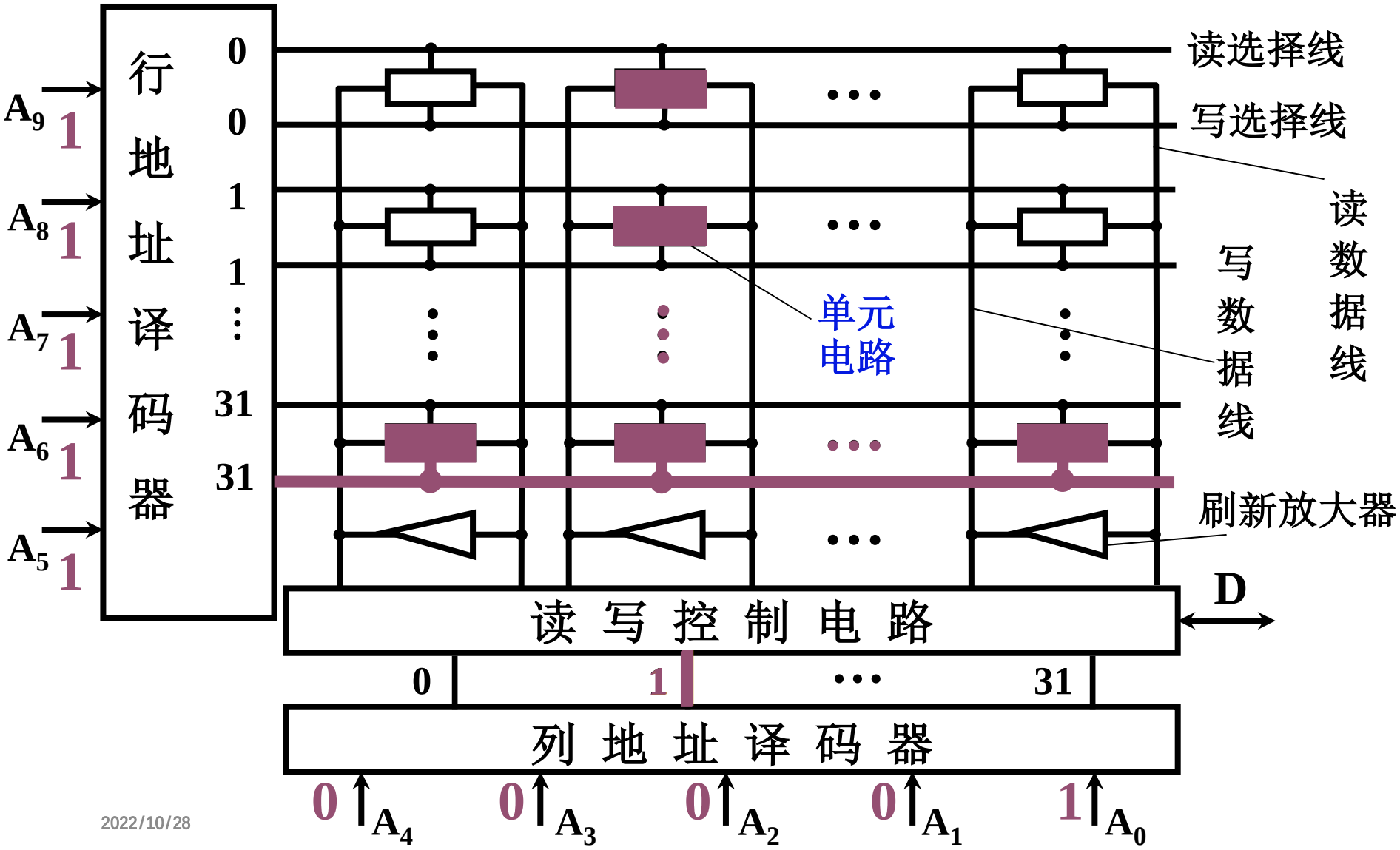
② 三管动态 RAM 芯片 (Intel 1103) 写



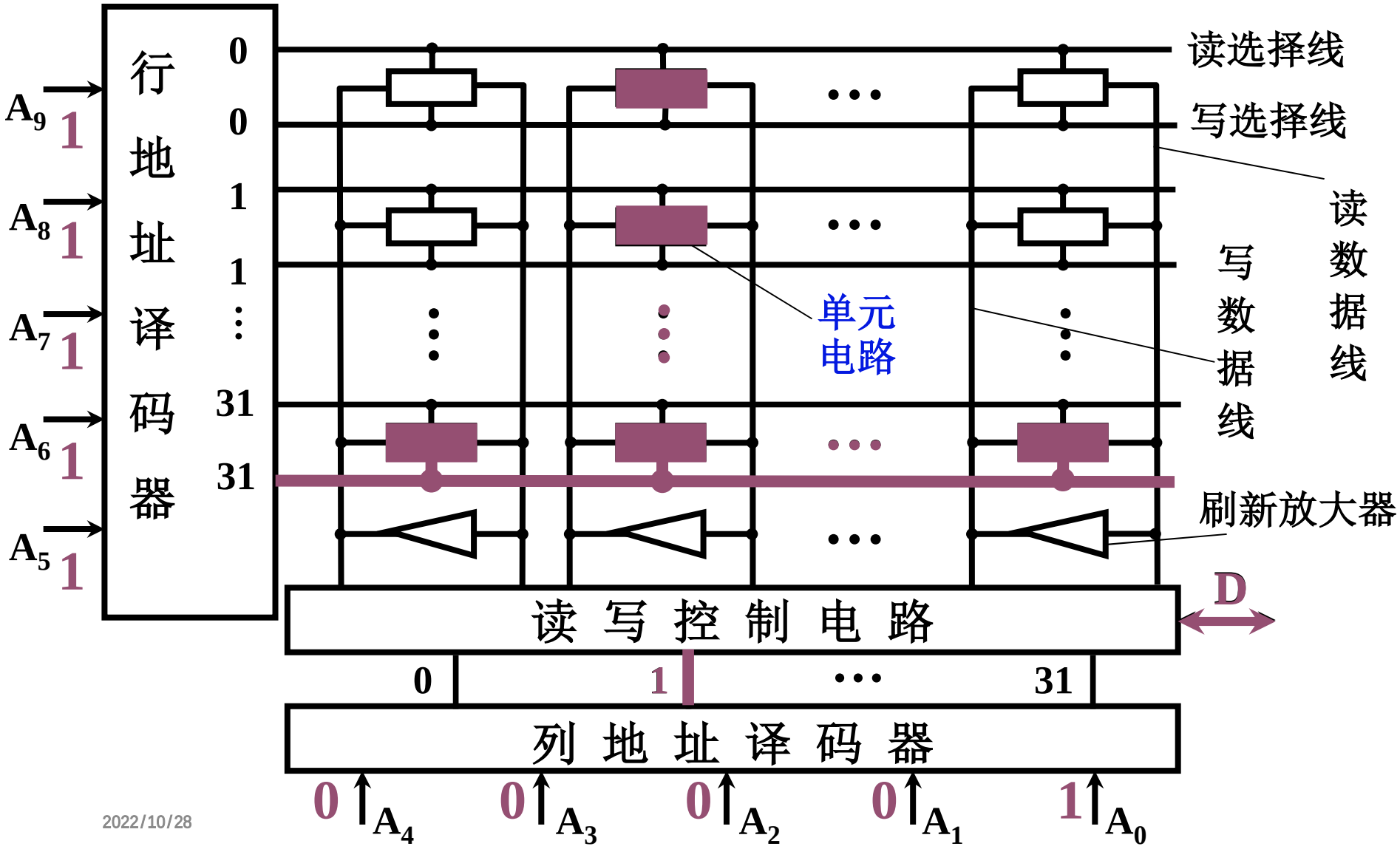
② 三管动态 RAM 芯片 (Intel 1103) 写



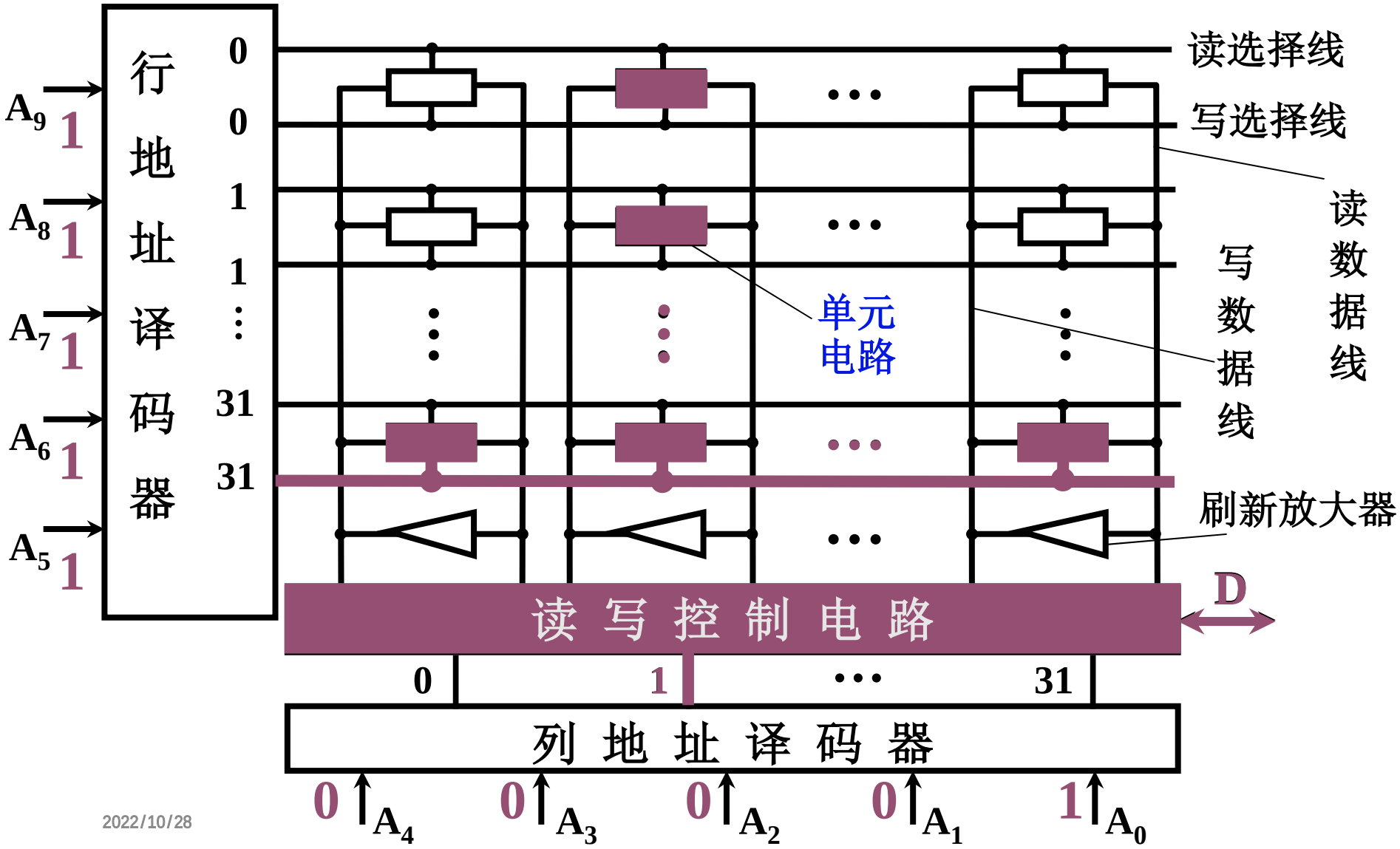
② 三管动态 RAM 芯片 (Intel 1103) 写



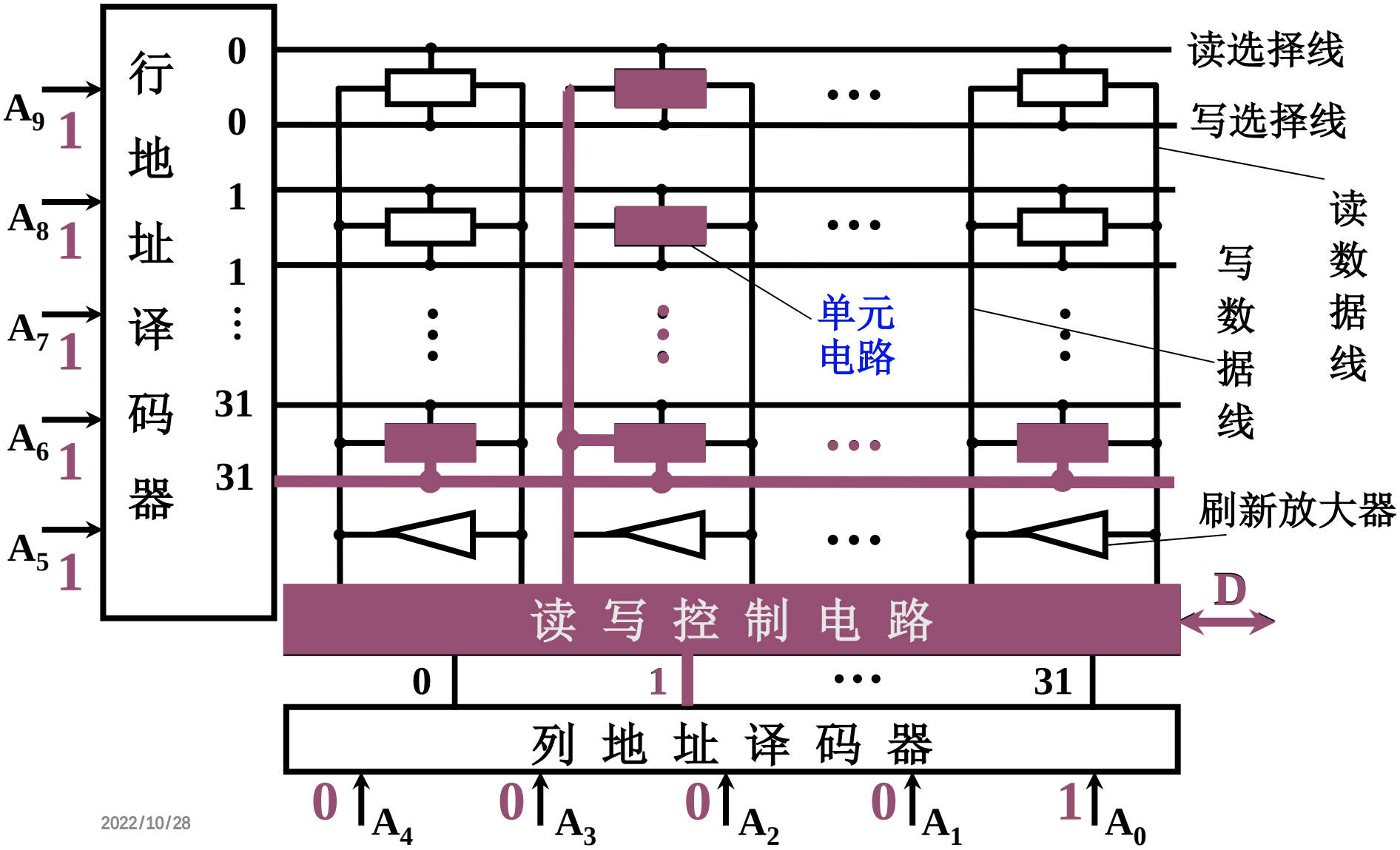
② 三管动态 RAM 芯片 (Intel 1103) 写



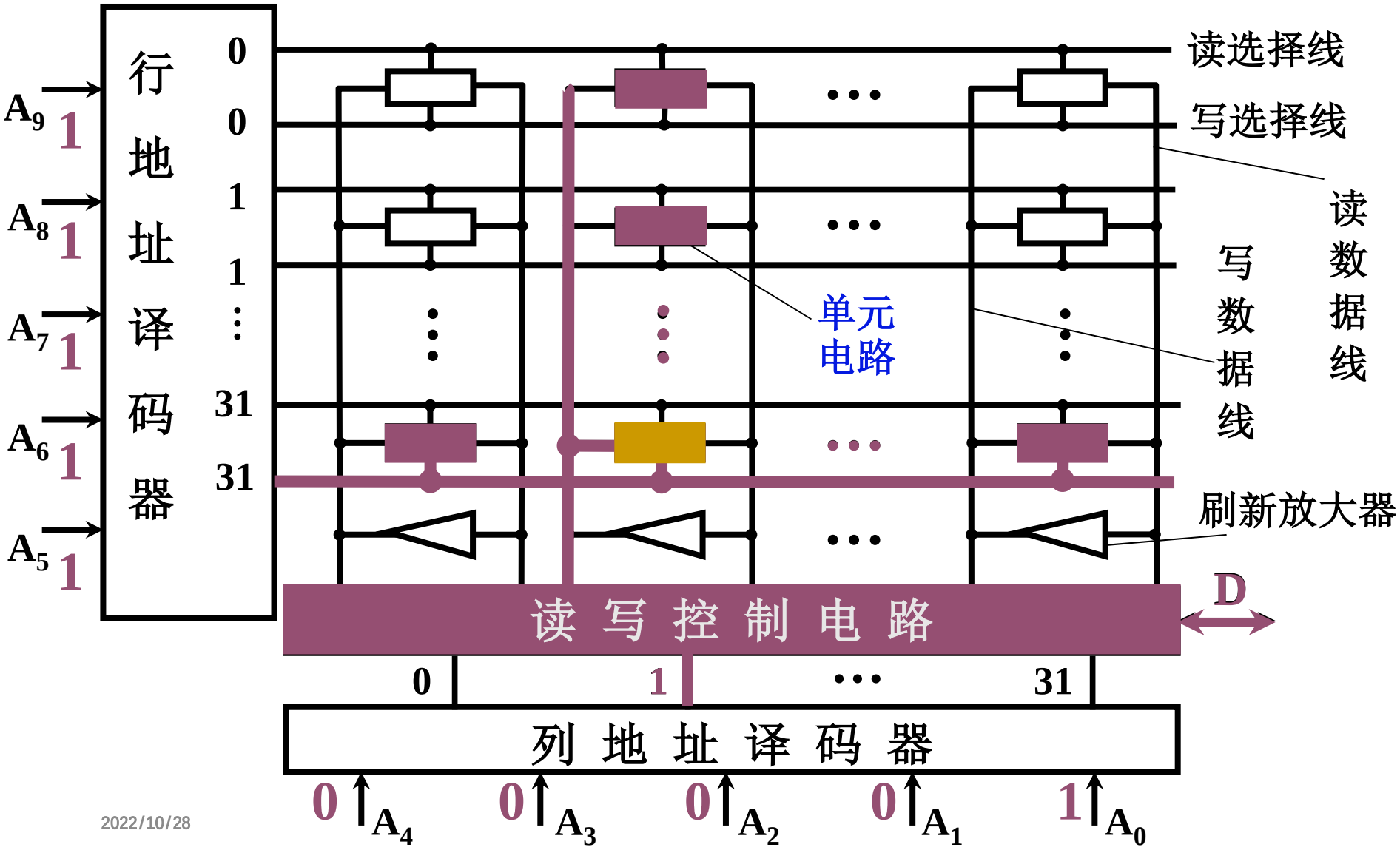
② 三管动态 RAM 芯片 (Intel 1103) 写



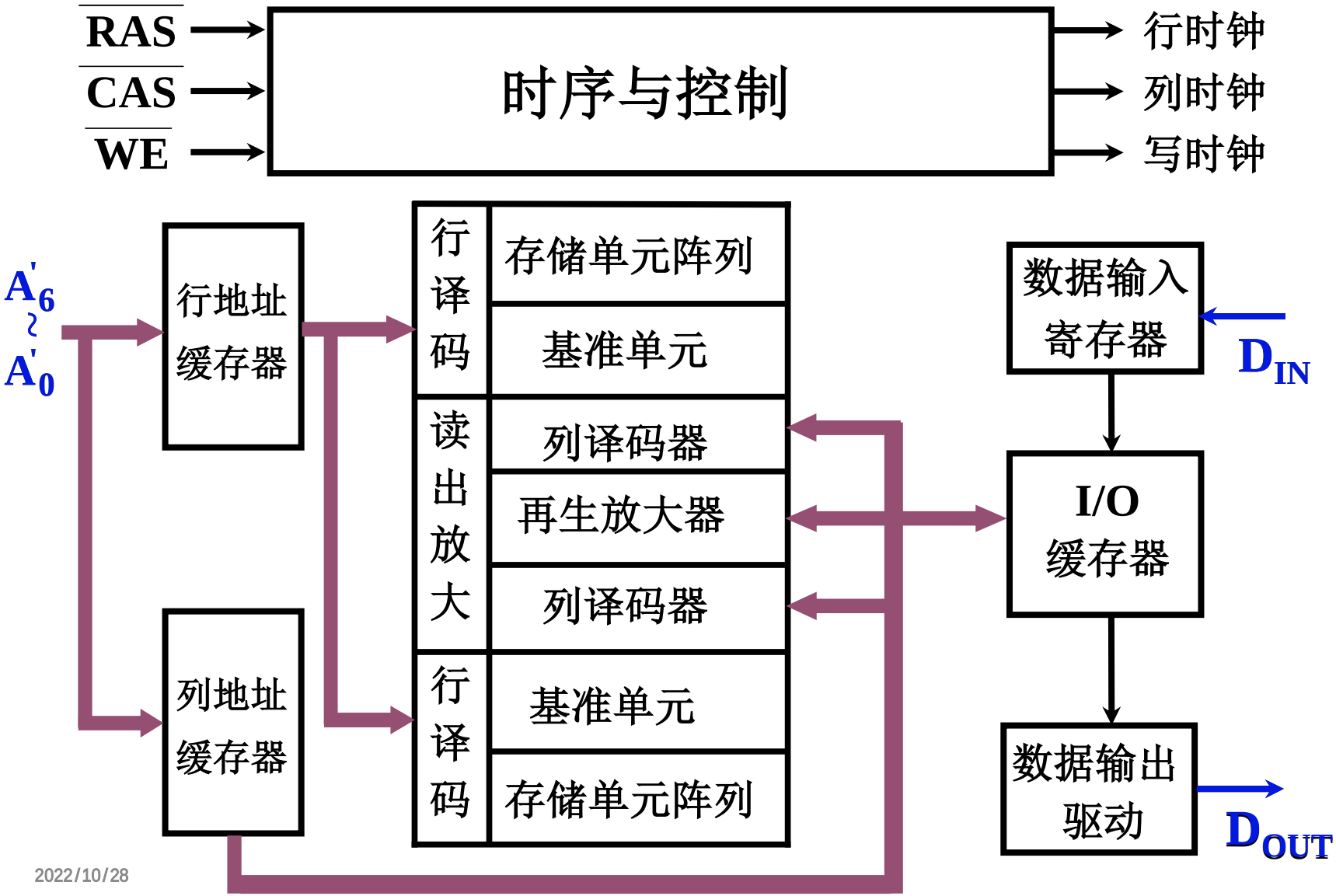
② 三管动态 RAM 芯片 (Intel 1103) 写



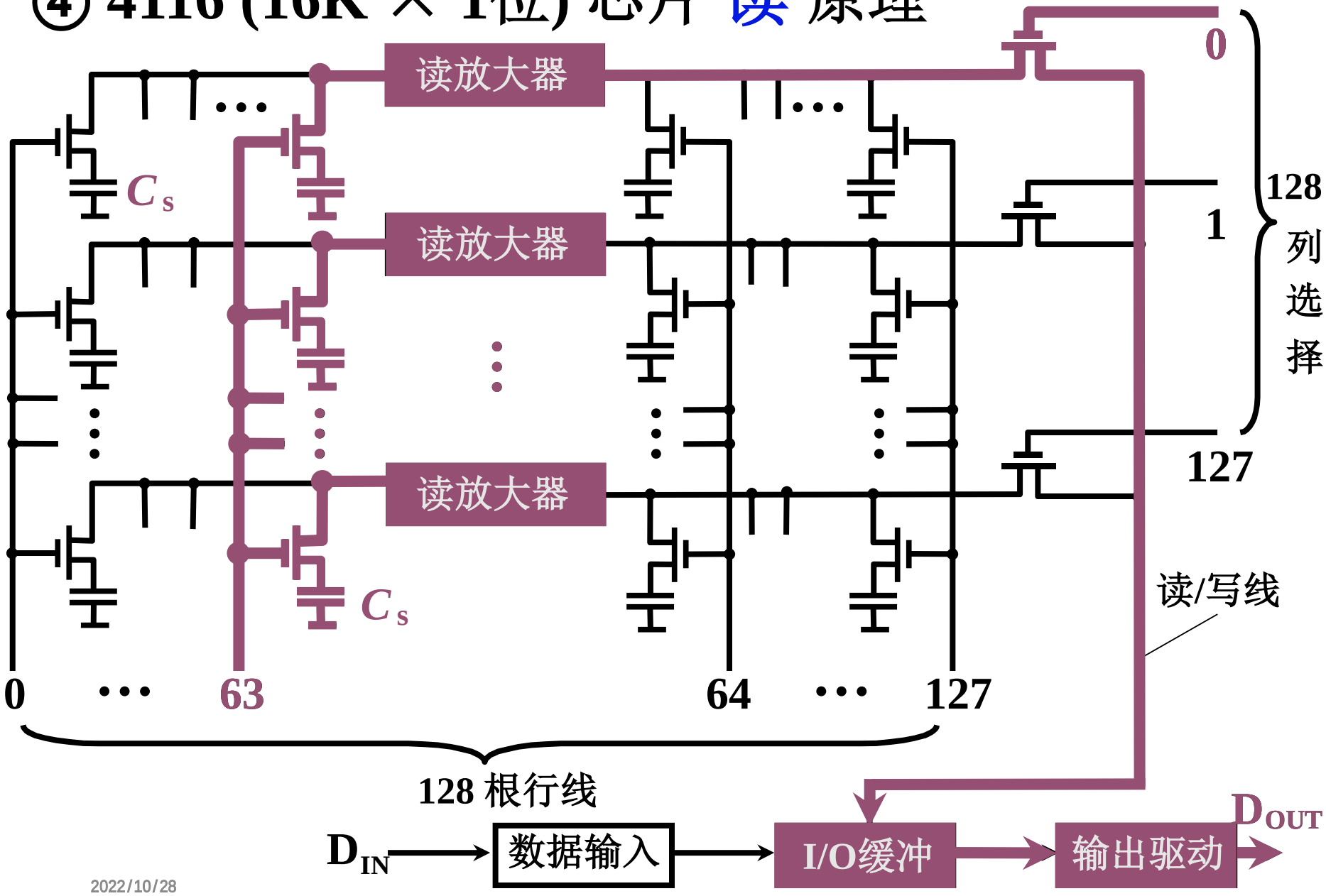
② 三管动态 RAM 芯片 (Intel 1103) 写



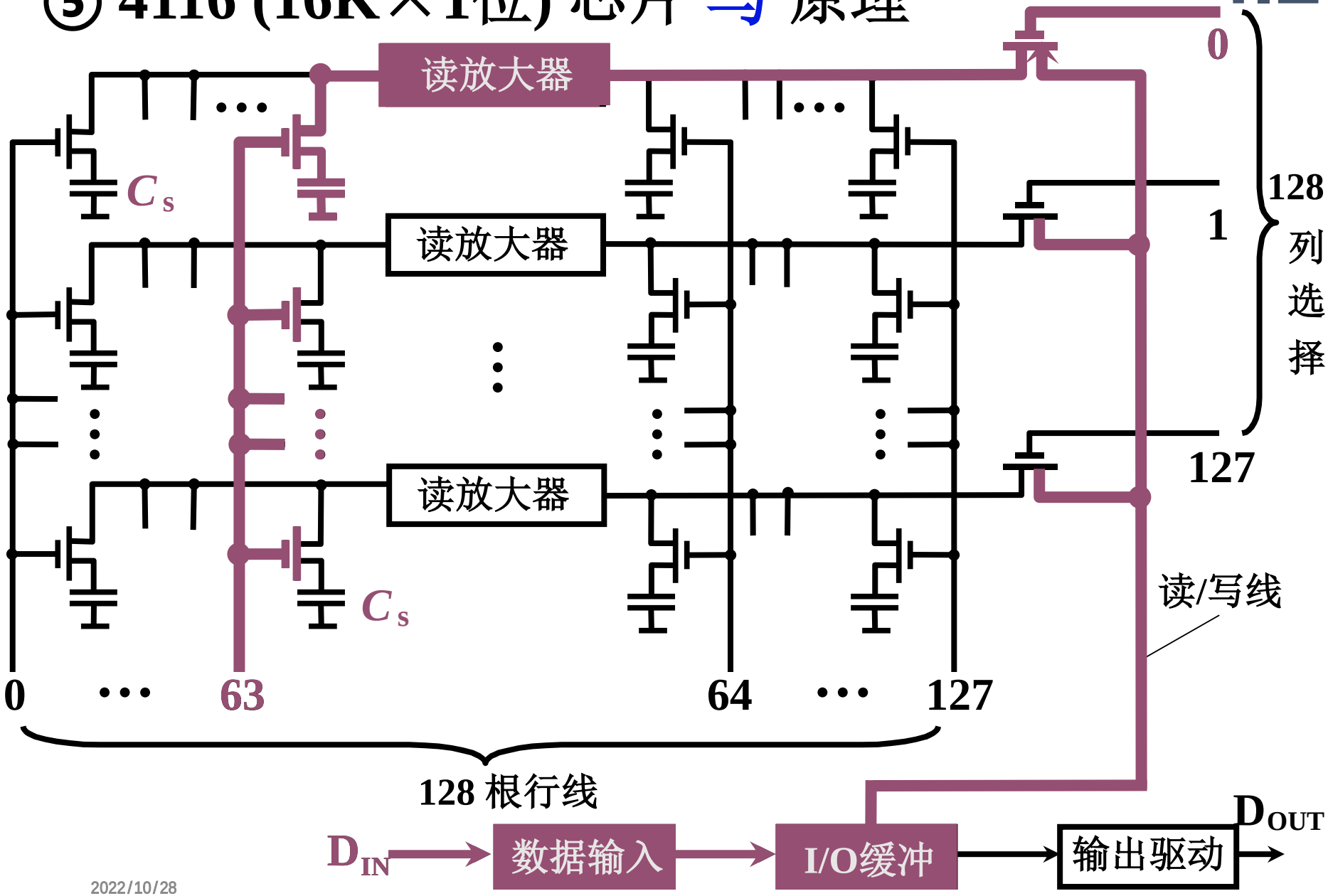
③ 单管动态 RAM 4116 (16K × 1位) 外特性 4.2



④ 4116 (16K × 1位) 芯片 读 原理



⑤ 4116 (16K×1位) 芯片 写 原理



(3) 动态 RAM 时序

行、列地址分开传送

读时序

行地址 $\overline{\text{RAS}}$ 有效

写允许 $\overline{\text{WE}}$ 有效(高)

列地址 $\overline{\text{CAS}}$ 有效

数据 D_{OUT} 有效

写时序

行地址 $\overline{\text{RAS}}$ 有效

写允许 $\overline{\text{WE}}$ 有效(低)

数据 D_{IN} 有效

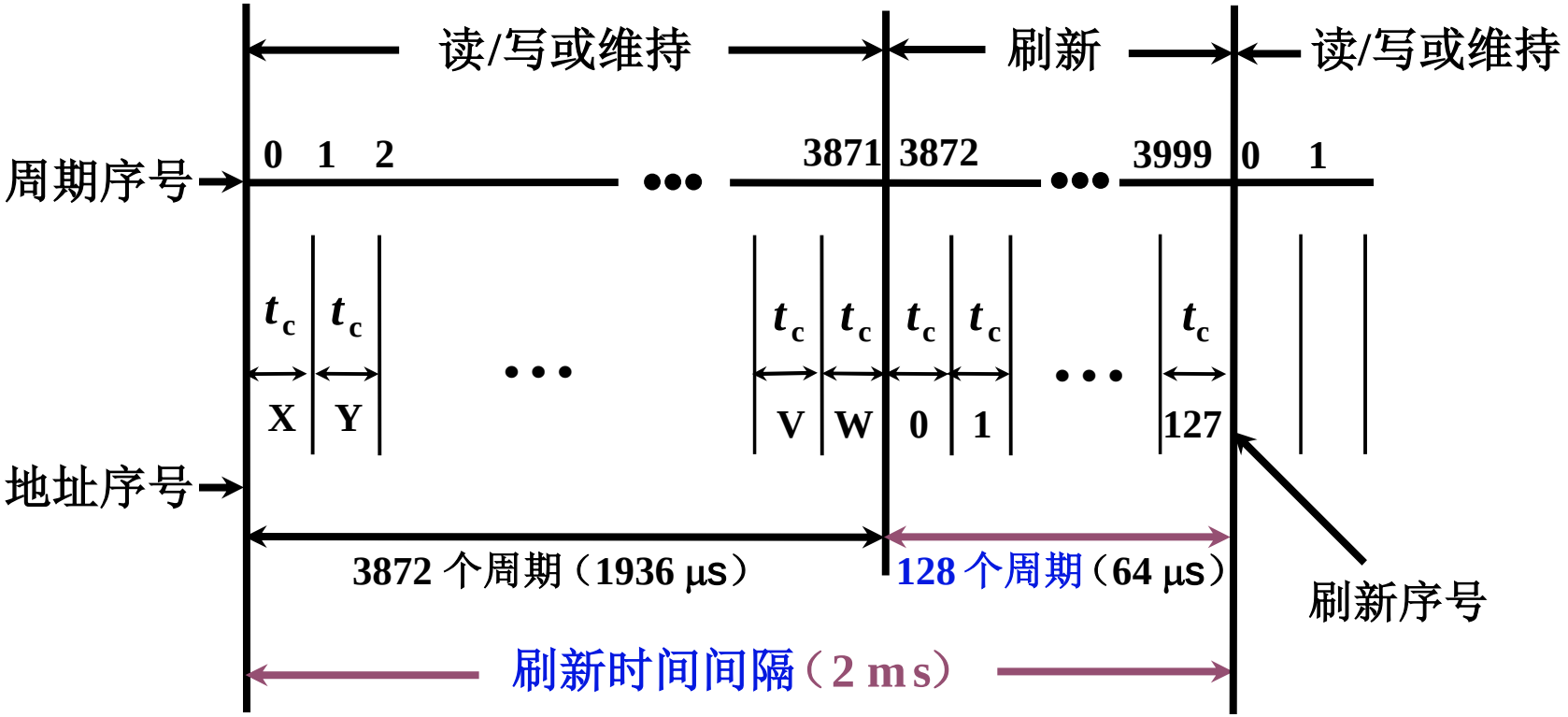
列地址 $\overline{\text{CAS}}$ 有效

(4) 动态 RAM 刷新

4.2

刷新与行地址有关

① 集中刷新 (存取周期为0.5 μs)以128 × 128 矩阵为例

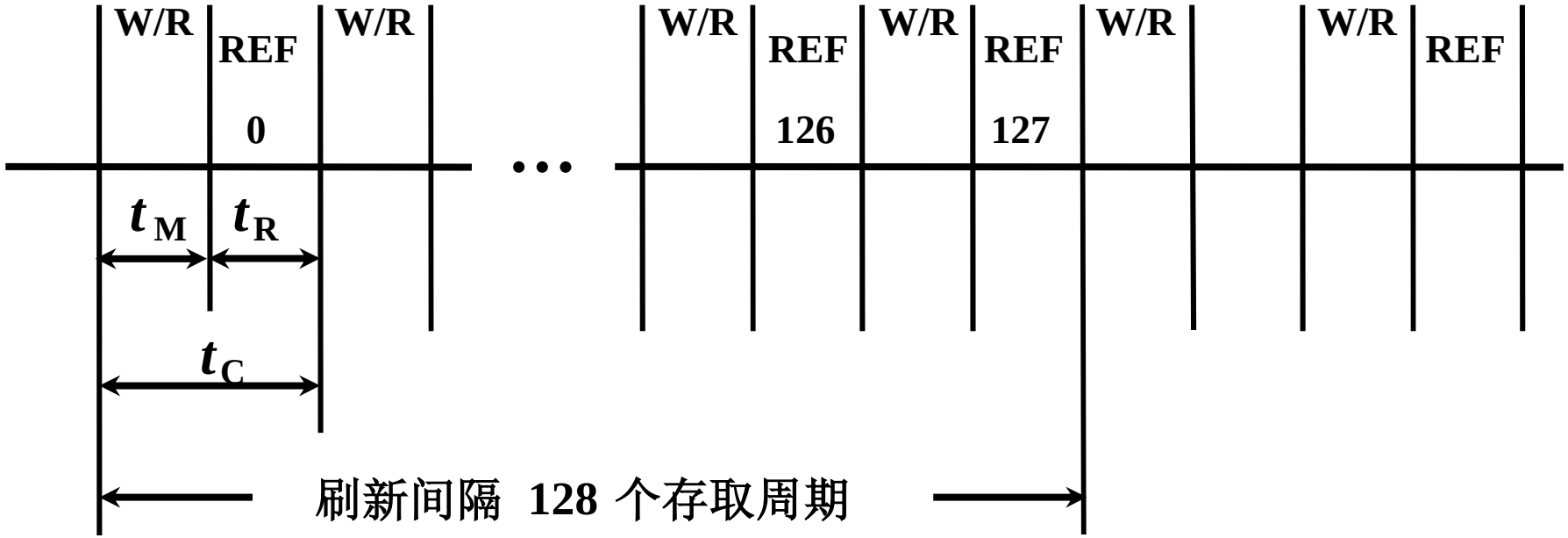


“死区” 为 0.5 μs ×128 = 64 μs

“死时间率” 为 128/4 000 ×100% = 3.2%

② 分散刷新（存取周期为1μs）

以 128 ×128 矩阵为例



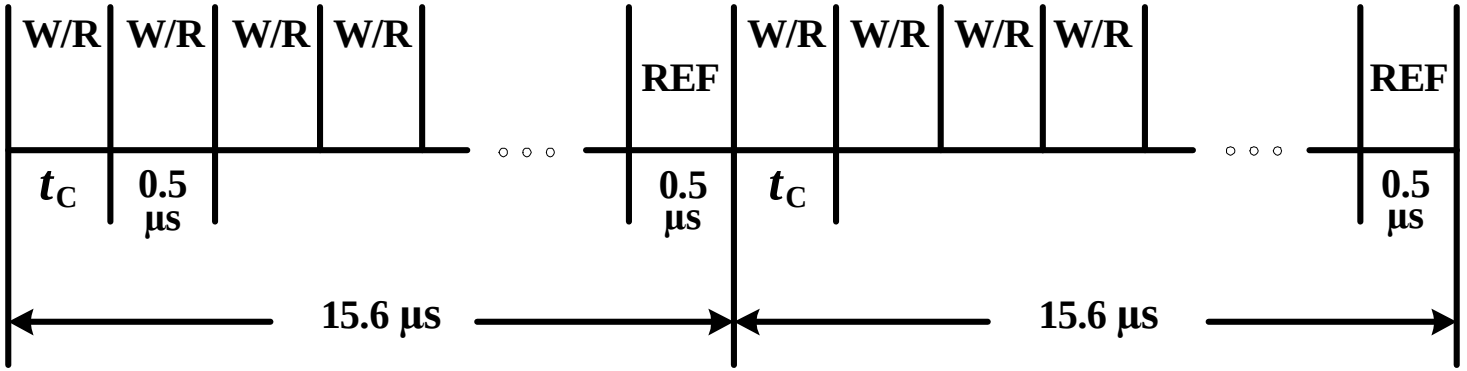
$t_C = t_M + t_R$　　无“死区”

读写　刷新　　(存取周期为 0.5 μs + 0.5 μs)

③ 分散刷新与集中刷新相结合（异步刷新）

对于 128 ×128 的存储芯片（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”

3. 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

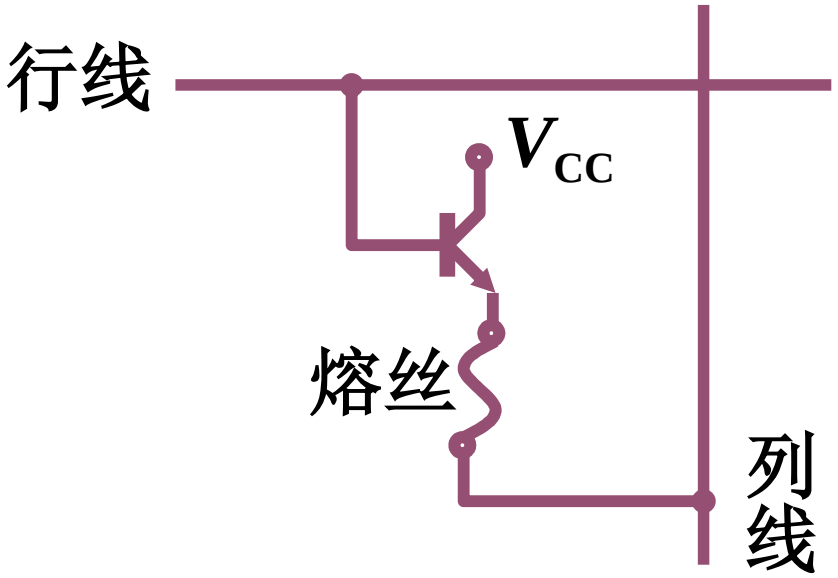
四、只读存储器（ROM）

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”

2. PROM (一次性编程)

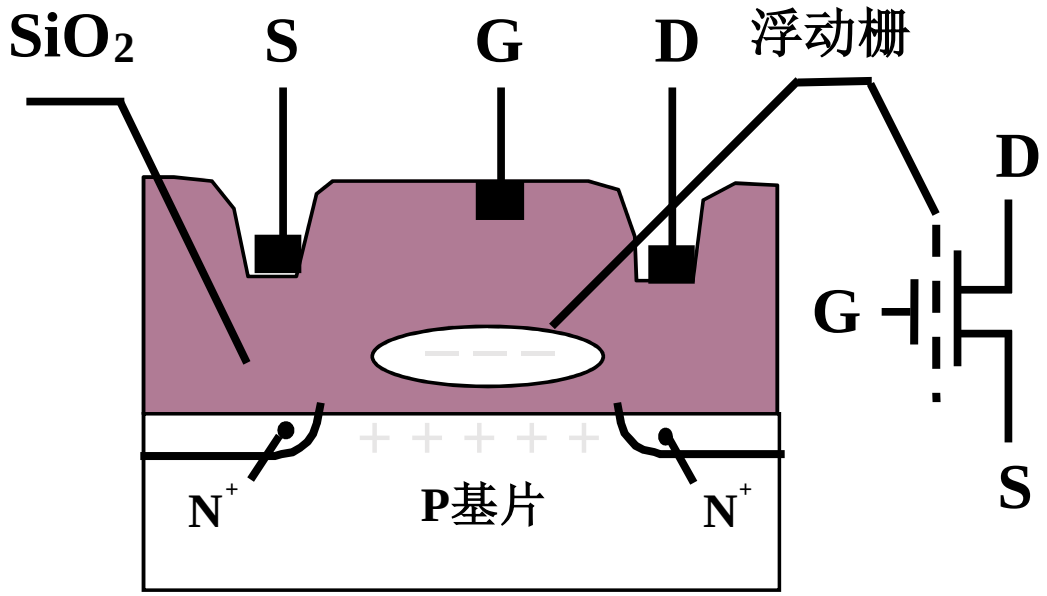


熔丝断　　为“0”

熔丝未断　为“1”

3. EPROM (多次性编程)

(1) N型沟道浮动栅 MOS 电路



G 栅极

S 源

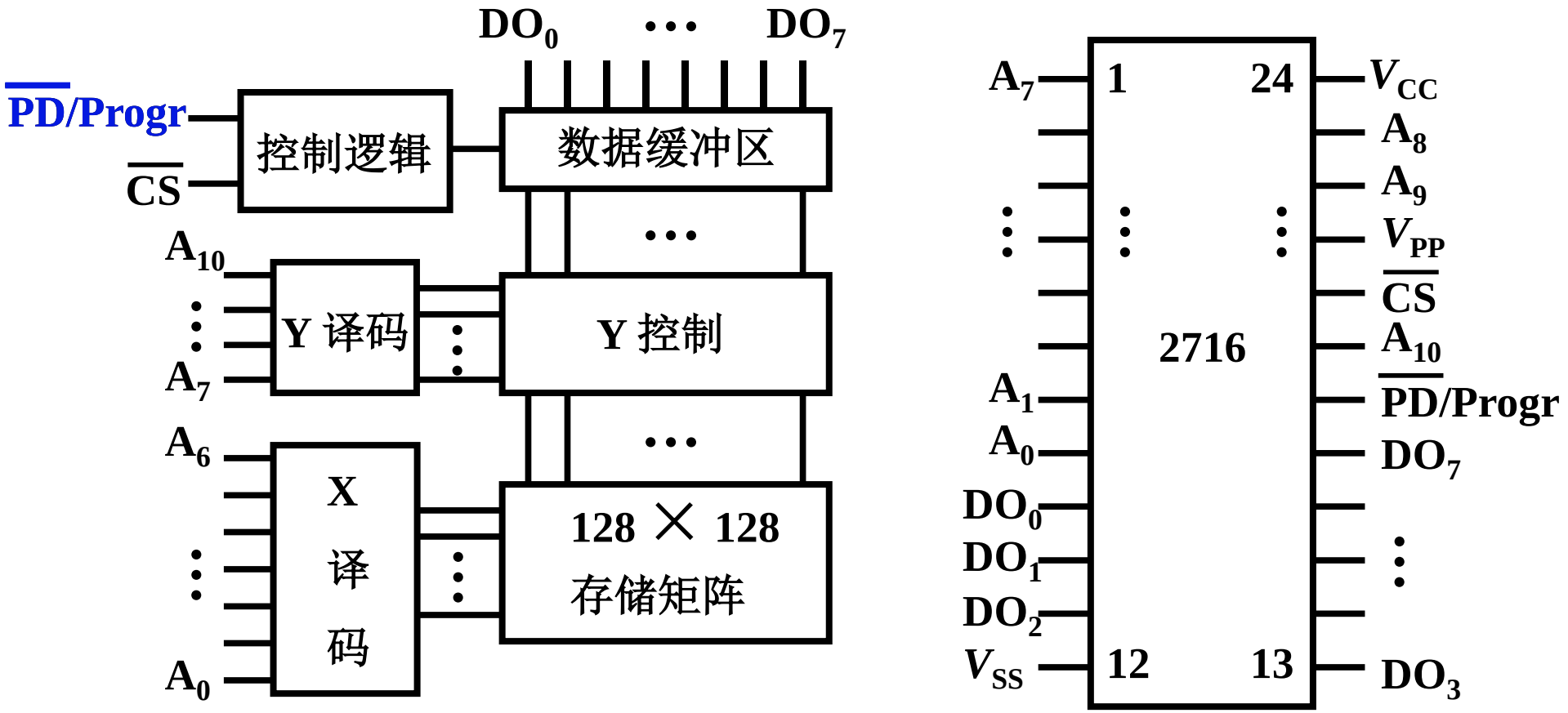
D 漏

紫外线全部擦洗

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”

(2) 2716 EPROM 的逻辑图和引脚



$\overline{PD}$/Progr 功率下降 / 编程输入端　读出时 为 低电平

4. EEPROM (多次性编程)

电可擦写

局部擦写

全部擦写

5. Flash Memory (闪速型存储器)

EPROM 价格便宜 集成度高

EEPROM 电可擦洗重写

比 **EEPROM**快 具备 **RAM** 功能

五、存储器与 CPU 的连接

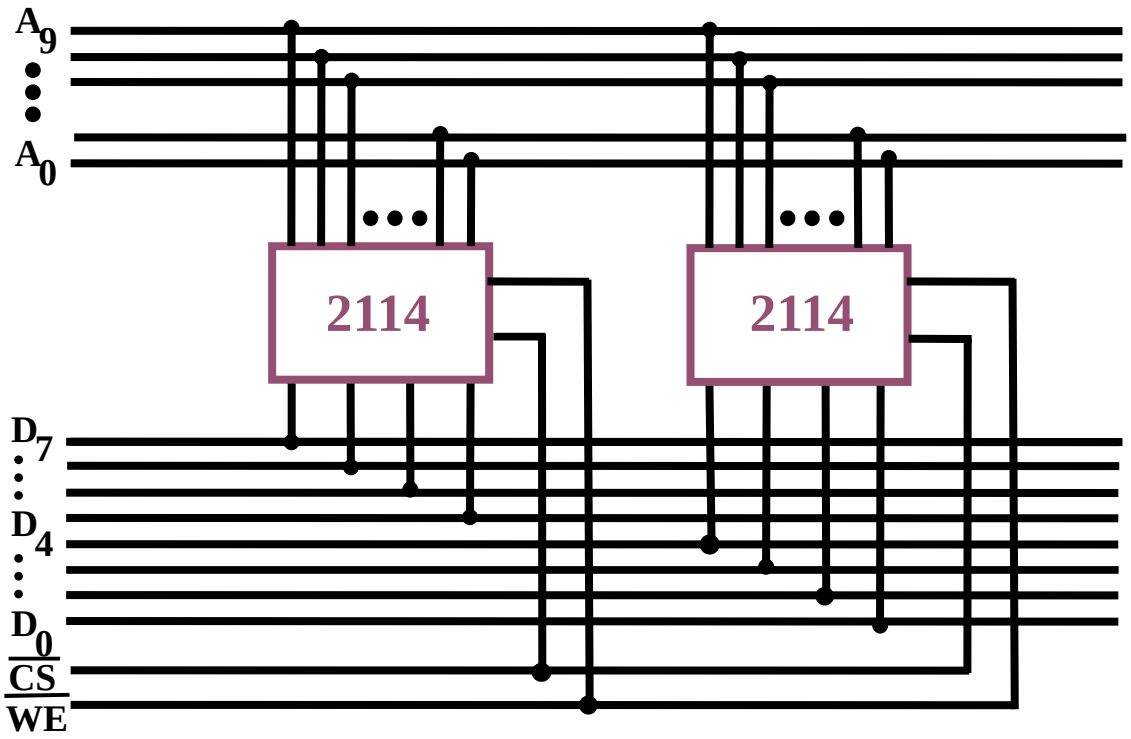
1. 存储器容量的扩展

(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

10根地址线

8根数据线

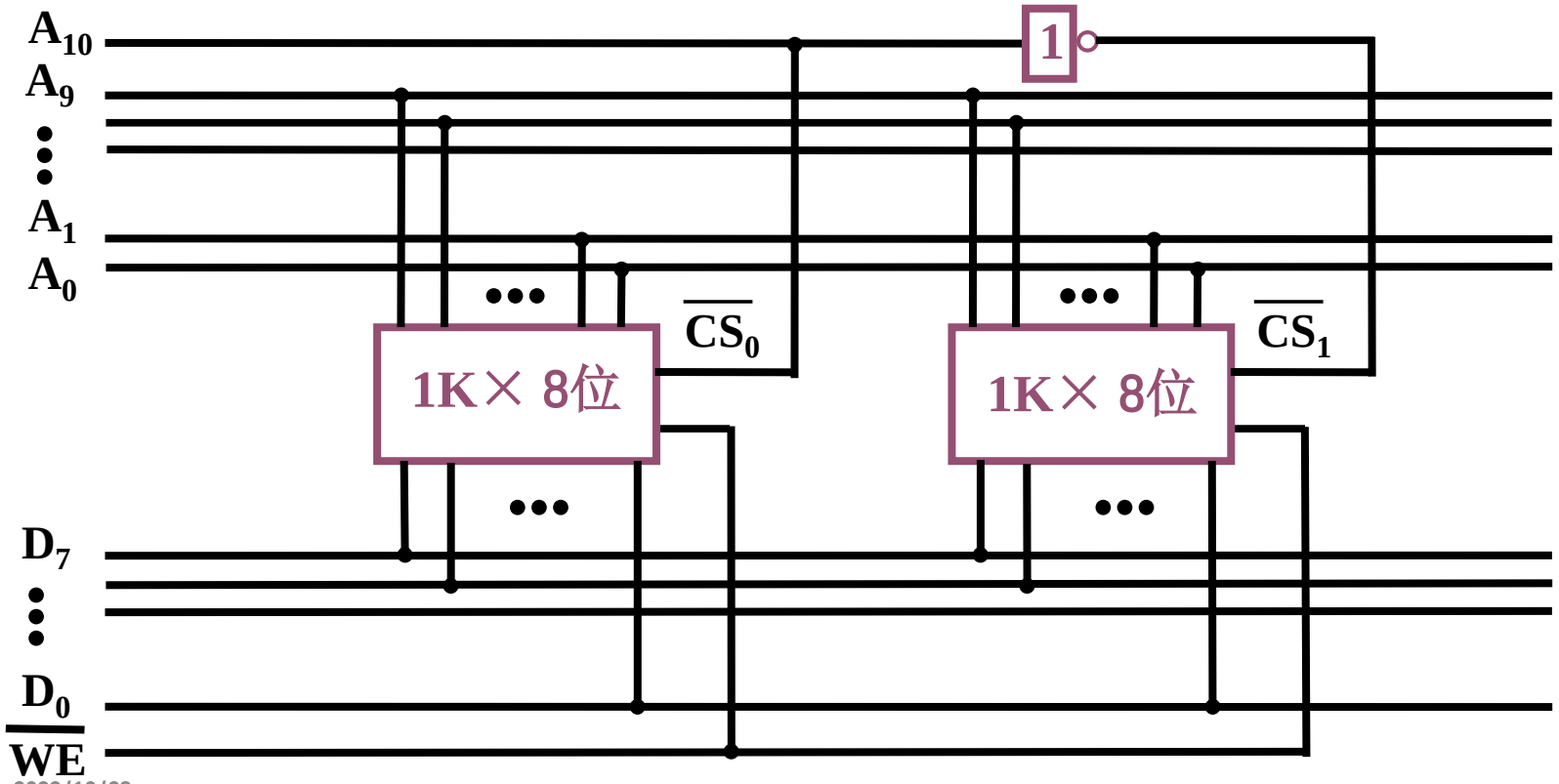


(2) 字扩展（增加存储字的数量）

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

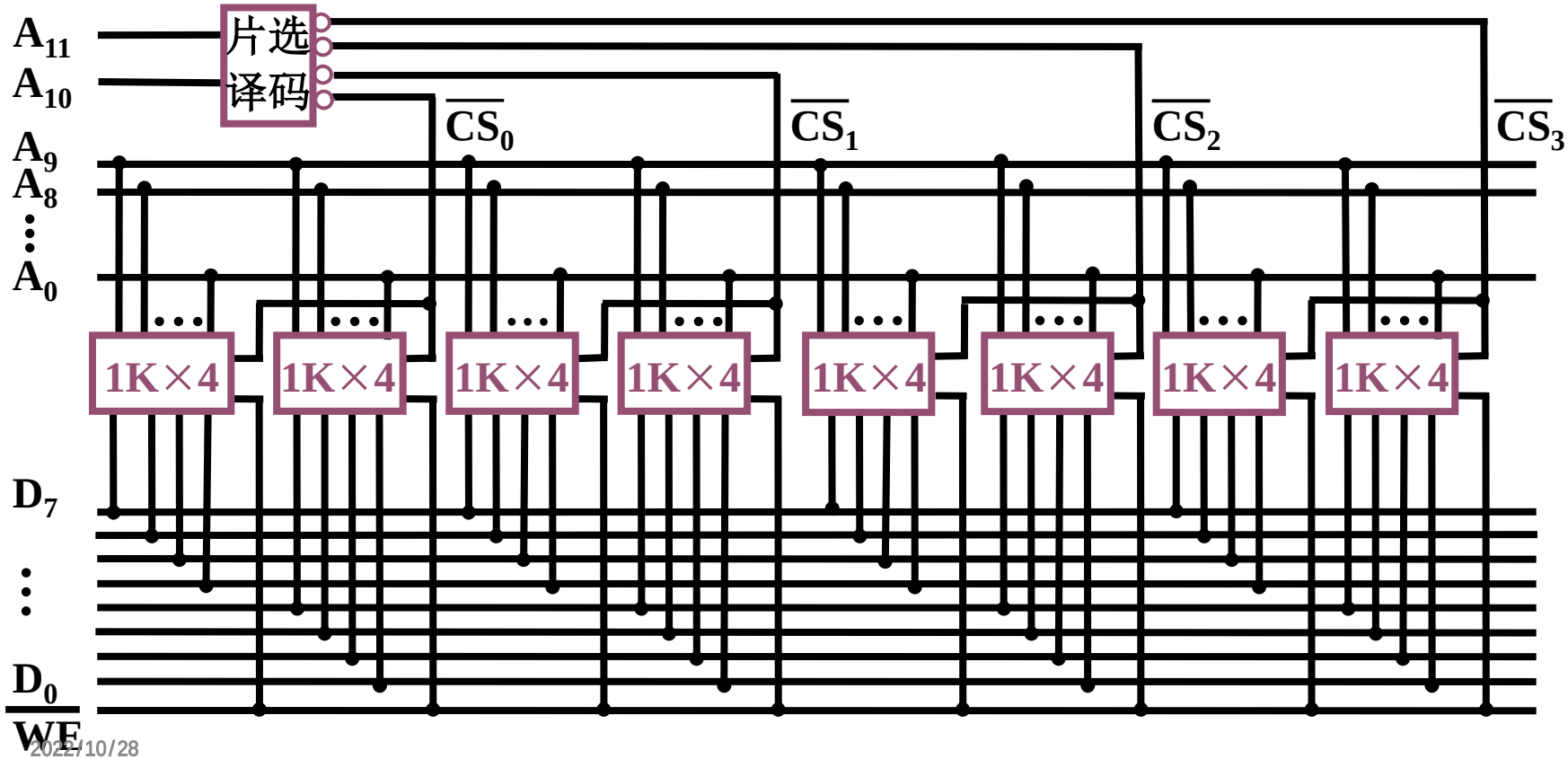


(3) 字、位扩展

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



2. 存储器与 CPU 的连接

(1) 地址线的连接

(2) 数据线的连接

(3) 读/写命令线的连接

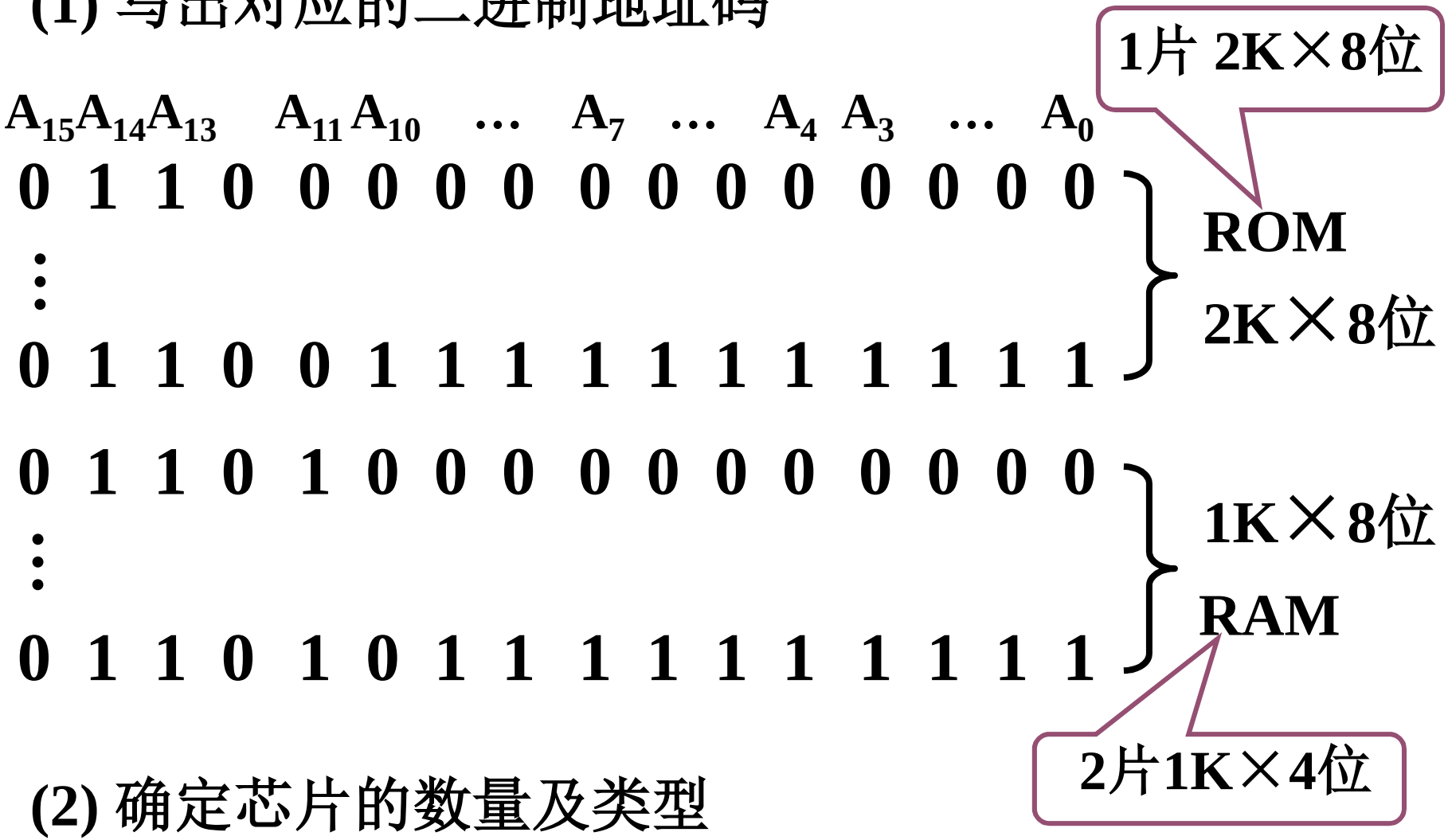
(4) 片选线的连接

(5) 合理选择存储芯片

(6) 其他　　时序、负载

例4.1 解:

(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

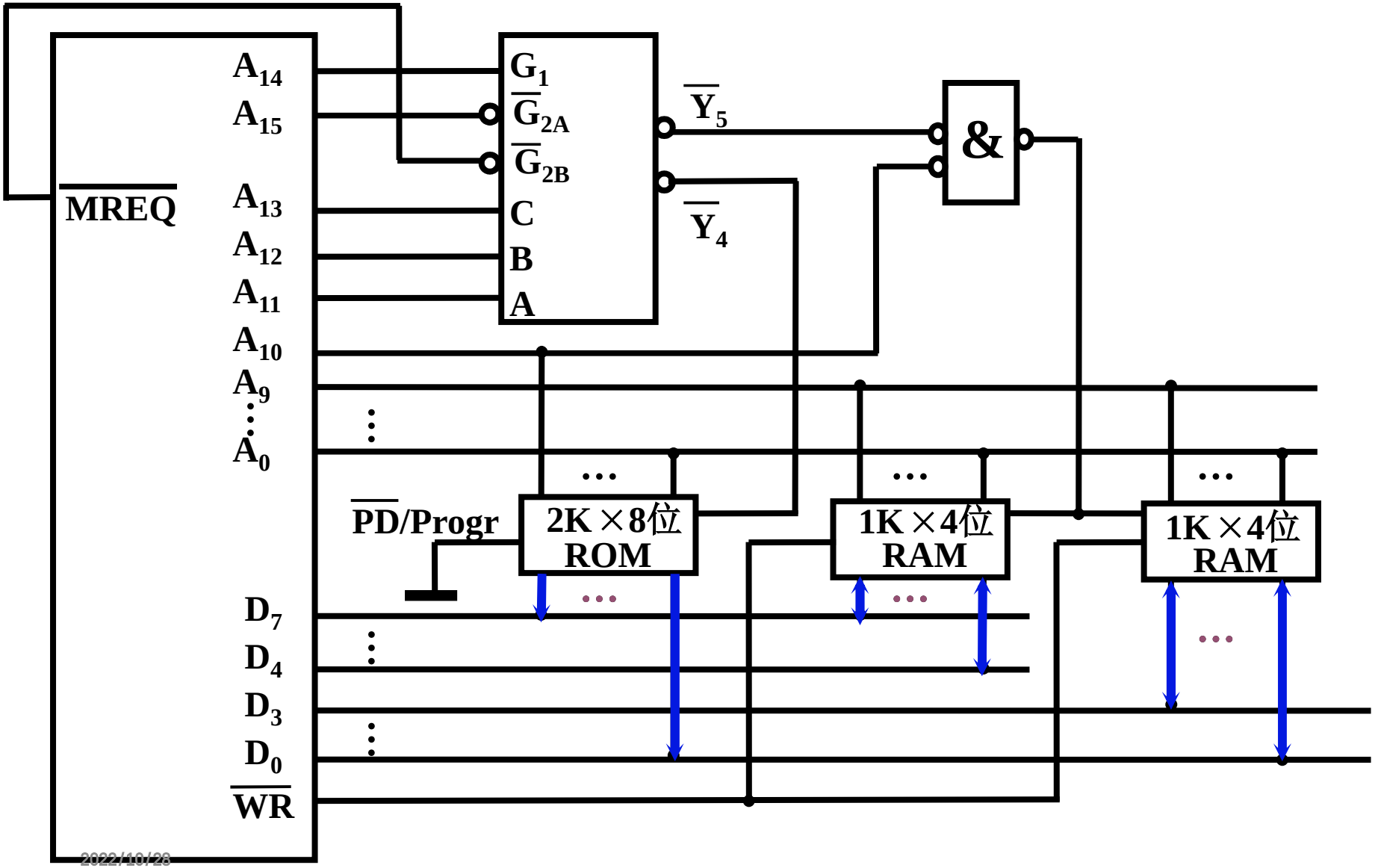
A_{15}	A_{13}	A_{11}	A_{10}	…	A_7	…	A_4	A_3	…	A_0	
0 1	1	0	0	0	0 0	0	0	0	0	0	1片 ROM
⋮											2K × 8位
0 1	1	0	0	1	1 1	1	1	1	1	1	
0 1	1	0	1	0	0 0	0	0	0	0	0	2片RAM
⋮											1K × 4位
0 1	1	0	1	0	1 1	1	1	1	1	1	
	C	B	A								

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 4.1 CPU 与存储器的连接图



例4.2 假设同前，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

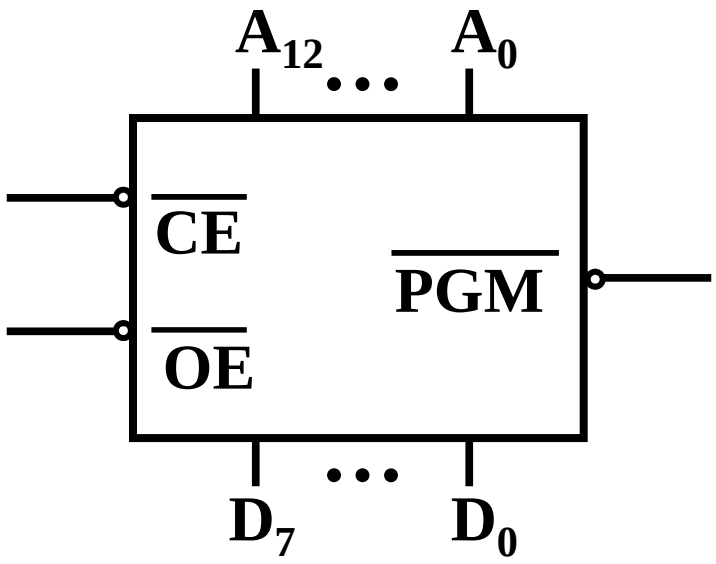
1片 4K×8位 ROM 2片 4K×8位 RAM

(3) 分配地址线

A_{11}~A_0 接 ROM 和 RAM 的地址线

(4) 确定片选信号

例 4.3 设 CPU 有 20 根地址线，8 根数据线。并用 IO/$\overline{M}$ 作访存控制信号。$\overline{RD}$ 为读命令，$\overline{WR}$ 为写命令。现有 2764 EPROM (8K × 8位), 外特性如下：



$\overline{CE}$ 片选信号

$\overline{OE}$ 允许输出

$\overline{PGM}$ 可编程端

用 138 译码器及其他门电路（门电路自定）画出 CPU 和 2764 的连接图。要求地址为 F0000H~FFFFFH , 并写出每片 2764 的地址范围。

六、存储器的校验

4.2

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　$L=3$

D —— 检测错误的位数　具有 一位 纠错能力

C —— 纠正错误的位数

汉明码是具有一位纠错能力的编码

2. 汉明码的组成

组成汉明码的三要素

汉明码的组成需增添？位检测位

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

检测位的位置？

$$2^i \ (i = 0，\ 1，\ 2，\ 3，\cdots)$$

检测位的取值？

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 所承担的检测小组为

C_1 检测的 g_1 小组包含第 1，3，5，7，9，11，…

C_2 检测的 g_2 小组包含第 2，3，6，7，10，11，…

C_4 检测的 g_3 小组包含第 4，5，6，7，12，13，…

C_8 检测的 g_4 小组包含第 8，9，10，11，12，13，14，15，24，…

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4.4 求 0101 按“偶校验”配置的汉明码

解：∵ $n = 4$

根据 $2^k \geq n + k + 1$

得 $k = 3$

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**

练习1　按配偶原则配置 0011 的汉明码　4.2

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4.5　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

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

$P_2 = 2 \oplus 3 \oplus 6 \oplus 7 = 1$　有错

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

$\therefore \quad P_4P_2P_1 = 110$

第 6 位出错，可纠正为 0100101，

故要求传送的信息为 0101。

练习2 写出按偶校验配置的汉明码

0101101 的纠错过程

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

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

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

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

练习3 按配奇原则配置 0011 的汉明码

配奇的汉明码为 0101011

4.2 主存储器

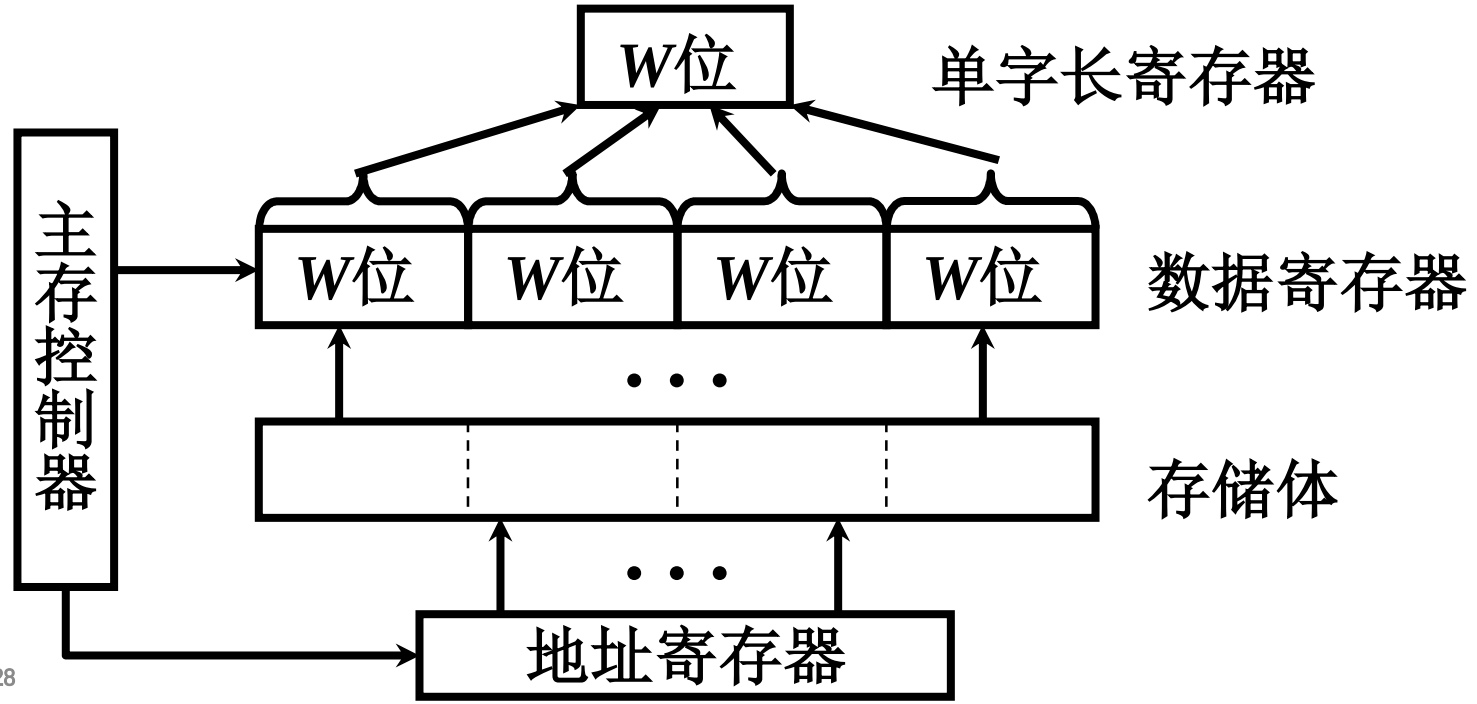
- 一、概述
- 二、半导体存储芯片简介
- 三、随机存取存储器 (RAM)
- 四、只读存储器（ROM）
- 五、存储器与 CPU 的连接
- 六、存储器的校验
- 七、提高访存速度的措施

七、提高访存速度的措施

- 采用高速器件
- 采用层次结构 Cache –主存
- 调整主存结构

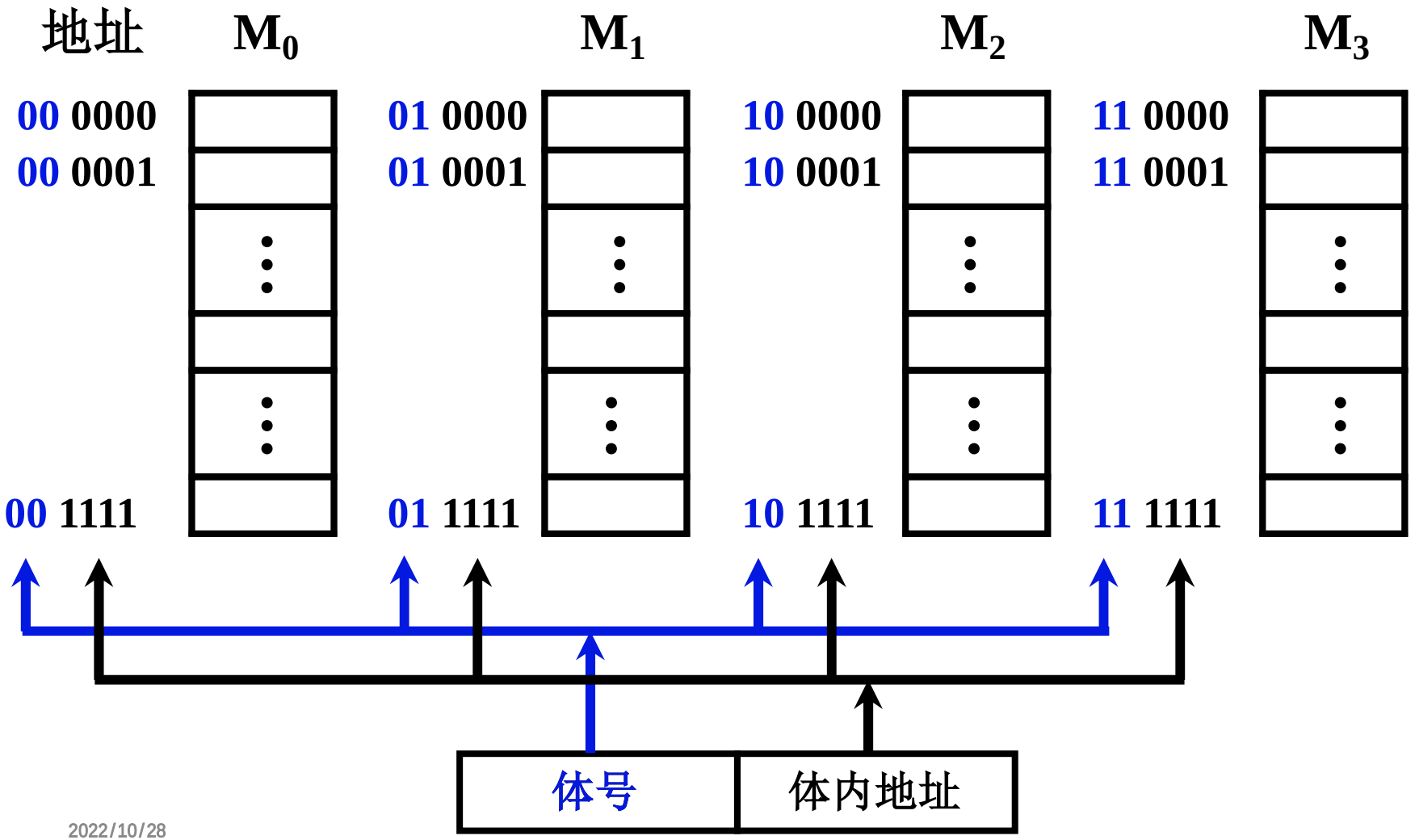
1. 单体多字系统

增加存储器的带宽

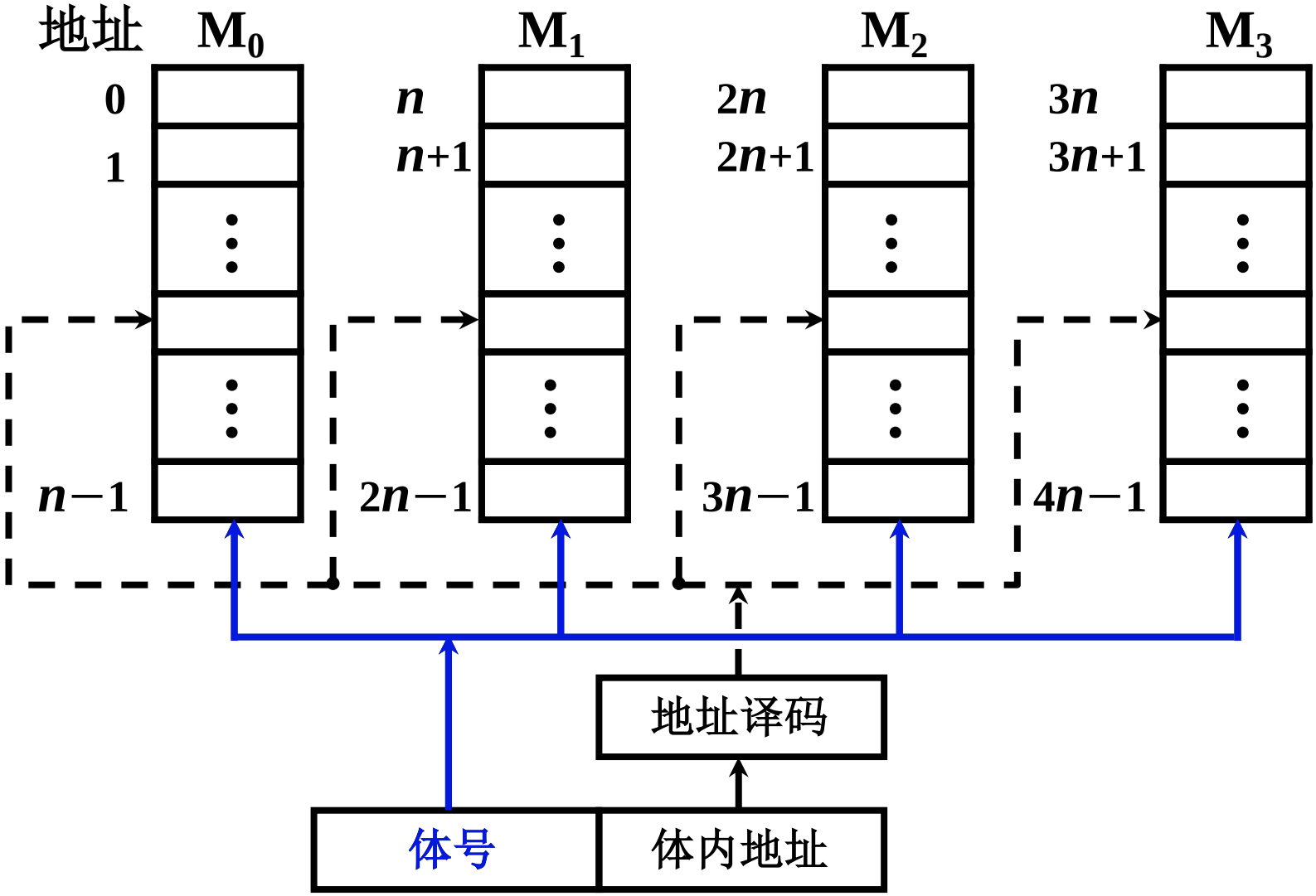


2. 多体并行系统

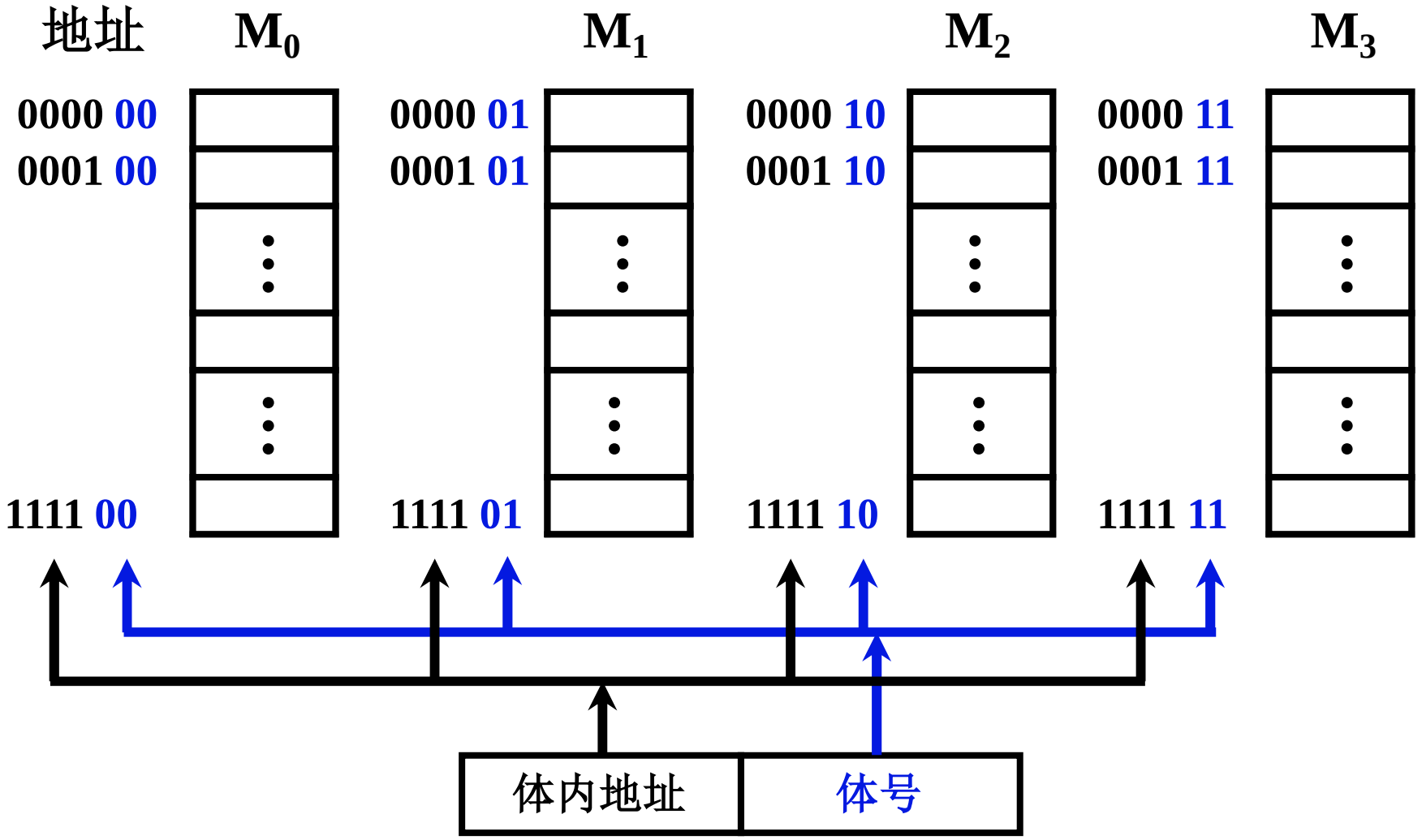
(1) 高位交叉　　顺序编址



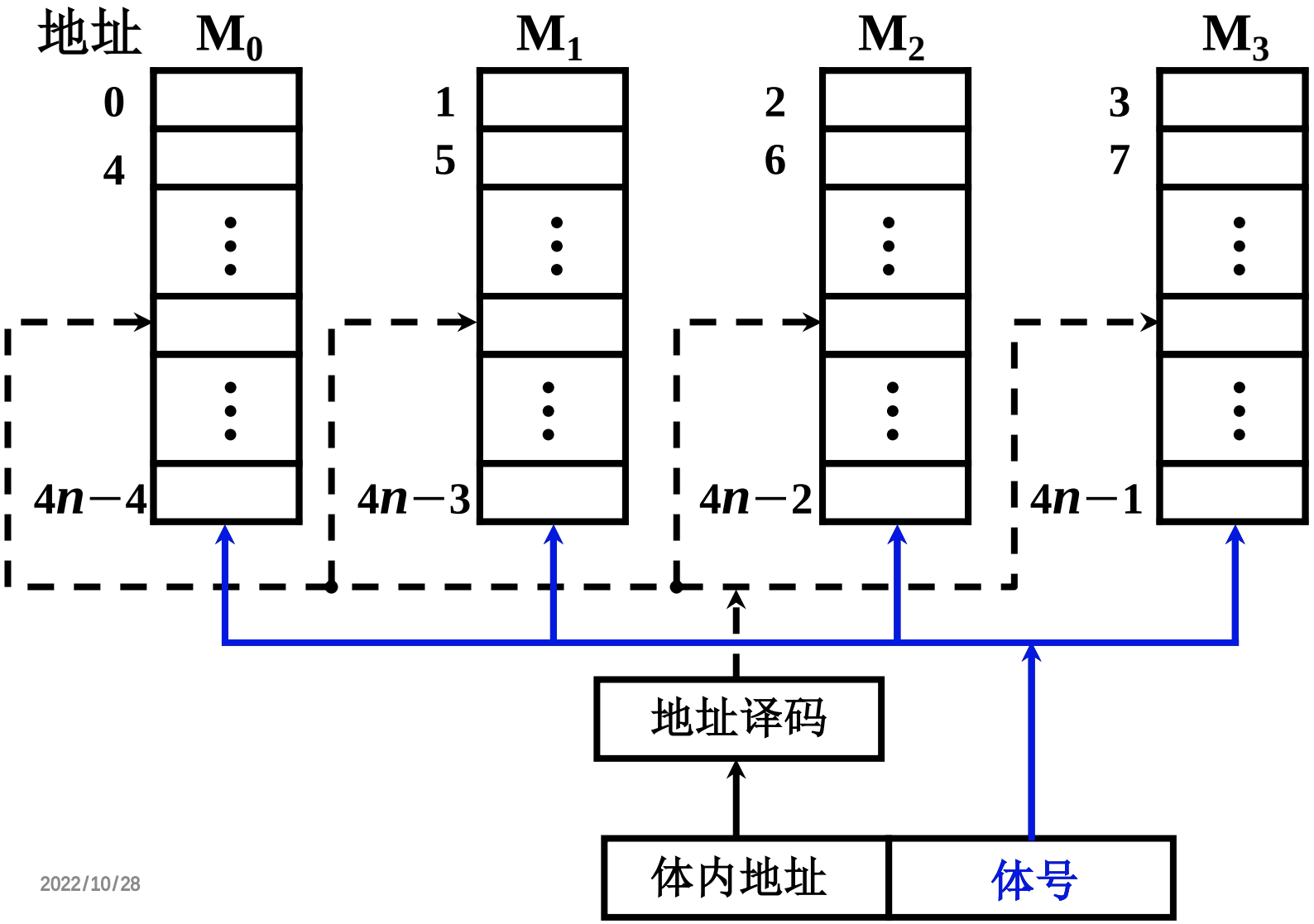
(1) 高位交叉　各个体并行工作



(2) 低位交叉　各个体轮流编址

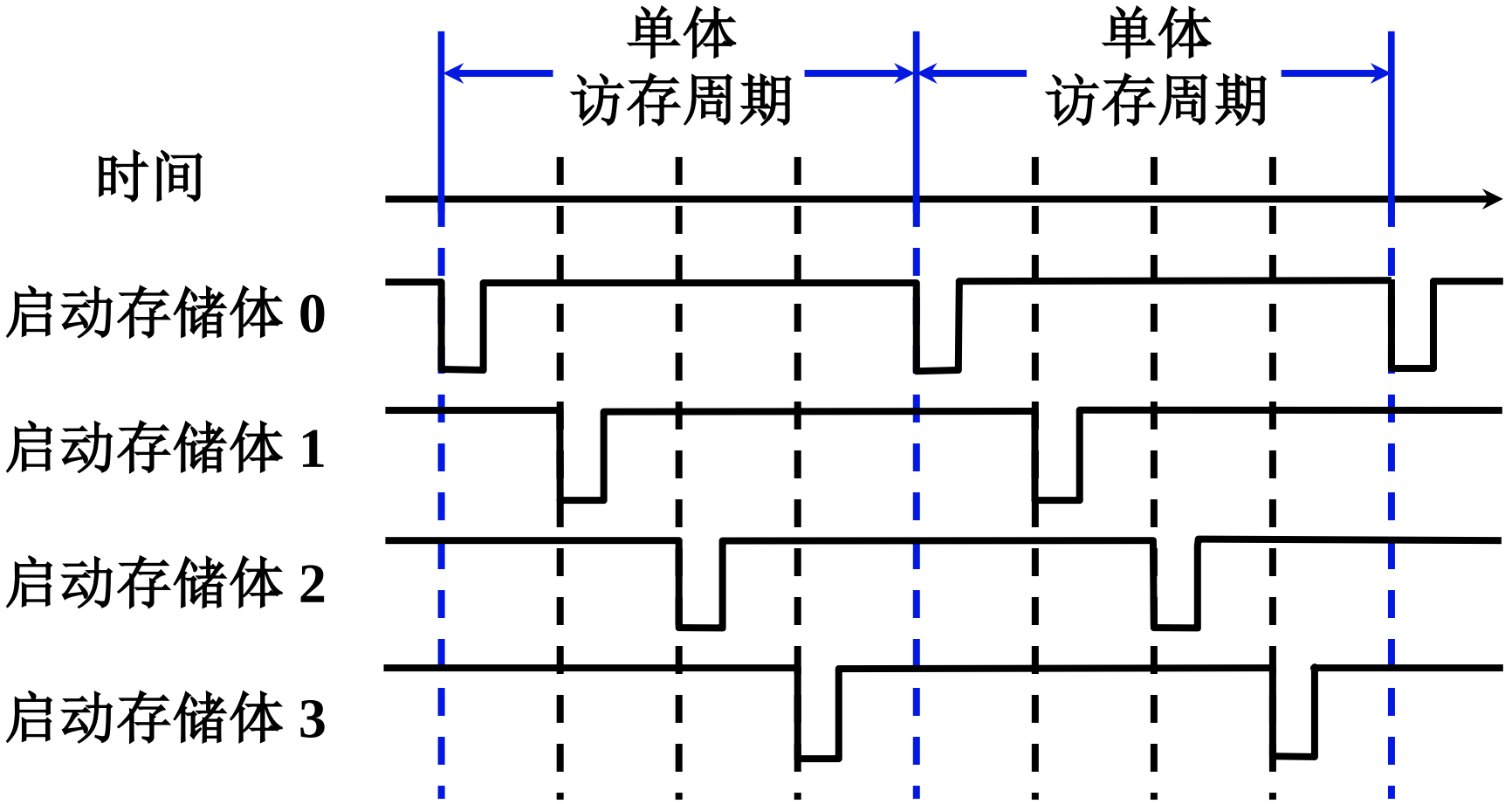


(2) 低位交叉　各个体轮流编址

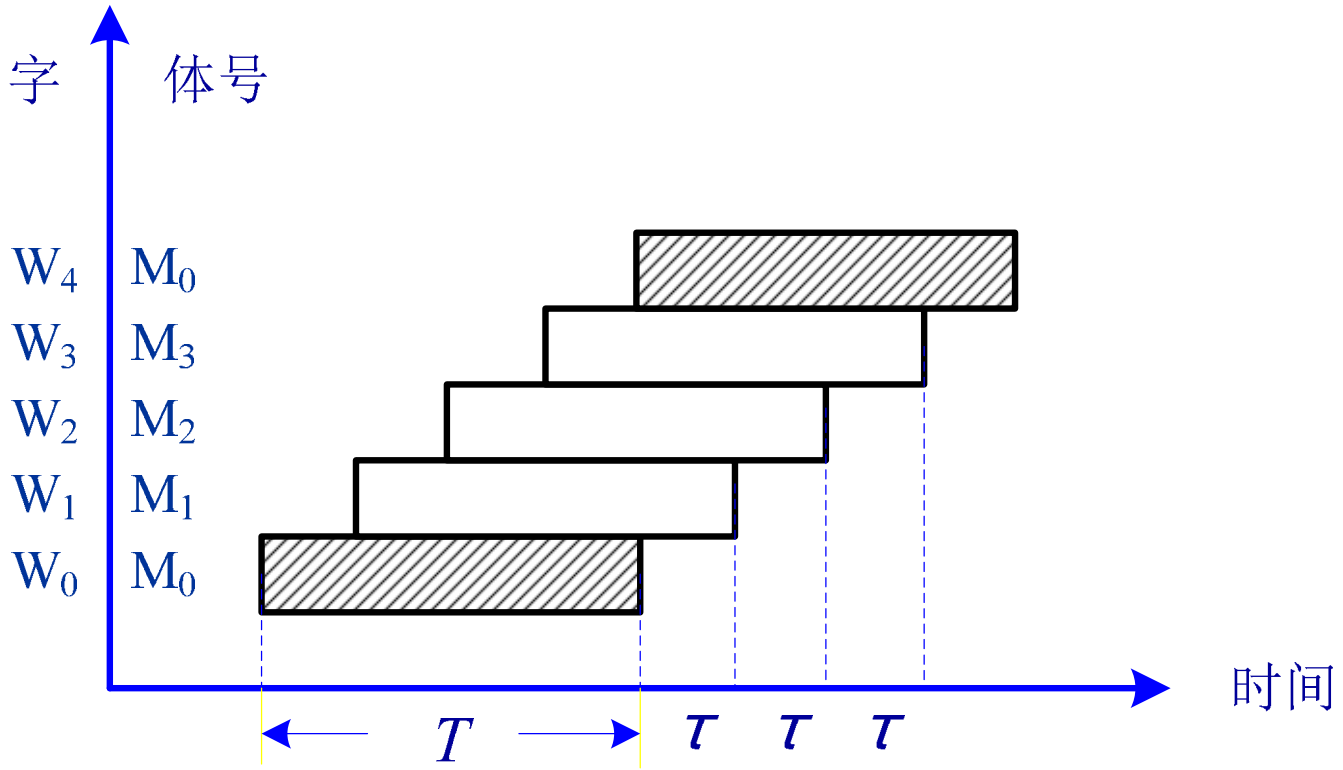


低位交叉的特点

在不改变存取周期的前提下，增加存储器的带宽

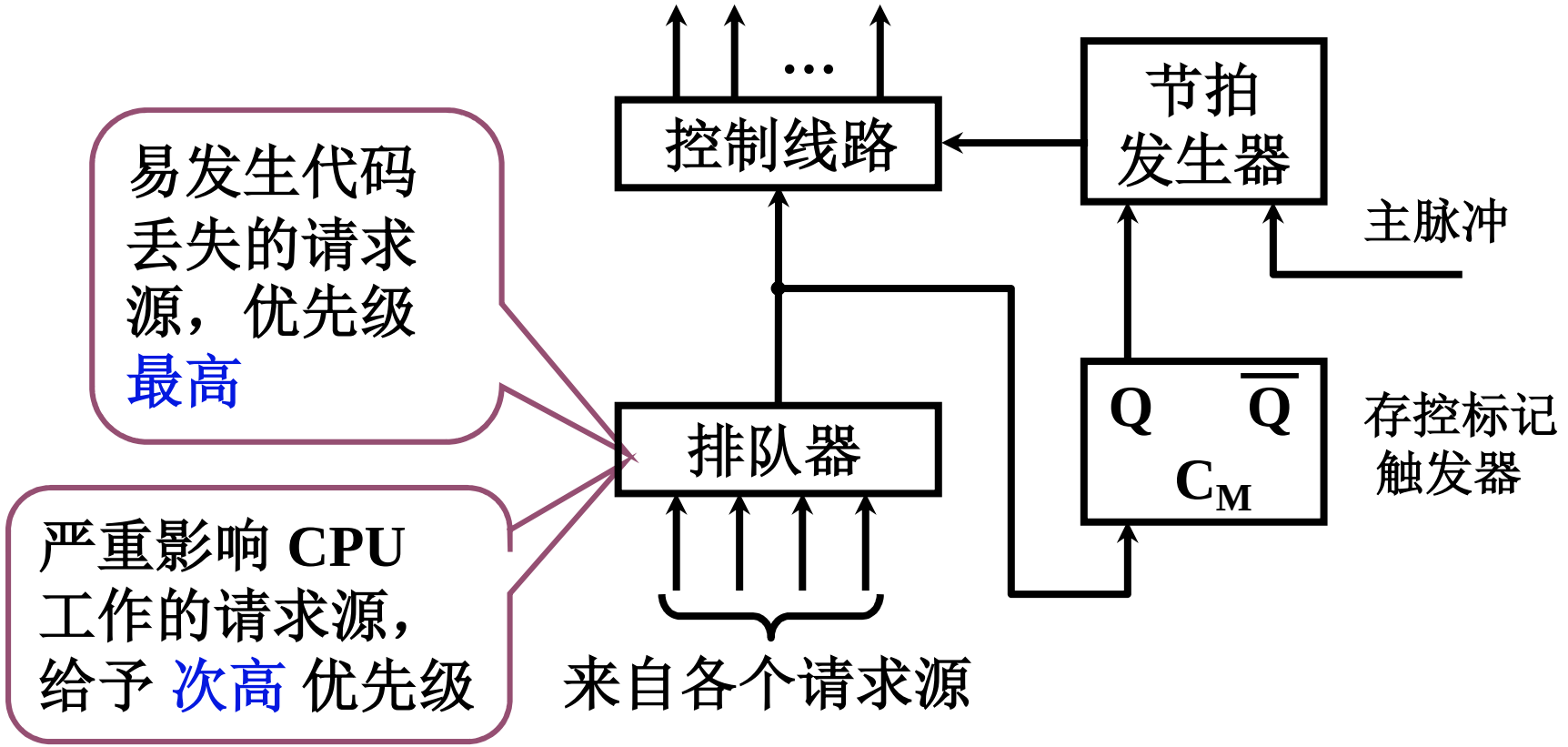


设四体低位交叉存储器，存取周期为T，总线传输周期为τ，为实现流水线方式存取，应满足 $T = 4\tau$。



连续读取 4 个字所需的时间为 $T+(4-1)\tau$

(3) 存储器控制部件（简称存控）



3.高性能存储芯片

(1) 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

(2) RDRAM

由 Rambus 开发，主要解决 存储器带宽 问题

(3) 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

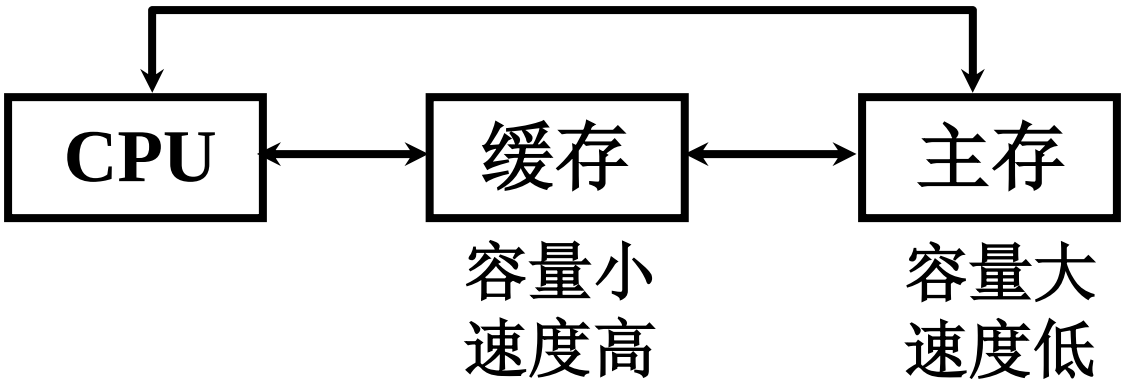
4.3 高速缓冲存储器

一、概述

1. 问题的提出

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

(1) 程序访问的局部性原理

对于绝大多数程序来说，程序所访问的指令和数据在地址上不是均匀分布的，而是相对簇聚的。

程序访问的局部性包含两个方面：

- 时间局部性：程序马上将要用到的信息很可能就是现在正在使用的信息。
- 空间局部性：程序马上将要用到的信息很可能与现在正在使用的信息在存储空间上是相邻的。

(2) 局部性举例

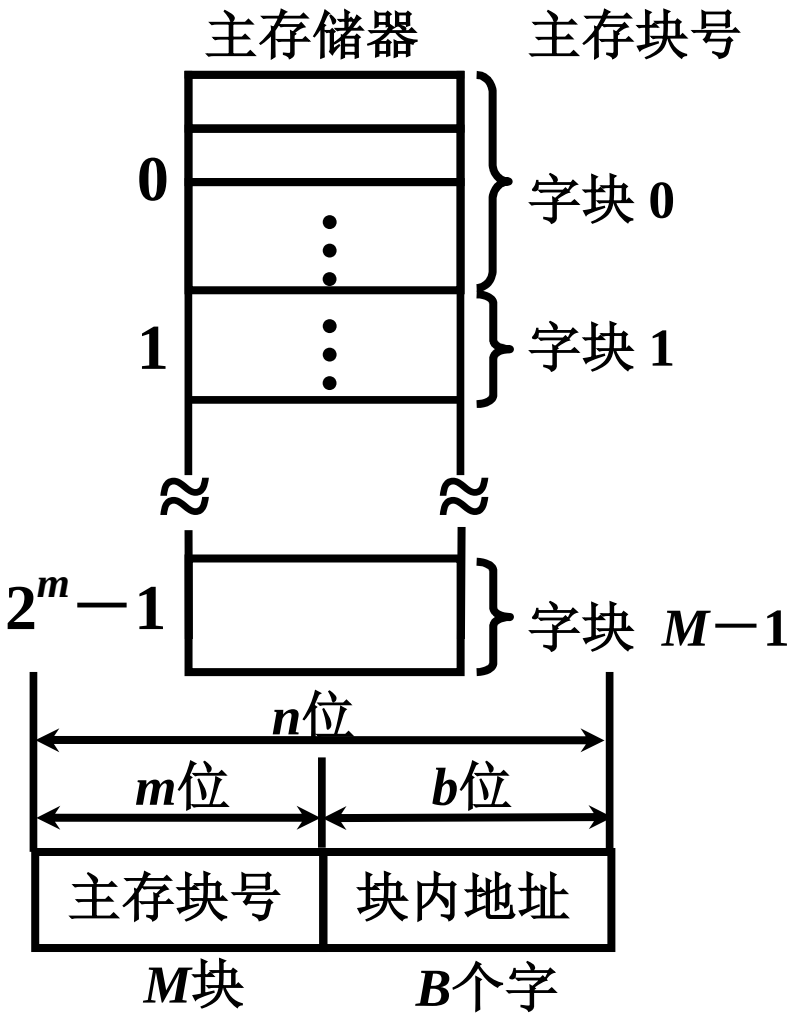

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

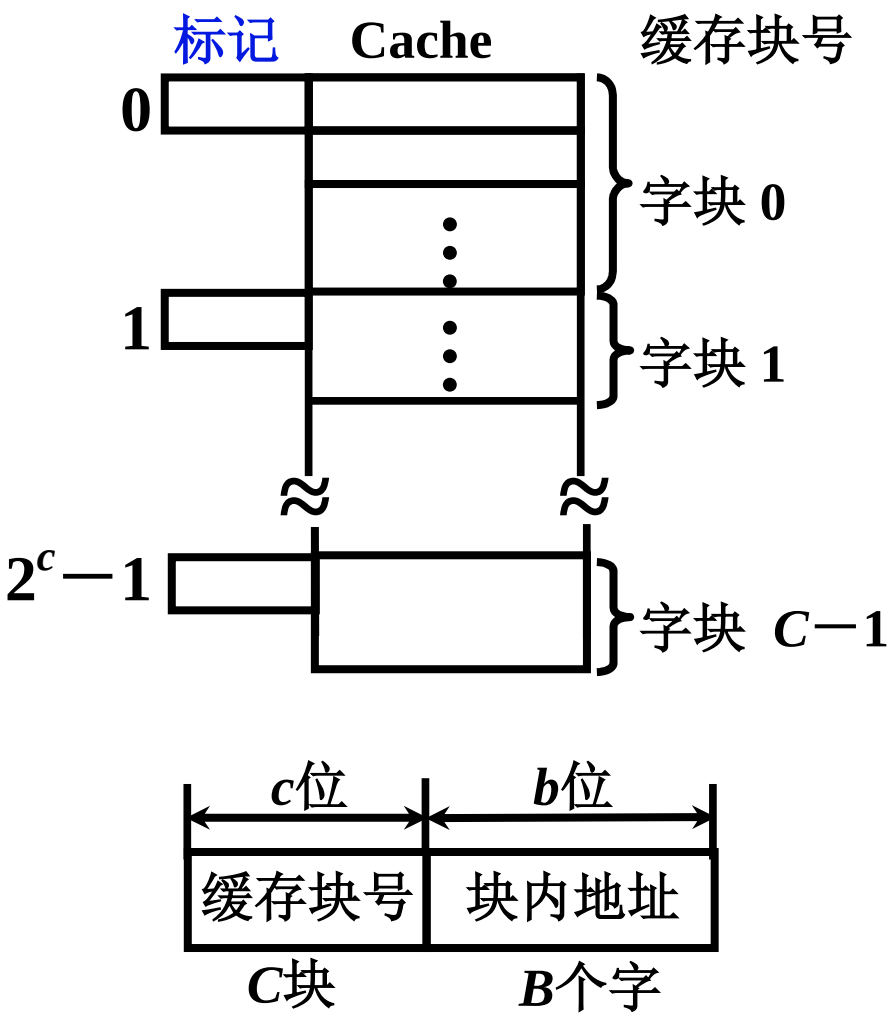
- 对数据的引用
  - 顺序访问数组元素
    (步长为1的引用模式)　　空间局部性
  - 变量sum在每次循环迭代中被引用一次　　时间局部性

- 对指令的引用
  - 顺序读取指令　　空间局部性
  - 重复循环执行for循环体　　时间局部性

2. Cache 的工作原理

(1) 主存和缓存的编址





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

块长取一个存取周期内从主存调出的信息长度

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

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

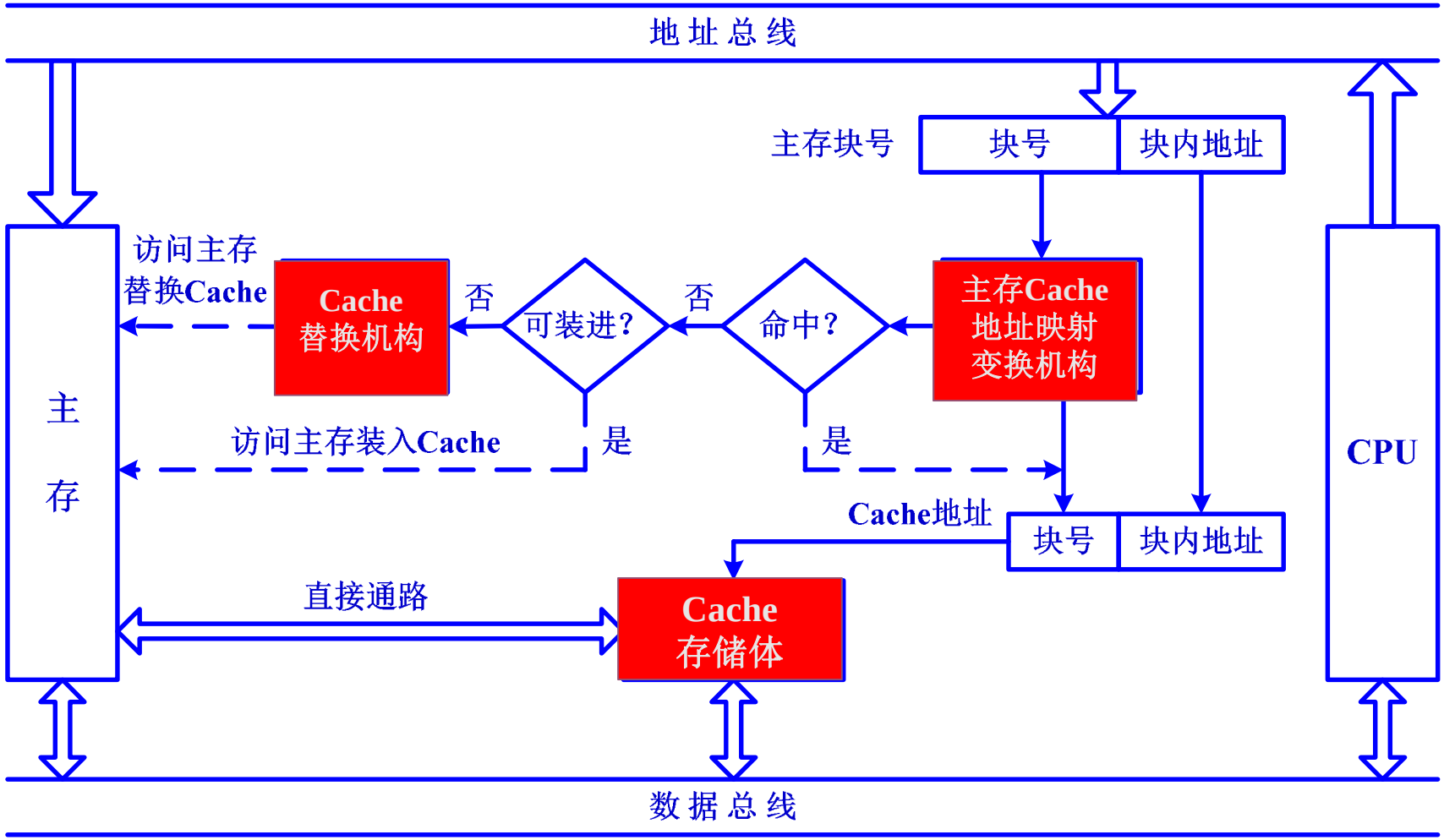
设 Cache 命中率 为 h，访问 Cache 的时间为 t_c，
访问 主存 的时间为 t_m

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

存储层次的四个问题

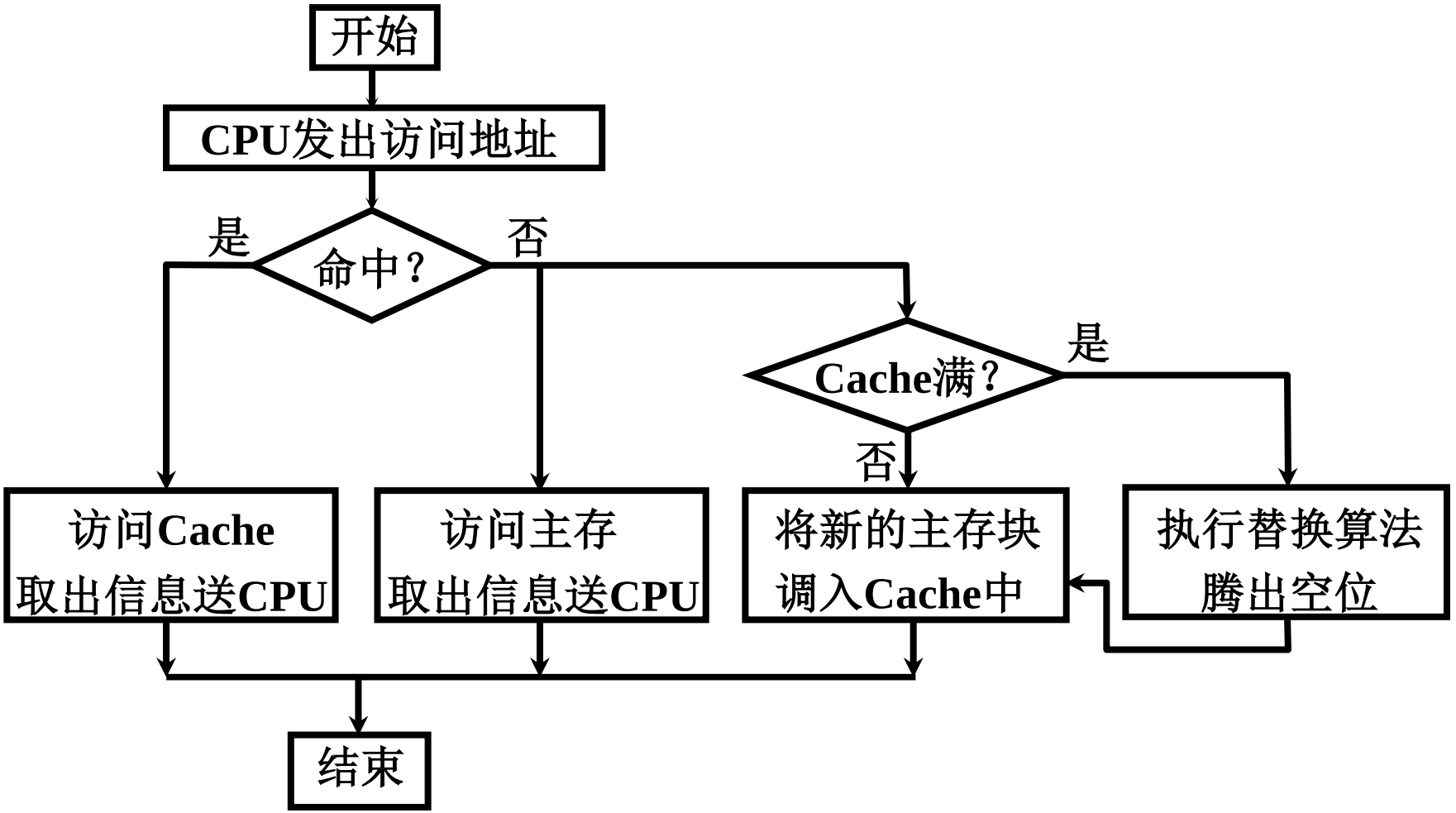
1. 当把一个块调入高一层(靠近CPU)存储器时，可以放在哪些位置上?
   (映象规则　调入块可以放在哪些位置)
2. 当所要访问的块在高一层存储器中时，如何找到该块?
   (查找算法　如何在映象规则 规定的候选位置查找)
3. 当发生失效时，应替换哪一块?
   (替换算法 规定的候选位置均被别的块占用)
4. 当进行写访问时，应进行哪些操作?
   (写策略　如何处理写操作)

3. Cache 的基本结构



4. 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

5. Cache 的改进

(1) 增加 Cache 的级数

片载（片内）Cache

片外 Cache

(2) 统一缓存和分立缓存

指令 Cache　　数据 Cache

与指令执行的控制方式有关　　是否流水

Pentium	8K 指令 Cache	8K 数据 Cache
PowerPC620	32K 指令 Cache	32K 数据 Cache

二、Cache – 主存的地址映射 4.3

1. 直接映射

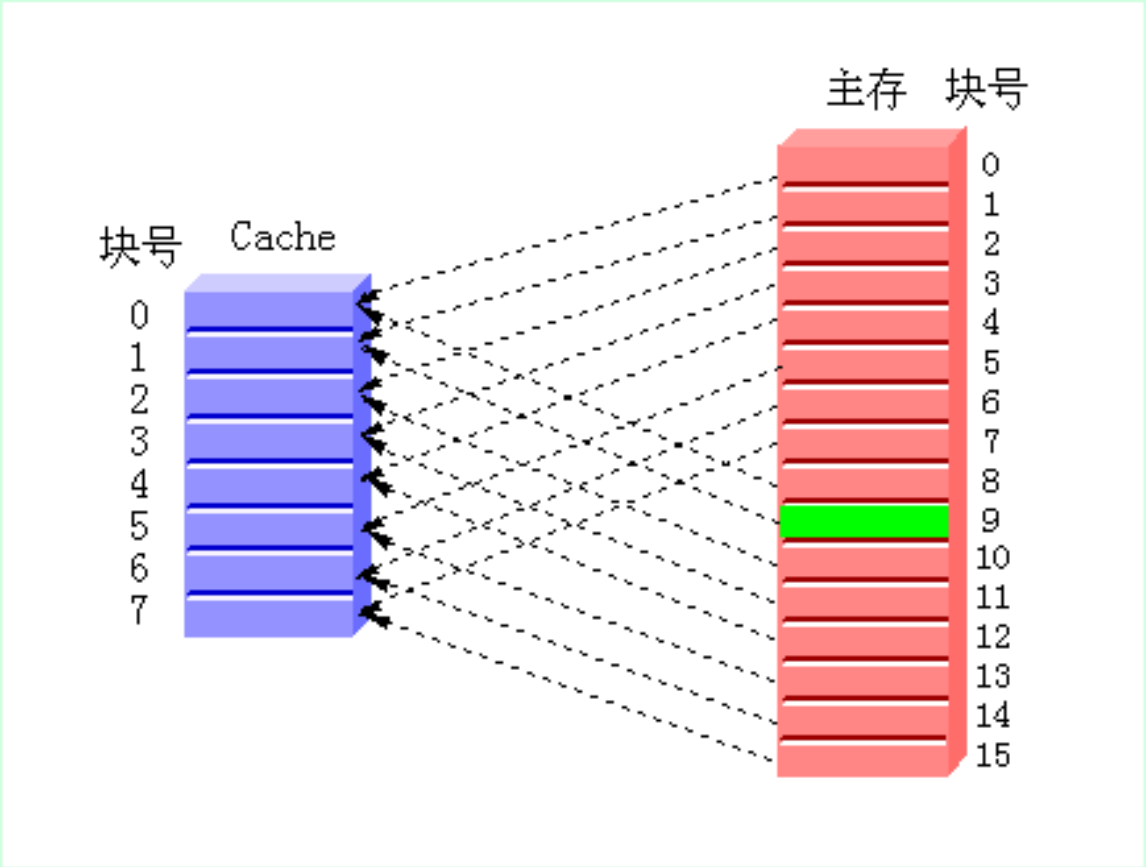
直接映射：主存中的每一块只能被放置到Cache中唯一的一个位置。 (循环分配)

对比：阅览室位置 --只有一个位置可以坐

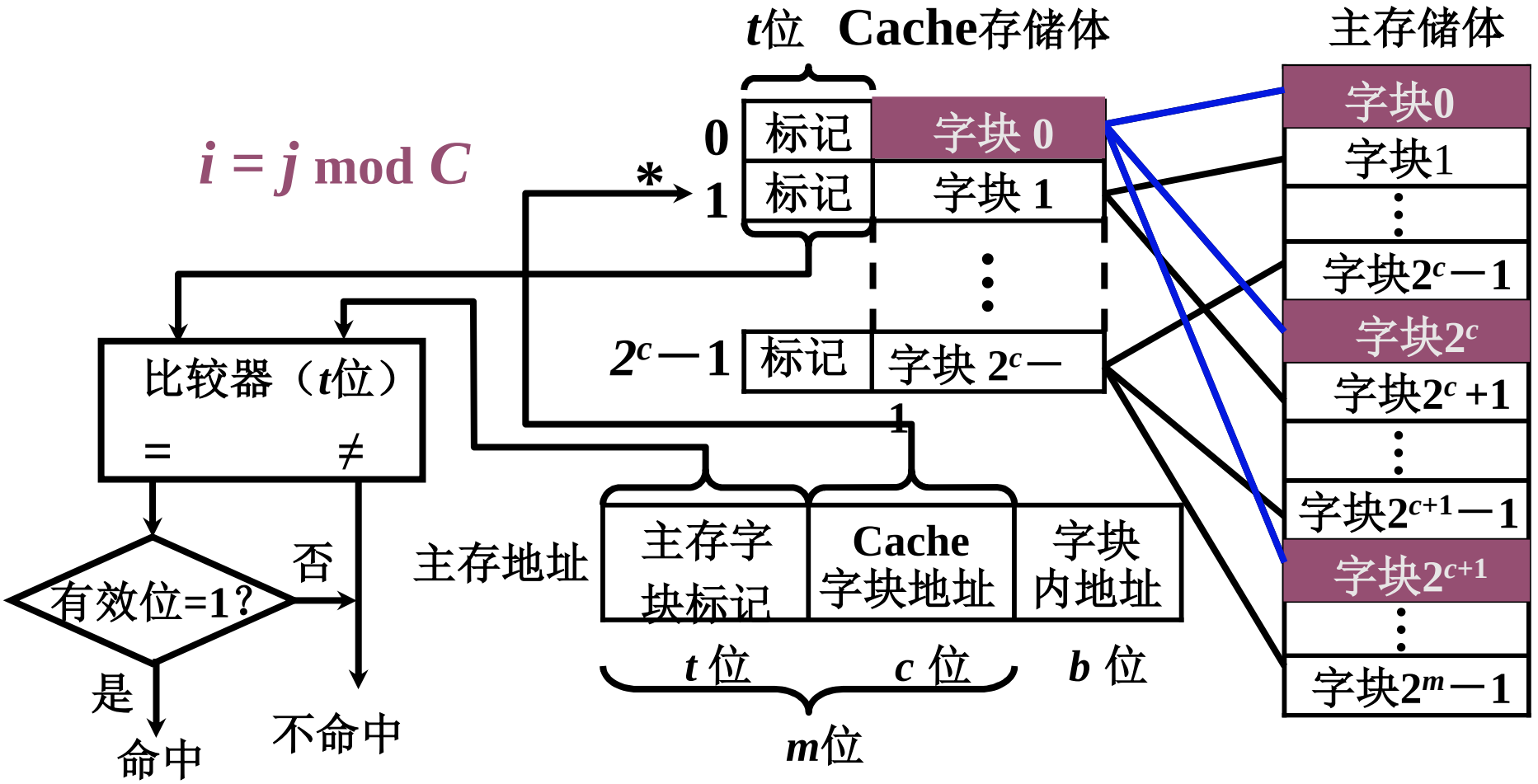
特点：空间利用率最低，冲突概率最高，实现最简单。

直 接 映 射

（举 例）



1. 直接映射



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

2. 全相联映射

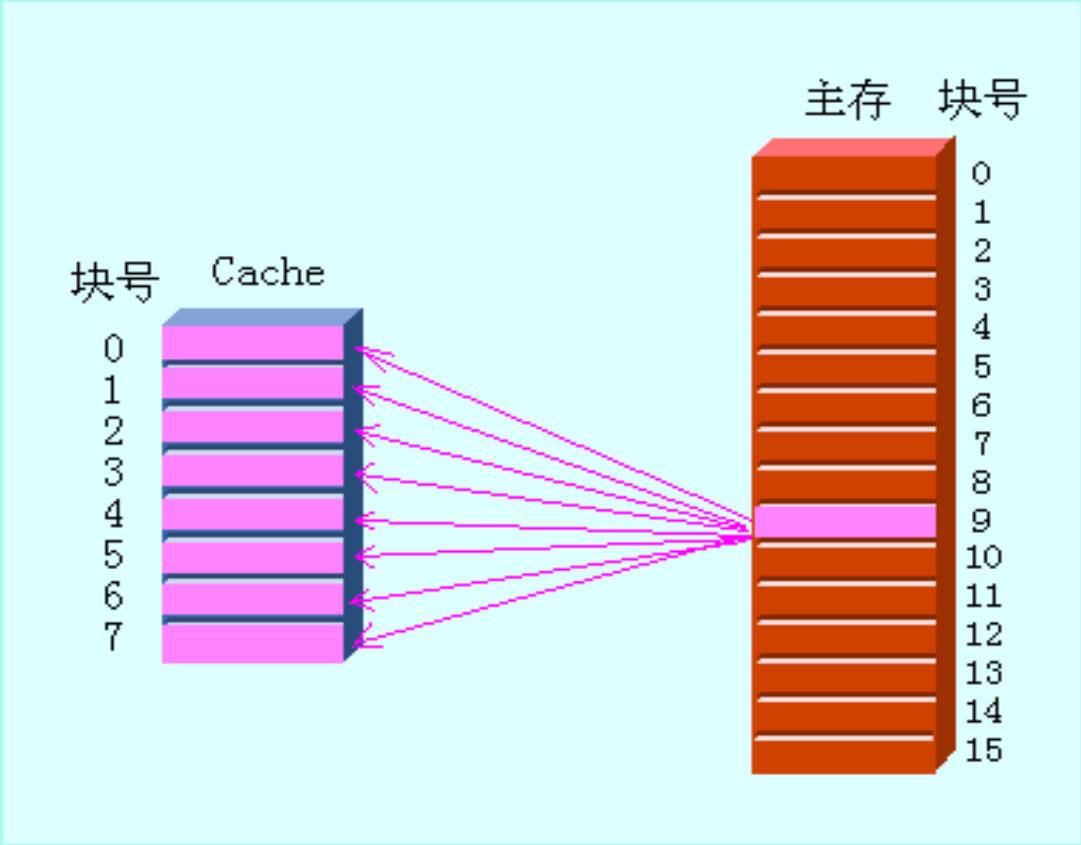
全相联：主存中的任一块可以被放置到Cache中的任意一个位置。

对比：　阅览室位置--随便坐

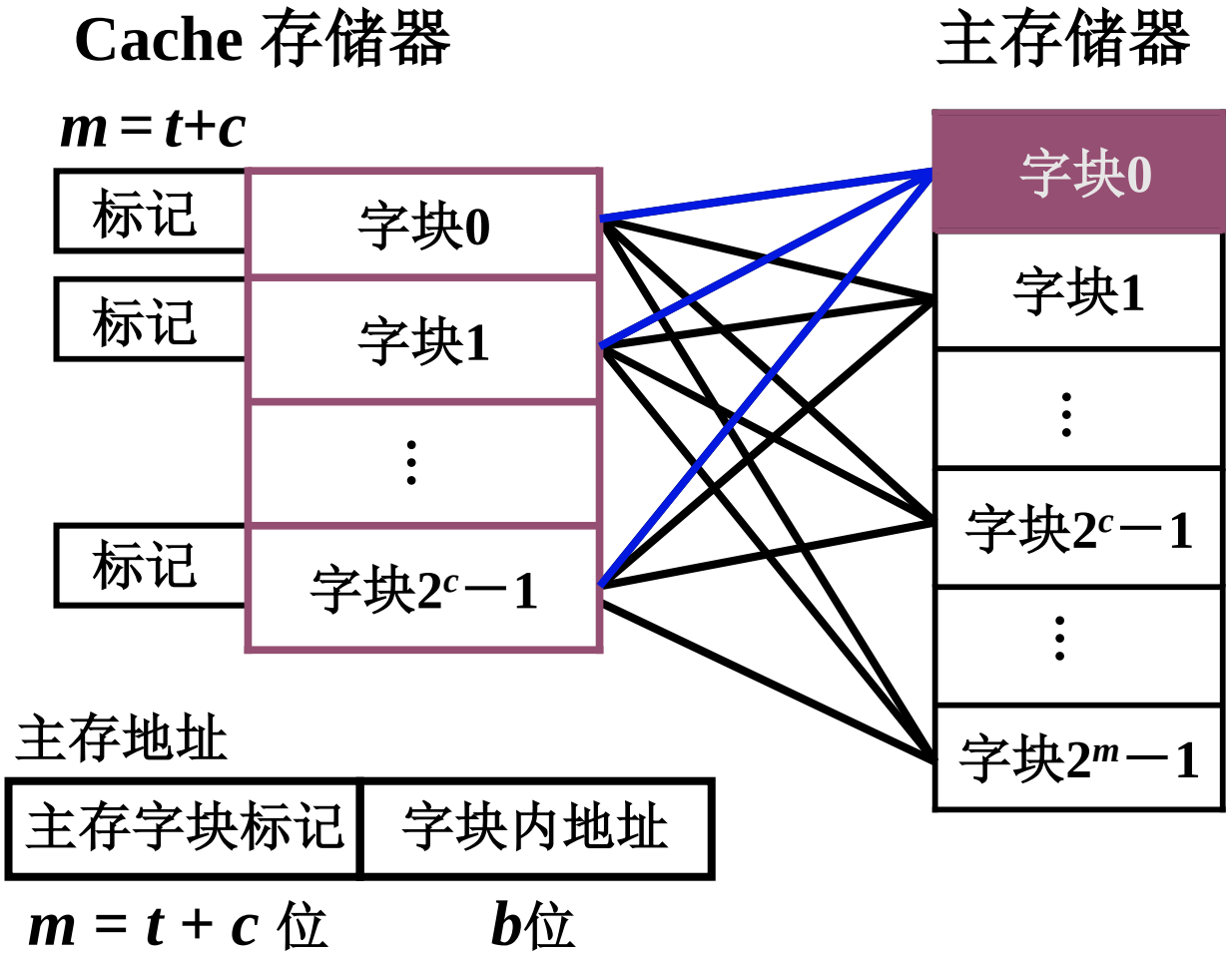
特点：　空间利用率最高，冲突概率最低，实现最复杂。

全相联映射

（举例）



2. 全相联映射

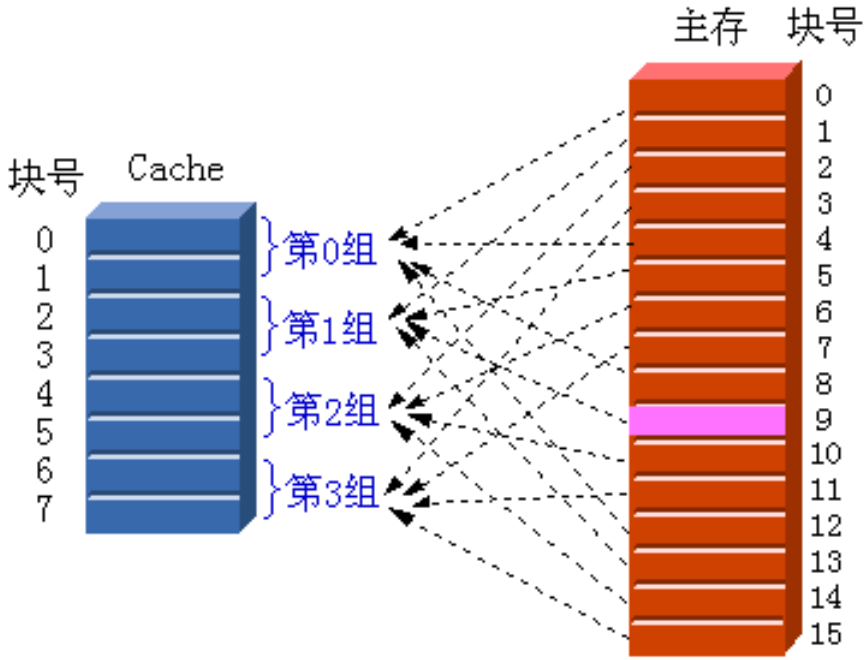


主存 中的 任一块 可以映射到 缓存 中的 任一块

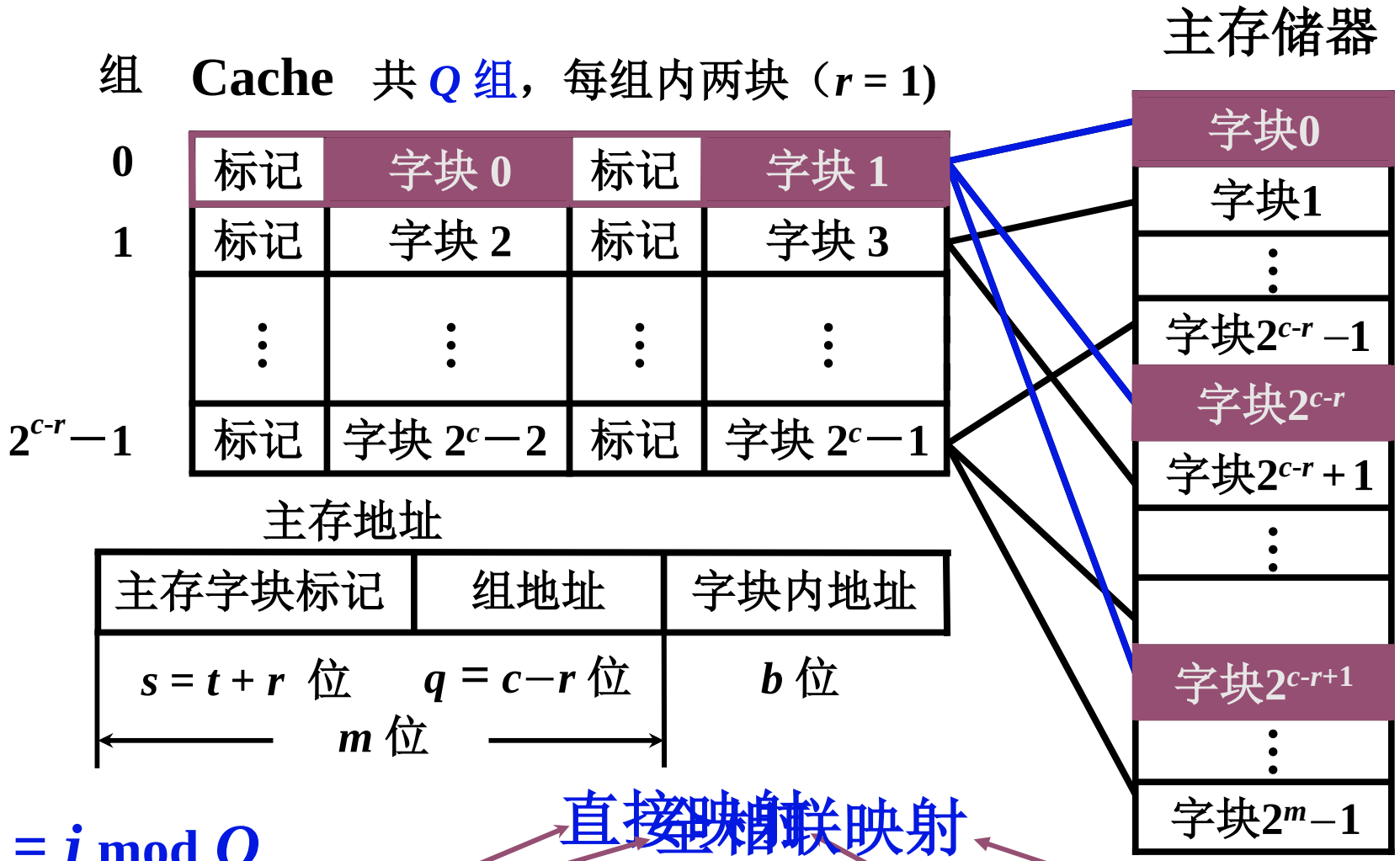
3. 组相联映射

组相联：主存中的每一块可以被放置到Cache中唯一　的一个组中的任何一个位置。

组相联是直接映象和全相联的一种折中



3. 组相联映射



$i = j \bmod Q$

直接映射 全相联映射

某一主存块 j 按模 Q 映射到 缓存 的第 i 组中的 任一块

- ***n*** 路组相联：每组中有***n*** 个块(***n*** = ***M/G***)，***n*** 称为相联度

相联度越高，Cache空间的利用率就越高，块冲突概率就越低，失效率也就越低。

	n (路数)	***G*** (组数)
全相联	***M***	1
直接映象	1	***M***
组相联	$1<n<M$	$1<G<M$

- 大多数计算机的Cache: $n \leq 4$

想一想：相联度是否越大越好？

三、替换算法

- **最近最少使用法LRU**

- 选择近期最少被访问的块作为被替换的块。
  （实现比较困难）
- 实际上：选择最久没有被访问过的块作为被替换的块。
- 优点：命中率较高

- **LRU和随机法分别因其不命中率低和实现简单而被广泛采用。**

- 模拟数据表明，对于容量很大的Cache，LRU和随机法的命中率差别不大。

时间 t	1	2	3	4	5	6	7	8	9	10	命中次数
页地址流	P1	P2	P1	P5	P4	P1	P3	P4	P2	P4	
先进先出算法（FIFO）	1	1	1	1*	4	4	4*	4*	2	2	2
		2	2	2	2*	1	1	1	1*	4	
				5	5	5*	3	3	3	3*	
	调入	调入	命中	调入	替换	替换	替换	命中	替换	替换	
最久没有使用算法（LRU）	1	1	1	1	1	1	1	1*	2	2	4
		2	2	2*	4	4	4*	4	4	4	
				5	5	5*	3	3	3	3*	
	调入	调入	命中	调入	替换	命中	替换	命中	替换	命中	
最优替换算法（OPT）	1	1	1	1	1	1*	3	3	3	3	5
		2	2	2	2	2	2	2	2	2	
				5*	4	4	4	4	4	4	
	调入	调入	命中	调入	替换	命中	替换	命中	命中	命中	

四、写策略 4.3

1. “写”操作所占的比例

Load指令：26%　　Store指令：9%

“写”在所有访存操作中所占的比例：

9%/(100%＋26%＋9%)≈7%

“写”在访问数据Cache操作中所占的比例：

9%/(26%＋9%)≈25%

2. “写”操作必须在确认是否命中后才可进行

3. “写”访问有可能导致Cache和主存内容的不一致

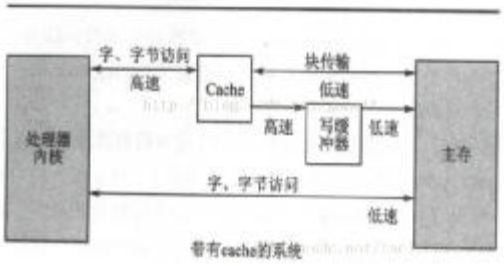

写策略
写命中
写回法（write-back）
写直达（write-through）
写不命中
写分配法（write-allocate）
非写分配法（not-write-allocte）

4．两种写策略

- 写直达法：执行“写”操作时，不仅写入Cache，而且也写入下一级存储器，全写法。
- 写回法：执行“写”操作时，只写入Cache。仅当Cache中相应的块被替换时，才写回主存。（设置“脏位”）

5．两种写策略的比较

- 写回法的优点：速度快，占用存储器频带低
- 写直达法的优点：易于实现，一致性好



6.写缓冲器

7. “写”操作时的调块

- 按写分配(写时取)：写失效时，先把所写单元所在的块调入Cache，再行写入。
- 不按写分配(绕写法)：写失效时，直接写入下一级存储器而不调块。

8. 写策略与调块

写回法 —— 按写分配

写直达法 —— 不按写分配

五、Cache结构举例 4.3

例子：DEC的Alpha AXP21064中的内部数据Cache

1．简介

容量：8KB

块大小：32B

块数：256

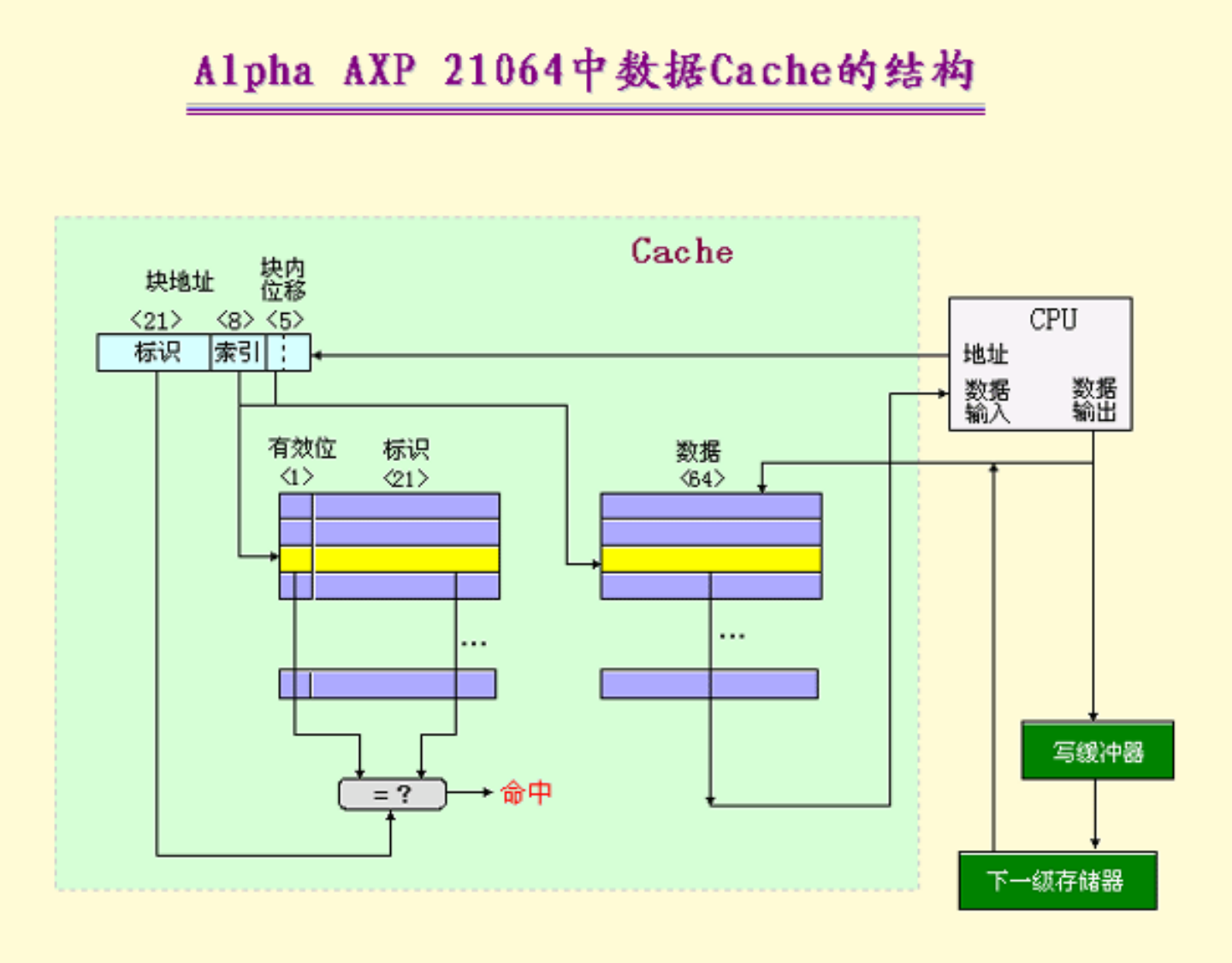
映射方法：直接映射

“写”策略：写直达—不按写分配

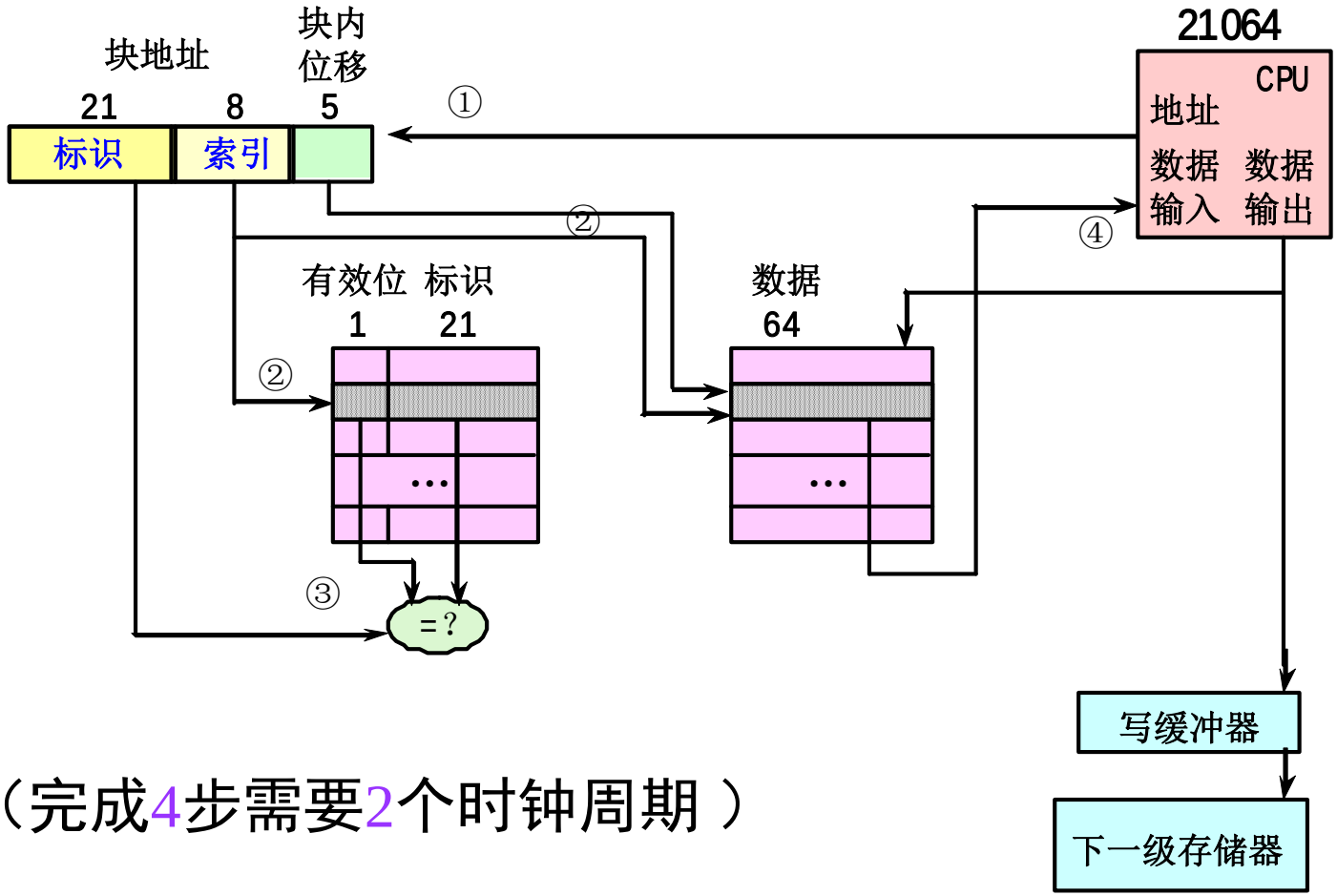
写缓冲器大小：4个块

内存地址：34位（块地址29位，块内地址5位）

结构图


Alpha AXP 21064中数据Cache的结构
Cache
块地址
块内位移
〈21〉
〈8〉
〈5〉
标识
索引
有效位
〈1〉
标识
〈21〉
数据
〈64〉
CPU
地址
数据输入
数据输出
写缓冲器
下一级存储器
= ?
命中

2. 工作过程



（完成4步需要2个时钟周期）

2. 工作过程

① 处理器传送给Cache物理地址

② 由索引选择标识的过程

- 根据索引从目录项中读出相应的标识和有效位

③ 从Cache中读出标识之后，用来同从CPU发来的块地址中标志域部分进行比较

- 为了保证包含有效的信息，必须要设置有效位

④ 如果有一个标识匹配，且标志位有效，则此次命中

- 通知CPU取走数据

- “写”访问命中
  - 前三步一样，只有在确认标识匹配后才把数据写入
- 设置了一个写缓冲器

  （提高“写”访问的速度）
  - 按字寻址的，它含有4个块，每块大小为4个字。
  - 当要进行写入操作时，如果写缓冲器不满，那么就把数据和完整的地址写入缓冲器。对CPU而言，本次“写”访问已完成，CPU可以继续往下执行。由写缓冲器负责把该数据写入主存。
  - 在写入缓冲器时，要进行写合并检查。即检查本次写入数据的地址是否与缓冲器内某个有效块的地址匹配。如果匹配，就把新数据与该块合并 。

发生读不命中与写不命中时的操作 4.3

- **读不命中：** 向CPU发出一个暂停信号，通知它等待，并从下一级存储器中新调入一个数据块（32字节）
  - Cache与下一级存储的数据通路宽度为16B，传送一次需5个周期，因此，一次传送需要10个周期
- **写不命中：** 将使数据“绕过”Cache，直接写入主存。
  - 写直达 –不按写分配
- 因为是写直达，所以替换时不需要写回

例题

- **某计算机字长32位，采用直接映射cache,主存容量4MB，cache数据存储体容量为4KB，字块长度为8个字。**
  1. 画出直接映射方式下主存地址划分情况。
  2. 设cache初始状态为空，若CPU顺序访问0-99号单元，并从中读出100个字，假设访问主存一次读一个字，并重复此顺序10次，请计算cache命中率。
  3. 如果cache的存取时间是2ns，主存访问时间是20ns，平均访问时间是多少。
  4. cache-主存系统访问效率。

六、改进Cache性能

平均访存时间＝命中时间＋失效率×失效开销

可以从三个方面改进Cache的性能：

(1) 降低失效率

例如：增加块大小、提高相联度

(2) 减少失效开销

例如：多级Cache、写缓冲及写合并、

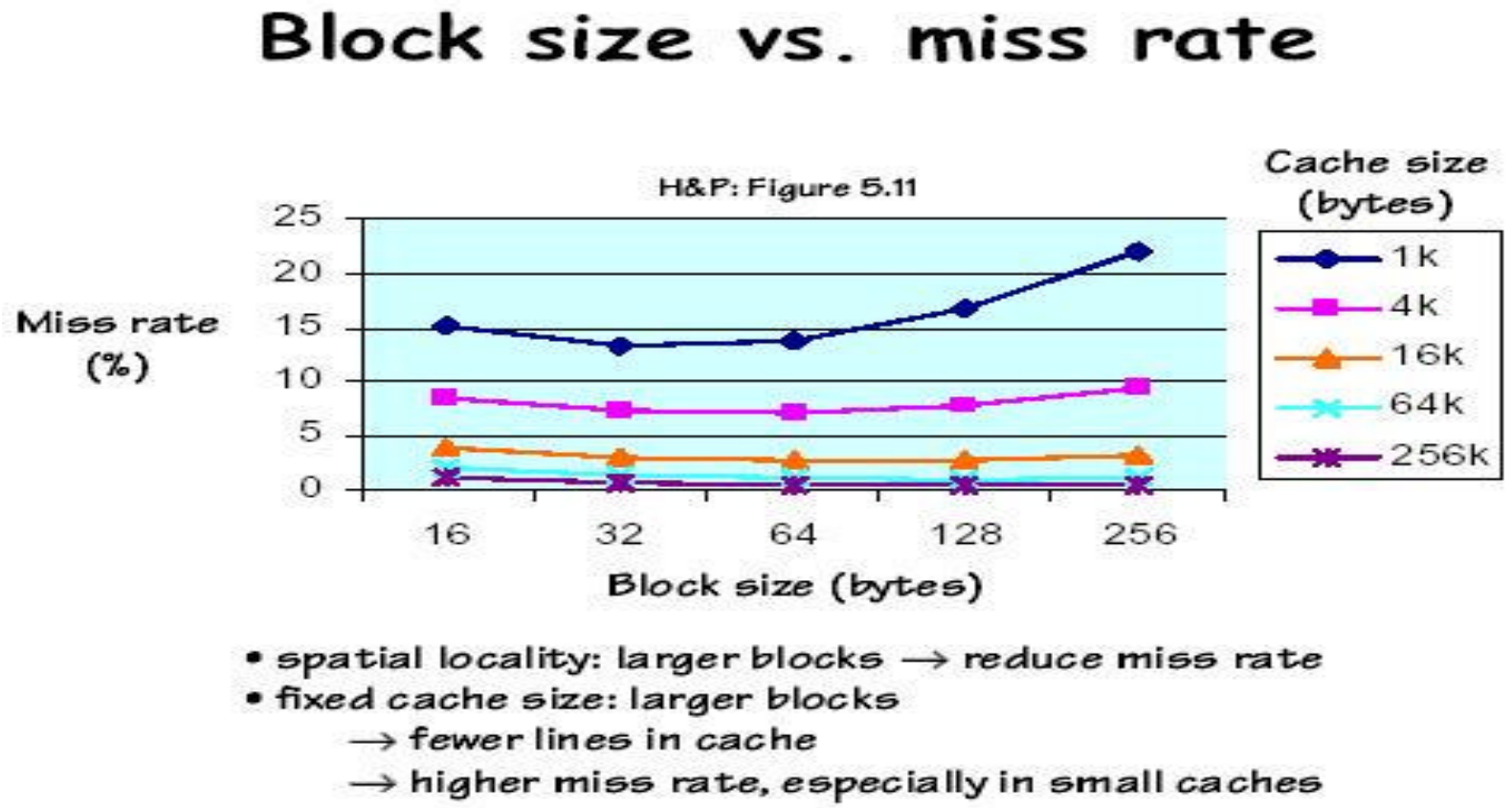
请求字优先

(3) 减少Cache命中时间

例如：容量小且结构简单的Cache

Cache大小、Block大小和缺失率的关系 4.3

缺失率与Cache大小、Block大小等有关



Cache**大小**：Cache**越大**，Miss**率越低，但成本越高！**

Block**大小**：Block**大小与**Cache**大小有关，且不能太大，也不能太小！**

访存时间(时钟周期数）和Cache容量及相联度的关系（举例）

Cache容量(KB)	相联度（路）			
	1	2	4	8
1	7.65	6.60	6.22	5.44
2	5.90	4.90	4.62	4.09
4	4.60	3.95	3.57	3.19
8	3.30	3.00	2.87	2.59
16	2.45	2.20	2.12	2.04
32	2.00	1.80	1.77	1.79
64	1.70	1.60	1.57	1.59
128	1.50	1.45	1.42	1.44

Cache系统

- 刚引入Cache时只有一个Cache。近年来多Cache系统成为主流
- 多Cache系统中，需考虑两个方面：

[1] 单级/多级？

外部(Off-chip)Cache:不做在CPU内而是独立设置一个Cache

片内(On-chip)Cache: 将Cache和CPU作在一个芯片上

单级Cache：只用一个片内Cache

多级Cache：同时使用L1 Cache和L2 Cache，甚至有L3 Cache，
L1 Cache更靠近CPU，其速度比L2快，其容量比L2小

[2] 联合/分立？

分立：指数据和指令分开存放在各自的数据和指令Cache中

一般L1 Cache都是分立Cache

L1 Cache的命中时间比命中率更重要！

联合：指数据和指令都放在一个Cache中

一般L2 Cache都是联合Cache

L2 Cache的命中率比命中时间更重要！

多级cache的性能 4.3

- **采用**L2 Cache**的系统，其缺失损失的计算如下：**
  - 若L2 Cache包含所请求信息，则缺失损失为L2 Cache访问时间
  - 否则访问主存，并取到L1 Cache和L2 Cache（缺失损失更大）
- **例：某处理器在无**cache**缺失时**CPI**为**1**，时钟频率为**5GHz**。假定访问一次主存的时间（包括所有的缺失处理）为**100ns**，平均每条指令在**L1 Cache**中的缺失率为**2%**。若增加一个**L2 Cache**，其访问时间为**5ns**，而且容量足够大到使全局缺失率减为**0.5%**，问处理器执行指令的速度提高了多少？**

  解：如果只有一级Cache，则缺失只有一种。即L1缺失(需访问主存)，
  其缺失损失为：100nsx5GHz=500个时钟，CPI=1+500x2%=11.0。

  如果有二级Cache，则有两种缺失：

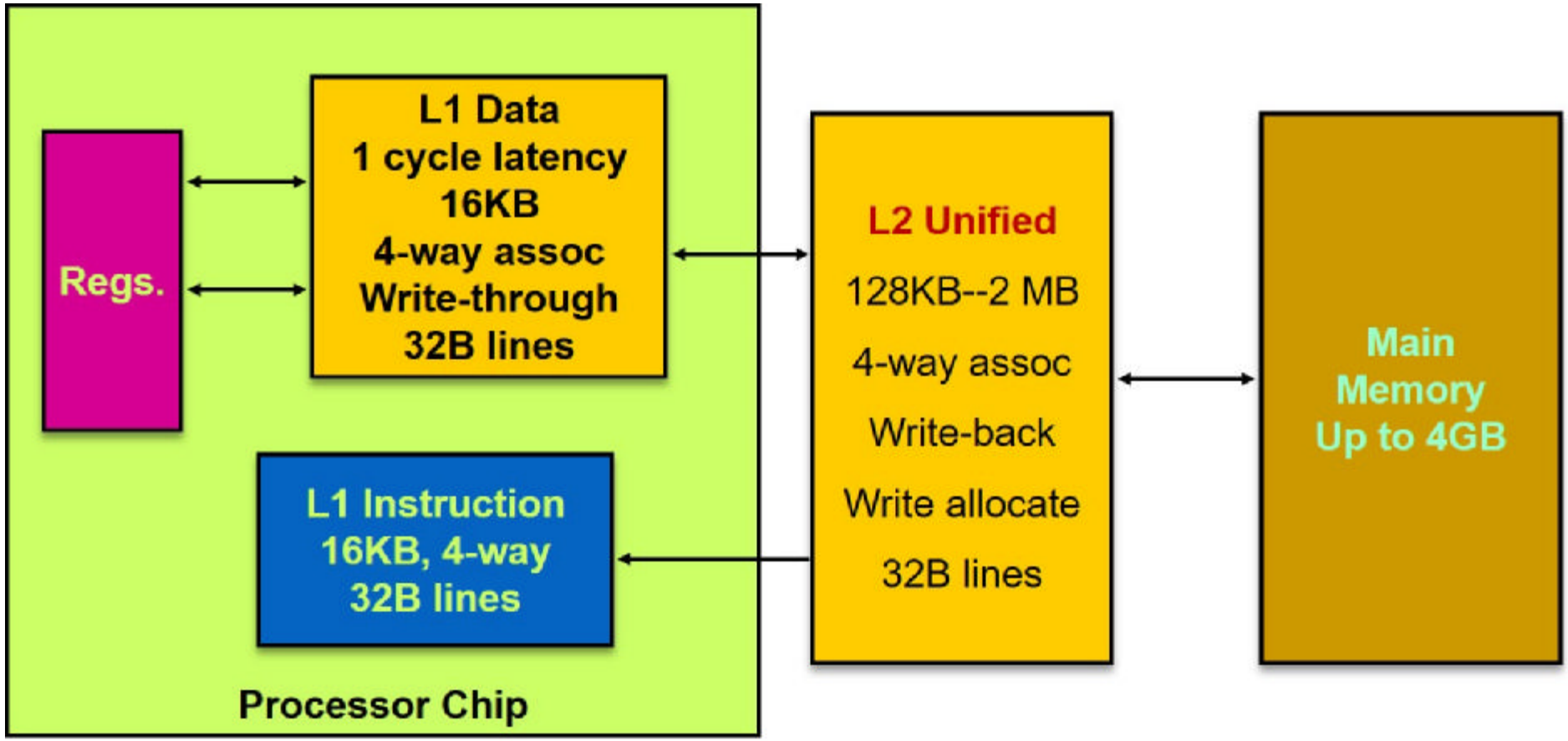
  L1缺失(需访问L2 Cache)：5nsx5GHz=25个时钟

  L1和L2都缺失(需访问主存)：500个时钟

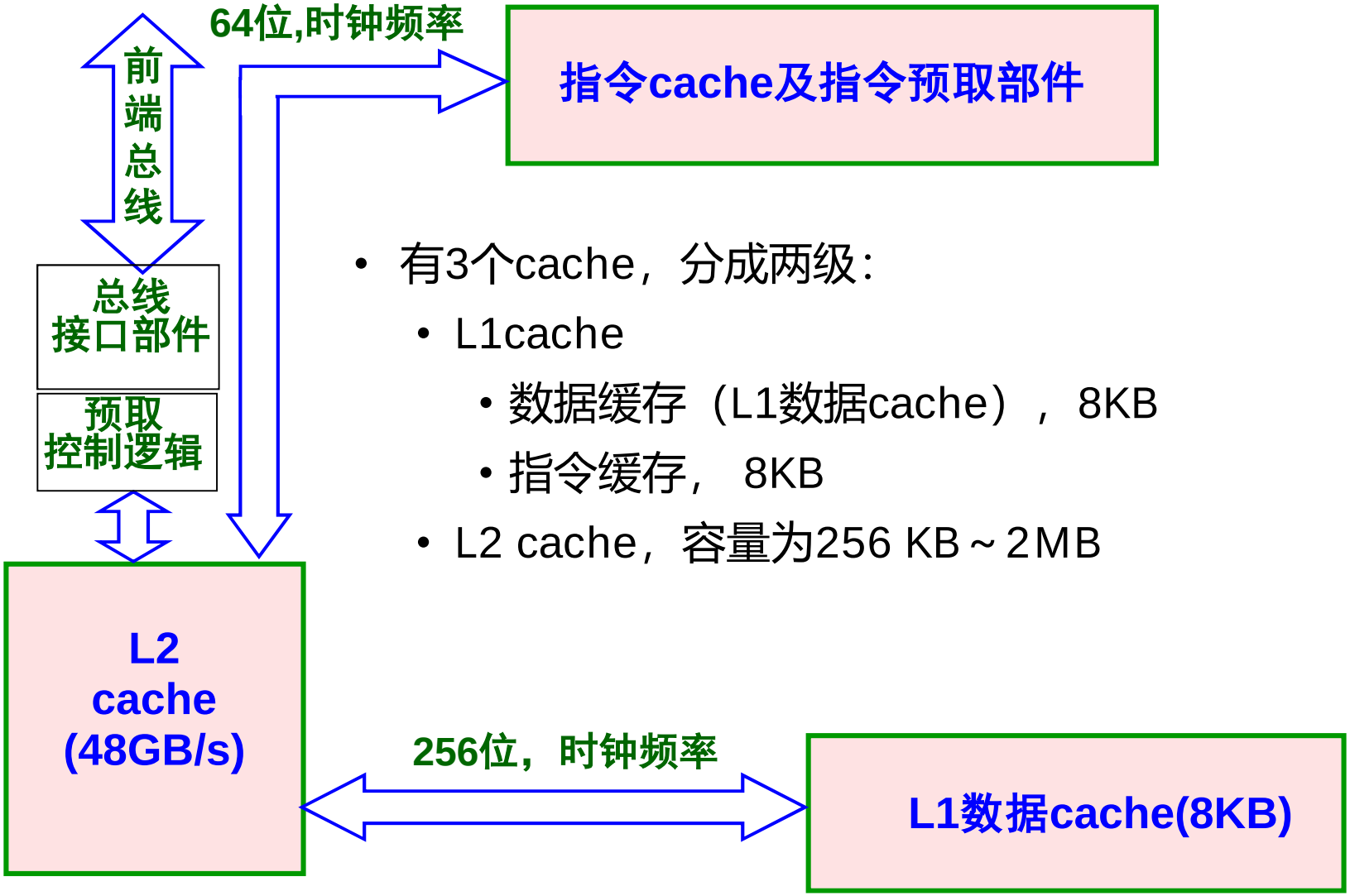
  因此，CPI=1+25x2%+500x0.5%=4.0

  二者的性能比为11.0/4.0=2.8倍！

Intel Pentium cache hierarchy



实例：Pentium 4的cache存储器

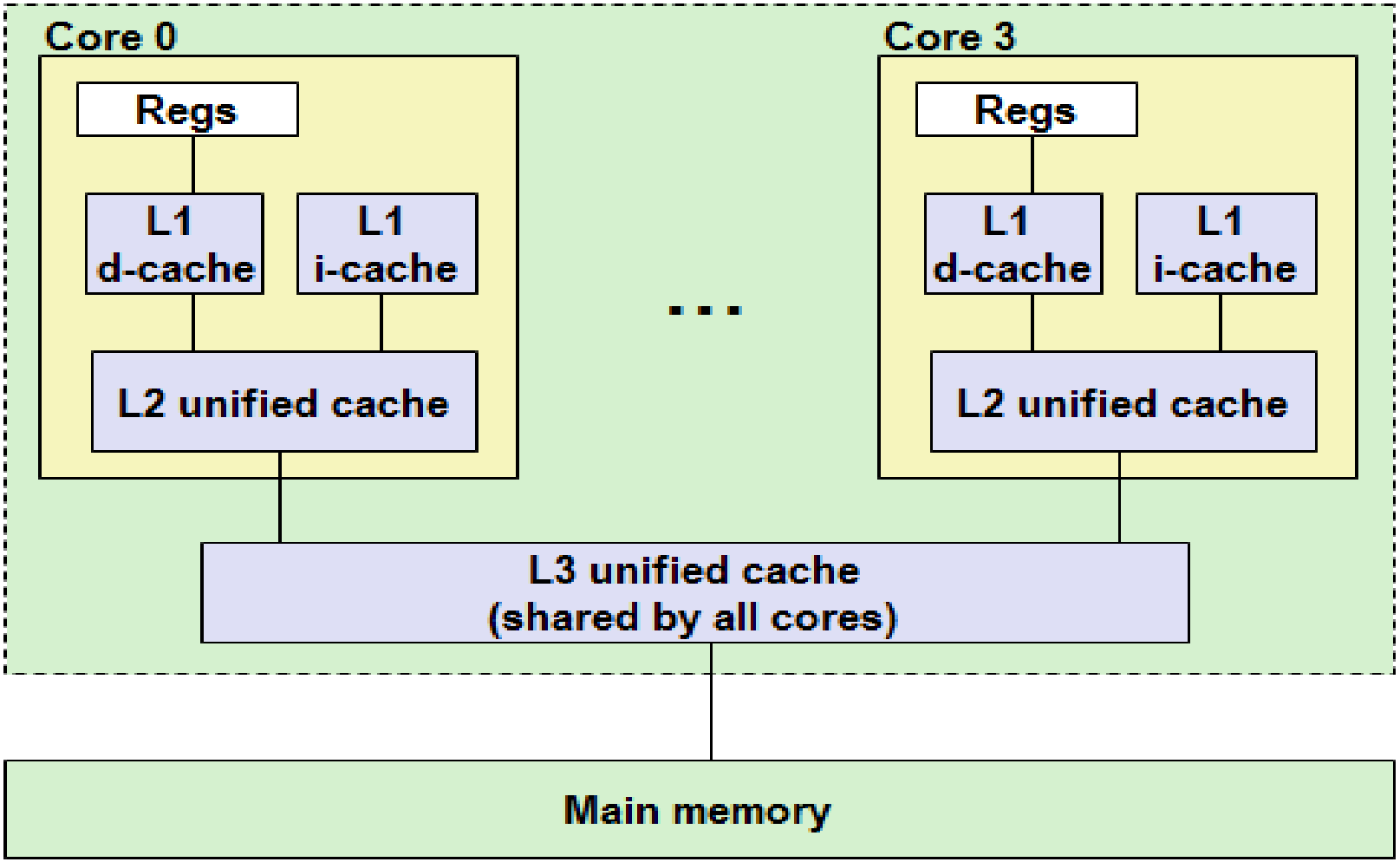


- 有3个cache，分成两级：
  - L1cache
    - 数据缓存（L1数据cache），8KB
    - 指令缓存， 8KB
  - L2 cache，容量为256 KB～2MB

Intel Pentium 4, 2.2 GHz Processor

Component	Access Speed	Size of Component
Registers	1 cycle = 0.5 ns	32 寄存器
L1 Cache	3 cycles =1.5 ns	指令数据cache分离，各 8 KB
L2 Cache	20 cycles =10 ns	256 Kbytes，8路 组相联
L3 Cache	30 cycles =15 ns	512 Kbytes 8路 组相联
Memory	400 cycles =200 ns	16 Gigabytes

实例：Intel Core i7处理器的cache结构 4.3

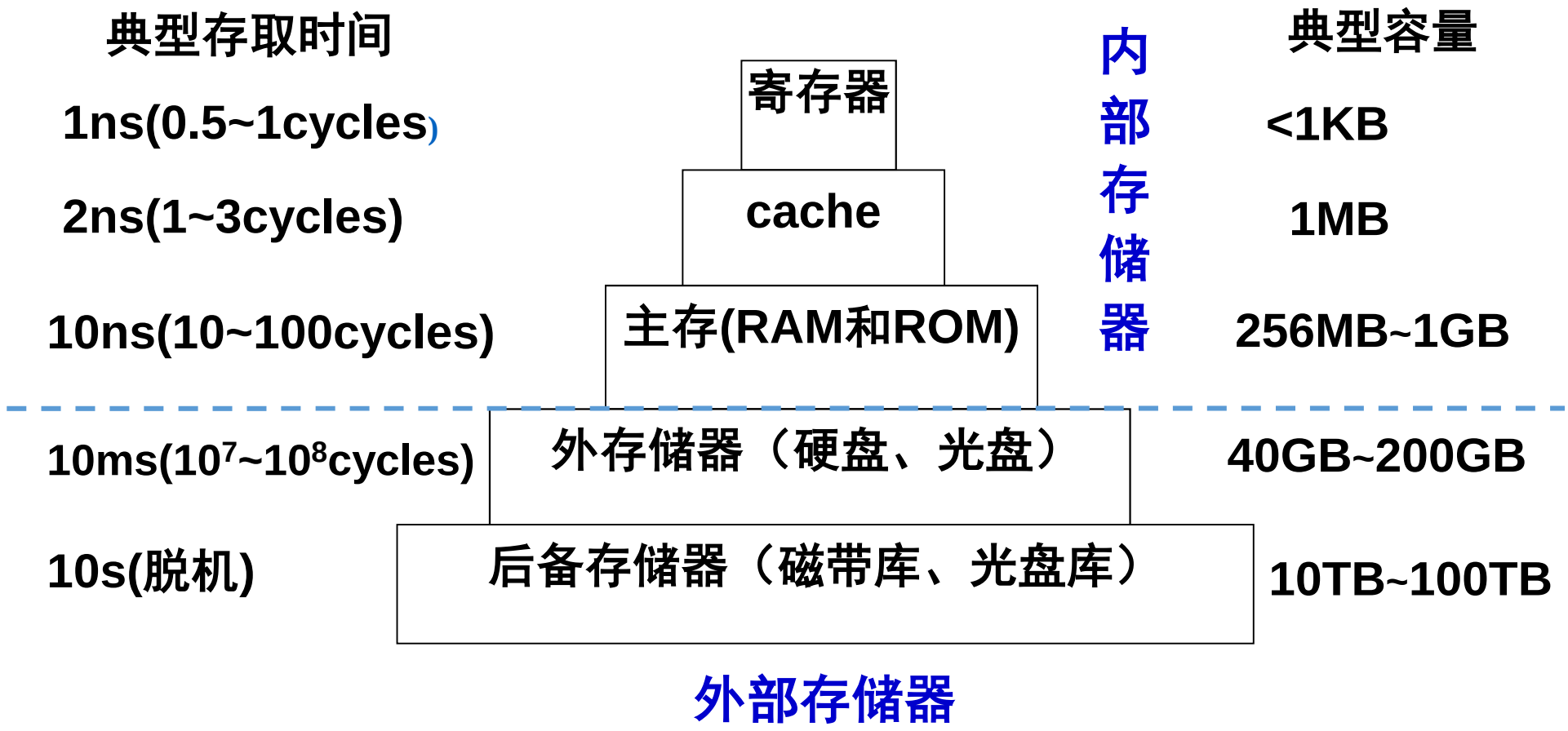


i-cache和d-cache都是32KB、8路、4个时钟周期；L2 cache：256KB、8路、11个时钟周期。所有核共享的L3 cache：8MB、16路、30~40个时钟周期。Core i7中所有cache的块大小都是64B

4.4 虚拟存储器

- **虚拟存储器概述**
- 页式虚拟存储器及页表管理
- 虚拟存储器地址映射与变换

• 存储器的层次结构



列出的时间和容量会随时间变化，但数量级相对关系不变。

4.4

• 虚拟存储器

- 比尔·盖茨(1981) “ 无论对谁来说,640K内存都足够了”
- 如何在有限的主存空间运行较多的较大的用户程序？
  - 初衷： 采用虚存来扩大寻址空间
  - 现在： 主要用于存储保护，程序共享
- 主存、辅存在OS和硬件的管理之下，提供更大的存储空间
- 虚存空间是靠辅存（磁盘）来支持的
- 虚拟存储采用与cache类似原理，提高速度，降低成本
- 用户感觉到的是一个速度接近主存而容量极大的存储器

• 虚拟存储系统的基本概念

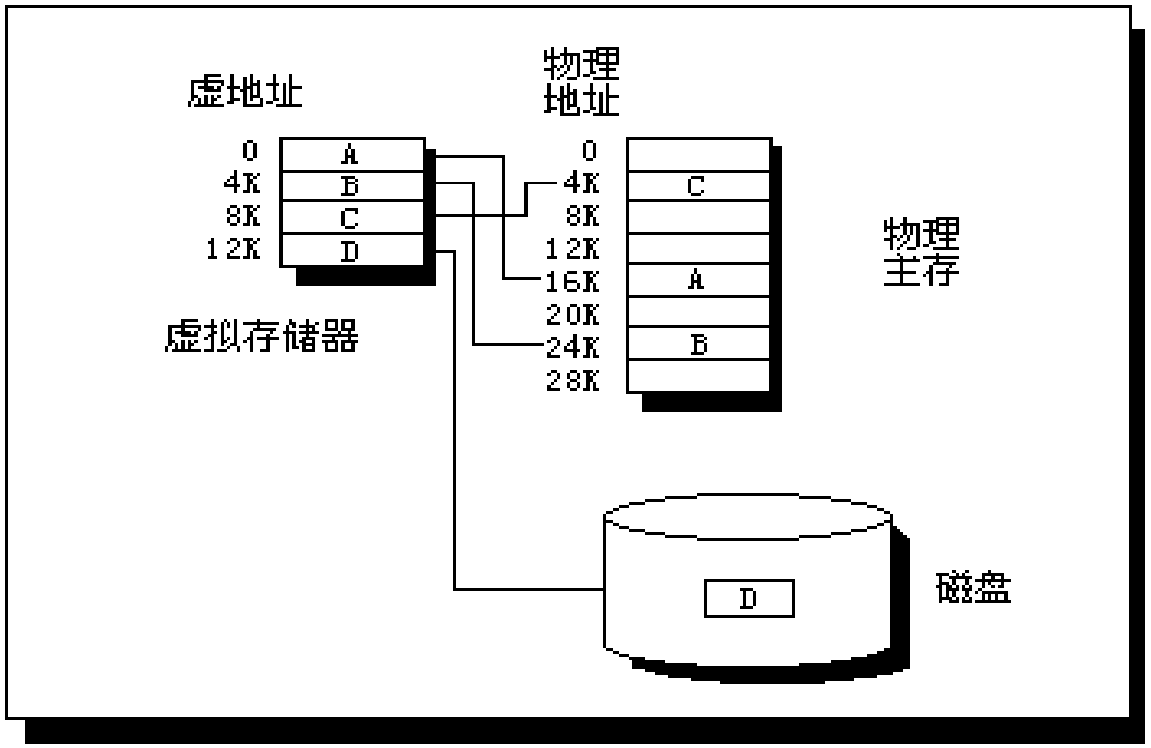
- 虚拟存储技术的引入用来解决一对矛盾
  - 一方面，由于技术和成本等原因，主存容量受到限制
  - 另一方面，系统程序和应用程序要求主存容量越来越大
- 虚拟存储技术的实质
  - 程序员在比实际主存空间大得多的逻辑地址空间中编写程序
  - 程序执行时，把当前需要的程序段和相应的数据块调入主存，其他暂不用的部分存放在磁盘上
  - 指令执行时，通过硬件将逻辑地址（也称虚拟地址或虚地址）转化为物理地址（也称主存地址或实地址）
  - 在发生程序或数据访问失效(缺页)时，由操作系统进行主存和磁盘之间的信息交换
- 虚拟存储器机制由硬件与操作系统共同协作实现，涉及到操作系统中的许多概念，如进程、进程的上下文切换、存储器分配、虚拟地址空间、缺页处理等。

• 虚拟存储器基本原理

4.4

虚拟存储器的特点

- ◆程序员可以利用巨大的逻辑空间，而不必做存储管理工作
- ◆多个进程可以共享主存空间
- ◆ 采用动态再定位，简化了程序的装入



• 虚拟存储器基本原理

4.4

1、虚存管理方式--页式管理

- 页式虚存把主存与外存均划分成等长的页面。
- 常用页大小为4KB～64KB。
- 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

• 虚拟存储器基本原理

2、虚存管理方式--段式管理

- 段式虚存把空间划分为可变长的块，称为段，段的分界与程序的自然分界相对应，段最小长度为1个字节，最大因机器而异，常为2^{16}B～2^{32}B。
- 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

• 虚拟存储器基本原理

3、虚存管理方式--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

- “Cache－主存”与“主存－辅存”层次的区别 4.4

存储层次 / 比较项目	“Cache - 主存”层次	“主存 - 辅存”层次
目　　的	弥补主存速度的不足	弥补主存容量的不足
存储管理实现	专用硬件实现	软件实现
访问速度的比值(第一级和第二级)	几比一	几百比一
典型的块(页)大小	几十个字节	几百~几千个字节
CPU对第二级的访问方式	可直接访问	均通过第一级
失效时CPU是否切换	不切换	切换到其他进程

- 存储层次中cache和虚存的典型指标

参数	一级cache	虚拟存储器
块（页）大小	16~128字节	4096~65536字节
命中时间	1~3个时钟周期	50~150个时钟周期
失效开销	8~150个时钟周期	10^6~10^7个时钟周期
失效率	0.1%~10%	0.00001%~0.001%
地址映射关系	25~45位物理地址映射到14~20位cache地址	32~64位虚地址映射到25~45位物理地址

有关虚拟存储器的四个问题 4.4

（1）映象规则

全相联，以降低失效率为主要目标。

（2）查找算法

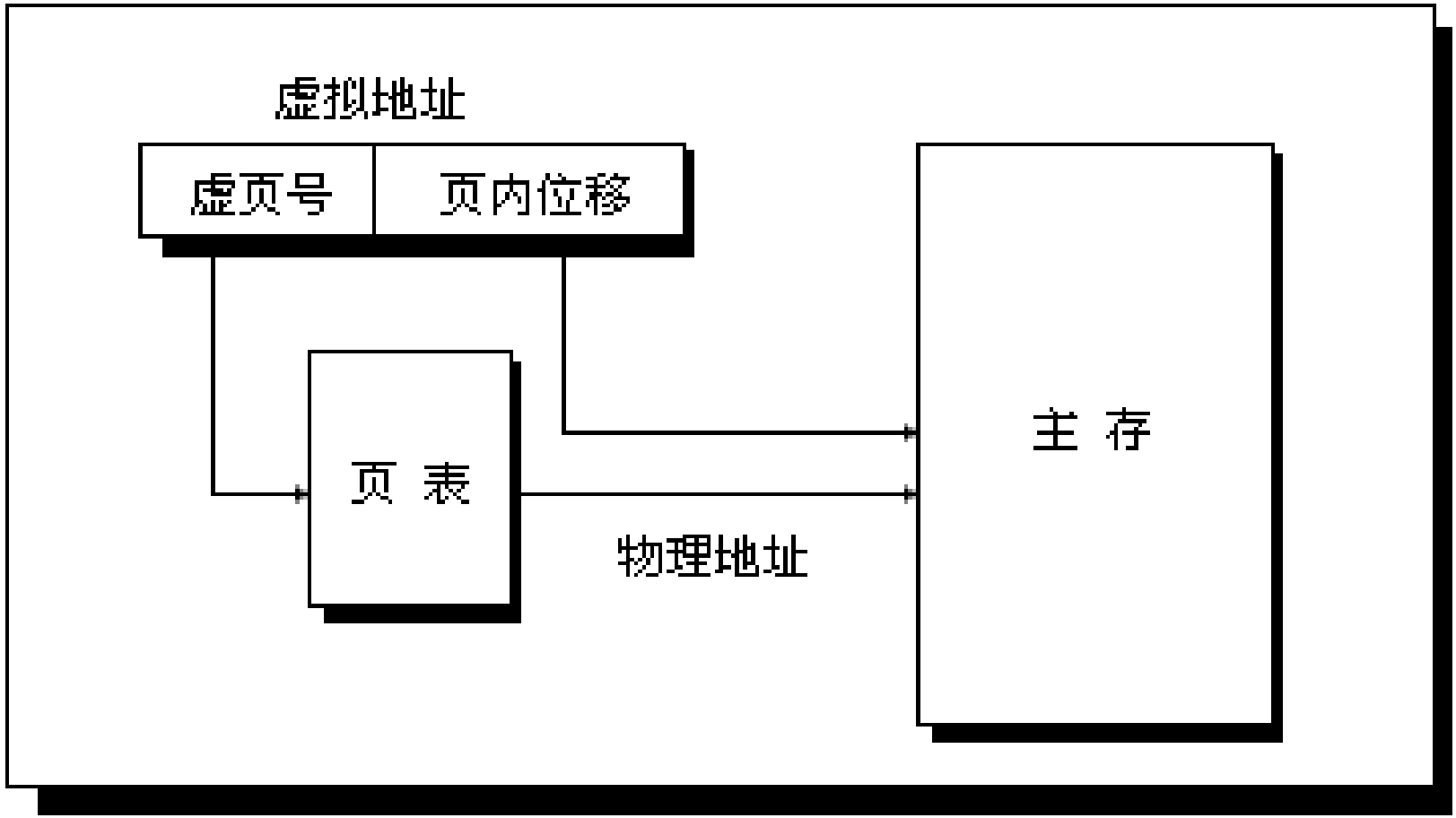
页表，段表，TLB

页表和段表：索引页号或段号的数据结构，含有所要查找的块的物理地址。

段式系统：段内位移加上段的物理地址就是最终的物理地址。

页式系统：只需简单地将页内位移拼接在相应页面的物理地址之后。

页表



有关虚拟存储器的四个问题 4.4

(3) **替换算法**

LRU等

尽可能减少页故障，OS为每个页面设置“使用位”。

(4) **写策略**

写回法

4.4 虚拟存储器

- 虚拟存储器概述
- **页式虚拟存储器及页表管理**
- **虚拟存储器地址映射与变换**

4.4

- 分页（Paging）

- 基本思想：
  - 内存被分成固定长且比较小的存储块（页框、实页、物理页）
  - 每个进程也被划分成固定长的程序块（页、虚页、逻辑页）
  - 程序块可装到存储器中可用的存储块中,无需用连续页框来存放一个进程
  - 操作系统为每个进程生成一个页表,通过页表(page table)实现逻辑地址向物理地址转换（Address Mapping ）
- 逻辑地址（Logical Address）：
  - 程序中指令所用地址(进程所在地址空间)，也称为虚拟地址（Virtual Address，简称VA）
- 物理地址（Physical Address，简称PA）：
  - 存放指令或数据的实际内存地址，也称为实地址、主存地址。

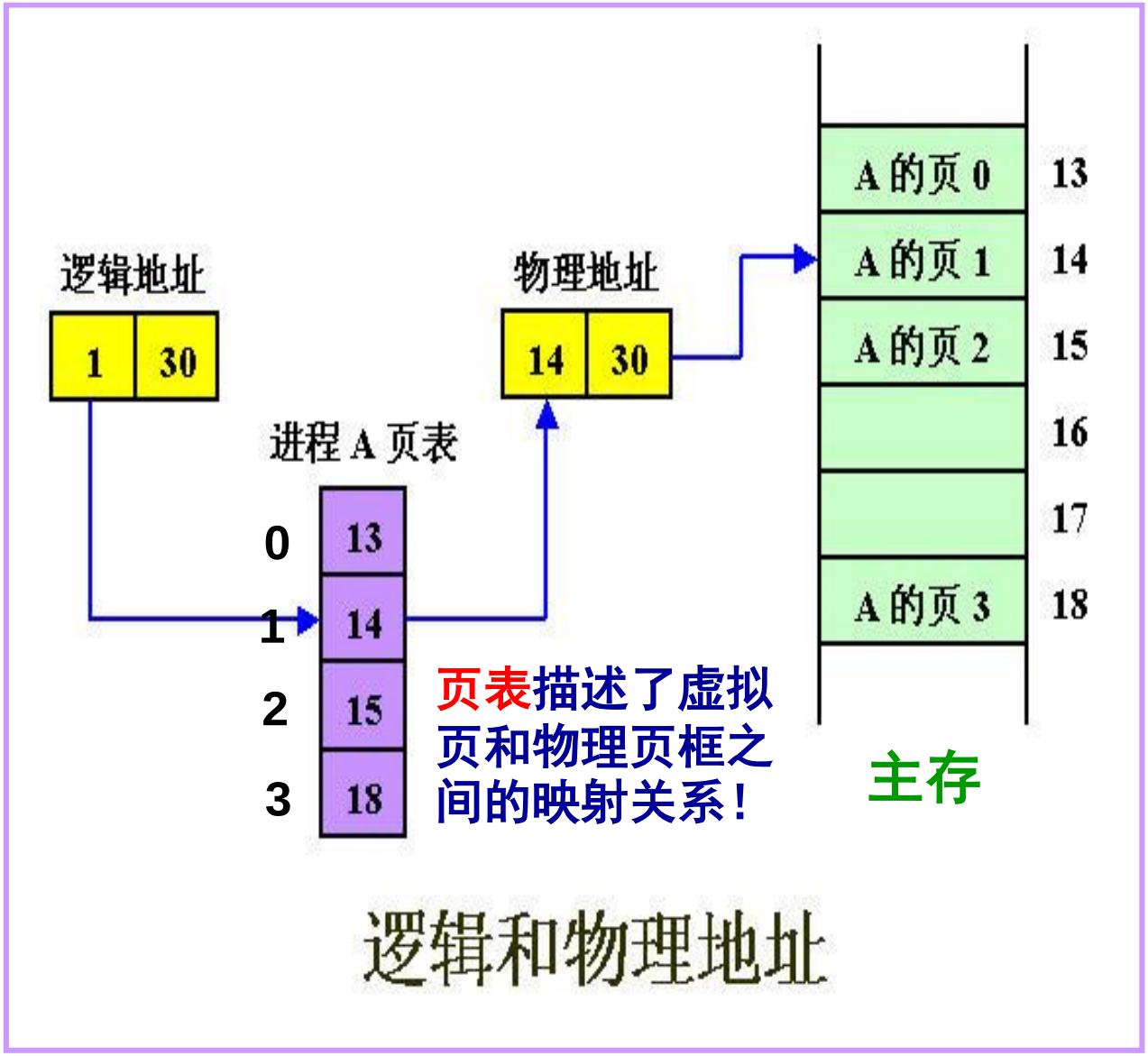
• 分页（Paging）

问题：是否需要将一个进程的全部都装入内存?

根据程序访问局部性可知：可把当前活跃的页面调入主存，其余留在磁盘上!

采用 “按需调页 Demand Paging” 方式分配主存！这就是虚拟存储管理概念

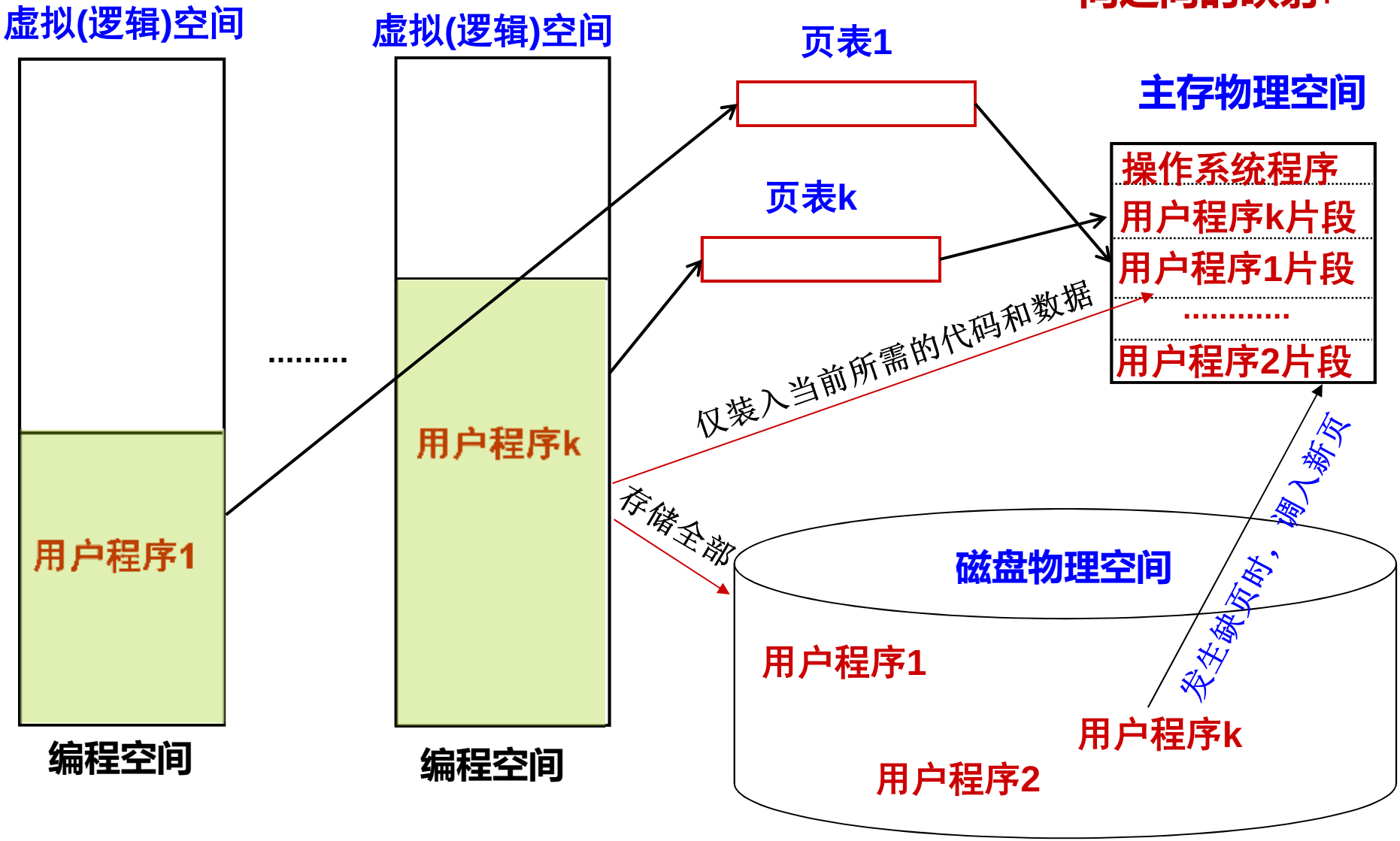
优点：浪费的空间最多是最后一页的部分!



逻辑和物理地址

页表--虚拟空间与物理空间的映射

通过页表建立虚拟空间和物理空间之间的映射!

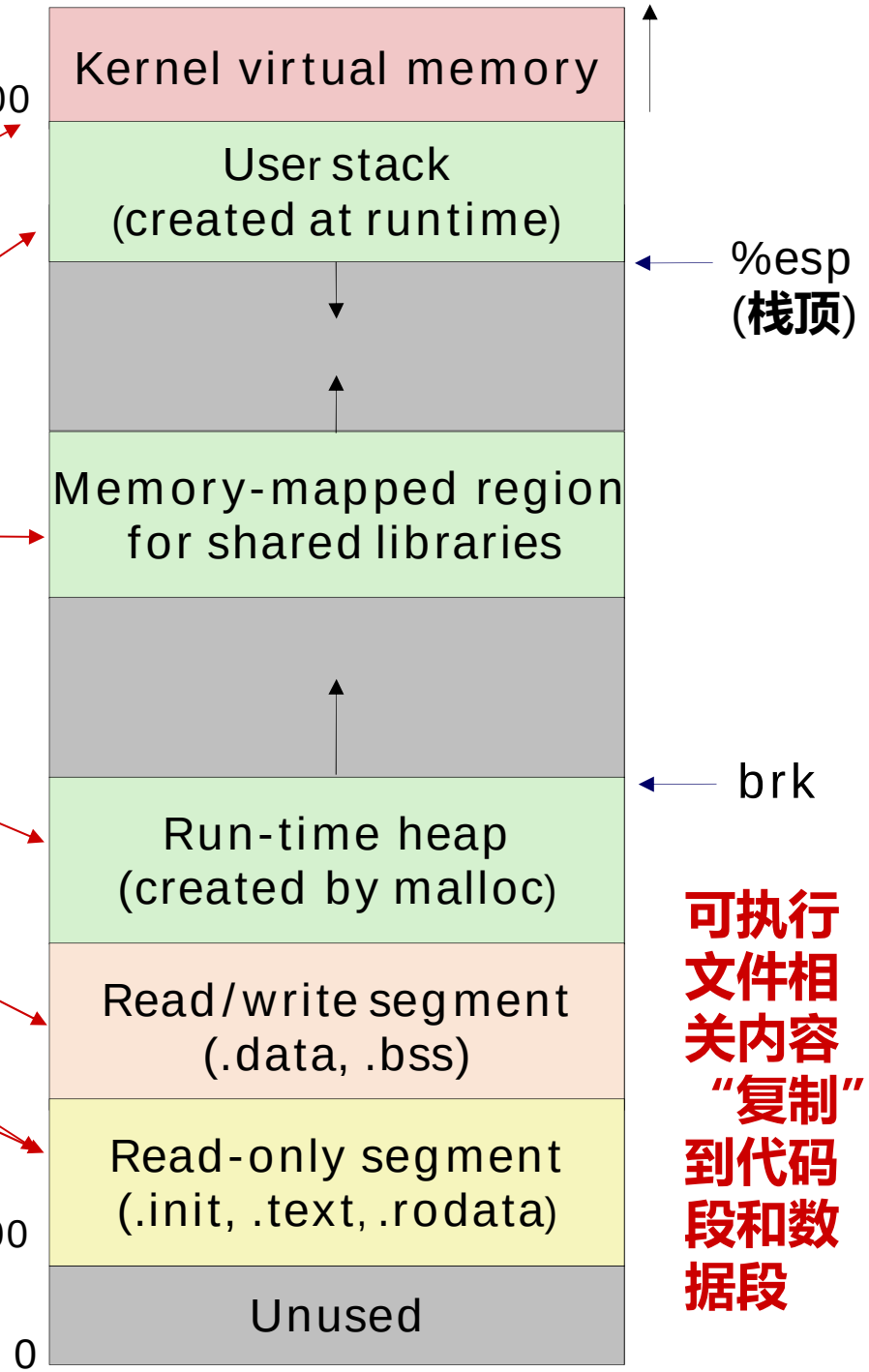


虚拟地址空间

- Linux在X86上的虚拟地址空间

  (其他Unix系统的设计类此)

  - 内核空间（Kernel）
  - 用户栈（User Stack）
  - 共享库（Shared Libraries）
  - 堆（heap）
  - 可读写数据（Read/Write Data）
  - 只读数据（Read-only Data）
  - 代码（Code）



问题：加载时是否真正从磁盘调入信息到主存?

实际上不会从磁盘调入，只是将虚拟页和磁盘上的数据/代码建立对应关系，称为“映射”。

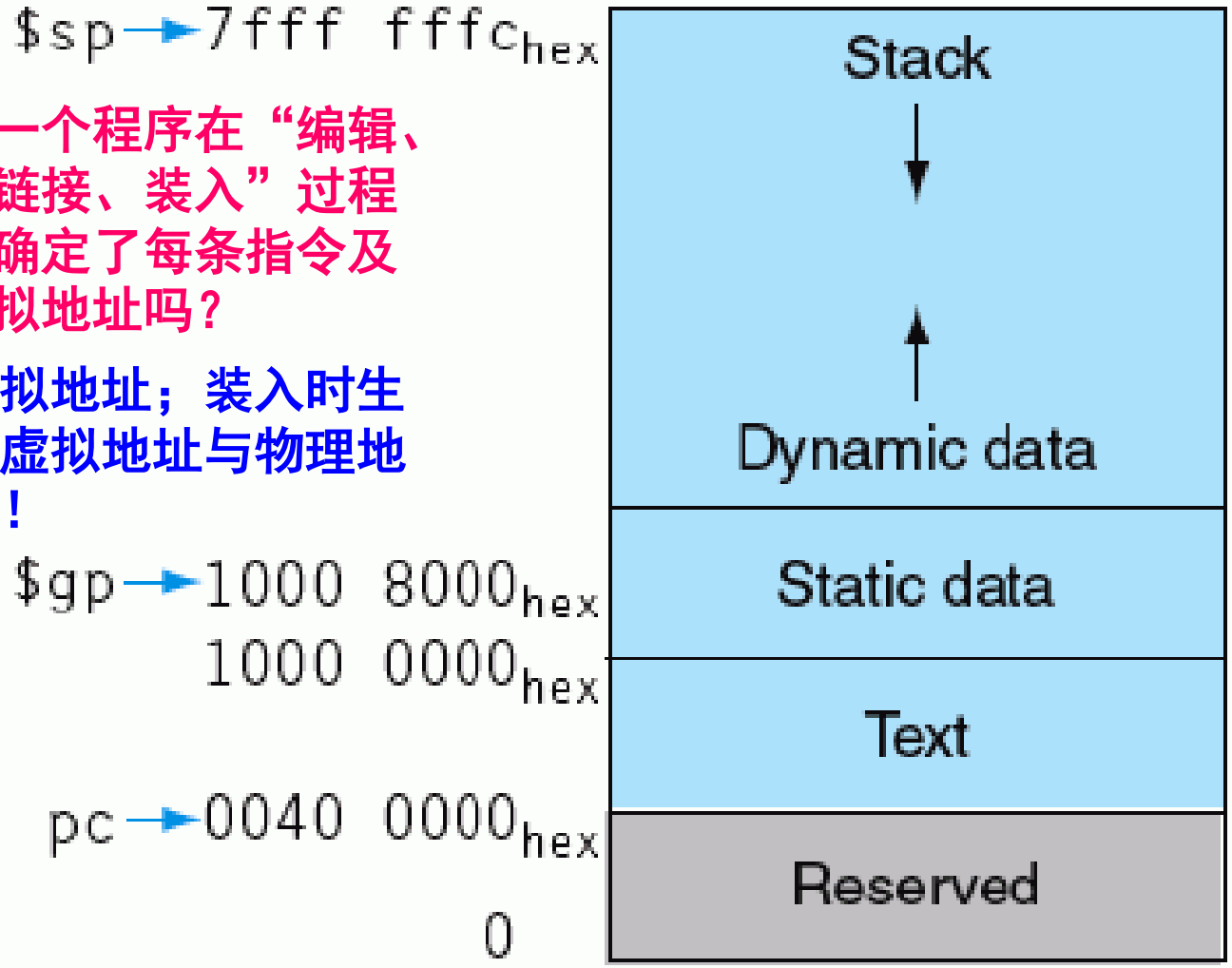
可执行文件相关内容“复制”到代码段和数据段

MIPS程序和数据的存储器分配

问题：你知道一个程序在“编辑、编译、汇编、链接、装入”过程中的哪个环节确定了每条指令及其操作数的虚拟地址吗？

链接时确定虚拟地址；装入时生成页表以建立虚拟地址与物理地址之间的映射！

请参考《深入理解计算机系统》和有关Linux内核分析方面的资料。



每个用户程序都有相同的虚拟地址空间！

这就是每个进程的虚拟（逻辑）地址空间！

页表结构

◆ **页表首址记录在页表基址寄存器中**

页表首地址

	装入位	修改位	替换控制位	其他	实页号（8进制）
0 虚页	1				11
1 虚页	1				13
2 虚页	1				16
3 虚页	1				10
4 虚页	1				14

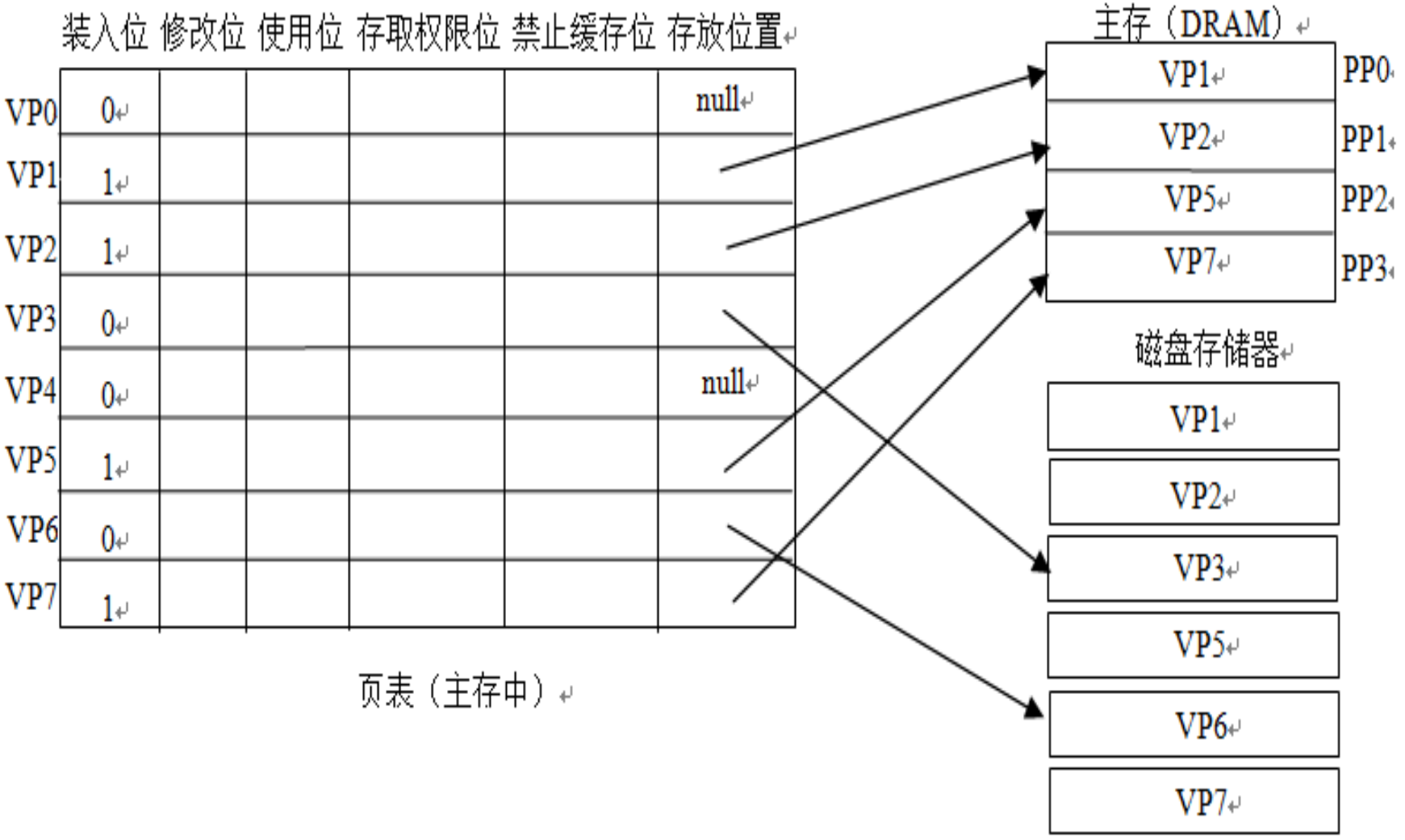
- 每个进程有一个页表，其中有装入位、修改（Dirt）位、替换控制位、访问权限位、禁止缓存位、实页号。
- 一个页表的项数由什么决定？

理论上由虚拟地址空间大小决定。

- 每个进程的页表大小一样吗？

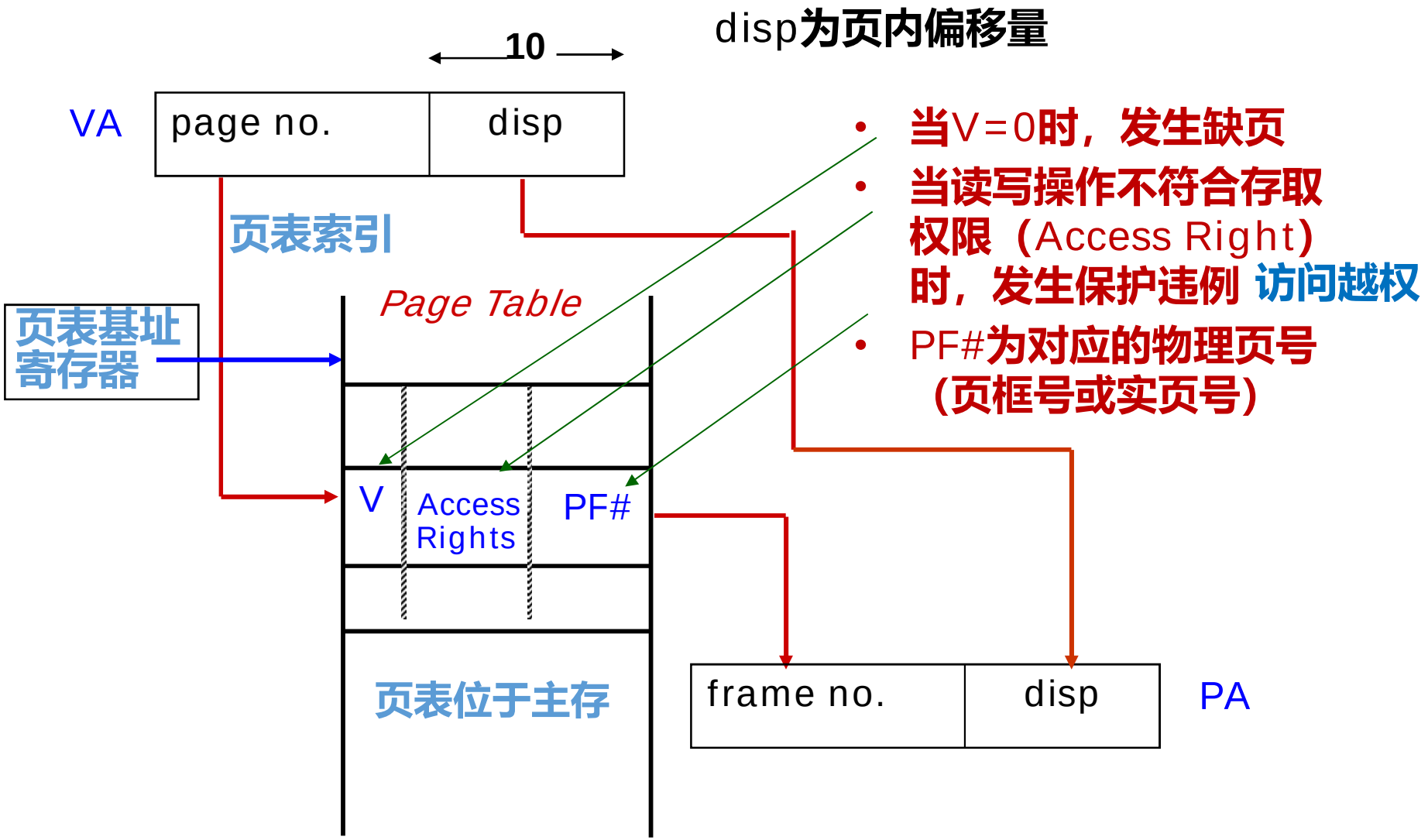
各进程有相同虚拟空间，故理论上一样。实际大小看具体实现方式，如“空洞”页面如何处理等

主存中的页表示例



- **未分配页**：进程的虚拟地址空间中“空洞”对应的页（如VP0、VP4）
- **已分配的缓存页**：有内容对应的已装入主存的页（如VP1、VP2、VP5等）
- **已分配的未缓存页**：有内容对应但未装入主存的页（如VP3、VP6）

逻辑地址转换为物理地址的过程



• 信息访问中可能出现的异常情况 4.4

可能有两种异常情况：

1）**缺页（** page fault**）**

产生条件：当 Valid**（有效位 / 装入位）为** 0 **时**

相应处理：从磁盘读到内存，若内存没有空间，则还要从内存选择一页替换到磁盘上，替换算法类似于 Cache**，采用写回法，淘汰时，根据“**dirty**”位确定是否要写磁盘**

当前指令执行被阻塞，当前进程被挂起，处理结束回到原指令继续执行

2）**保护违例（** protection_violation_fault **）或访问违例**

产生条件：当 Access Rights **(存取权限)与所指定的具体操作不相符时**

相应处理：在屏幕上显示“内存保护错”或“访问违例”信息

当前指令的执行被阻塞，当前进程被终止

Access Rights **(存取权限)可能的取值有哪些?**

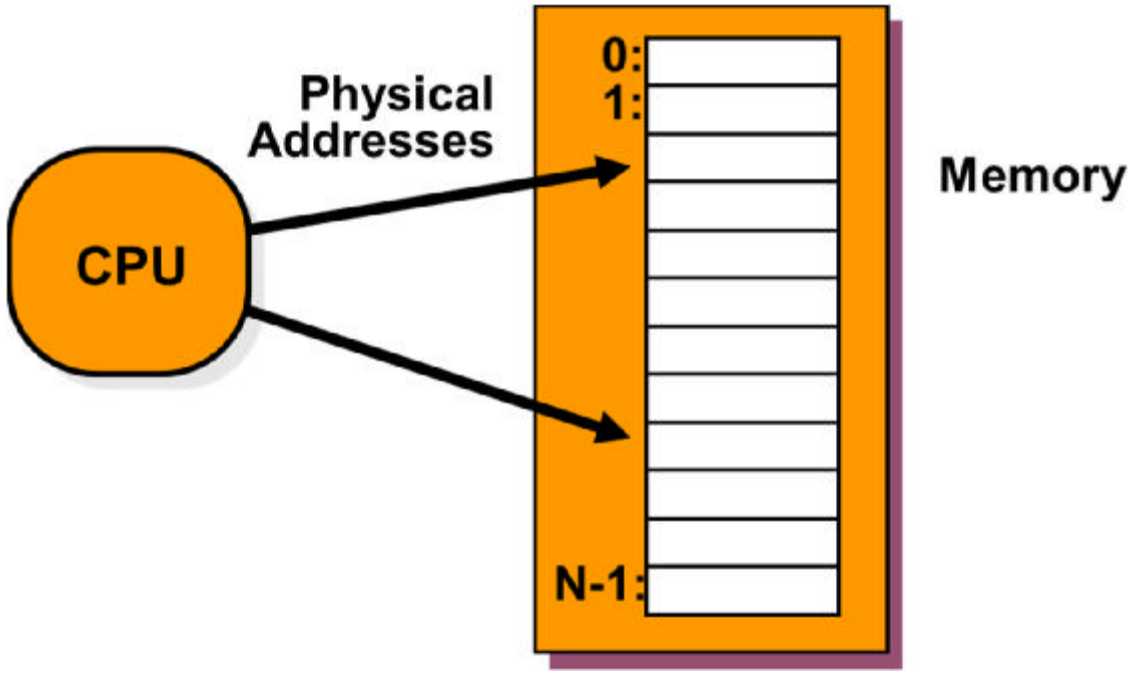
R = Read-only, R/W = read/write, X = execute only

4.4 虚拟存储器

- 虚拟存储器概述
- 页式虚拟存储器及页表管理
- **虚拟存储器地址映射与变换**

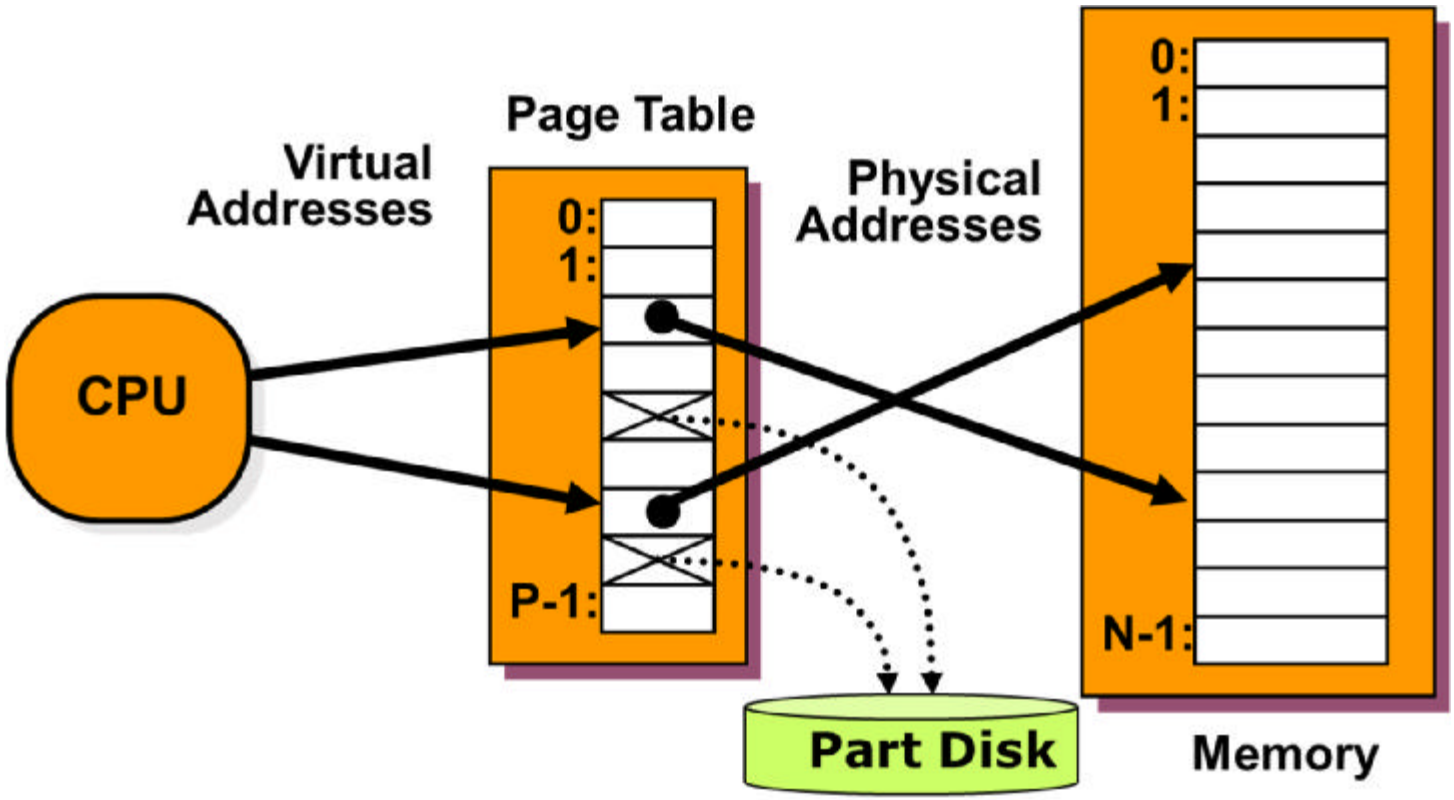
- 实地址计算机系统

- CPU地址：物理内存地址
  - 大多数Cray计算机，早期PC，大多数嵌入式系统



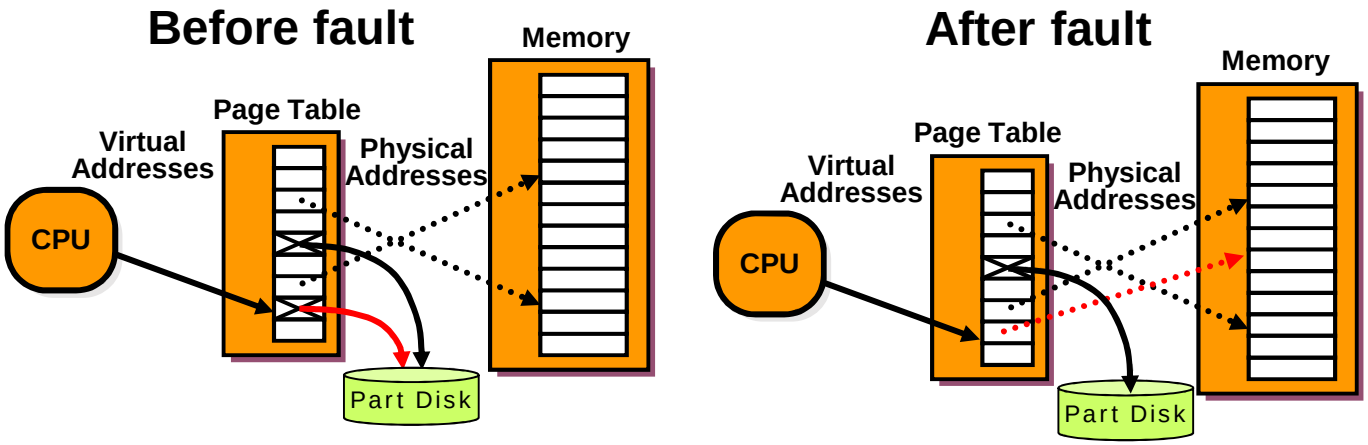
• 虚地址计算机系统

- CPU地址：虚拟地址
  - 需要虚实变换，硬件通过OS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

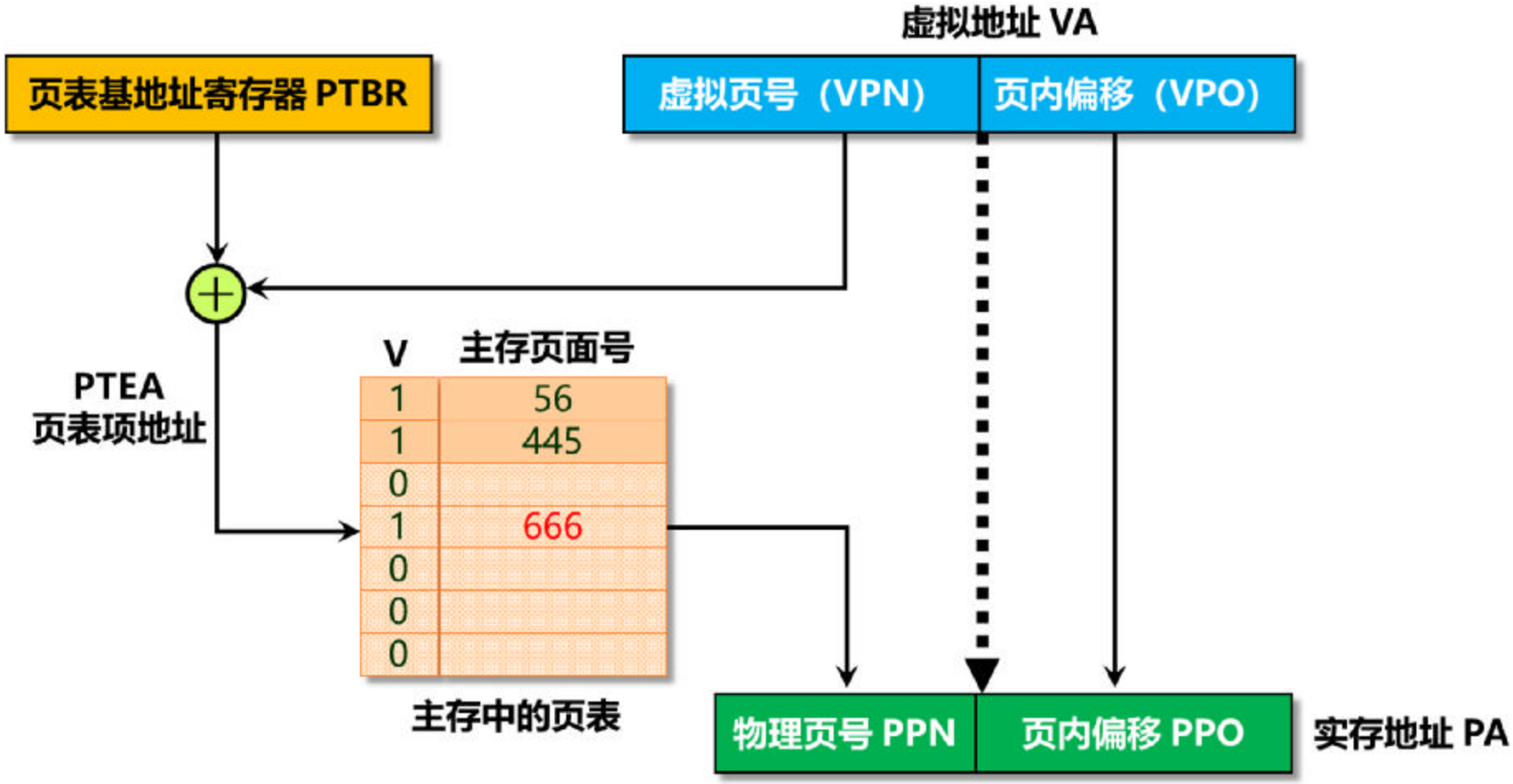


• 缺页 Page Faults

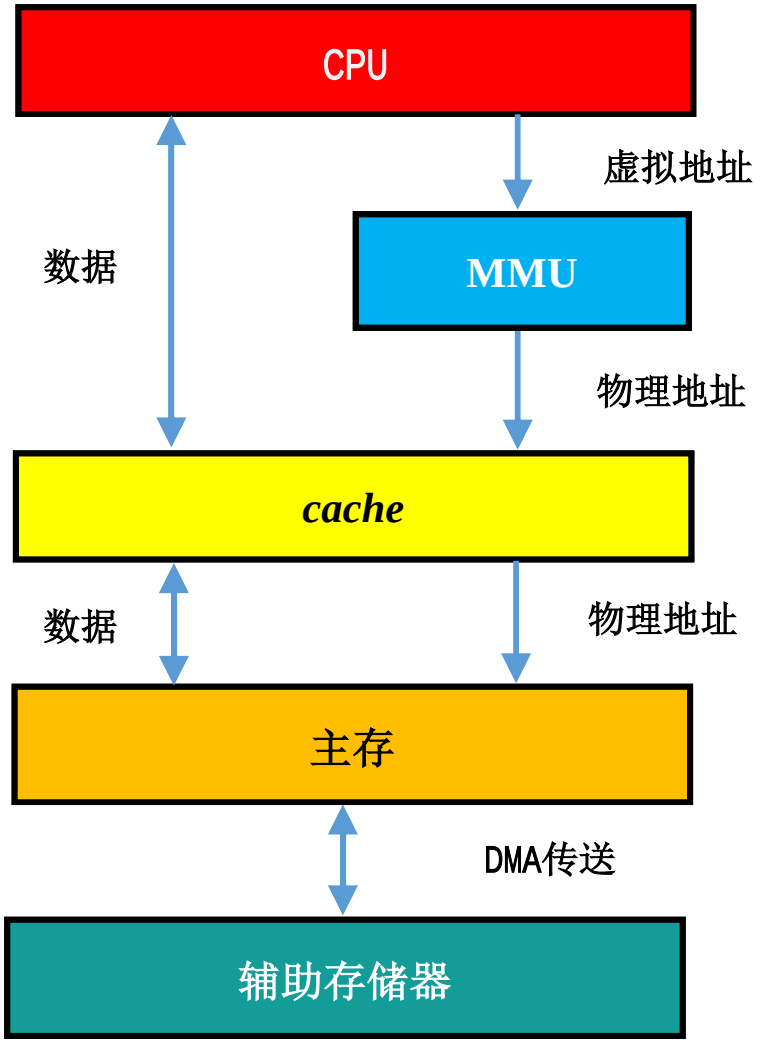
- 页表指示虚拟地址不在内存中
  - 操作系统负责将数据从磁盘迁移到内存中，当前进程挂起
  - 操作系统负责所有的替换策略，唤醒挂起进程



- 页式虚拟存储器结构

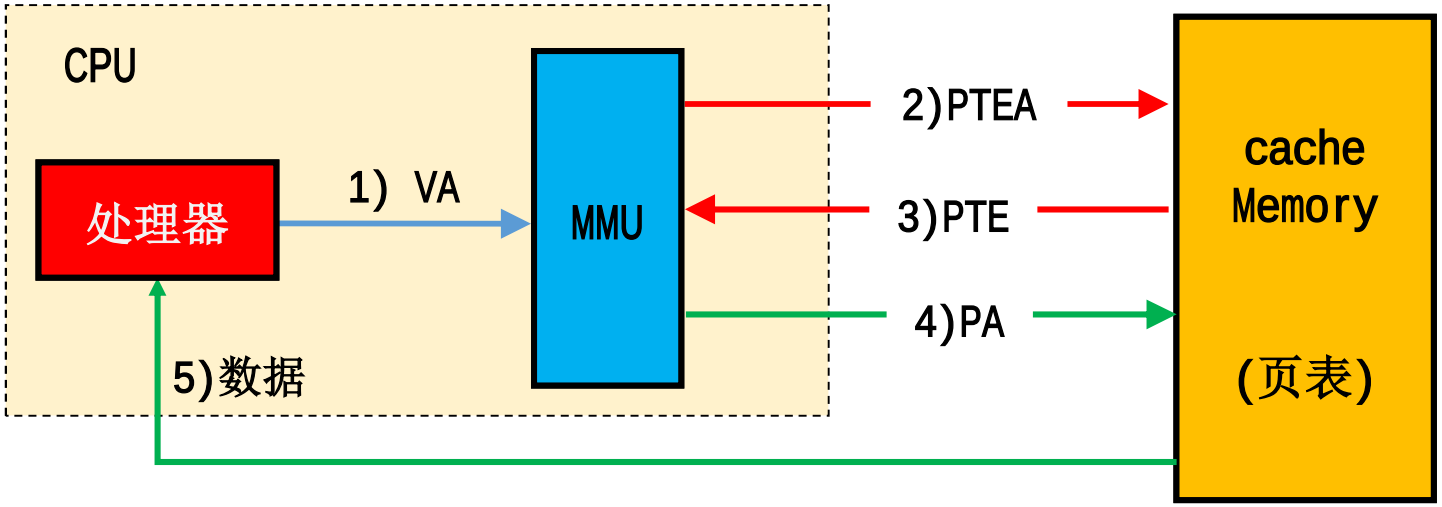

虚拟地址 VA
页表基地址寄存器 PTBR
虚拟页号（VPN）
页内偏移（VPO）
PTEA
页表项地址
V
主存页面号
1 56
1 445
0
1 666
0
0
0
主存中的页表
物理页号 PPN
页内偏移 PPO
实存地址 PA

层次性结构

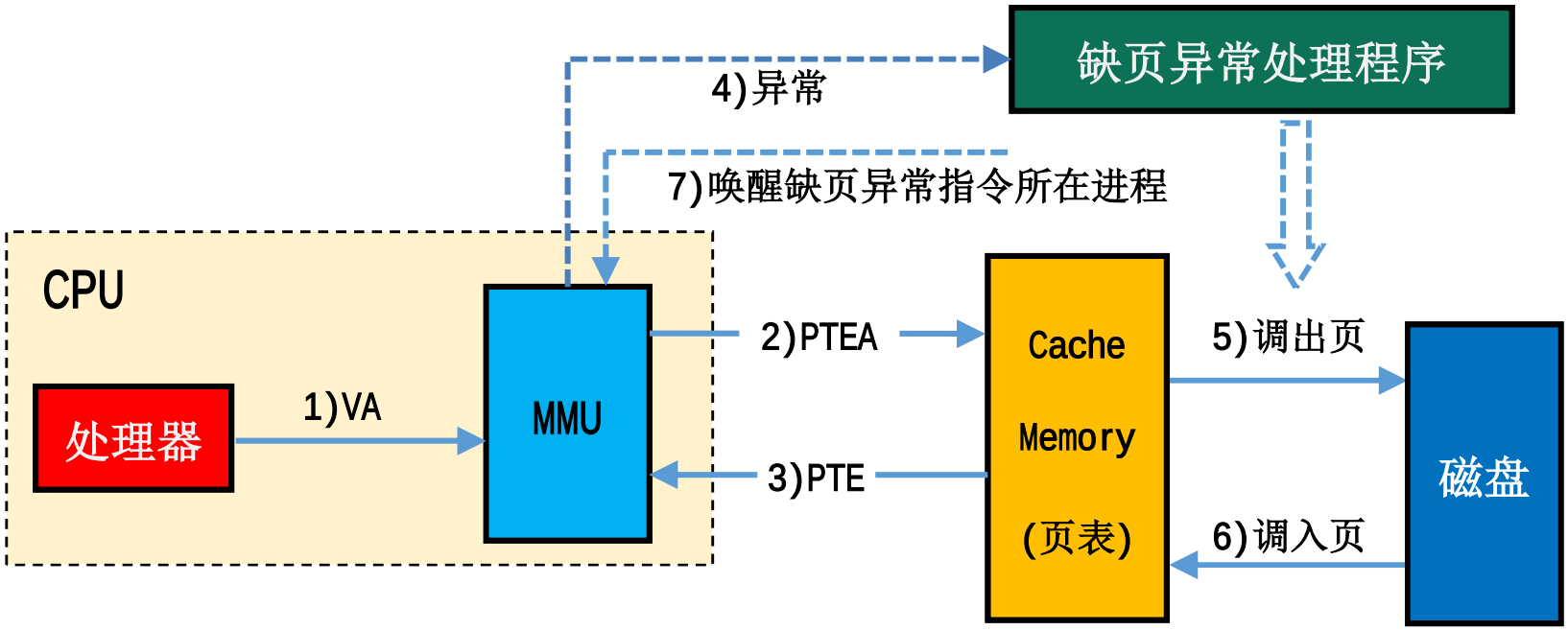


为什么cache用物理地址访问？

- 虚拟地址→物理地址 （页命中）



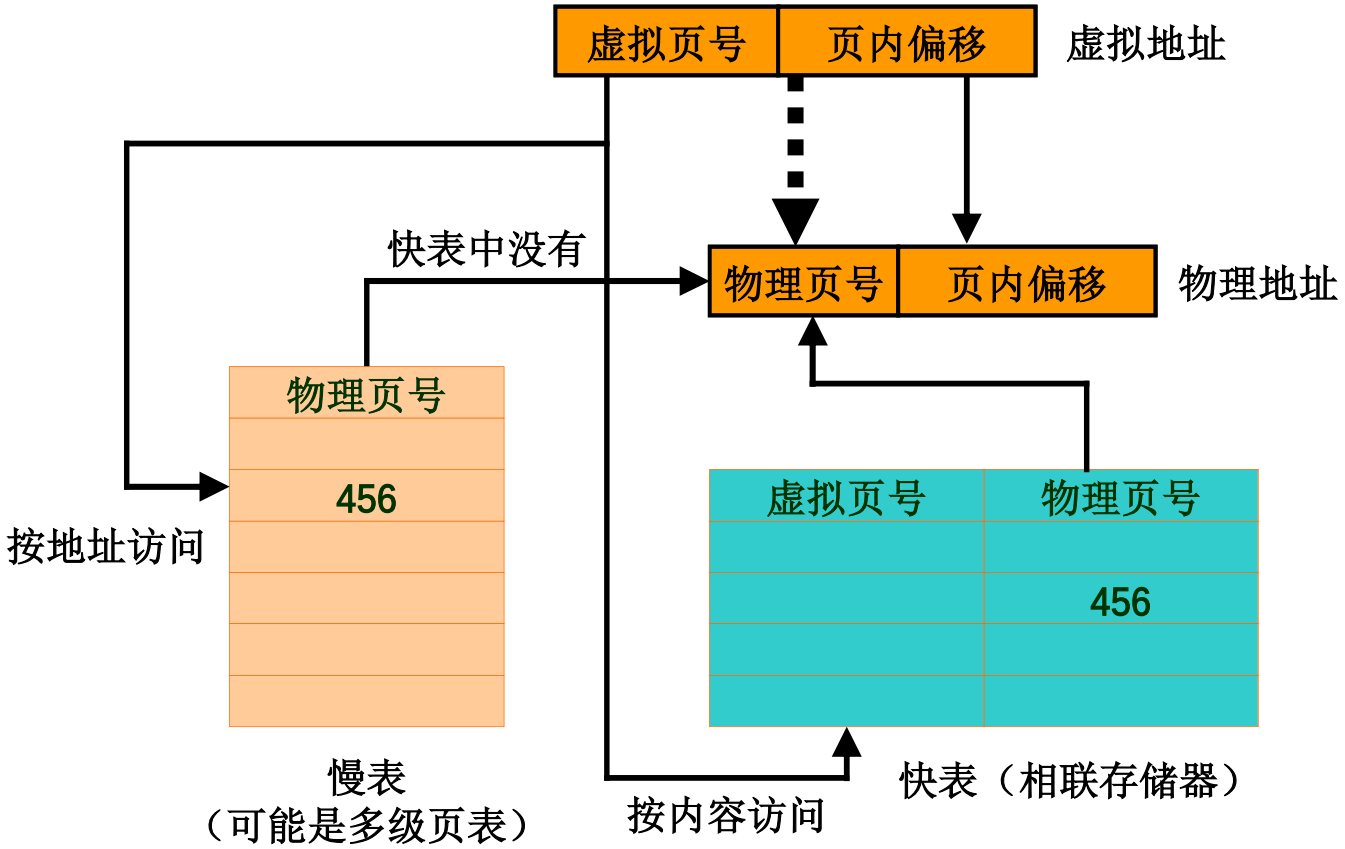
- 虚拟地址→物理地址（缺页）


缺页异常处理程序
4)异常
7)唤醒缺页异常指令所在进程
CPU
处理器
1)VA
MMU
2)PTEA
3)PTE
Cache
Memory
(页表)
5)调出页
6)调入页
磁盘

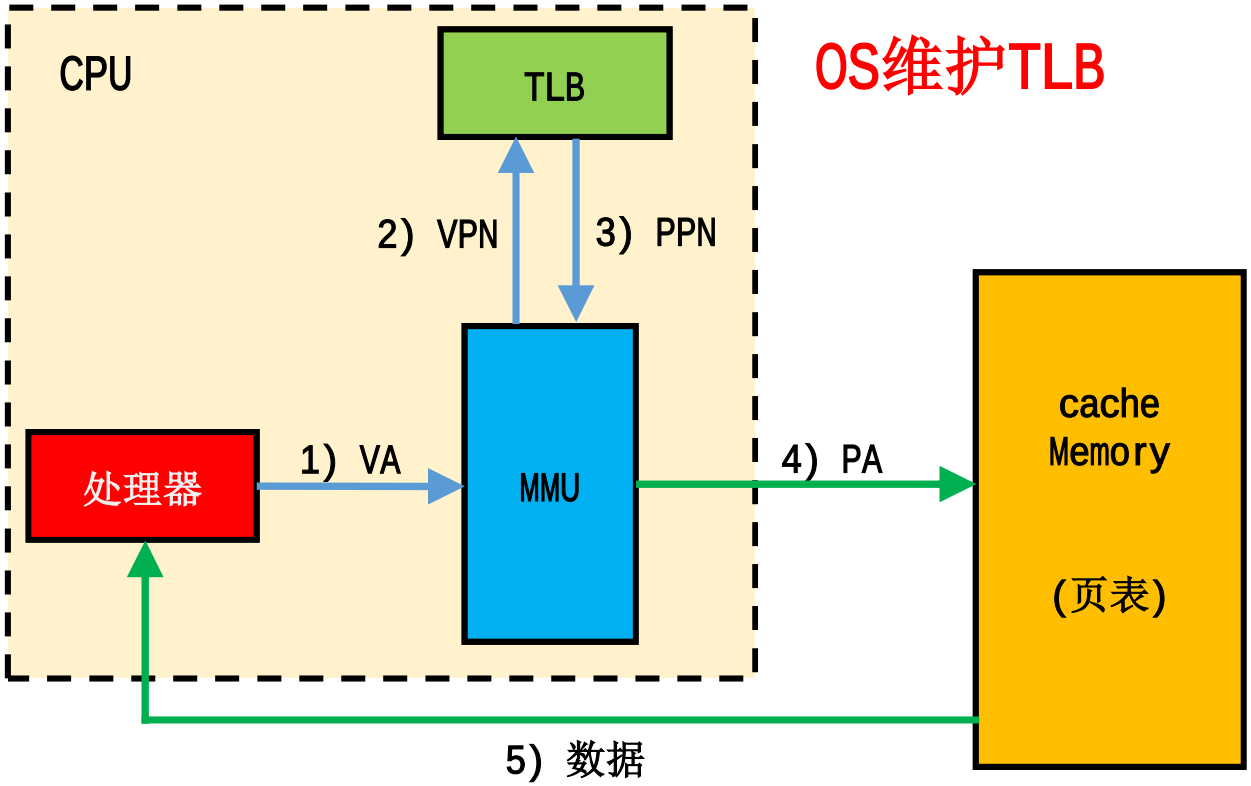
- 快表提高地址转换速度

  - 地址转换速度慢
    - 访问页表，访问数据，需2次访存，速度慢
    - 为缩小页表大小，OS普遍采用多级页表结构，速度更慢
  - 加速方法：引入一个体积小的快表TLB
    - 缓存页表中经常被访问的表项
    - (Valid，VPN，PPN)
  - 快表引入相联存储器机制，提高查找速度
    - 采用随机替换算法

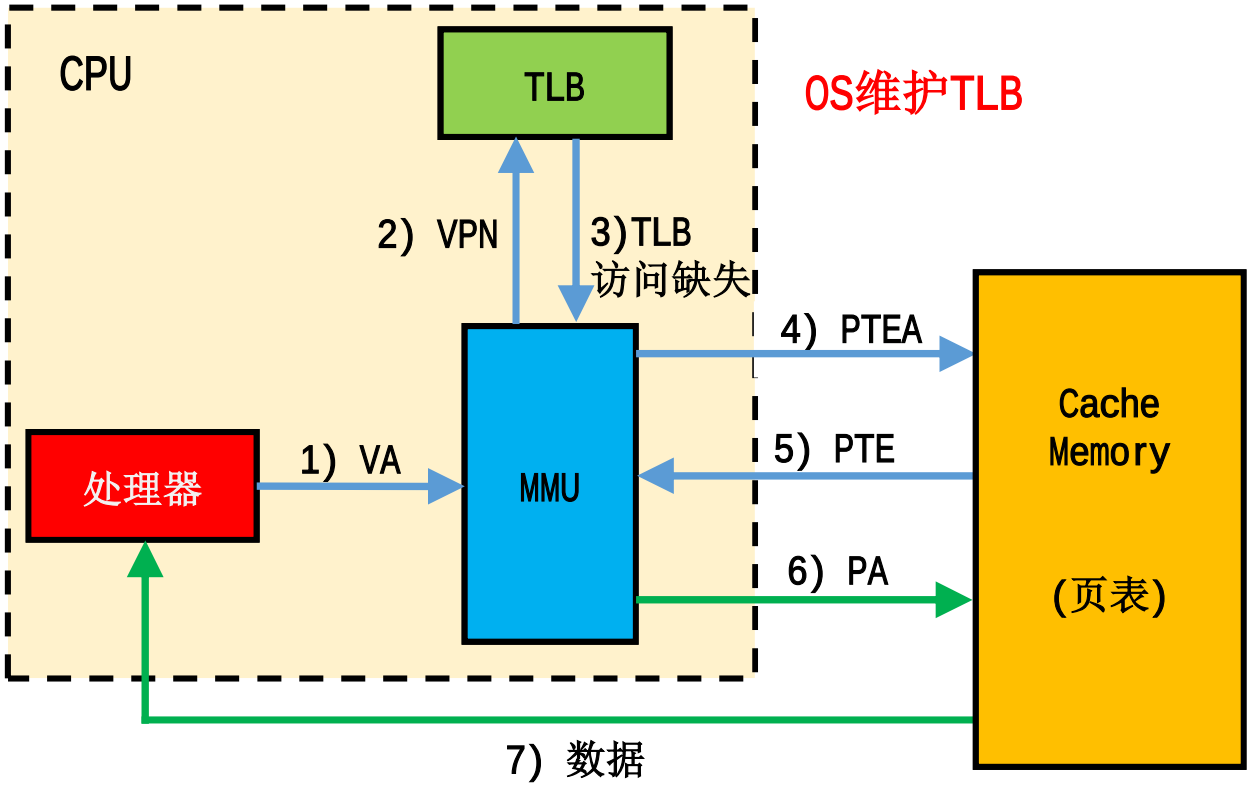
经快慢表实现内部地址转换



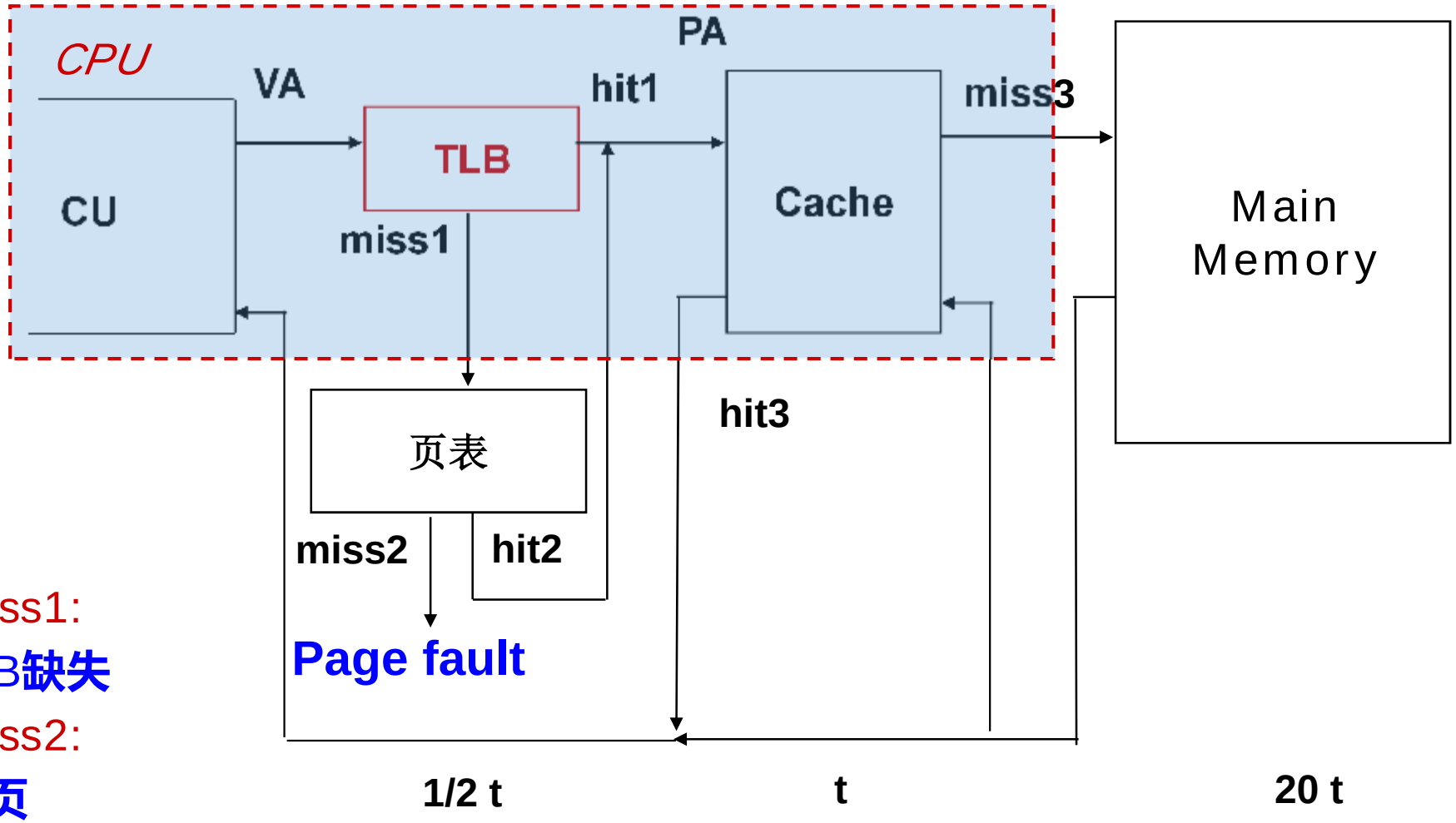
- TLB 命中



- TLB 缺失


CPU
TLB
OS维护TLB
2) VPN
3)TLB
访问缺失
4) PTEA
处理器
1) VA
MMU
5) PTE
Cache
Memory
6) PA
(页表)
7) 数据

Translation Look-Aside Buffers



Miss1:

TLB**缺失**

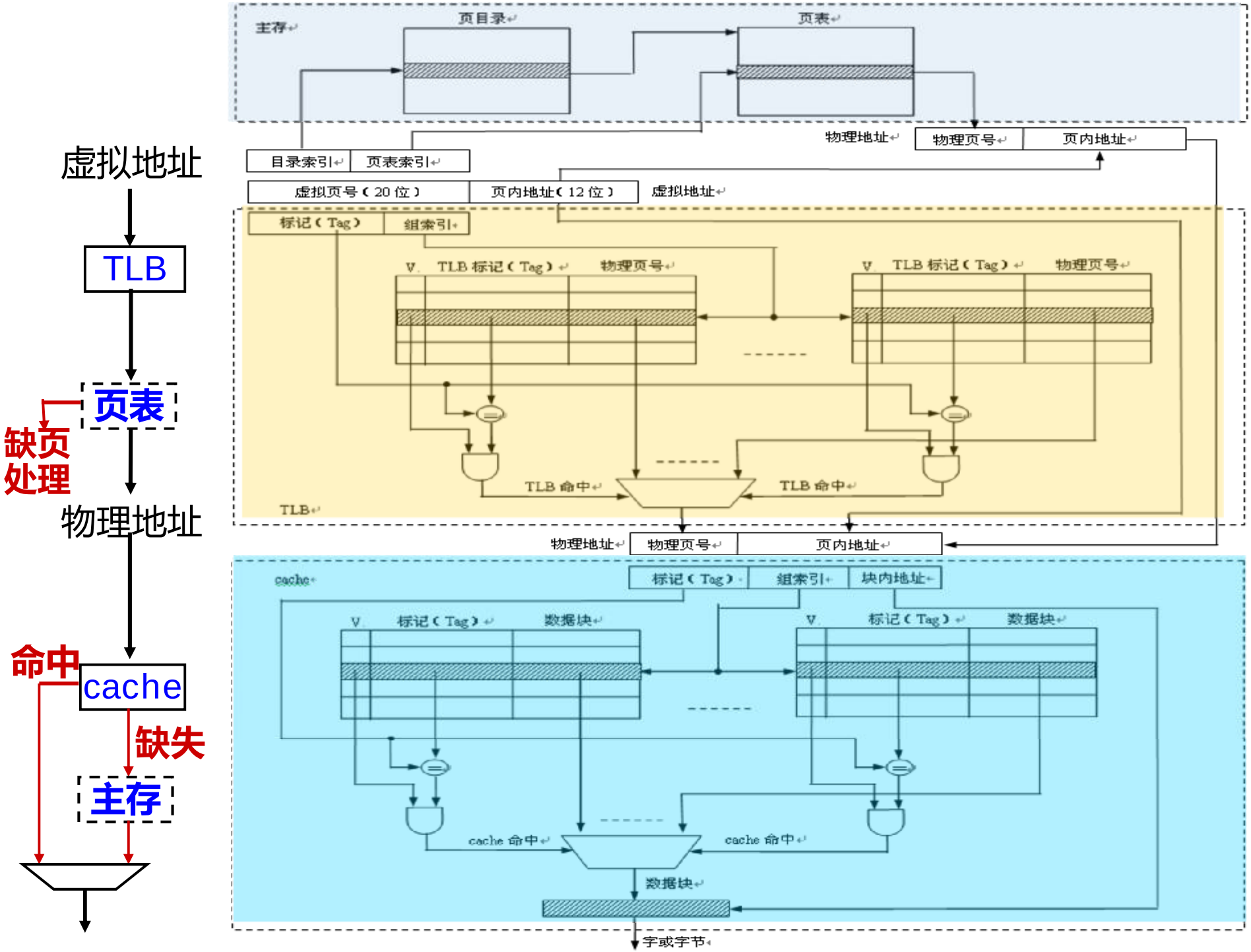
Miss2:

缺页

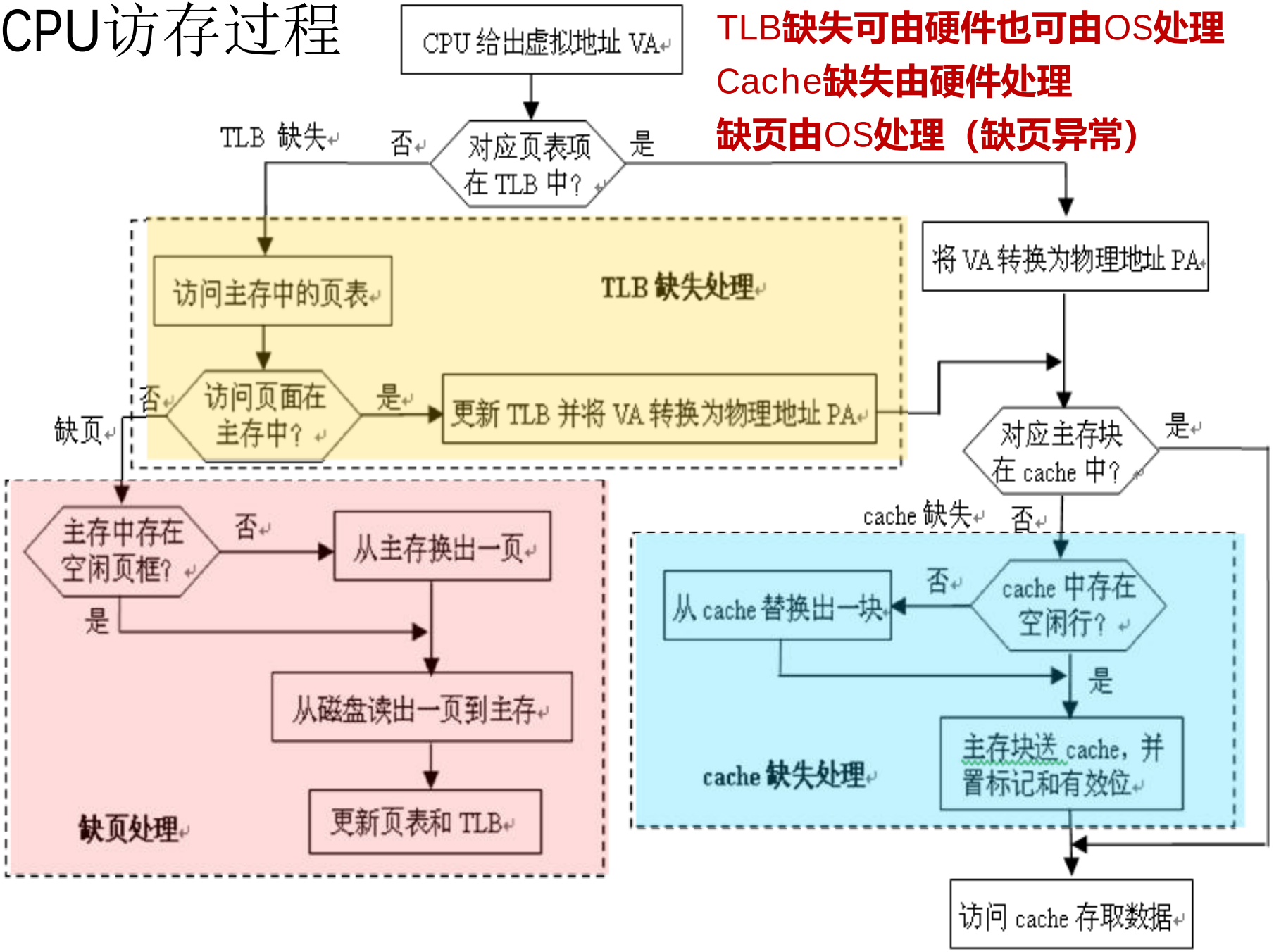
Miss3:

PA **在主存中，但不在**Cache**中**

TLB**冲刷指令和**Cache**冲刷指令**
都是操作系统使用的特权指令


虚拟地址
TLB
页表
缺页
处理
物理地址
命中
cache
缺失
主存
主存
页目录
页表
物理地址
物理页号
页内地址
目录索引
页表索引
虚拟页号（20 位）
页内地址（12 位）
虚拟地址
标记（Tag）
组索引
V
TLB 标记（Tag）
物理页号
TLB 命中
TLB
物理地址
物理页号
页内地址
cache
标记（Tag）
组索引
块内地址
V
标记（Tag）
数据块
cache 命中
数据块
字或字节

CPU访存过程



TLB缺失可由硬件也可由OS处理

Cache缺失由硬件处理

缺页由OS处理（缺页异常）

举例：三种不同缺失的组合

TLB	Page table	Cache	Possible? If so, under what circumstance?
hit	hit	miss	可能，TLB命中则页表一定命中，但实际上不会查页表
miss	hit	hit	可能，TLB缺失但页表命中，信息在主存，就可能在Cache
miss	hit	miss	可能，TLB缺失但页表命中，信息在主存，但可能不在Cache
miss	miss	miss	可能，TLB缺失页表缺失，信息不在主存，一定也不在Cache
hit	miss	miss	不可能，页表缺失，信息不在主存，TLB中一定没有该页表项
hit	miss	hit	同上
miss	miss	hit	不可能，页表缺失，信息不在主存，Cache中一定也无该信息

最好的情况是hit、hit、hit，此时，访问主存几次？　不需要访问主存！

以上组合中，最好的情况是？　hit、hit、miss和miss、hit、hit　访存1次

以上组合中，最坏的情况是？　miss、miss、miss　需访问磁盘、并访存至少2次

介于最坏和最好之间的是？　miss、hit、miss　不需访问磁盘、但访存至少2次

4.4 虚拟存储器

- 虚拟存储器概述
- 页式虚拟存储器及页表管理
- 虚拟存储器地址映射与变换
- **举例**

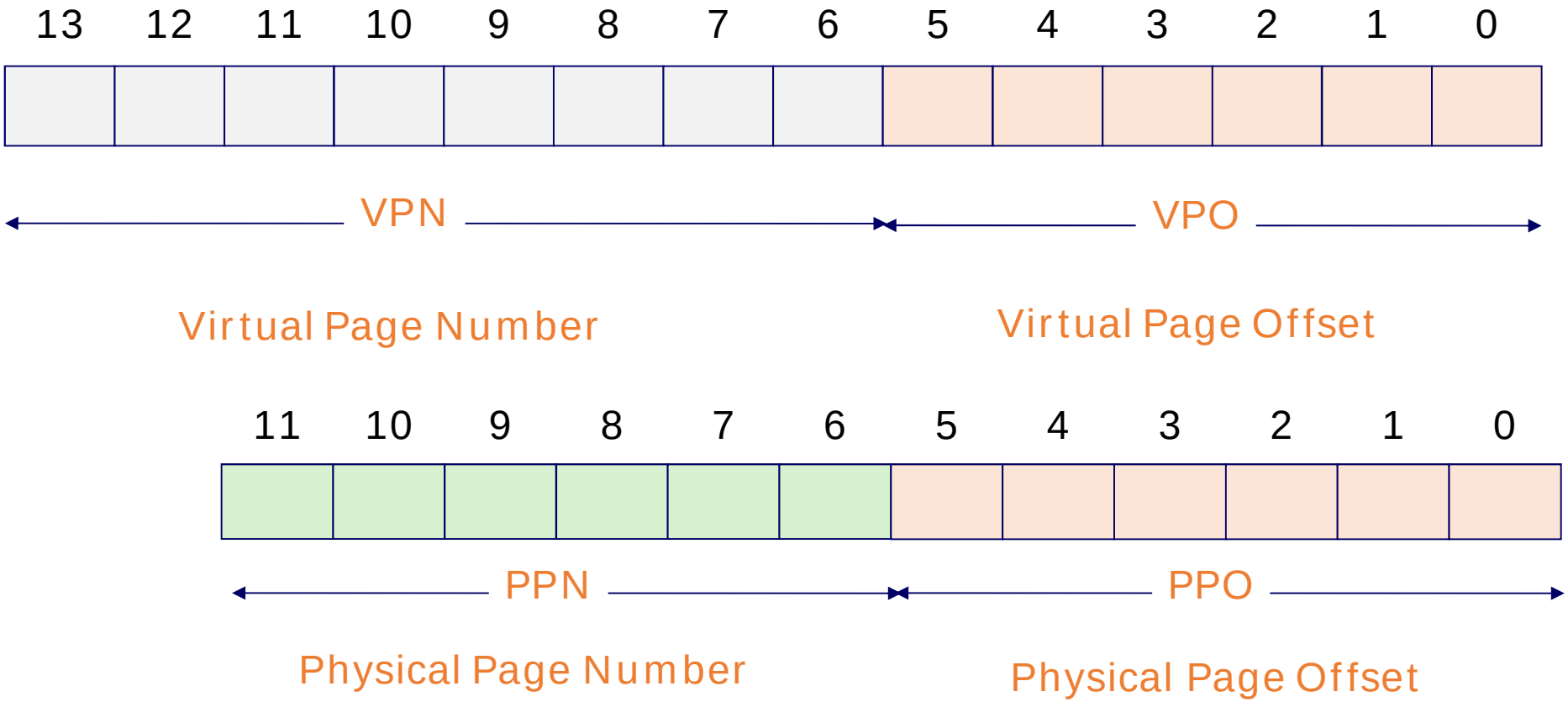
缩写的含义

- 基本参数（按字节编址）
  - $N = 2^n$: 虚拟地址空间大小
  - $M = 2^m$: 物理地址空间大小
  - $P = 2^p$: 页大小
- 虚拟地址 (VA)中的各字段
  - TLBI: TLB index（TLB索引）
  - TLBT: TLB tag（TLB标记）
  - VPO: Virtual page offset （页内偏移地址）
  - VPN: Virtual page number （虚拟页号）
- 物理地址(PA)中的各字段
  - PPO: Physical page offset (页内偏移地址)
  - PPN: Physical page number（物理页号）
  - CO: Byte offset within cache line（块内偏移地址）
  - CI: Cache index（cache索引）
  - CT: Cache tag（cache标记）

一个简化的存储系统举例

- 假定以下参数，则虚拟地址和物理地址如何划分？共多少页表项？
  - 14-bit virtual addresses（虚拟地址14位）
  - 12-bit physical address（物理地址12位）
  - Page size = 64 bytes（页大小64B）

页表项数应为：
$2^{14-6}=256$



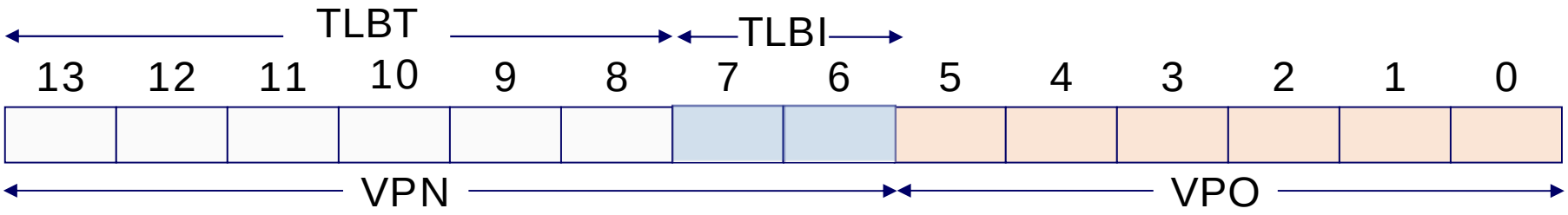
一个简化的存储系统举例（续）

假定部分页表项内容（十六进制表示）如右：

VPN	PPN	Valid
000	28	1
001	–	0
002	33	1
003	02	1
004	–	0
005	16	1
006	–	0
007	–	0

VPN	PPN	Valid
028	13	1
029	17	1
02A	09	1
02B	–	0
02C	–	0
02D	2D	1
02E	11	1
02F	0D	1

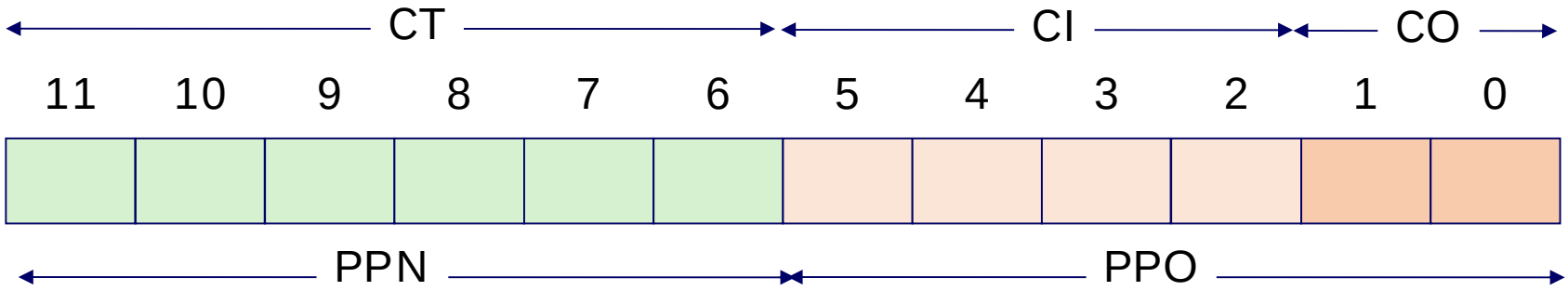
假定TLB如下：16 个TLB项，4路组相联，则TLBT和TLBI各占几位?



Set	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid	Tag	PPN	Valid
0	03	–	0	09	0D	1	00	–	0	07	02	1
1	03	2D	1	02	–	0	04	–	0	0A	–	0
2	02	–	0	08	–	0	06	–	0	03	–	0
3	07	–	0	03	0D	1	0A	34	1	02	–	0

一个简化的存储系统举例（续）

假定Cache的参数和内容（十六进制）如下：16 行，主存块大小为4B，直接映射，则主存地址如何划分?

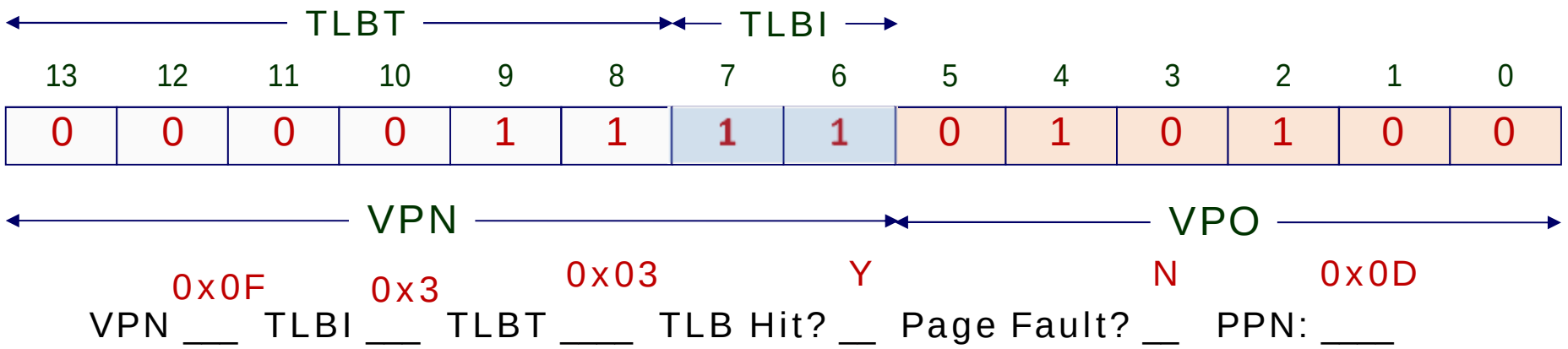


Idx	*Tag*	*V*	*B0*	*B1*	*B2*	*B3*
0	19	1	99	11	23	11
1	15	0	–	–	–	–
2	1B	1	00	02	04	08
3	36	0	–	–	–	–
4	32	1	43	6D	8F	09
5	0D	1	36	72	F0	1D
6	31	0	–	–	–	–
7	16	1	11	C2	DF	03

Idx	*Tag*	*V*	*B0*	*B1*	*B2*	*B3*
8	24	1	3A	00	51	89
9	2D	0	–	–	–	–
A	2D	1	93	15	DA	3B
B	0B	0	–	–	–	–
C	12	0	–	–	–	–
D	16	1	04	96	34	15
E	13	1	83	77	1B	D3
F	14	0	–	–	–	–

一个简化的存储系统举例（续）

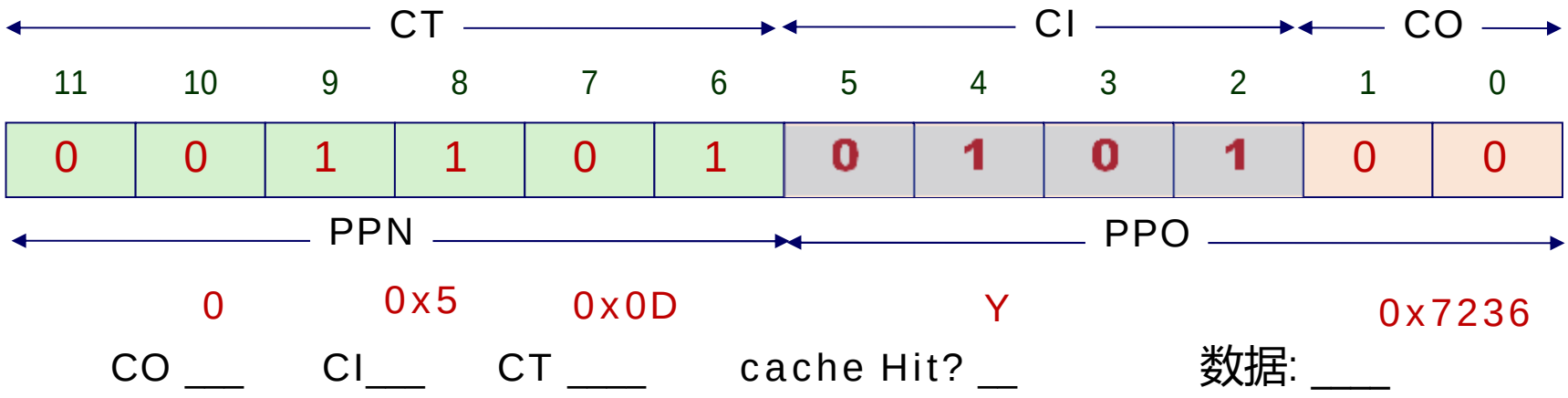
假设该存储系统所在计算机采用小端方式， CPU执行某指令过程中要求访问一个16位数据，给出的逻辑地址为0x03D4，说明访存过程。



物理地址为

问题：逻辑地址为0x0A7A、0x0507时的访存过程如何?

TLB缺失/cache缺失、TLB缺失/缺页



4.4 辅助存储器

一、概述

1. 特点　不直接与 CPU 交换信息

2. 磁表面存储器的技术指标

(1) 记录密度　　道密度 D_t　　位密度 D_b

(2) 存储容量　　$C = n \times k \times s$ (*盘面，磁道，每磁道数*)

(3) 平均寻址时间　寻道时间 + 等待时间

辅存的速度 { 寻址时间；磁头读写时间

(4) 数据传输率　　$D_r = D_b \times V$ （*记录密度，运动速度*）

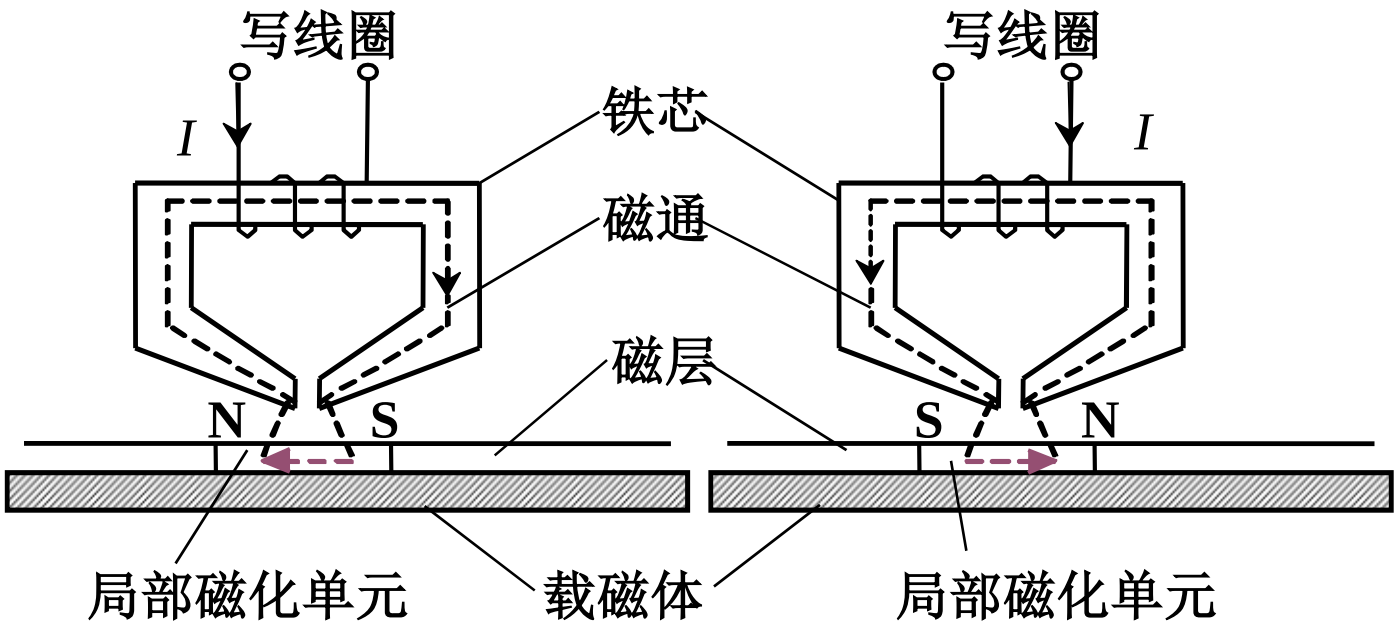
(5) 误码率　　出错信息位数与读出信息的总位数之比

二、磁记录原理和记录方式

4.4

1. 磁记录原理

写

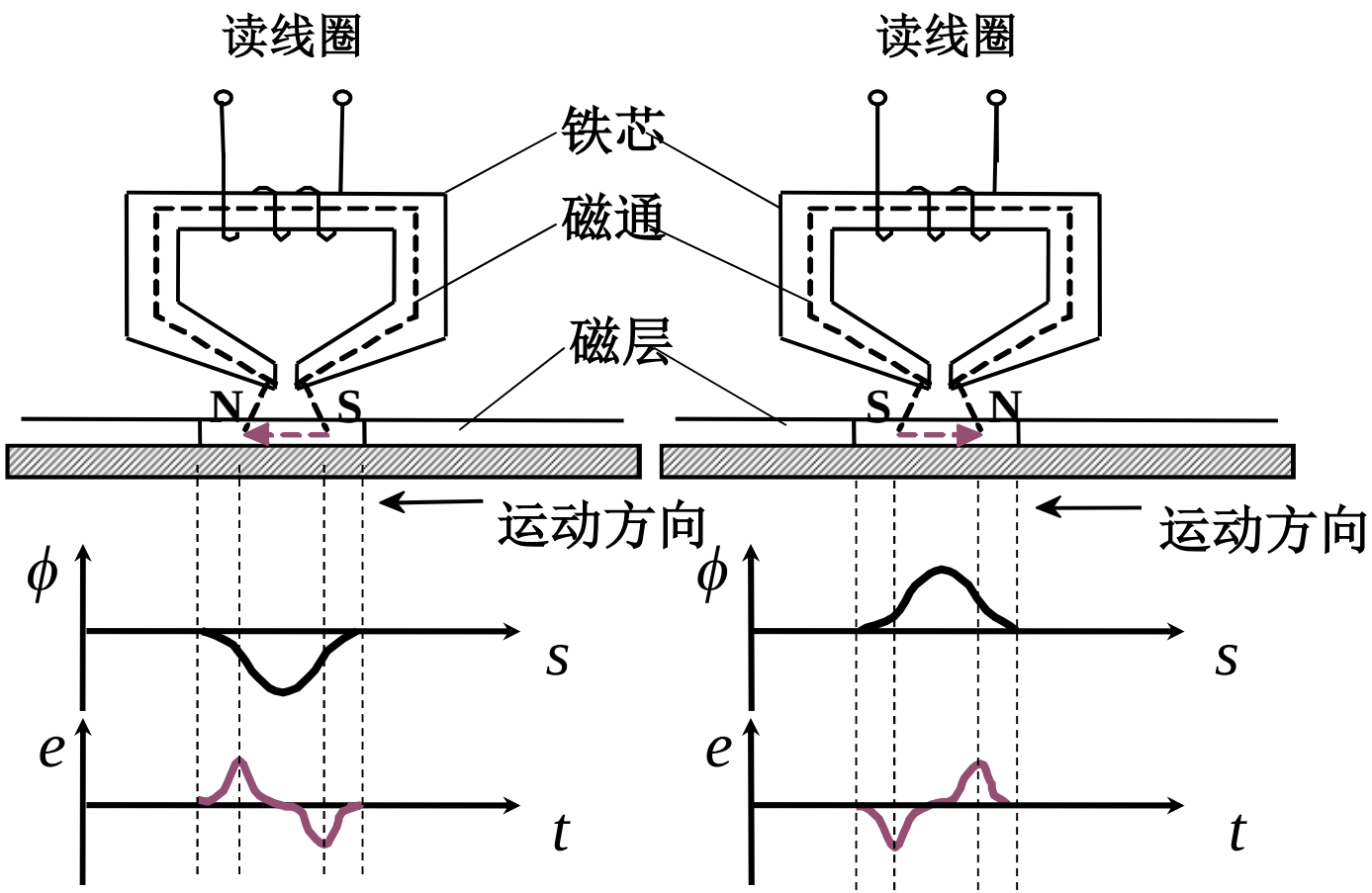


写入“0”

写入“1”

1. 磁记录原理

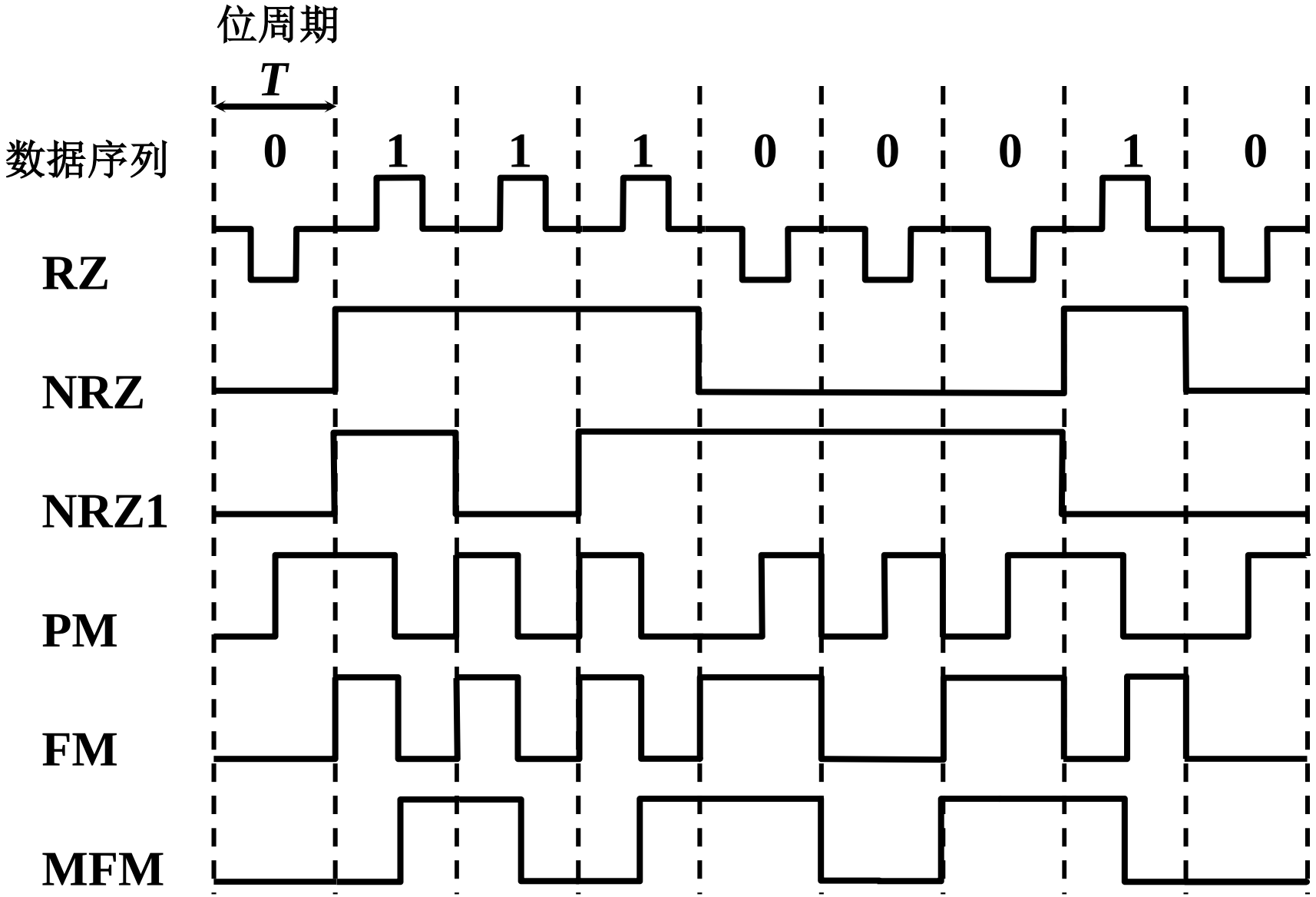
读



读出“0”

读出 “1”

2. 磁表面存储器的记录方式



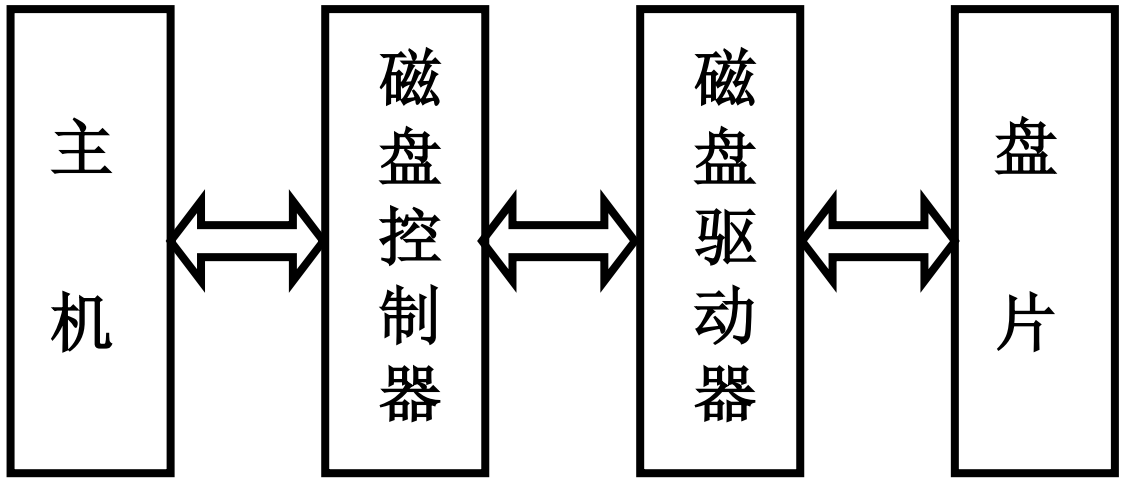
三、硬磁盘存储器 4.4

1. 硬磁盘存储器的类型

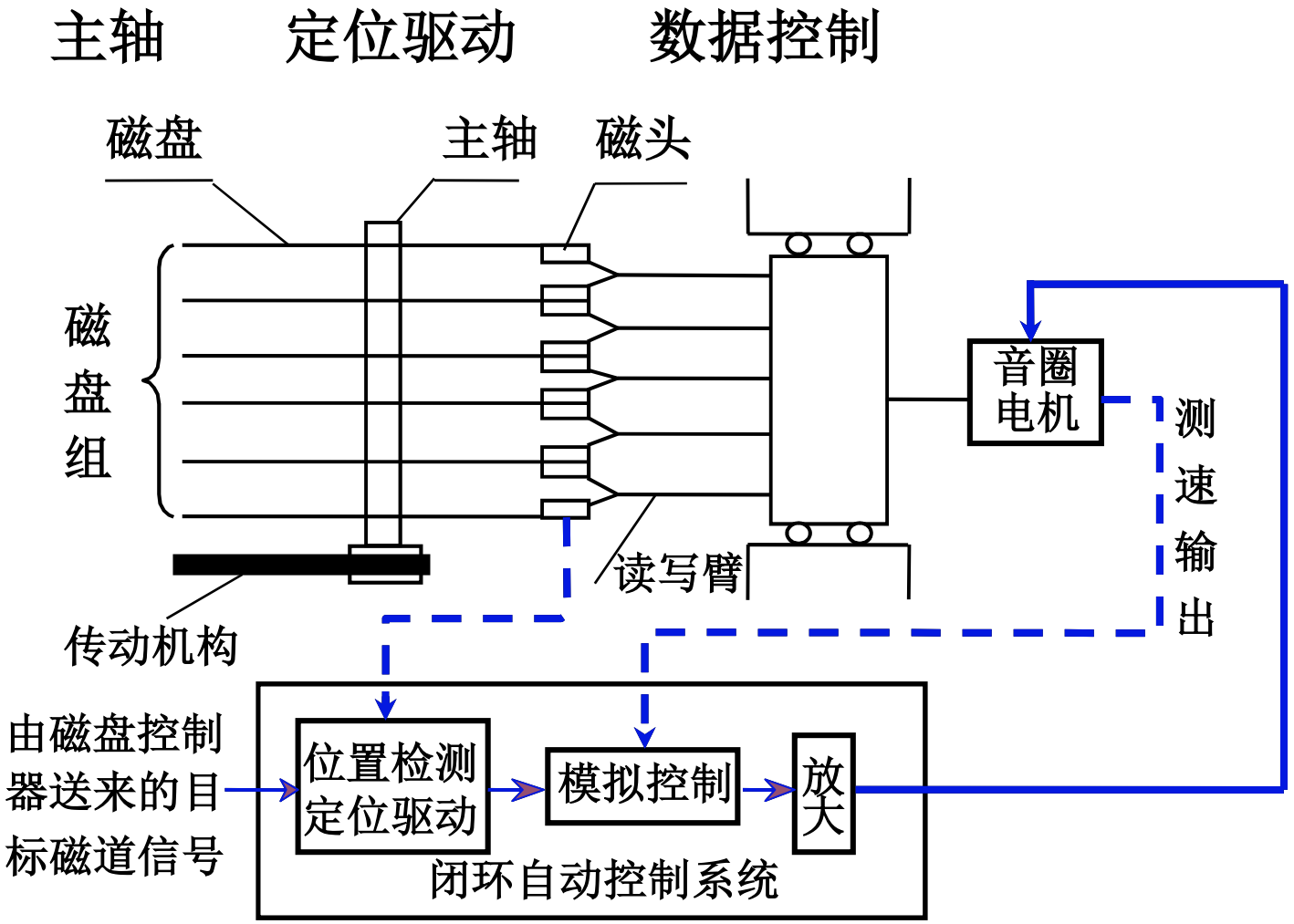
(1) 固定磁头和移动磁头

(2) 可换盘和固定盘

2. 硬磁盘存储器结构



(1) 磁盘驱动器


主轴
定位驱动
数据控制
磁盘
主轴
磁头
磁盘组
音圈电机
测速输出
读写臂
传动机构
由磁盘控制器送来的目标磁道信号
位置检测定位驱动
模拟控制
放大
闭环自动控制系统

(2) 磁盘控制器

- 接收主机发来的命令，转换成磁盘驱动器的控制命令
- 实现主机和驱动器之间的数据格式转换
- 控制磁盘驱动器读写

磁盘控制器 是

主机与磁盘驱动器之间的 接口 { 对主机 通过总线
对硬盘（设备）

(3) 盘片

由硬质铝合金材料制成

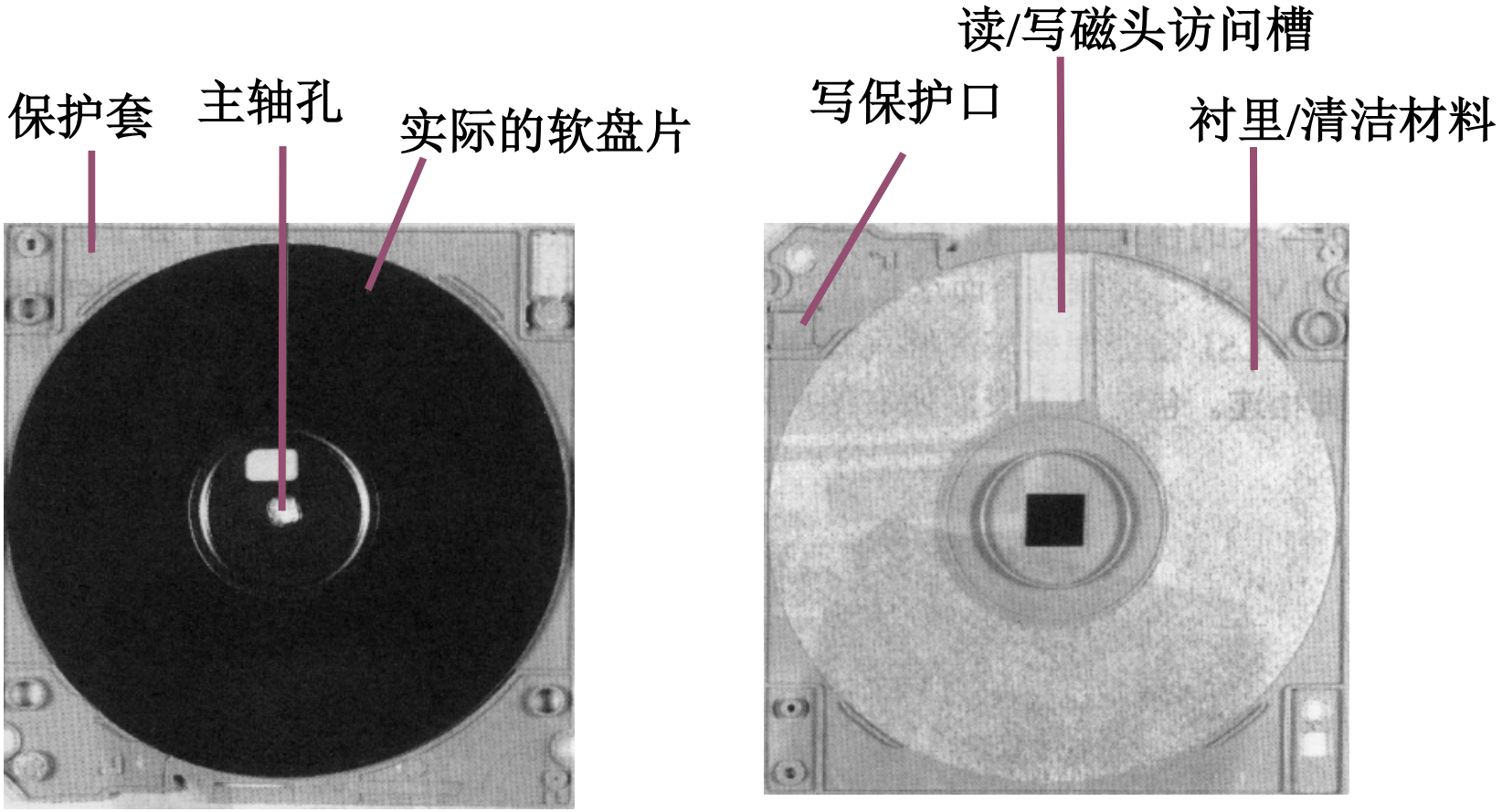
四、软磁盘存储器

1. 概述

	硬盘	软盘
速度	高	低
磁头	固定、活动	活动
	浮动	接触盘片
盘片	固定盘、盘组 大部分不可换	可换盘片
价格	高	低
环境	苛刻	

2. 软盘片

由聚酯薄膜制成



五、光盘存储器

1. 概述

采用光存储技术　　利用激光写入和读出

- 第一代光存储技术　　采用非磁性介质　　不可擦写
- 第二代光存储技术　　采用磁性介质　　可擦写

2. 光盘的存储原理

只读型和只写一次型　热作用（物理或化学变化）

可擦写光盘　　热磁效应

第 5 章　输入输出系统

5.1 概述

5.2 外部设备

5.3 I/O接口

5.4 程序查询方式

5.5 程序中断方式

5.6 DMA方式

5.1 概 述

一、输入输出系统的发展概况

1. 早期

分散连接

CPU 和 I/O设备 串行 工作　程序查询方式

2. 接口模块和 DMA 阶段

总线连接

CPU 和 I/O设备 并行 工作 { 中断方式 / DMA 方式

3. 具有通道结构的阶段

4. 具有 I/O 处理机的阶段

二、输入输出系统的组成 5.1

1. I/O 软件

(1) I/O 指令　　CPU 指令的一部分

操作码	命令码	设备码

(2) 通道指令　　通道自身的指令

指出数组的首地址、传送字数、操作命令

如 IBM/370 通道指令为 64 位

2. I/O 硬件

设备　　I/O 接口

设备　　设备控制器　　通道

三、I/O 设备与主机的联系方式 5.1

1. I/O 设备编址方式

(1) 统一编址　　用取数、存数指令

(2) 不统一编址　　有专门的 I/O 指令

2. 设备选址

用设备选择电路识别是否被选中

3. 传送方式

(1) 串行

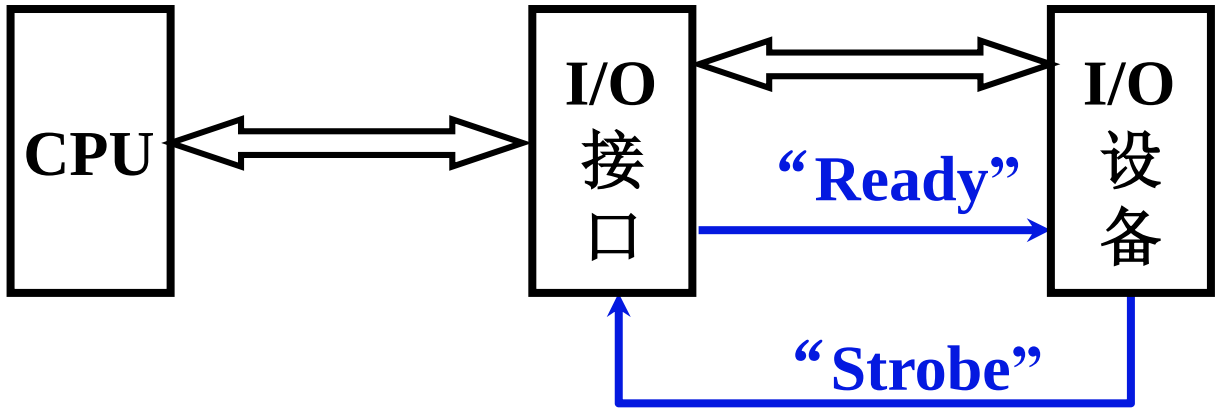
(2) 并行

4. 联络方式

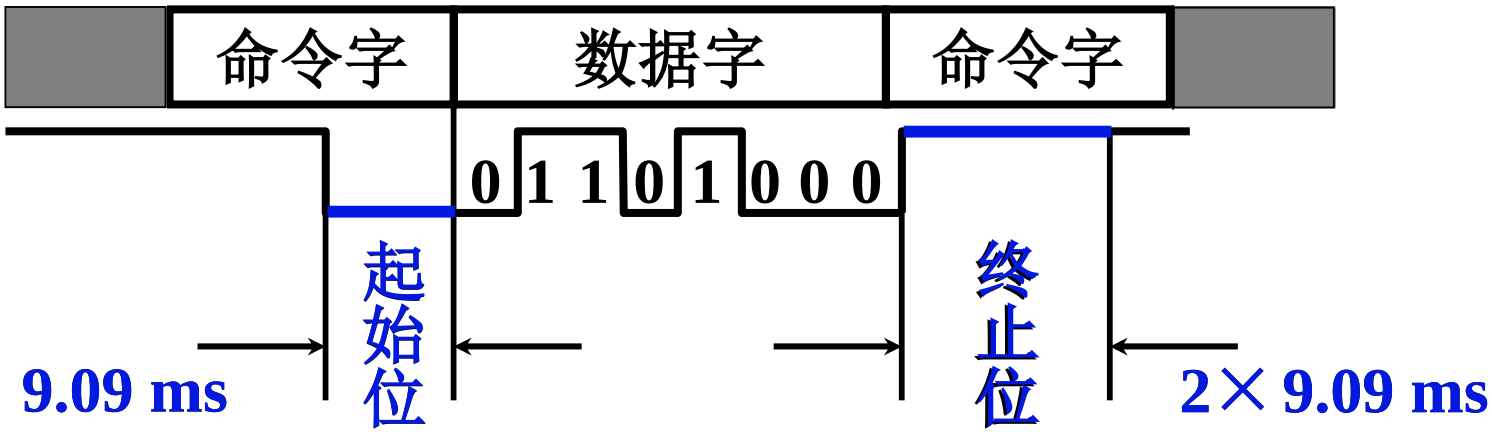
(1) 立即响应

(2) 异步工作采用应答信号

并行



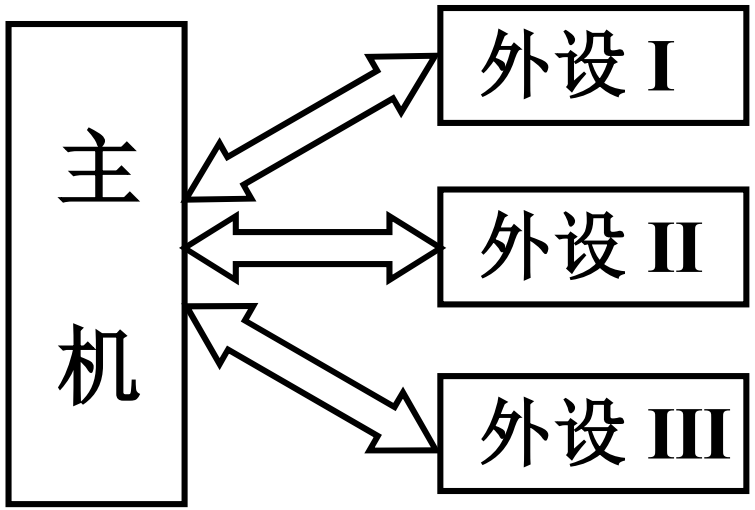
串行



(3) 同步工作采用同步时标

5. I/O 设备与主机的连接方式

(1) 辐射式连接



每台设备都配有一套

控制线路和一组信号线

不便于增删设备

(2) 总线连接

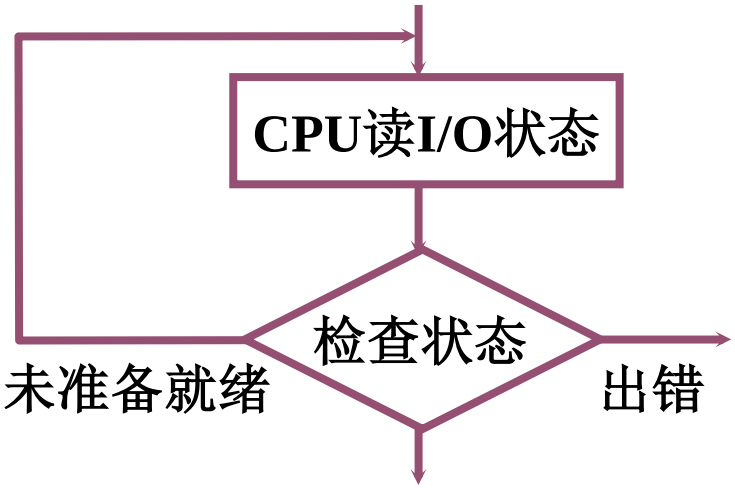
便于增删设备

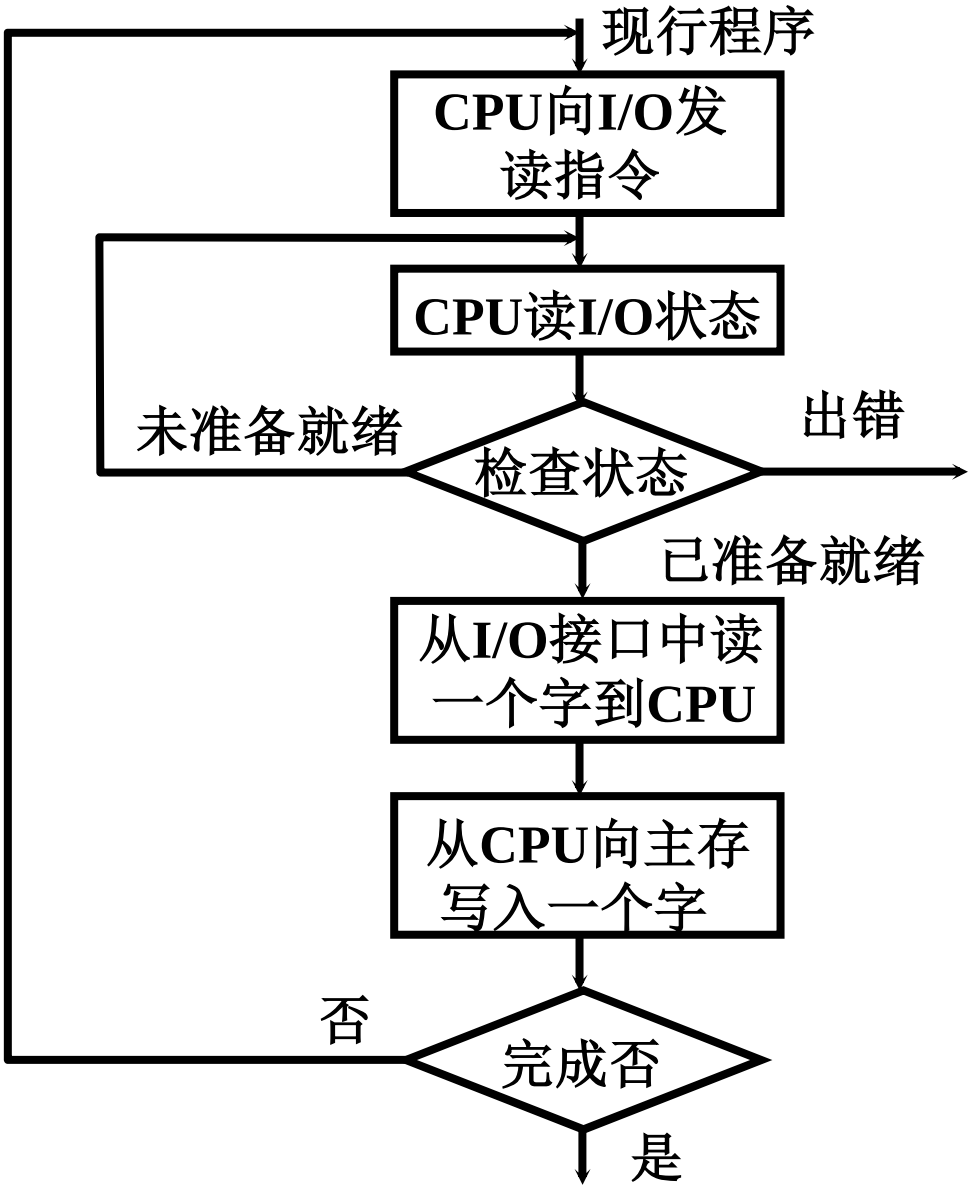
四、I/O设备与主机信息传送的控制方式 5.1

1. 程序查询方式

CPU 和 I/O 串行工作

踏步等待

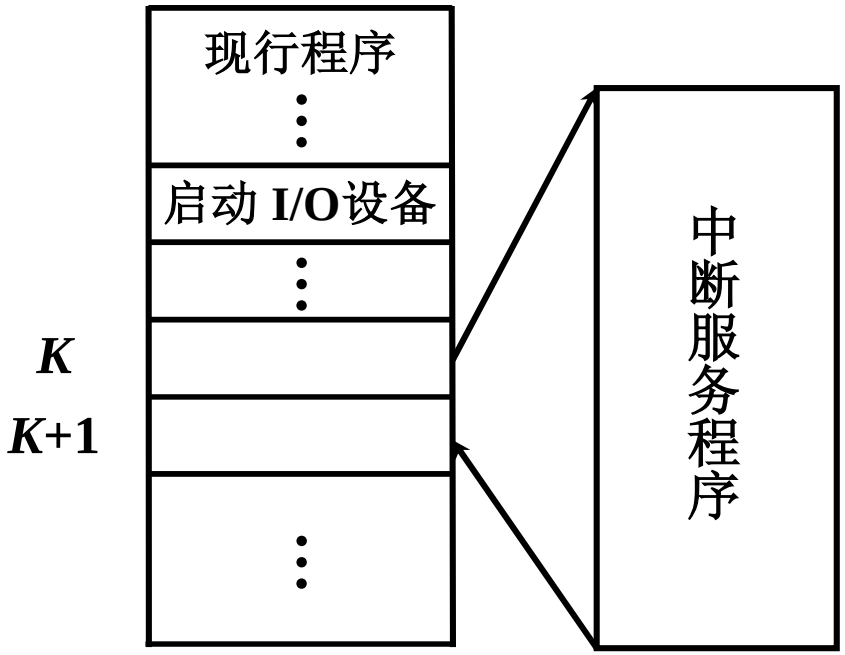




2. 程序中断方式

I/O 工作 {
- 自身准备　　CPU 不查询
- 与主机交换信息　CPU 暂停现行程序

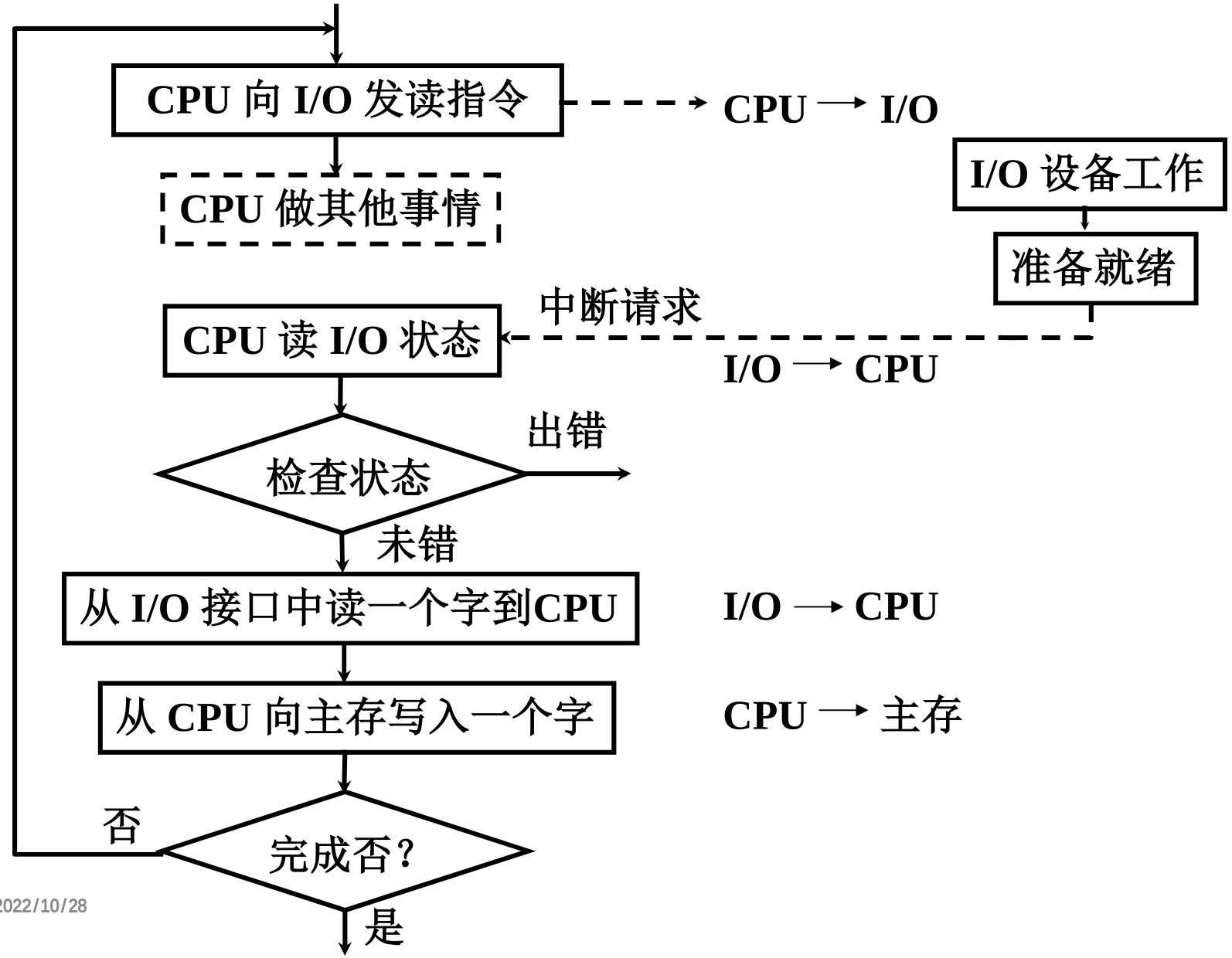
CPU 和 I/O 并行工作



没有踏步等待现象

中断现行程序

程序中断方式流程


CPU 向 I/O 发读指令
CPU → I/O
I/O 设备工作
CPU 做其他事情
准备就绪
中断请求
CPU 读 I/O 状态
I/O → CPU
出错
检查状态
未错
从 I/O 接口中读一个字到CPU
I/O → CPU
从 CPU 向主存写入一个字
CPU → 主存
否
完成否？
是

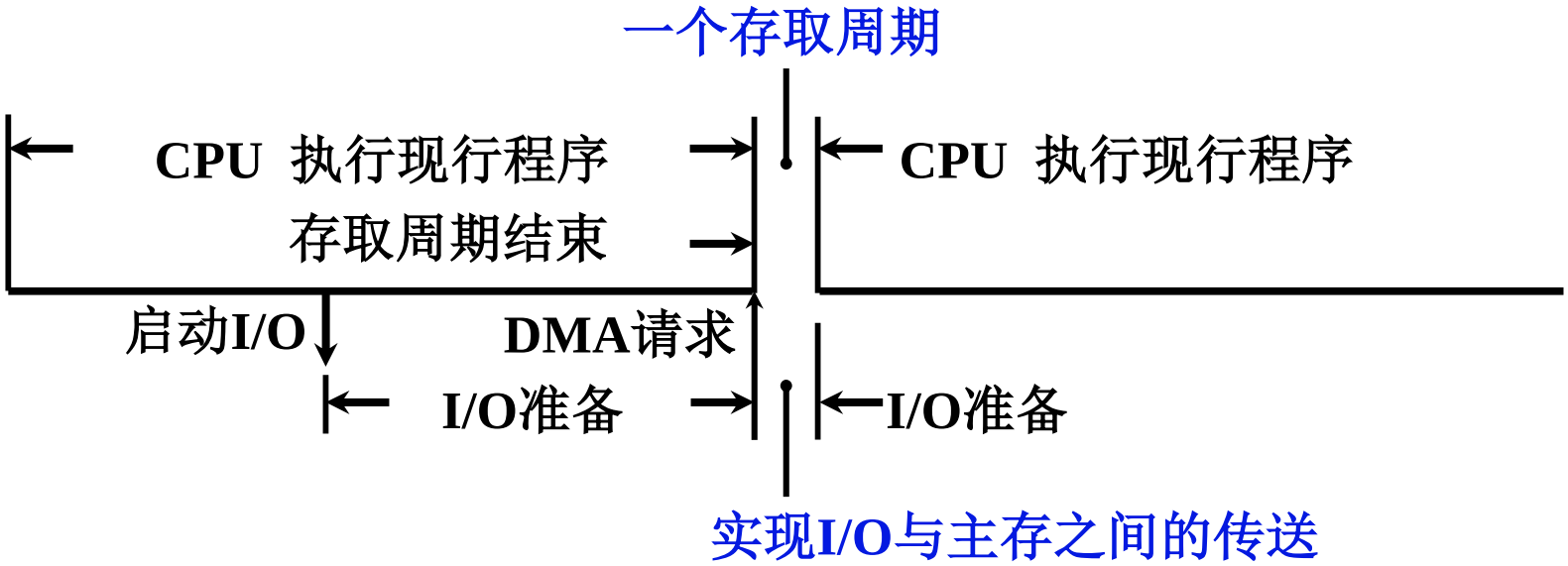
3. DMA 方式

主存和 I/O 之间有一条直接数据通道

不中断现行程序

周期挪用（周期窃取）

CPU 和 I/O 并行工作



三种方式的 CPU 工作效率比较