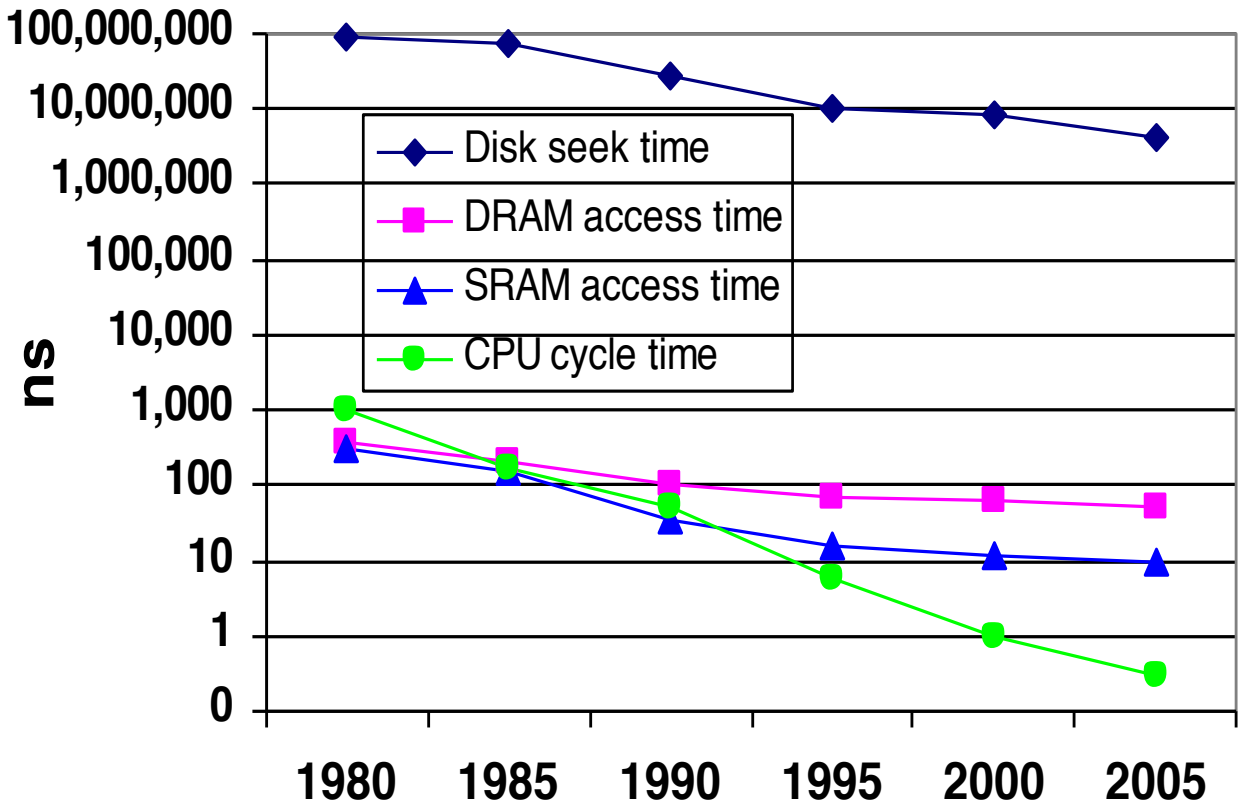
# 5.4 存储层次结构

刘 芳 副教授
国防科学技术大学计算机学院

# 5.4.1 并行主存系统

# 5.4.1 并行主存系统

## 存储器与CPU速度差距愈来愈大!!!




CPU工作速度很快，主存速度比较慢(差1～2个数量级)，CPU访问主存时，往往需要等待

**解决内存访问速度慢的措施：**

- 提高主存芯片本身的速度
- 在主存和CPU之间加入Cache
- 采用并行结构技术

# 5.4.1 并行主存系统

## 主存性能一直是计算机的重要性能指标之一，决定着计算机系统的整体性能

- 计算机系统期望主存：速度快、容量大、可靠性高、成本低
- 但从计算机实现技术来看，存在两个现实：
  - 主存速度的提高总是远远落后于CPU速度的增长
  - 主存容量的扩大总是远远满足不了软件的日益膨胀

结论：单从存储技术本身提高主存速度和容量，已经很难满足计算机系统的实际需求

# 5.4.1 并行主存系统

从存储系统结构上想办法？

1、层次式存储系统

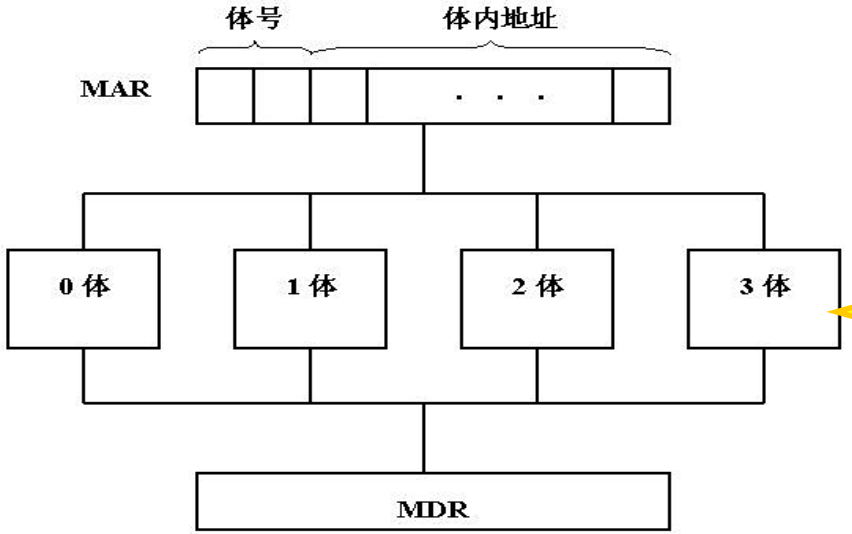
Cache —— 主存层次

主存 —— 辅存层次（虚拟存储技术，扩大用户编程的逻辑空间）

2、多体(多模块)并行交叉编址的主存储器

## 多体存储器

由若干个小存储体组成，共用MAR和MDR，同一套读/写控制电路、地址缓存、地址译码，**不能提高数据访问速度**



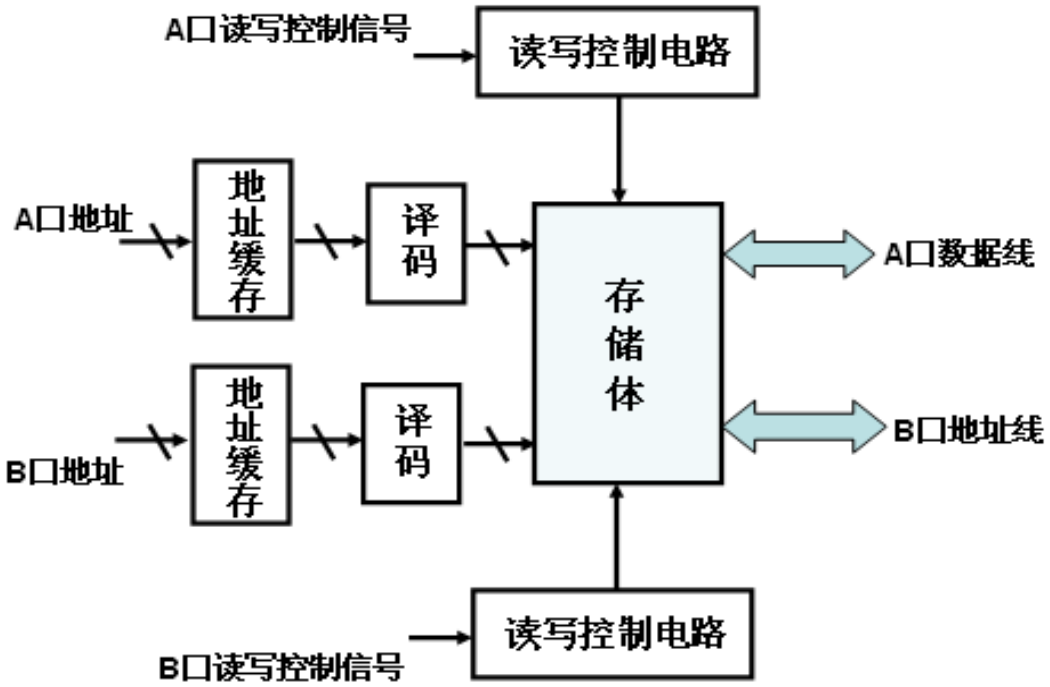

多体存储器结构

不能提高
访问速度！

## 多体存储器

- 一个存储器中提供两套独立工作的读/写控制电路、地址缓存、地址译码，两个读写端口，根据地址线和数据线能同时进行两个数据的读/写

- 通常作为双口RAM或指令预取部件



## 多模块存储器(多体交叉)

- 包含多个小存储体/模块，每个体有自己的MAR、MDR和读写电路
- 多个存储模块可独立并行工作
- 能够提高数据访问速度

**编址方式**

连续编址（高位交叉访问）

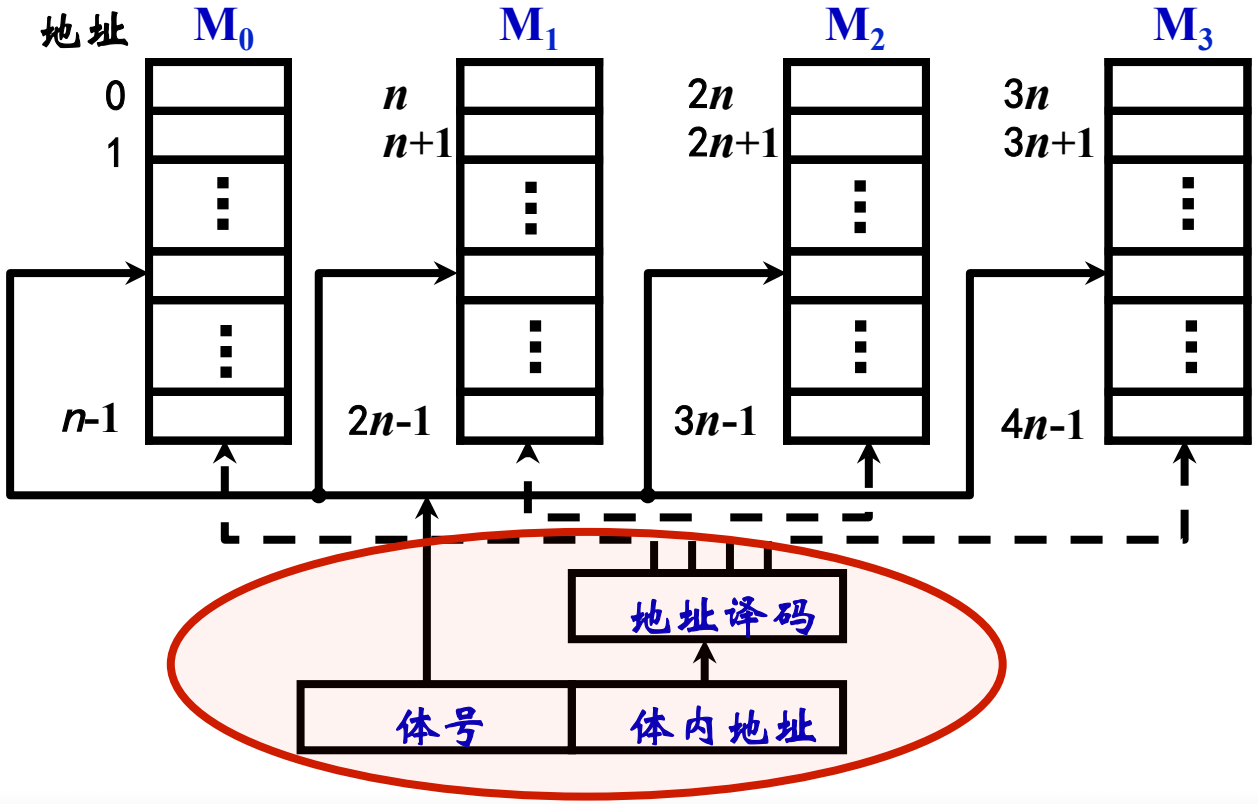
交叉编址（低位交叉访问）

## 连续编址方式 —— 高位交叉访问存储器

**实现方法：**用主存地址码的**高位区分**存**储体号，**低位表示模块内的地址

扩大了存储器容量连续字在一个模块中；

各个体并行工作



## 连续编址方式 —— 高位交叉访问存储器

存储地址连续的数据落在同一存储体内，容易发生访存冲突，并行存取的可能性很小

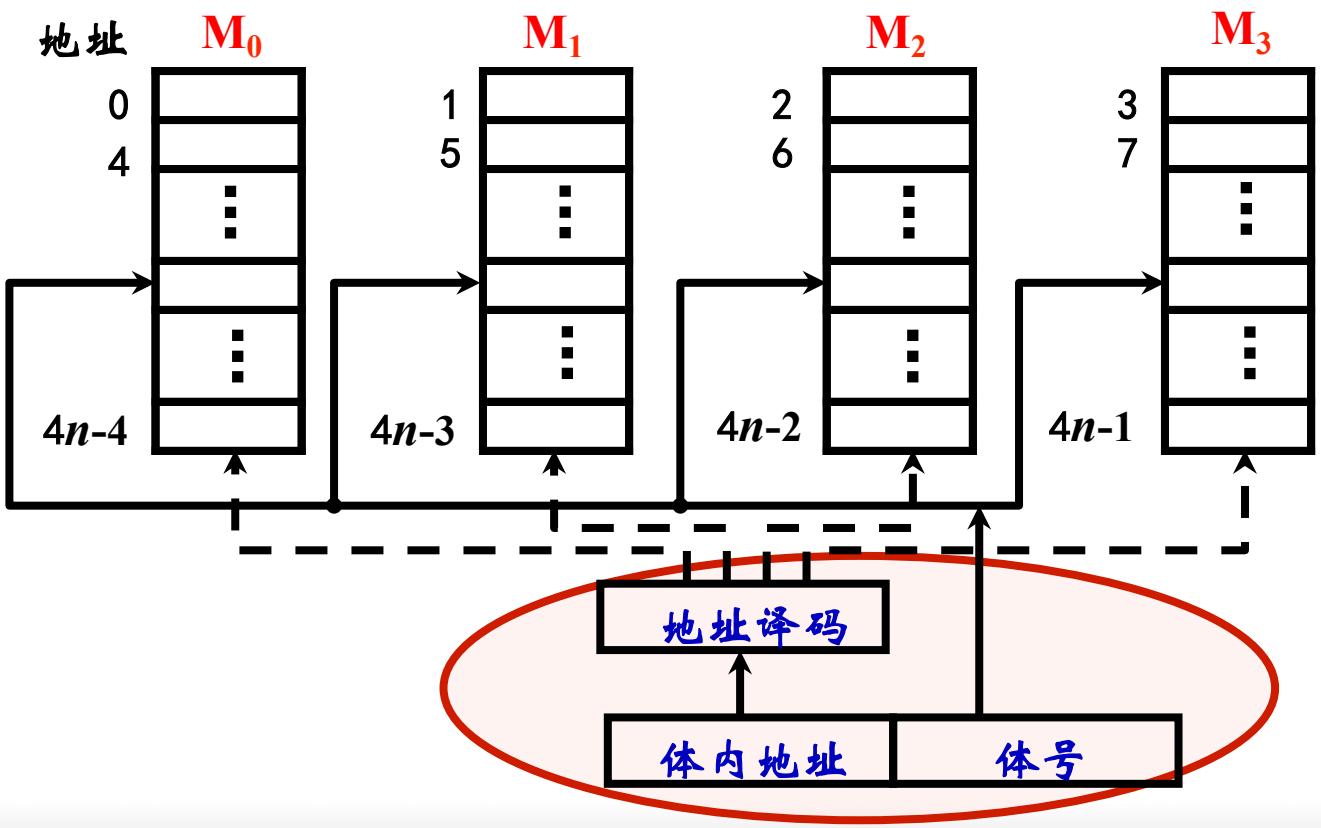
访存冲突，就是同时有两个或两个以上访存地址指向同一存储体，不能同时进行访存

用于非共享主存(即每个处理机仅享用统一编址主存的部分连续地址空间)和专用Cache的多处理机系统中

## 交叉编址方式 —— 低位交叉访问存储器

**实现方法：**用主存地址码的**低位区分**存**储体号，**高位表示模块内的地址

连续字分布在多个体中；**各个体轮流编址、并行工作**提高存储器访问速度



## 交叉编址方式 —— 低位交叉访问存储器

存储地址在同一存储体中不连续，以存储体个数（如：m）为模交叉编址

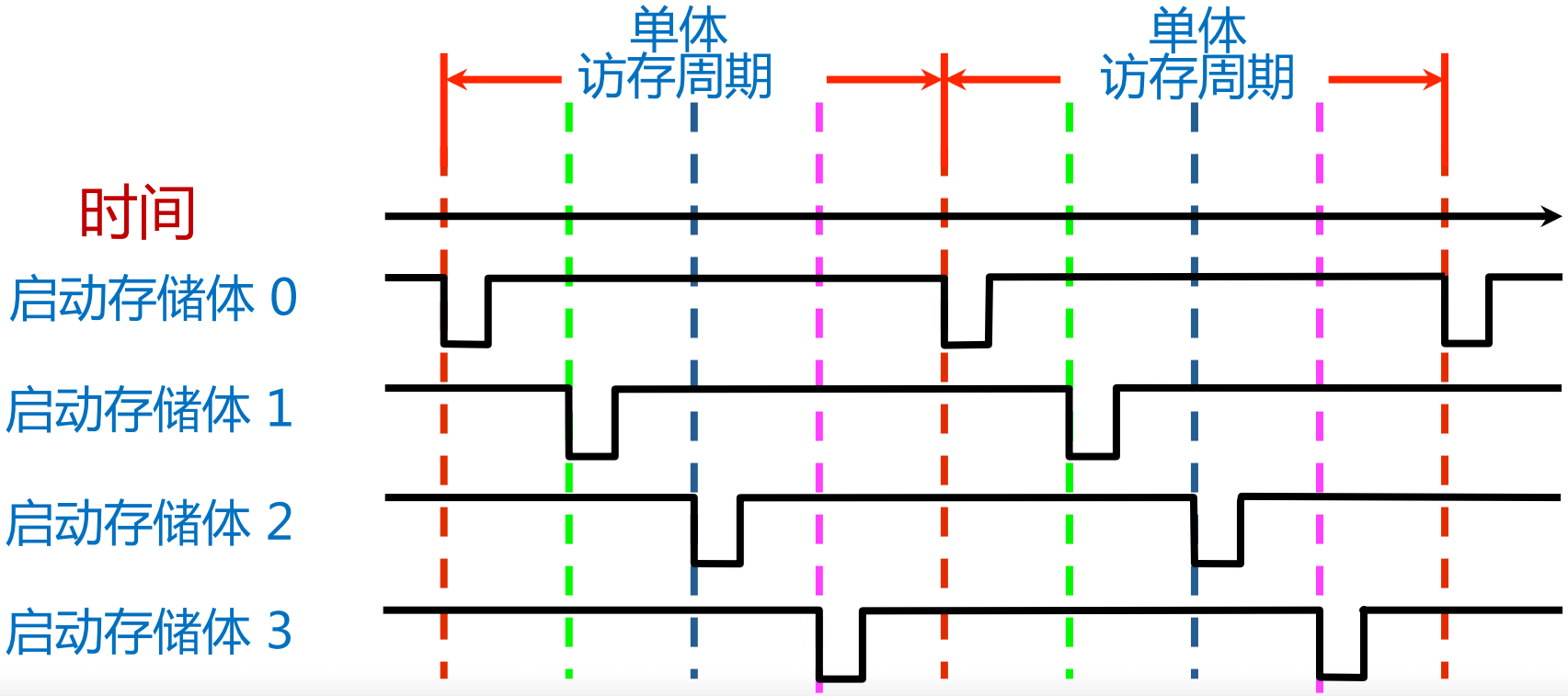
连续的程序或数据将交叉存放在m个存储体中，可实现以m为模的交叉并行存取，访存冲突的概率很小

为充分发挥并行性，多数计算机都采用低位交叉编址方式

## 交叉编址方式 —— 低位交叉访问存储器

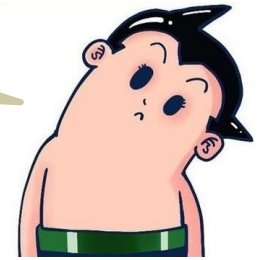
m个存储体分时启动：一般采用流水线方式工作。每存储体的启动间隔为：$t = \lfloor T_m / m \rfloor$

在不改变每个体的存取周期前提下，提高存储器的带宽



# 5.4.1 并行主存系统

什么条件下，多体低位交叉编址可增加存储带宽？



CPU访问主存的存储单元地址分布在不同存储体内

## 避免存储体访问冲突

- 软件通过编译程序做循环变换，可避免访问相同的存储体
- 硬件采用质数个存储体的低位交叉并行主存系统：一种无访问冲突的并行主存结构（实际带宽接近于最大带宽）
  - 余数定律证明
  - YH-1巨型机采用的是31个存储体构成的无冲突并行主存结构

## 支持Cache的存储器系统性能

指令执行时发生Cache失效/缺失/失靶，需到主存取数据或指令

在主存和Cache之间传输的单位是块(Block)

问题：何种存储器使Block传输最快(miss penalty最小)？

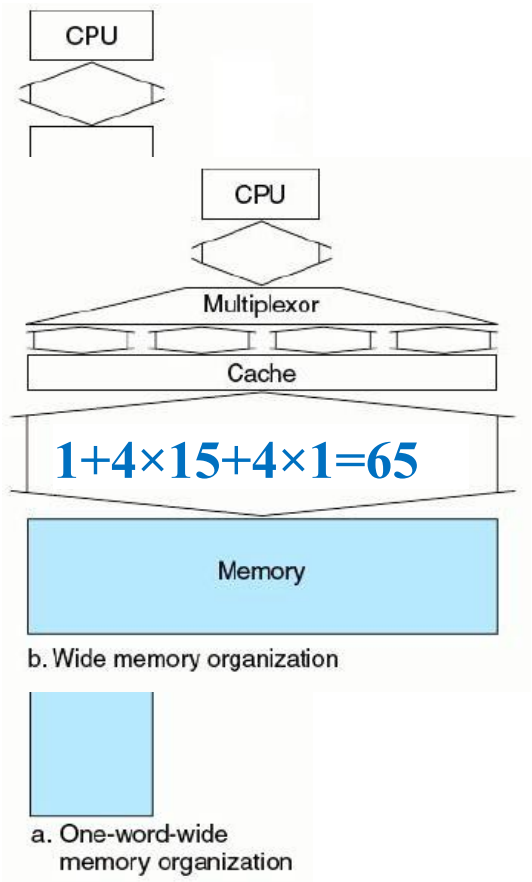
存储器访问过程：

- 发送地址到主存储器：1个总线时钟
- 访问主存的时间：15个总线时钟
- 从总线上传送一个字：1个总线时钟

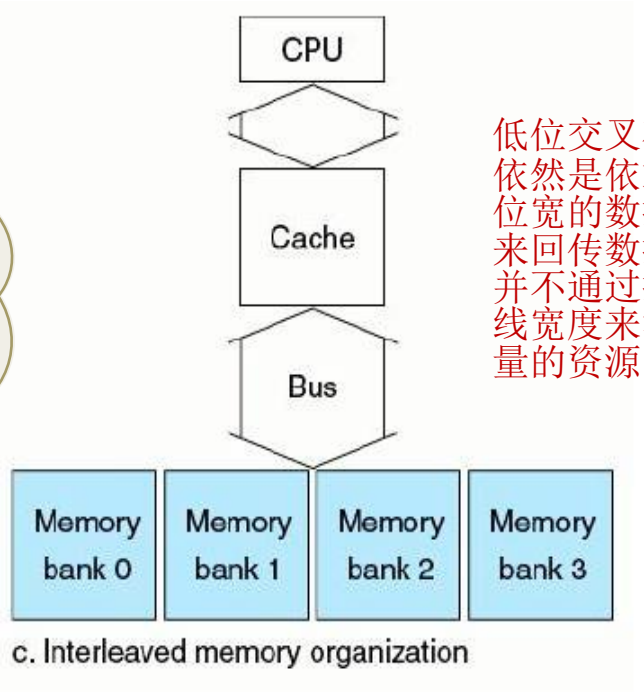
# 5.4.1 并行主存系统

**存储器访问：**

**(1)发送地址：1个；(2)访问主存：15个；(3)传送1个字：1个总线周期**




假定一个Block有4个字，则缺失损失各为多少？




低位交叉存储器依然是依靠原来位宽的数据总线来回传数据（而并不通过扩展总线宽度来消耗大量的资源）

**Four-word: 1+15+4=20**

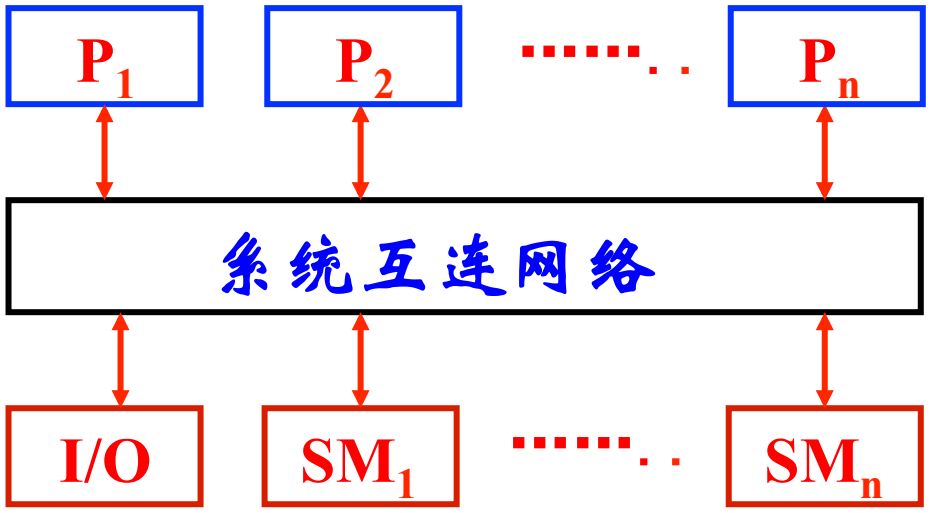
**Interleaved four banks**
**one-word: 1+1×15+4×1=20**

## 并行计算机的访存模型

### 1、UMA模型(Uniform Memory Access)：均匀存储访问模型

**特点**

- 物理存储器被所有处理器均匀共享
  （SM：Shared Memory）
- 访问任何存储字的时间相同
- 每台处理器可带私有高速缓冲
- 外围设备可以某种形式共享

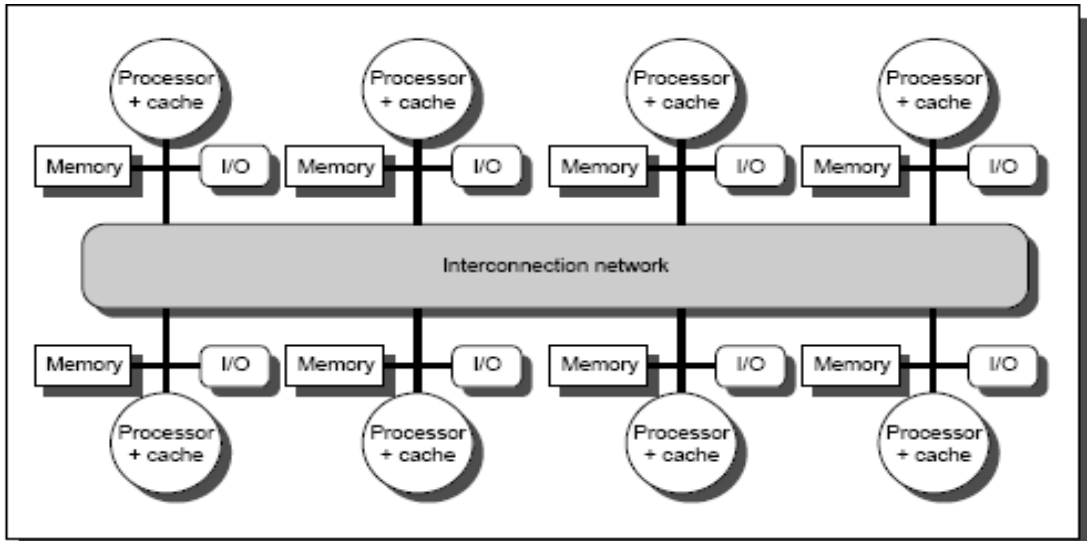


## 并行计算机的访存模型

### 2、NUMA模型(Non-Uniform Memory Access)：非均匀存储访

**特点**

- 被共享的存储器在物理上分布于各个处理器
- **处理器访问不同存储器的时间不一样**
- 每台处理器带私有高速缓冲
- 外围设备可以某种形式共享



# 5.4.2 多处理机的Cache一致性

# 5.4.2 多处理机的Cache一致性

**多核处理器**：在单芯片上的多个处理器，可能会共享主存中的一个公共的物理地址空间

**Cache提供共享数据的迁移(migration)和复制(replication)**

- **迁移**：数据项可以移入本地cache并以透明的方式使用。迁移减少访问远程共享数据项的延迟和对共享存储器带宽的需求
- **复制**：当共享数据被同时读取时，Cache在本地对数据项做了备份。复制减少了访问延迟和读取共享数据时的竞争现象

# 5.4.2 多处理机的Cache一致性

## Cache共享数据带来新的问题 —— Cache一致性问题

- 两个不同处理器所保存的存储器视图是通过各自的cache得到(采用写直达cache)
- 两个处理器可能分别得到两个不同的值

| 时间 | 事件 | $CPU_A$ cache内容 | $CPU_B$ cache内容 | 存储器位置X的内容 |
|---|---|---|---|---|
| 0 | | | | 0 |
| 1 | $CPU_A$读X | 0 | | 0 |
| 2 | $CPU_B$读X | 0 | 0 | 0 |
| 3 | $CPU_A$向X写1 | 1 | 0 | 1 |

多处理机系统中，由于多个处理器异步地相互操作，因此多个高速缓存中的同一副本可能不同

# 5.4.2 多处理机的Cache一致性

**多处理机系统的Cache一致性的解决方法**

**软件方法：**借助编译程序进行分析，使共享数据只存放在主存中，不允许放入高速缓存中

**硬件方法：**硬件协议维护，关键在于跟踪所有共享数据块的状态

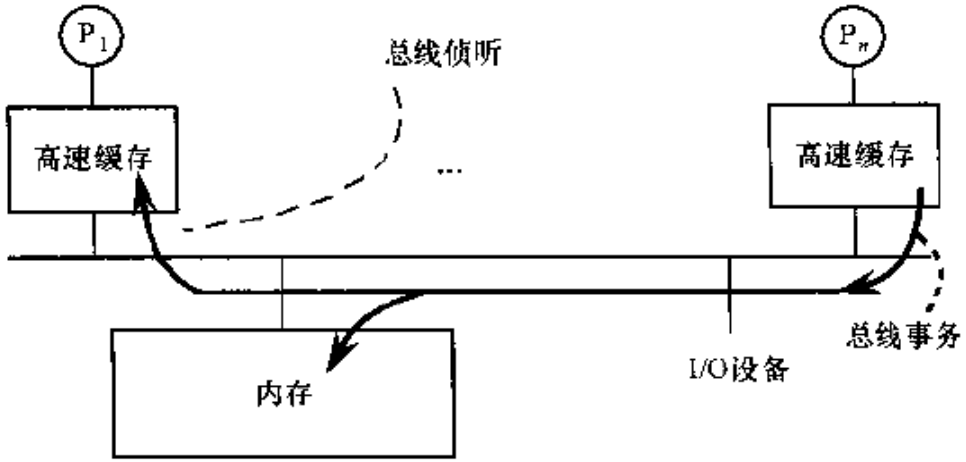
- 侦听协议(snooping)
- 目录协议

# 5.4.2 多处理机的Cache一致性

## 侦听协议

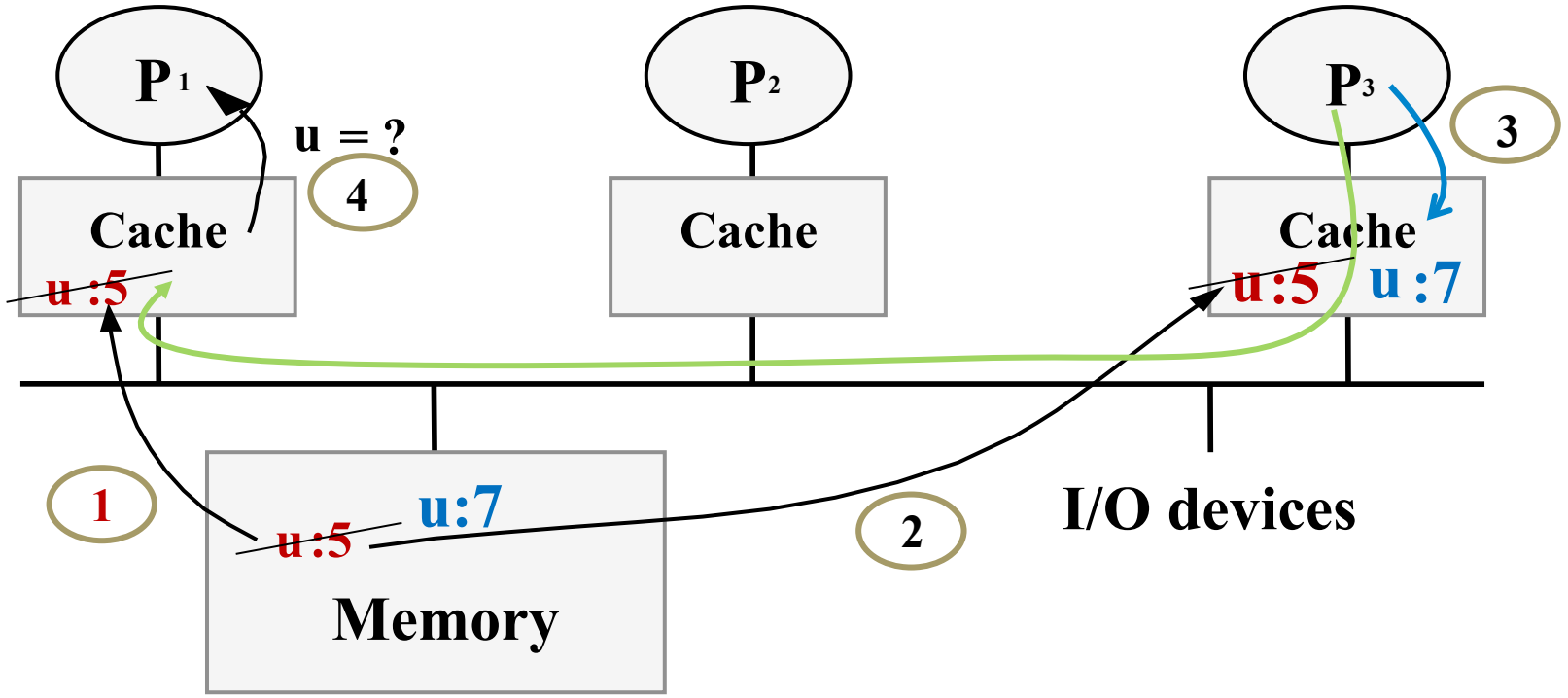
基于总线连接的多处理机系统中，每个处理器的高速缓存设置1个侦听部件，侦听总线上的事务活动

若侦听到主存中有1个单元被其它处理器修改，而在自己Cache中有该单元的副本时，则将自己Cache中的副本置为无效，或把该副本更新，保持与主存中相应单元的内容一致



# 5.4.2 多处理机的Cache一致性

一个基于总线侦听的写失效协议的例子(cache采用写直达机制)



# 5.4.2 多处理机的Cache一致性

## 目录协议(Directory Protocol)

- **在主存**中**设置一个目录表**，表中每一项**记录共享数据**的所有高速缓存行(数据块)的**位置和状态**，包括几个指示器(指示数据块的副本放在哪些处理器的高速缓存中)和指示位(指示是否已有高速缓存更新过此数据块的内容)
- 当一个处理器写入自己的高速缓存时，基于目录表，只需有选择地通知存有该数据块的处理器中的高速缓存，使相应副本被废弃或被更新