计算机组织结构

18 课程小结

任桐炜，吴海军，刘博涵

2021年12月23日

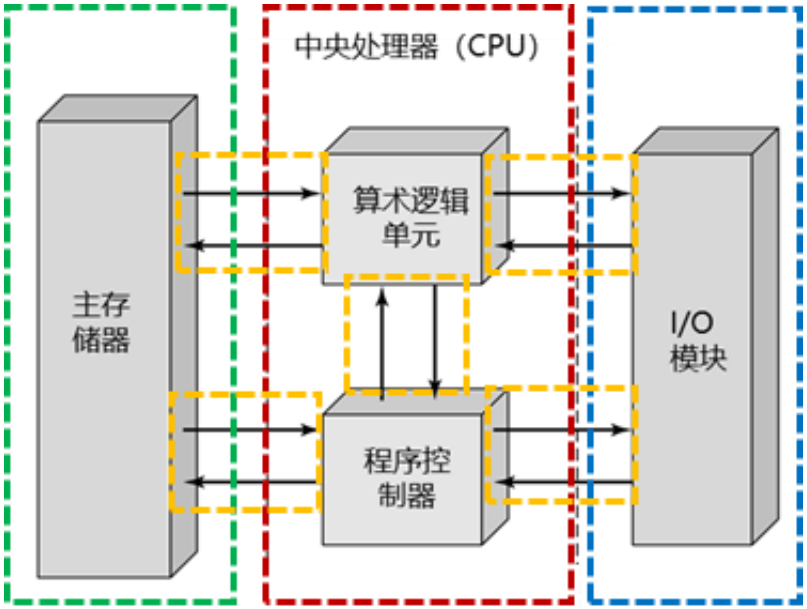
南京大学
NANJING UNIVERSITY

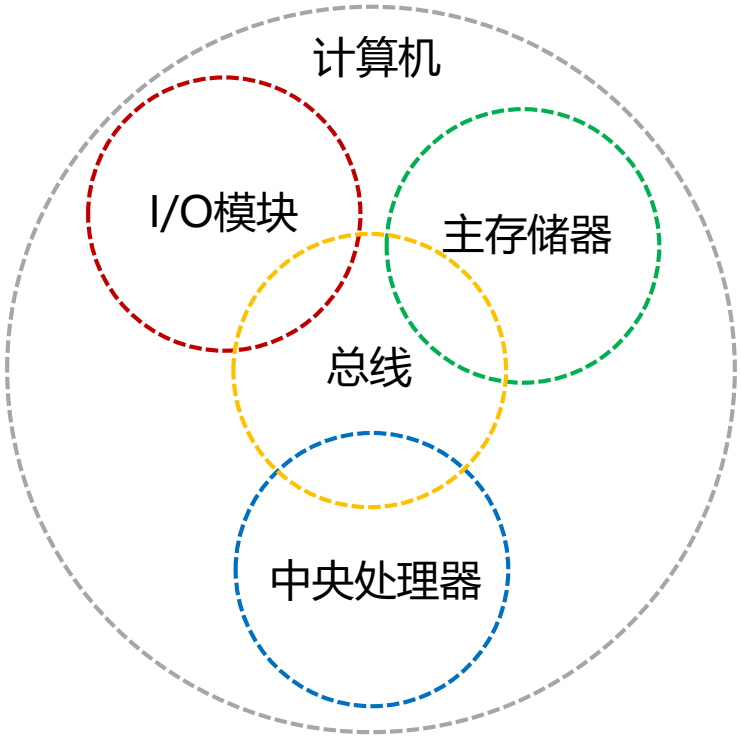
计算机，组织，结构

- **计算机**是指“通用电子数字计算机（general-purpose electronic digital computer）”
- **组织（Organization）**：对编程人员不可见
  - 操作单元及其相互连接
  - 包括：控制信号，存储技术，……
    - 例如：实现乘法是通过硬件单元还是重复加法?
- **结构（Architecture）**：对编程人员可见
  - 直接影响程序逻辑执行的属性
  - 包括：指令集，表示数据类型的位数，.....
    - 例如：是否有乘法指令?

冯·诺依曼结构

- 基本思想：存储程序





摩尔定律

- **摩尔定律** (Gordon Moore, 1965)
  - 单芯片上所能包含的晶体管数量每年翻一番 (1965-1969) / 1970年起减慢为每18个月翻一番
  - 影响
    - 更小的尺寸带来更多灵活性和可能性
    - 由于单个芯片的成本几乎不变，计算机逻辑电路和存储电路的成本显著下降
    - 减小了对电能消耗和冷却的要求
    - 集成电路上的内部连接比焊接更可靠，芯片间的连接更少

计算机性能

- 性能评价标准
  - CPU：速度
  - 存储器：速度，容量
  - I/O：速度，容量
  - ……
- 计算机设计的主要目标是：提高CPU性能
  - 系统时钟：时钟频率，时钟周期
  - 指令执行：CPI
  - 每秒百万条指令（MIPS）
  - 每秒百万条浮点数操作（MFLOPS）
  - 基准程序：算术平均，调和平均

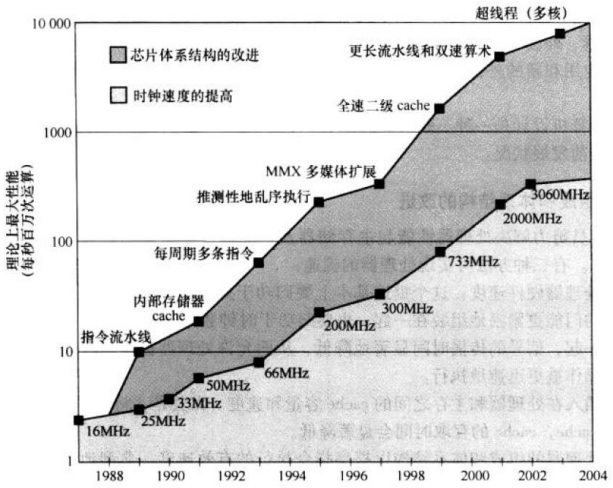


图 2-12　Intel 微处理器性能［GIBB04］

数的表示

- 整数：补码
- 浮点数：IEEE 754

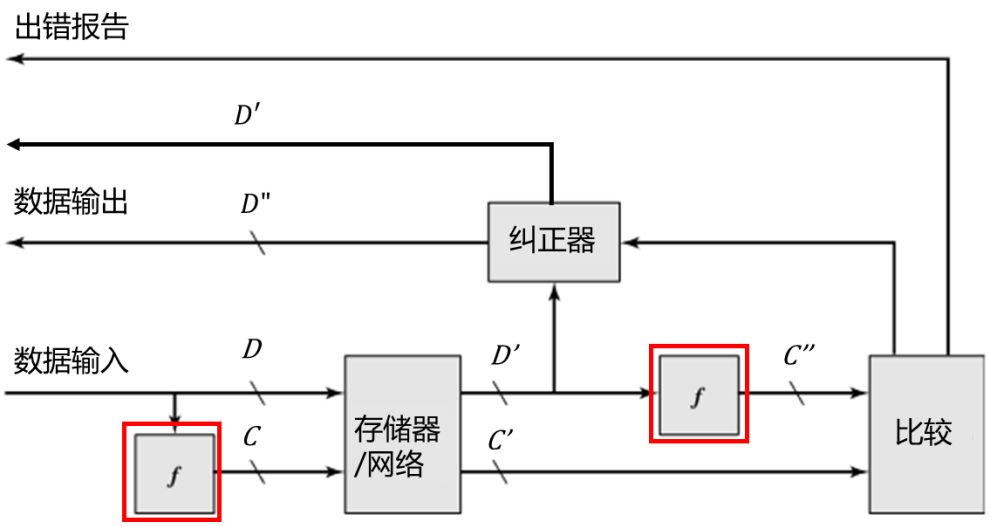
K位的二进制编码至多表示2^k个不同的值

	单精度（32位）				双精度（64位）			
	符号	移码阶值	小数	值	符号	移码阶值	小数	值
正零	0	0	0	0	0	0	0	0
负零	1	0	0	−0	1	0	0	−0
正无穷大	0	255 (all 1s)	0	∞	0	2047 (all 1s)	0	∞
负无穷大	1	255 (all 1s)	0	−∞	1	2047 (all 1s)	0	−∞
静默式非数	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
通知式非数	0 or 1	255 (all 1s)	≠0	NaN	0 or 1	2047 (all 1s)	≠0	NaN
正的规格化非零数	0	$0 < e < 255$	f	$2^{e-127}(1.f)$	0	$0 < e < 2047$	f	$2^{e-1023}(1.f)$
负的规格化非零数	1	$0 < e < 255$	f	$-2^{e-127}(1.f)$	1	$0 < e < 2047$	f	$-2^{e-1023}(1.f)$
正的非规格化数	0	0	$f \neq 0$	$2^{e-126}(0.f)$	0	0	$f \neq 0$	$2^{e-1022}(0.f)$
负的非规格化数	1	0	$f \neq 0$	$-2^{e-126}(0.f)$	1	0	$f \neq 0$	$-2^{e-1022}(0.f)$

- 二进制编码的十进制数表示：NBCD

数据校验码

- 基本思想：存储额外的信息以进行检错和校正
- 处理过程



- 常见类型
  - 奇偶校验码
  - 海明码
  - 循环冗余校验码

位的编号	12	11	10	9	8	7	6	5	4	3	2	1
位置编号	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
数据位	D8	D7	D6	D5		D4	D3	D2		D1		
校验位					C4				C3		C2	C1

指令系统

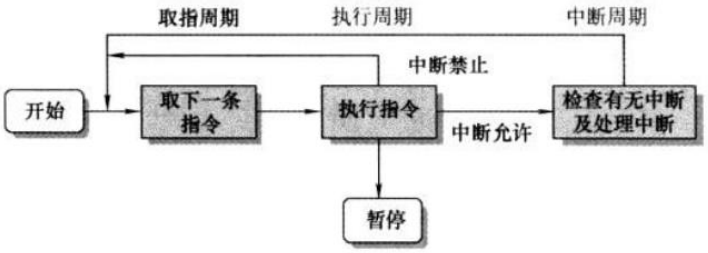
- 指令的要素：操作码+源操作数引用+结果操作数引用+下一条指令应用
  - 操作码：控制转移，……
  - 操作数
    - 寻址：立即，直接，间接，寄存器，寄存器间接，偏移，栈
    - 大端序 vs. 小端序
- 指令格式
  - 设计原则
  - 指令长度：位的分配，变长指令
- 指令集设计
  - 示例：不同地址数量的指令实现相同功能

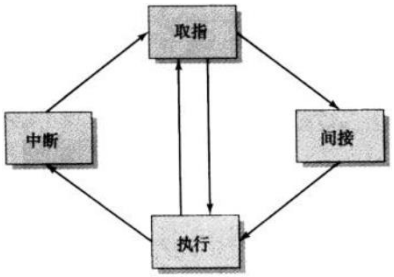
CPU的任务

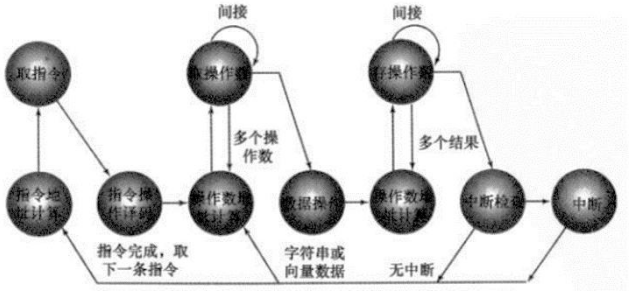
- 取指令：CPU必须从存储器（寄存器、cache、主存）读取指令
- 解释指令：必须对指令进行译码，以确定所要求的动作
- 取数据：指令的执行可能要求从存储器或输入/输出（I/O）模块中读取数据
- 处理数据：指令的执行可能要求对数据完成某些算术或逻辑运算
- 写数据：执行的结果可能要求写数据到存储器或I/O模块

CPU.指令执行

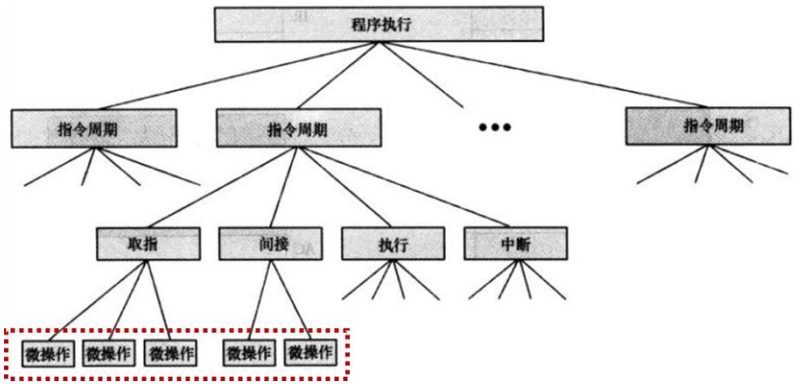
- 指令周期

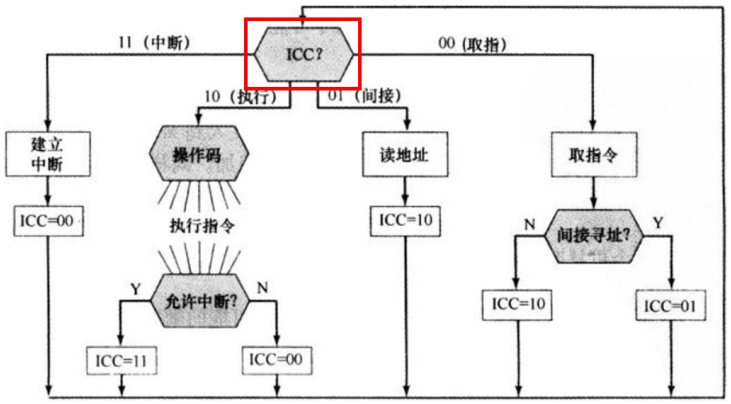






- 微操作
  - 分组原则：事件顺序，避免冲突
  - 指令周期代码（ICC）





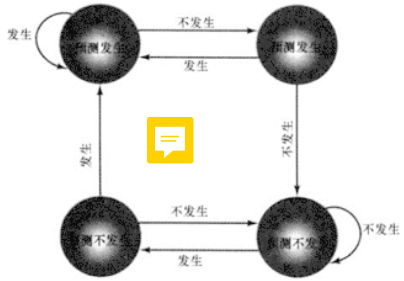
CPU.指令执行（续）

- 指令流水线：一条指令的处理过程分成若干个阶段，每个阶段由相应的功能部件完成

- 评价指标：加速比

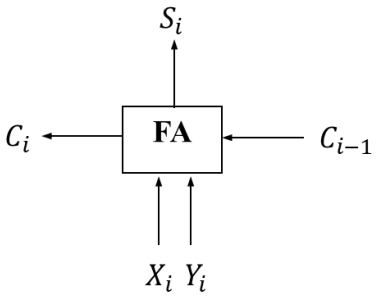
$$S_k = \frac{T_{1,n}}{T_{k,n}}$$

- 冒险
  - 结构冒险 / 硬件资源冲突：不同硬件资源，分时使用一个资源
  - 数据冒险 / 数据依赖性：nop，bubble，旁路，交换指令顺序
  - 控制冒险

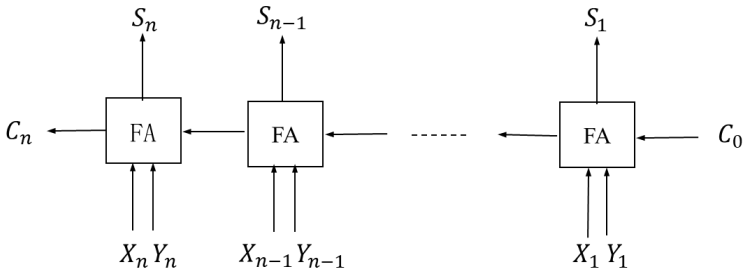


CPU.ALU：整数运算

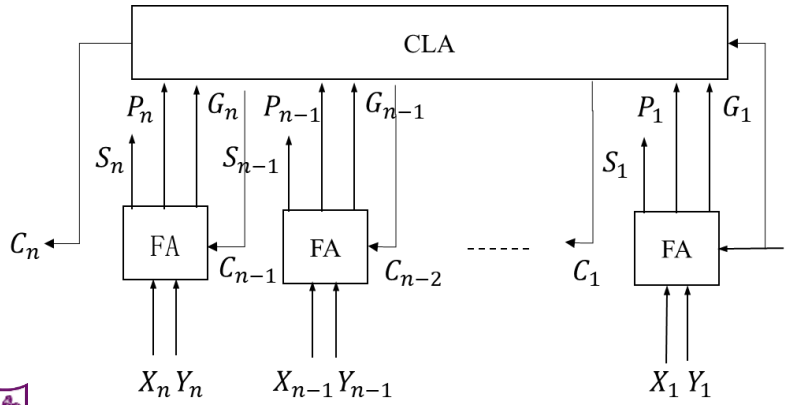
- 全加器



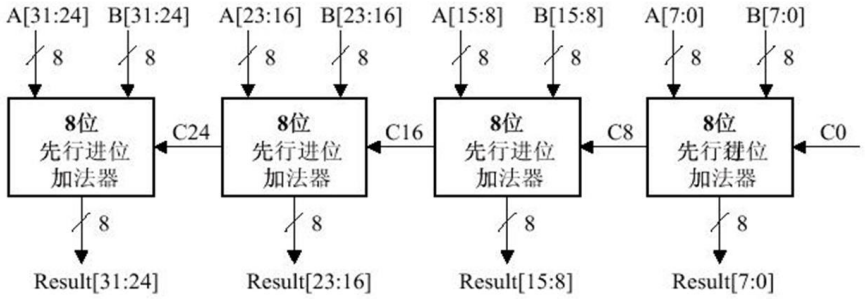
- 串行进位加法器



- 全先行进位加法器

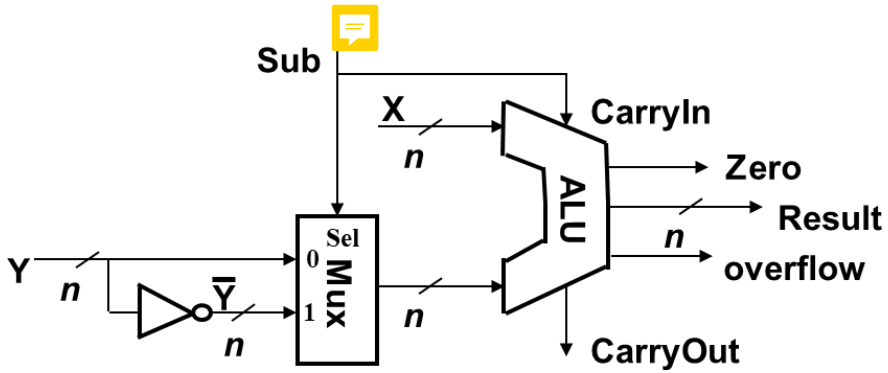


- 部分先行进位加法器



CPU.ALU：整数运算（续）

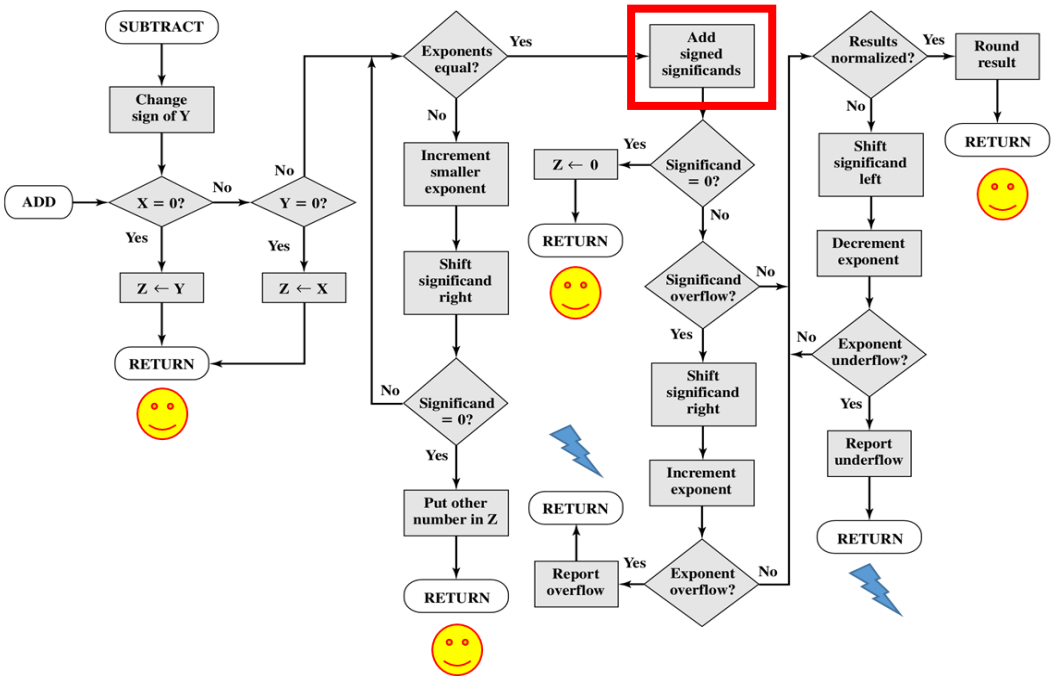
- 加法：溢出
  - $X_n = Y_n$且$S_n \neq X_n, Y_n$: $overflow = X_n Y_n \overline{S_n} + \overline{X_n}\,\overline{Y_n} S_n$
  - $C_n \neq C_{n-1}$: $overflow = C_n \oplus C_{n-1}$
- 减法：



- 乘法：布斯算法
- 除法：恢复余数，不恢复余数

CPU.ALU：浮点数运算

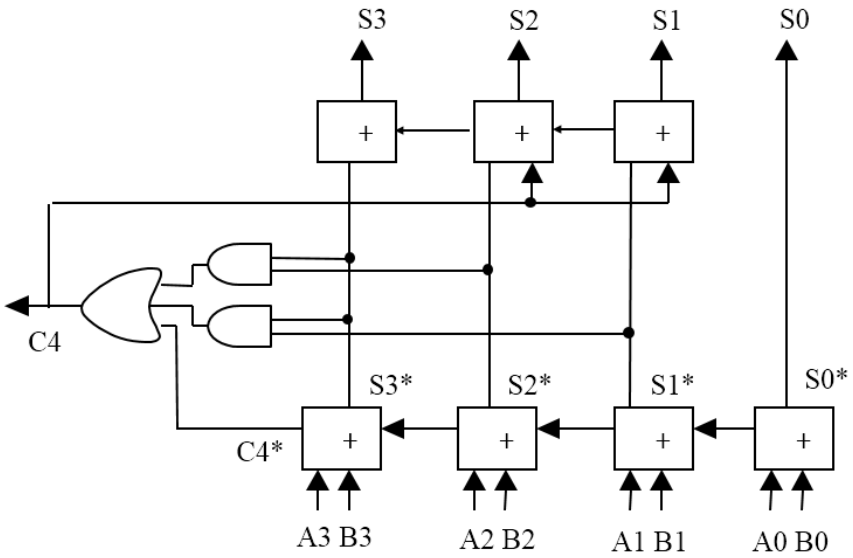
- 加法和减法



- 乘法
- 除法
- 精度考虑
  - 保护位
  - 舍入：就近舍入，朝+∞舍入，朝−∞舍入，朝 0 舍入

CPU.ALU：NBCD加法和减法

- 加法：通过+0110进行调整

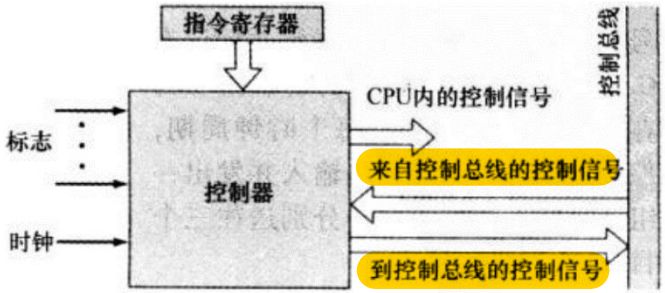


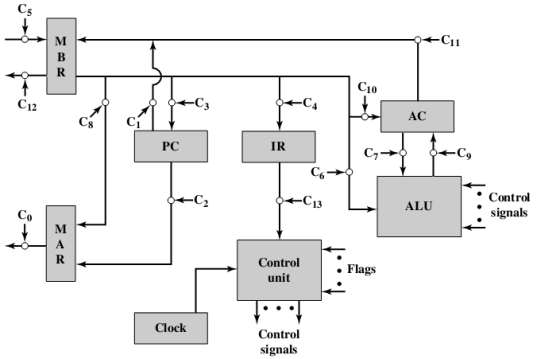
- 减法：参照补码减法，避免借位
  - $N_1 - N_2 = N_1 + (10^n - N_2) - 10^n$
    $= N_1 + (99\ldots9 - N_2 + 1) - 10^n$

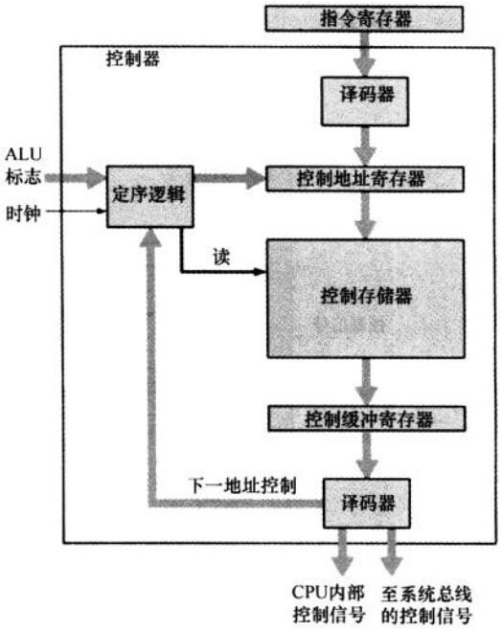
“反转”每一个数字，最后一位加1

CPU.控制器

- 控制器的输入输出
  - 输入：指令寄存器，标志，时钟，来自控制总线的控制信号
  - 输出：CPU内的控制信号，到控制总线的控制信号
- 控制信号
- 控制器的实现
  - 硬布线实现：组合电路
  - 微程序实现：相对简单的逻辑电路
    - 微程序
    - 构成
    - 工作流程

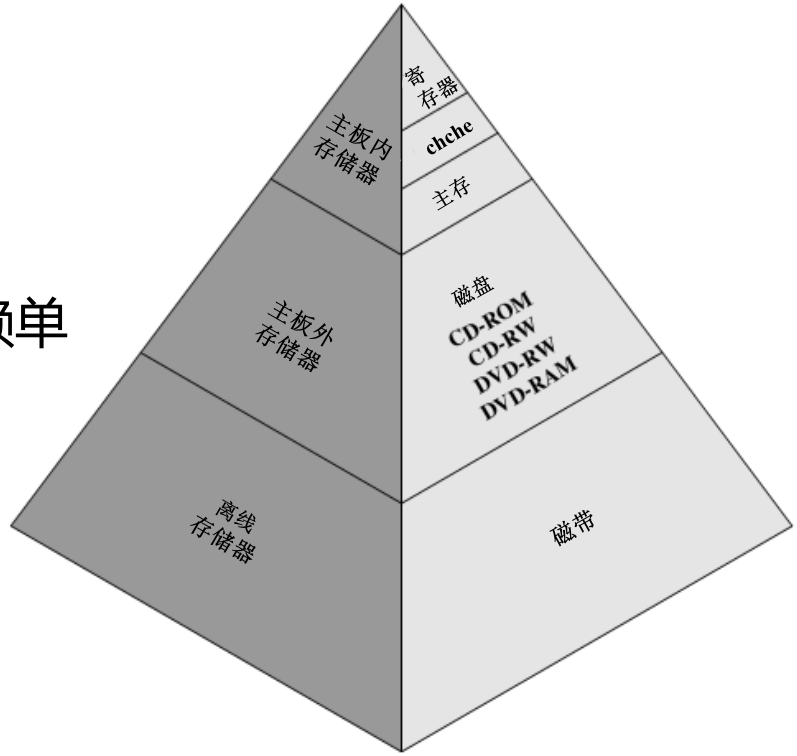




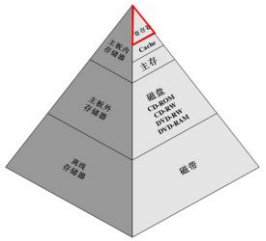


层次式存储结构

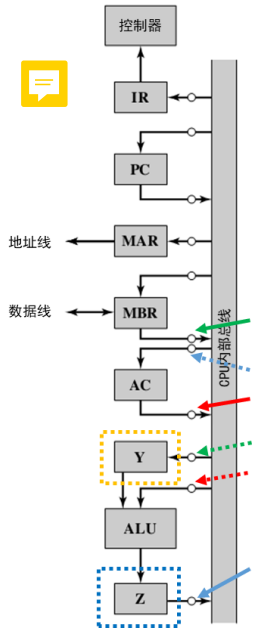
- 需求
  - 大容量数据存储
  - 高速性能
- 解决方案
  - 使用存储器层次结构而不是依赖单个存储器组件

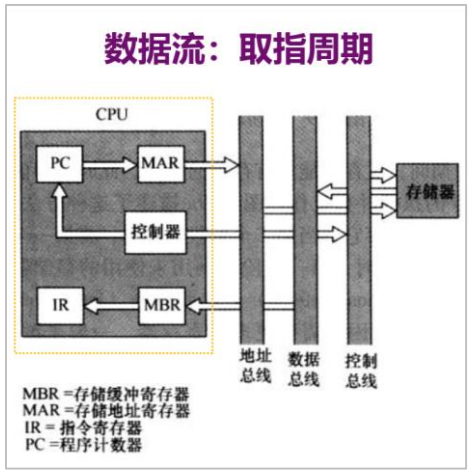


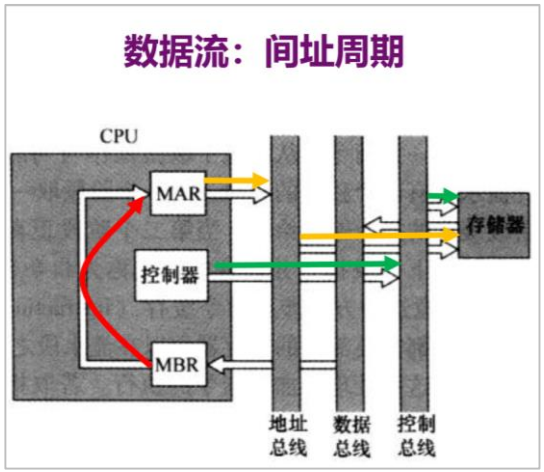
寄存器

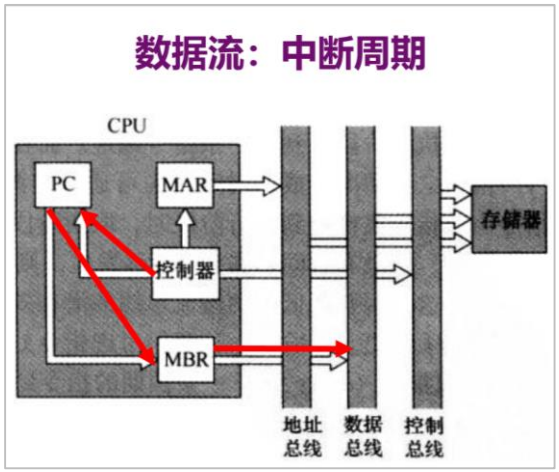


- CPU需要一些小容量的内部存储器
  - 在指令周期中临时保存指令和数据
  - 记录当前所执行指令的位置，以便知道从何处得到下一条指令
- 用户可见寄存器：允许编程人员访问，减少主存访问
  - 设计出发点：寄存器数量，寄存器长度
  - 保存和恢复
- 控制和状态寄存器：控制器控制CPU操作，操作系统程序控制程序执行
  - 常见：PC，IR，MAR，MBR，程序状态字（PSW）
  - 设计出发点：对操作系统的支持，控制信息的分配

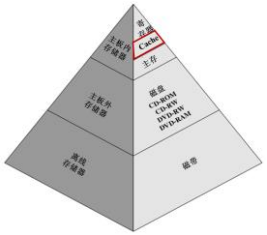




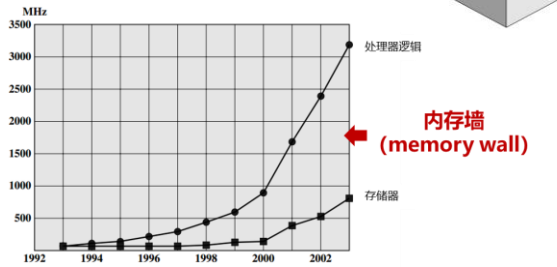




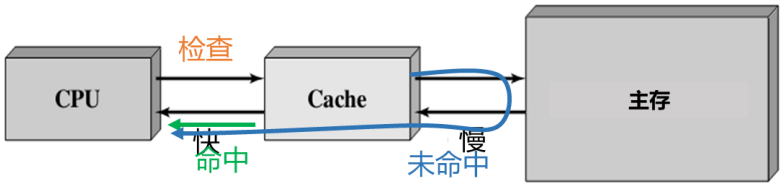
高速缓冲存储器 Cache



- 动机：内存墙（主存和CPU传输数据的速度跟不上CPU的速度）



- 工作流程：



- 关键问题
  - 如何判断是命中还是未命中？ Cache的tag
  - 如果未命中，为什么不直接把所需要的字从内存传送到CPU？
  - 如果未命中，为什么从内存中读入一个块而不只读入一个字？

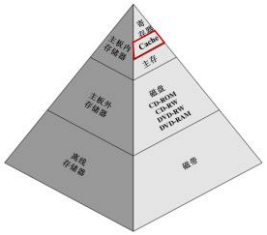
程序访问的局部性原理

  - 使用Cache后需要更多的操作，为什么还可以节省时间？

平均访问时间

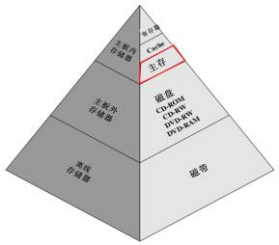
$$T_A = p \times T_C + (1-p) \times (T_C + T_M)$$
$$= T_C + (1-p) \times T_M$$

高速缓冲存储器 Cache（续）



- 设计要素
  - Cache容量
  - 映射功能：直接映射，关联映射，组关联映射
  - 替换算法：FIFO，LRU，LFU，Random
  - 写策略：写回法，写直达
  - 行大小
  - Cache数目

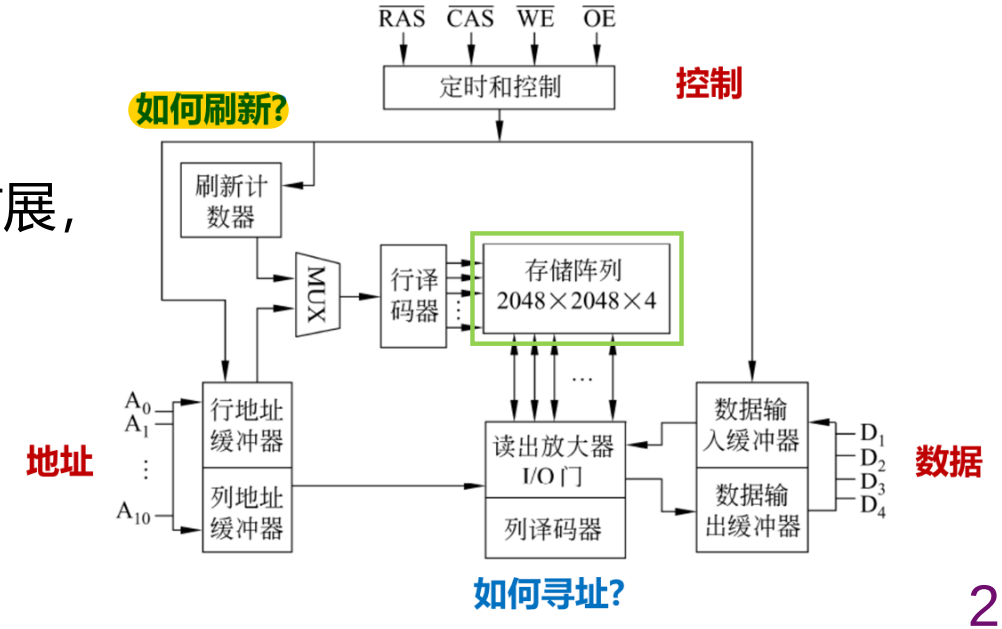
内部存储器



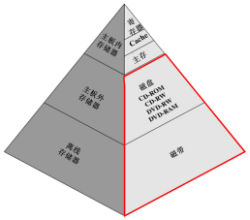
- **位元**：半导体存储器的基本元件，用于存储1位数据
- 半导体存储器类型
  - DRAM vs. SDRAM
  - DDR

存储器类型	种类	可擦除性	写机制	易失性
随机存取存储器（RAM）	读-写存储器	电可擦除，字节级	电	易失
只读存储器（ROM）	只读存储器	不可能	掩膜	非易失
可编程ROM（PROM）			电	
可擦除PROM（EPROM）	主要进行读操作的存储器	紫外线可擦除，芯片级		
电可擦除PROM（EEPROM）		电可擦除，字节级		
快闪存储器		电可擦除，块级		

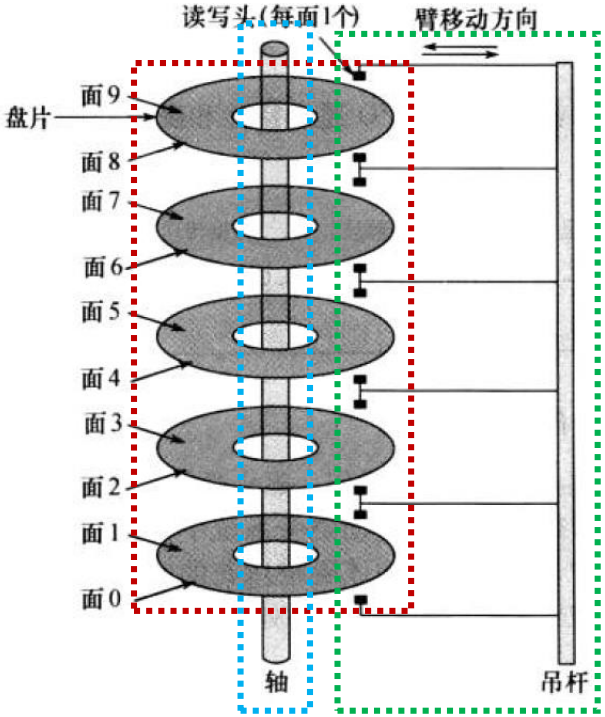
- 从位元到主存
  - 寻址单元
  - 存储阵列：位扩展，字扩展，位字同时扩展
  - 芯片
  - 模块组织
  - 主存



外部存储器



- 磁盘
  - 结构：基材（玻璃），磁头
  - 读写机制
  - 数据组织：磁道，扇区（恒定角速度 vs. 多带式记录，默认大小），间隙，柱面
  - 格式化
  - I/O访问时间：寻道，旋转延迟，传送
  - 磁头寻道 / 磁盘调度
- 光盘：CD，DVD
- 磁带：顺序读取
- 快闪存储器：U盘，SSD



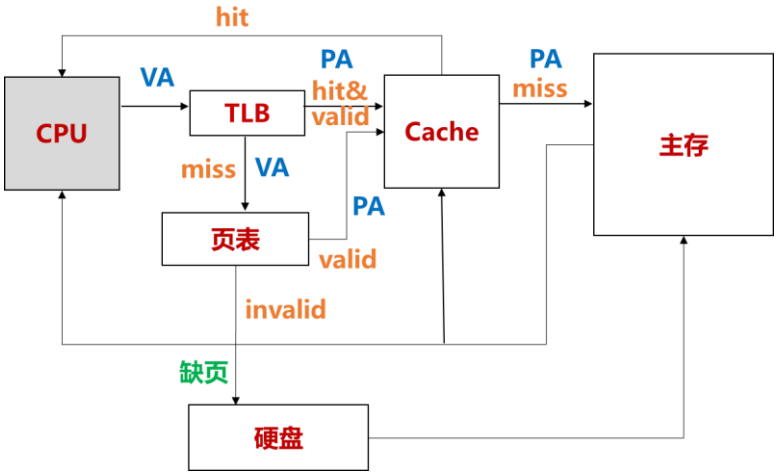
冗余磁盘阵列 RAID

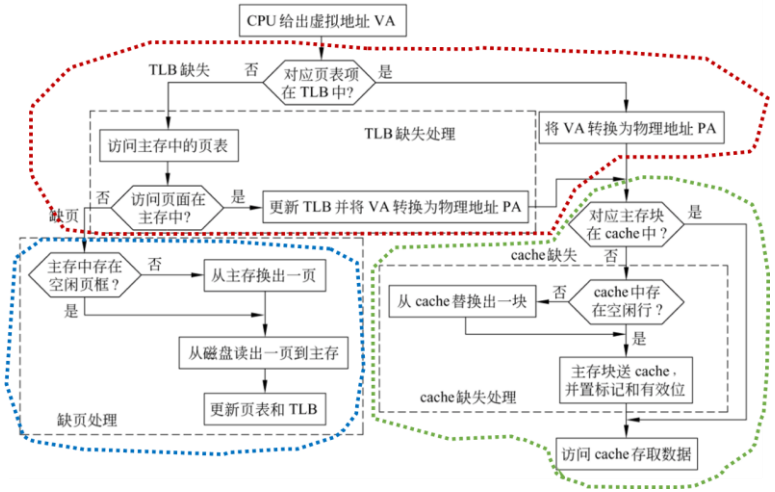
- 基本思想
  - 将多个独立操作的磁盘按某种方式组织成磁盘阵列，以增加容量
  - 将数据存储在多个盘体上，通过这些盘并行工作来提高数据传输率
  - 采用数据冗余来进行错误恢复以提高系统可靠性
- 分类

种类	级别	描述	磁盘要求	数据可用性	大 I/O 数据传输能力	小 I/O 请求速率
条带化	0	非冗余	N	比单盘低	很高	读和写都很高
镜像	1	镜像	$2N$	比 RAID 2、3、4、5 高；比 RAID 6 低	读比单盘高；写与单盘类似	读高达单盘的两倍；写与单盘类似
并行存取	2	汉明码冗余	$N+m$	比单盘高很多，与 RAID 3、4、5 差不多	列表各级中最高	接近于单盘的两倍
	3	位交错奇偶校验	$N+1$	比单盘高很多；与 RAID 2、4、5 差不多	列表各级中最高	接近于单盘的两倍
独立存取	4	块交错奇偶校验	$N+1$	比单盘高很多；与 RAID 2、3、5 差不多	读与 RAID 0 类似；写低于单盘	读与 RAID 0 类似；写显著低于单盘
	5	块交错分布式奇偶校验	$N+1$	比单盘高很多；与 RAID 2、3、4 差不多	读与 RAID 0 类似；写低于单盘	读与 RAID 0 类似；写显著低于单盘
	6	块交错分布式奇偶校验	$N+2$	列表各级中最高	读与 RAID 0 类似；写比 RAID 5 低	读与 RAID 0 类似；写显著低于 RAID 5

虚拟存储器

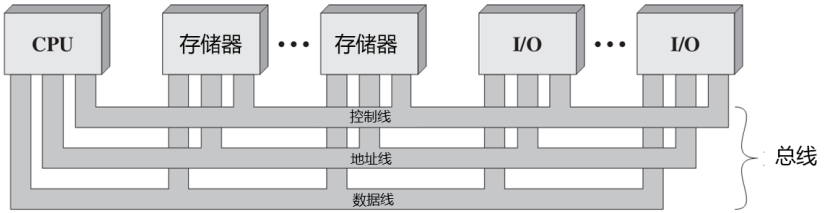
- 存储器管理：将更多任务装入主存
  - 分区：固定分区，可变长分区
  - 分页：
    - 页框，页，页表
    - 逻辑地址，物理地址
- 虚拟存储器
  - 基本思想：请求分页
  - 类型
    - 分页式虚拟存储器
      - 页表，快表，地址转换
    - 分段式虚拟存储器
    - 段页式虚拟存储器



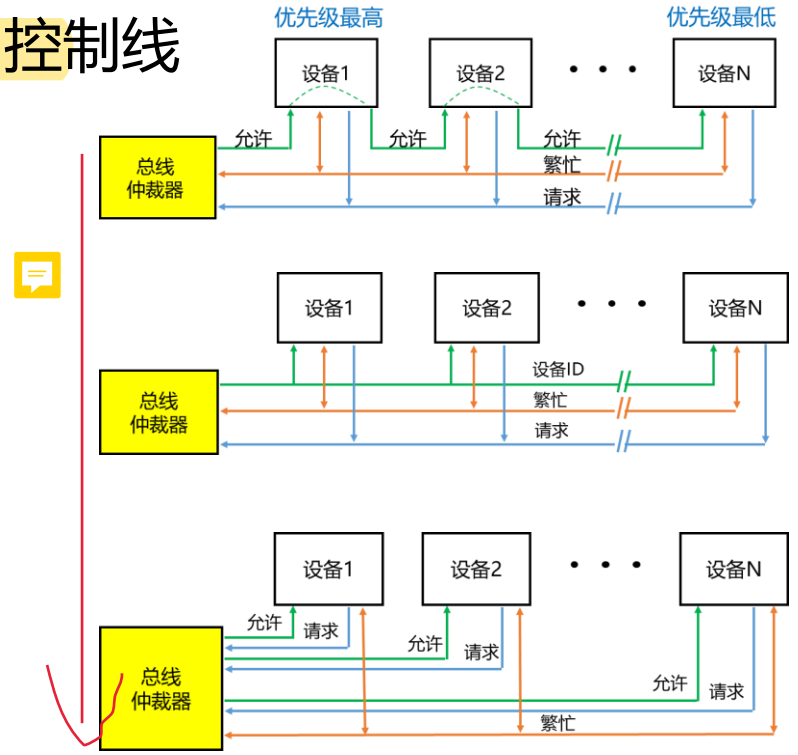


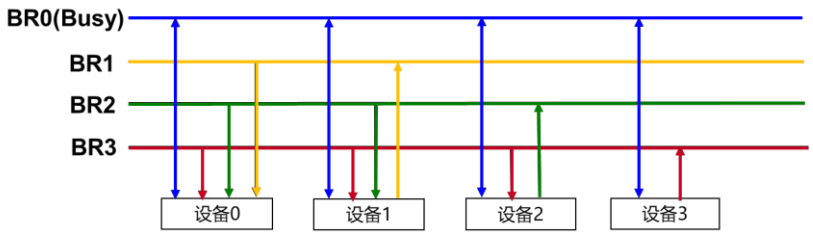
总线

共享传输介质
简化互连布局和处理器控制



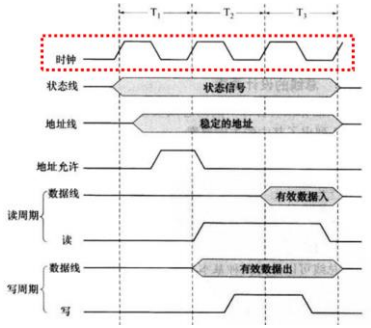
- 问题：计算机部件互连复杂
- 类型：芯片内部总线，系统总线，通信总线
- 构成：地址线，数据线（复用），控制线
- 用途：专用总线，复用总线
- 仲裁（arbitration）
  - 集中式：菊花链，计数器查询，独立请求
  - 分布式：自举式，冲突检测

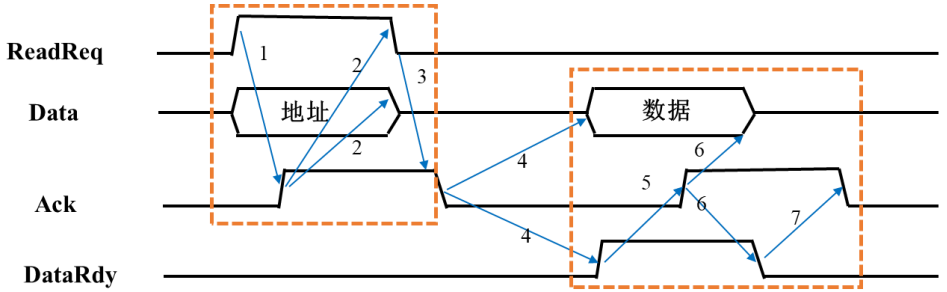




总线（续）

- 时序（timing）
  - 同步，异步，半同步，分离事务

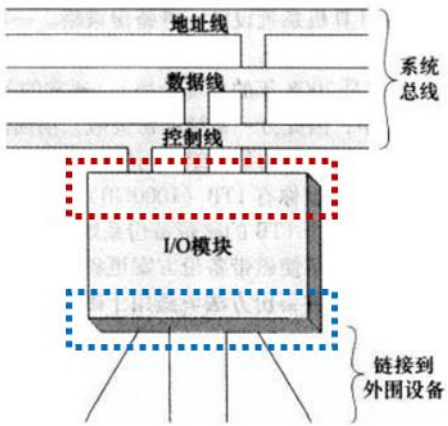


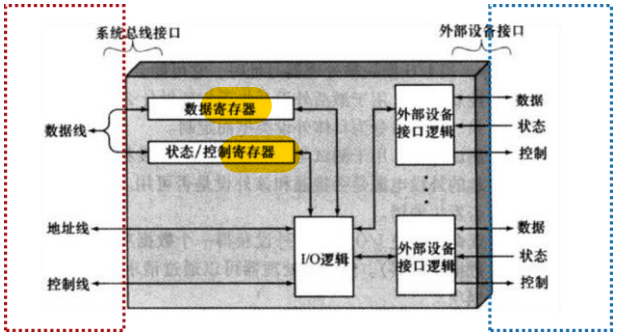


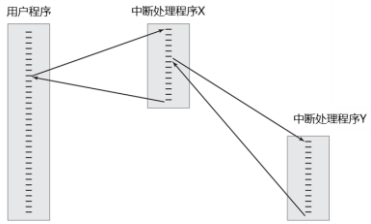
- 总线带宽（bandwidth）和数据传输速率（data transfer rate）
  - 同步 vs. 异步
  - 不同数据块大小
- 总线层次结构
  - 单总线，双总线，多总线

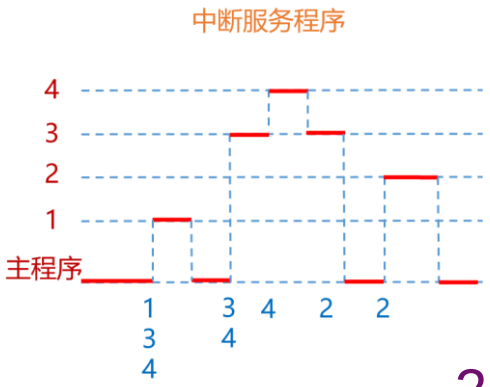
输入输出

- 外围设备
- I/O模块
  - 计算机内部系统和外设之间的桥梁
  - 功能：处理器通信，设备通信，数据缓冲，控制和定时，检错
  - 结构：外设接口并行 vs. 串行
- I/O操作技术
  - 编程式 I/O
  - 中断驱动式 I/O：响应优先级 vs. 处理优先级
  - 直接存储器读取（DMA）：内存访问
- I/O模块的演变









谢谢

rentw@nju.edu.cn

南京大學
NANJING UNIVERSITY