前言：该资料主要整理了rtw课件中需要背书的部分，包括各种算法/策略/方案的“报菜名”以及它们的优缺点，还有诸如“能不能无限增加xxx”一类的问题的收集。所有内容全部来自rtw的课件以及录播，希望对大家做部分大题和选择题时带来帮助！

一.计算机系统概述

计算机组织与结构：

组织：对编程人员不可见，包括控制信号、存储技术。例如：乘法指令的内部实现

结构：对编程人员可见，包括指令集、位数。例如：是否提供乘法指令

计算机的演变：第一代：真空管；第二代：晶体管；第三代及以后：集成电路

冯诺依曼结构：

三原则：二进制；存储程序；五个组成部分：主存储器，处理单元，控制单元，输入单元，输出单元

摩尔定律的影响：

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；

减少了对电能损耗和冷却的要求；

集成电路上的内部连结比焊接更可靠，芯片间的连接更少

性能评价的标准：

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：提高CPU的性能

二.计算机的顶层视图

CPU频率提高的好处与局限：

好处：增加处理速度

局限：

理论限制：mos管开关，脉冲通过门电路需要时间；为了信号同步，每个脉冲信号需要持续一定时间；……

制造限制：芯片面积越来越大，连线延迟越来越大，需要保证信号在指定时钟周期内能完成任何的移动；频率越高CPU会费电和散热高

解决方案：指令流水线、Cache、多核……

存储器速度、容量与成本间的约束：

容量：肯定越大越好；

速度：需要跟上处理器，不能过慢；

成本：相对于其他组件合理；

约束直接的关系：更短的访问时间意味着更高的每比特成本（代表更小的容量）

计算机体系结构遇到的问题及解决方案：

CPU的频率不能无限提高：改进CPU芯片结构

内存墙的存在：采用高速缓存

CPU等待I/O：采用中断机制

兼顾存储速度、容量和成本：采用层次式存储结构

I/O设备间传输速率差异大：采用缓冲区和改进I/O操作技术

计算机内部互连复杂：采用总线

三.数据的机器级表示

采用二进制的原因：

多种物理器件可以表示两种稳定的状态，用于表示0和1；

二进制的编码和运算规则简单；

1和0可以表示逻辑上的真和假

使用补码的原因：二进制原码、反码、移码进行加法运算时都会导致不必要的硬件需求

二进制编码的十进制表示：

浮点运算的限制：精度限制、转换成本高；

应用需要：长数字串的计算：会计，……

四.数据校验码

差错：数据在计算机内部进行计算、存取和传送过程中，由于元器件故障或噪音干扰等原因，会出现差错

硬故障：永久性的物理故障，由恶劣的环境、制造缺陷和旧损引起

软故障：随机非破坏性事件，由电源问题或α粒子引起

奇偶校验码：

优点：代价低，只需要1位额外数据，计算简单

缺点：不能发现出错位数为偶数的情形；不能纠错

适用：对较短长度（1B）的数据进行检错

海明码：

优点：1位错误时可以纠错

缺点：额外成本很大；要求把数据分为字节

循环冗余校验：

适用：适用于以流格式存储和传输大量数据

五.整数运算

全加器：实现简单；但随着位数增加会变慢

全先行进位加法器：速度快；但随着位数增加电路会越来越复杂

部分先行进位加法器：取得计算时间和硬件复杂度的权衡

八.内部存储器

半导体存储器类型：RAM（DRAM、SRAM）、ROM、PROM、EPROM、EEPROM、快闪

DRAM和SRAM的对比：

相同点：易失，需要电源供电来保持位值

不同点：DRAM比SRAM具有更简单、更小的位元，但要求能够刷新的电路

DRAM比相应的SRAM密度更高、价格更低

SRAM通常比DRAM快

DRAM通常满足大容量存储器的需求，SRAM一般用于高速缓存，DRAM用于主存

高级的DRAM架构：

问题：传统的DRAM芯片受到其内部架构和处理器内存总线接口的限制；传统DRAM是异步的

类型：SDRAM；DDR SDRAM

ROM：

特性：非易失；可读但不可写

应用：[微程序](#微程序)设计、库子程序、系统程序、函数表

问题：无出错处理机会，如果有一位出错，整批的ROM芯片只能报废；用户无法写入数据，唯一的数据写入机会在出厂时完成

PROM：

特性：非易失；只能被写入一次，写入内容通过“编程”实现

与ROM的对比：PROM提供了灵活性和方便性；ROM在大批量生产领域仍具有吸引力

主要进行读操作的存储器：

特性：非易失；写操作较读操作困难

应用：读操作比写操作频繁得多的场景

类型：EPROM、EEPROM、快闪

EPROM：

特性：光擦除，电写入

与PROM对比：EPROM更贵，但具有可多次改写的优点

EEPROM：

特性：可以随时写入而不删除之前的内容；只更新寻址到的一个或多个字节；写操作每字节需要几百微秒

与EPROM对比：EEPROM更贵，且密度低，支持小容量芯片

快闪存储器：

特性：

电可擦除：与EEPROM相同，优于EPROM

擦除时间为几秒：优于EPROM，低于EEPROM

可以在块级擦除，不能在字节级擦除：优于EPROM，不如EEPROM

达到与EPROM相同的密度：优于EEPROM

价格和功能介于EPROM和EEPROM之间

内存的刷新策略：

集中式刷新：

方式：停止读写操作，刷新每一行

缺点：刷新时无法操作内存

分散式刷新：

方式：每个存储周期内，读写操作完成时进行刷新

对集中式的改进：用户不会感受到内存停止

缺点：会增加每个存储周期的时间

异步刷新：

方式：每行各自以64ms间隔刷新；两个时间间隔内保证每一行被刷新一次

优点：刷新不需要占用读写时间，效率高，常用

内存的模块组织策略：

位扩展：地址线不变，数据线增加

使用8个4\*1b芯片组成4\*8b存储器（注意此时每个地址都会同时选中八个芯片上的对应位置）

字扩展：数据线不变，地址线增加

使用8个4\*8b芯片组成32\*8b存储器（高3位用于选片）

字、位扩展：数据线和地址线同时增加

使用8个4\*4b芯片组成16\*8b存储器（高2位选片；每次同时选中两片）

九.高速缓冲存储器

程序访问的局部性原理：

时间局部性：在相对较短的时间周期内，重复访问特定的信息

空间局部性：在相对较短的时间周期内，访问相邻位置的数据

顺序局部性：当数据被线性排列和访问（一维数组等）时，空间局部性的特殊情况

扩大Cache容量的好处与局限：

好处：增加了命中率

局限：增大了Cache的开销和访问时间

Cache的映射策略：

直接映射：

关联度：1

优点：简单、快速映射、快速检查

缺点：抖动现象

适用：大容量的Cache

关联映射：

关联度：C

优点：避免抖动

缺点：实现起来较为复杂；搜索代价大

适用：容量较小的Cache

组关联映射：

关联度：K

优点：结合了直接映射和关联映射的优点

缺点：结合了直接映射和关联映射的缺点

适用：容量中等的Cache

三种映射策略的比较：

关联度越低，命中率越低，判断是否命中的时间越短，标记额外空间越小。

命中率：直接映射<组关联映射<关联映射

命中时间：直接映射<组关联映射<关联映射

标记长度：直接映射<组关联映射<关联映射

Cache的替换策略：通过硬件实现

LRU实现：对于2路组关联映射，每行包含一个USE位

FIFO实现：时间片轮转法或环形缓冲技术

LFU实现：每一行设置一个计数器

随机替换实现：随机替换，性能上只稍逊于其他算法

Cache的写策略：

写直达：

优点：确保主存和Cache的内容总是一致的

缺点：产生大量的主存访问，减慢写操作

写回法：

优点：减少了主存访问的次数

缺点：部分主存数据可能不是最新的：I/O模块存取时可能会无法获得最新的数据，为解决该问题会使得电路设计更加复杂且有可能带来性能瓶颈

Cache的行大小和命中率的关系（增加行大小的影响）：

一开始，增大行大小会增大命中率，因为每一块有更多的内容被载入Cache，利用了空间局部性；当行大小变得较大以后，继续增加行大小，命中率会下降，其一是Cache容量一定的情况下，装入Cache的行数变少，数据块频繁替换，其二是每个数据块的数据在主存中位置较远，被使用的可能性减小。

行大小和命中率间的关系较为复杂。

Cache的数目：一级vs多级；统一vs分立：

一级：减少处理器在外部总线上的访问，从而减少执行时间

多级：L1未命中时，减少处理器对总线上DRAM和ROM的访问（现代多级Cache多集成在芯片上）

统一：更高的命中率，在获取指令和数据的负载间自动平衡；只需要设计和实现一个Cache

分立：消除Cache在指令的取值、移码单元和执行单元之间的竞争，在指令流水线的设计中很重要（[流水线的不同阶段](#并行访问)可以分别访问不同的Cache）

十.外部存储器

外部存储器的类型：磁盘存储器、光存储器、磁带、U盘、固态硬盘SSD

磁盘使用玻璃基材的优势：

改善磁膜表面的均匀性，提高可靠性

减少表面瑕疵，从而减少读写错误

能过支持磁头较低的飞行高度

更高的硬度，转动时更稳定

更强的抗冲击和抗损伤能力

磁盘增加数据密度的局限：需要更窄的磁头和更窄的磁道，增加出错风险

磁头的数量：

单磁头：读写公用同一个磁头（软盘，早期硬盘）

双磁头：读写分别用一个磁头（当代硬盘）

磁盘扇区的划分策略：

恒定角速度：增大盘片区域上信息位的间隔，是磁盘能以恒定角速度扫描信息

优点：能以磁道号和扇区号直接寻址各个数据块

缺点：存储容量受到最内侧磁道的最大数据密度的限制

多带式记录：将盘面划分为多个同心圆区域，每个区域各磁道的扇区数量相同；外侧磁道的扇区数多于内侧磁道

优点：提升存储容量

缺点：需要更复杂的电路

磁盘的磁道调度算法：

先来先服务（FCFS）：

优点：公平简单

缺点：如果有大量分散的请求会性能不佳

最短寻道时间优先（SSTF）：

优点：每次寻道时间最短（局部最优），平均寻道时间缩短

缺点：可能产生饥饿现象，尤其是位于两端的磁道请求

扫描/电梯（SCAN）：

优点：性能较好，平均寻道时间较短，不会产生饥饿现象

缺点：只有到最边上的磁道才能改变磁头方向，对各个磁道响应时间不平均

循环扫描（C-SCAN）：

优点：与SCAN算法相比：对各个磁道的响应频率平均

缺点：平均寻道时间比SCAN长

LOOK：

优点：减少了SCAN需要移动到边缘的额外时间

缺点：对新加入的任务不友好（尤其是边缘处的）

C-LOOK：

优点：减少了C-SCAN算法的时间

缺点：略慢于LOOK算法

光存储器的类型：CD、CD-ROM、CD-R、CD-RW、DVD、DVD-R、DVD-RW、蓝光DVD

（R：recordable，可刻录；RW：rewritable，可重写）

CD和CD-ROM的优缺点：

优点：可以大规模复制；可以更换；CD-ROM比CD更耐用且有纠错功能

缺点：只读；存取时间比磁盘长得多

DVD vs CD：

DVD上的位组装更精密；DVD采用双层结构；DVD-ROM可以用两面记录数据

磁带和磁盘的区别：

读取：磁带采用顺序读取，磁盘采取直接读取

记录：磁带采用蛇形（串行）记录或并行记录，目前采取串行记录

U盘 vs软盘、光盘：

U盘使用[快闪存储器](#快闪)，体积小，容量大，非常方便，寿命长达数年

固态硬盘 vs 硬磁盘存储器：

抗震性好，无噪声，能耗低，发热量低

十一.冗余磁盘阵列

RAID0：

优点：将I/O负载分散，提高性能；无奇偶计算开销；设计简单易实现

缺点：只要有一个驱动器失效就会全部数据丢失

应用：任何要求高带宽的应用，如视频编辑、图像处理、应用压缩

RAID1：

优点：数据100%冗余，某些环境下能承受多个驱动器同时失效；最简单RAID子系统设计

缺点：磁盘数开销最大

应用：任何要求高可用性的应用，如统计、工资单、财务

RAID10：先做RAID1，再做RAID0

RAID01：先做RAID0，再做RAID1，两者主要区别在于对磁盘损坏的容错能力（类比逻辑上的与门和或门）

RAID2：

优点：可能有极高的数据传输率；数据传输率要求得越高，数据盘对ECC盘的比值越好；控制器实现比RAID3，4，5简单

缺点：短字长时很低效；冗余盘成本很高；只适用于多磁盘易出错环境

应用：无商品实现的存在/无商业化应用

RAID3：

优点：很高的读写传输率；磁盘失效时对吞吐率无显著影响；ECC对数据盘的低比率意味着高效率

缺点：最好情况下的事务率等于单盘的事务率；控制器设计相当复杂

应用：任何要求高吞吐率的应用，如直播、视频剪辑、图像处理、应用压缩

RAID4：

优点：很高的读数据事务率；ECC对数据盘的低比率意味着高效率

缺点：十分复杂的控制器设计；最差的写事务率和写聚集传输率；磁盘失效事件中，数据重构困难且低效

应用：无商品实现的存在/无商业化应用（同RAID2）

RAID5：

优点：最高的读数据事务率；ECC对数据盘的低比率意味着高效率；好的聚集传输速率

缺点：最复杂的控制器设计；磁盘失效事件中，重构数据困难（和RAID1相比）

应用：用途最多的RAID级；用于各种服务器

RAID50：对多组RAID5彼此构成条带访问，性能更高但容量利用率更低

RAID6：

优点：数据故障容忍能力最高，能承受多个驱动器同时失效

缺点：较复杂的控制器设计；计算奇偶校验地址的控制器开销非常高

应用：丢失数据严重的应用

（分析RAID优缺点的角度：数据传输率（分读和写）、聚集传输率、控制器设计复杂度、对数据故障的容忍能力/数据可用性、磁盘数量，校验盘带来的额外开销）

RAID的类型：条带化：0；镜像：1；并行存取：2，3；独立存取：4，5，6

磁盘要求：1(2N)>2(N+m)>6(N+2)>3=4=5(N+1)>0=无(N)

数据可用性：6>1>2≈3≈4≈5>无>0

大I/O数据传输能力：读：2=3>0≈4≈5≈6>无，1>无

写：2=3>0>1≈无>4≈5>6

小I/O请求速率：读：0≈4≈5≈6（很高）>无，1≈2≈3>无（接近两倍）

写：0（很高）>无，2≈3（接近单盘两倍）>无≈1>4≈5>6

十二.虚拟存储器

各种分区方式：

简单固定分区：

优点：简单

缺点：浪费主存空间

可变长分区：

优点：提高了主存的利用率

缺点：时间越长，存储器中的碎片越多

虚拟存储器：

解决问题：内存大小有限而对内存需求增加的矛盾

优点：在不扩大物理内存的前提下载入更多的任务；编写程序时不需要考虑物理内存的状态；可以在大于物理内存的逻辑地址空间中编程

TLB、页表和Cache的缺失组合：页表缺失时TLB、Cache必定缺失，页表缺失时Cache和TLB的缺失无关联

分页式虚拟存储器：

优点：实现简单，开销少

缺点：数据可能跨页面

分段式虚拟存储器：

优点：段的分界与程序的自然分界相对应，易于编译、管理、修改和保护

缺点：段的长度不固定

段页式虚拟存储器：

优点：程序按段实现共享和保护

缺点：需要多次查表

十三.总线

总线的用途：

专用总线：

优点：高吞吐量，减少总线冲突

缺点：增加了系统的规模和成本

复用总线：

优点：布线少，节省空间和成本

缺点：需要更复杂的控制电路，且共享可能会降低性能

总线仲裁的平衡因素：

优先级：优先级高的设备优先被服务

公平性：优先级低的设备不能一直被延迟

集中式的仲裁方案：

链式查询（菊花链）：

优点：确定优先级很简单；可以灵活添加设备

缺点：不能保证公平性；对电路故障敏感；限制总线的速度

计数器查询：

优点：通过使用不同的初始计数，灵活地确定优先级；对电路故障不敏感

缺点：需要添加设备ID线；需要解码和比较ID信号；限制总线的速度

独立请求：

优点：快速响应；可编程的优先级

缺点：复杂的控制逻辑；更多的控制线路

分布式的仲裁方案：

总线的时序：

同步时序：

优点：更容易实现和测试

缺点：所有设备共享一个时钟；总线长度受到时钟偏差的限制

异步时序：

优点：可以灵活地协调速度不同的设备

缺点：接口逻辑复杂（三次握手）；对噪声敏感

半同步时序：结合了同步时序和异步时序的优点

分离事务：

优点：增加总线利用率

缺点：增加每个总线事务的持续时间和系统复杂度

十四.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，下一指令引用

指令类型：数据传送，算术运算，逻辑运算，转换，输入输出，系统控制，控制转移

增加每条指令的地址数的好处与局限：

好处：指令条数减少，执行时间更短，程序更简单

局限：指令长度变长，需要更复杂的CPU

操作数：

数值型：[二进制整数](#二进制整数)，二进制浮点数，[十进制数](#十进制数)

字符型：ASCII，EBCDIC，Unicode

各种操作数的寻址方式：

立即寻址：操作数=A

优点：获得操作数不需要额外的存储器访问

缺点：数的大小受限于地址字段的长度

直接寻址：EA=A

优点：只要求一次存储器访问，且无需为生成地址而专门计算

缺点：有限的地址空间（地址的大小受限于地址字段的长度）

间接寻址：EA=（A）

优点：扩大了地址空间

缺点：取操作数需要访问两次存储器

解释：地址引用的数量限制可能是有益的（？）

寄存器寻址：EA=R

优点：指令中只需要一个较小的寄存器字段，且不需要存储器访问

缺点：地址空间十分有限（寄存器数量限制）

解释：寄存器寻址只有在被有效利用时才有意义

寄存器间接寻址：EA=（R）

优点：扩大了地址空间，比间接寻址减少一次存储器访问

缺点：比寄存器寻址多一次存储器访问

偏移寻址：EA=（R）+A

相对寻址：EA=（PC）+A

优点：利用程序局部性原理，节省指令中地址的位数

用法：大多数存储器访问都靠近正在执行的指令

基址寄存器寻址：EA=（B）+A

用法：虚拟内存空间的程序重定位

变址寻址：EA=A+（R）（前两者以A作为偏移量，这里以寄存器作为偏移量）

用法：为完成重复操作提供一种高效机制

扩展：前变址EA=（A+（R））；后变址EA=（A）+（R）；基址带变址寻 址EA=A+（B）+（R）

栈寻址：栈指针保存在寄存器中，实际上是寄存器间接寻址方式

（提问：寻址方式属于计算机的组织还是结构？）

指令格式：必须包含一个操作码，以及隐式或显式的一个或多个操作数；必须显式或隐式指定寻址方式；大多数指令集使用不止一种指令格式

指令格式的设计原则：

指令尽量短——程序占用存储空间小

有足够的操作码位数——要为操作类型不断增加预留

操作码的编码必须有唯一的解释——要么合法要么不合法

指令长度是字节的整数倍——与内存寻址对应，便于指令读取和地址计算

合理选择地址字段个数——涉及到指令长度和规整性，是时空开销权衡的结果

指令尽量规整——简化硬件实现

指令长度的权衡：在强有力的指令清单和节省空间中进行；和传送长度与存储单元长度呈倍数关系

增长指令的好处与局限：

好处：对编程人员有利；

局限：存储空间变大，数据传送时间变长

指令位数的分配：

对定长指令，在操作数数量和寻址能力间权衡

寻址位数的考虑因素：寻址方式种数；操作数数量；寄存器和存储器比较；寄存器组数；地址范围；寻址粒度

变长指令：

优点：易于提供大的操作码清单；寻址方式能更灵活

缺点：增加了CPU的复杂程度

指令集设计的基本原则：

完备性：操作类型要尽可能完备，但太复杂了也会给硬件实现带来困难

兼容性：能够兼容旧指令系统，为软件充分利用带来方便

均匀性：能对多种数据类型进行处理

可扩充性：操作码要预留一定的编码空间

指令集设计的基本问题：

操作指令表：应提供多少和什么样的操作，操作有多复杂

数据类型：对哪几种数据进行操作

[指令格式](#指令格式)：指令的位长度、地址数目、各个字段的大小等

[寄存器](#寄存器)：能被指令访问的寄存器数目和用途

[寻址](#寻址)：寻址方式的种类和有效地址的计算

下一条指令地址的确定：通常用PC寄存器

十五.指令周期和指令流水线

流水线的方法：

两阶段方法：取指令和执行指令

问题：执行时间一般长于取指时间；主存访问冲突；条件分支指令

六阶段方法：FI，DI，CO，FO，EI，WO（理解几个缩写的含义）

问题：不是所有指令都包含六个阶段，如LOAD没有WO，只能假设每个指令按六阶段完成；不是所有阶段可以并行完成，如FI、FO、WO都涉及[存储器访问](#分立Cache)；若每个阶段时间不同，则会涉及等待

限制：条件分支指令、中断

流水线阶段数提高的局限：每个阶段转移数据存在开销；处理内存和寄存器依赖以及优化管道所用的控制逻辑数量随着阶段的增加而增加

流水线冒险：

结构冒险：硬件资源冲突

问题：已进入流水线的不同指令在同一时刻访问相同的硬件资源

解决：使用[不同的硬件资源](#分立Cache)；或分时访问同一资源

数据冒险：数据依赖性

问题：未生成指令所需要的数据

解决：插入nop指令；插入bubble；转发/旁路；交换指令顺序

控制冒险：指令顺序被更改

问题：条件转移、中断、异常、调用/返回

解决：

取多条指令：多个指令流；预取分支目标；循环缓冲器

分支预测：静态预测；动态预测

动态预测：转移状态图；转移历史表

十六.控制器

寄存器分类：

用户可见寄存器：

类型：通用寄存器、数据寄存器、地址寄存器、条件码寄存器/标志寄存器

寄存器的数量能不能无限增加：

太少的寄存器会导致更多的存储器访问

太多的寄存器又不能显著减少存储器访问

寄存器的长度：应能保存大部分数据类型的值；某些机器允许两个相邻寄存器合并

保存和恢复：子程序调用时作为CPU的责任，其余时候作为程序员的责任

控制和状态寄存器：

例子：PC、IR、MAR、MBR、PSW

其他机器上可能存在的：指针寄存器、中断向量寄存器、系统栈指针、页表指 针寄存器、[控制地址寄存器、控制缓存寄存器](#控制寄存器)

设计的出发点：对操作系统的支持；控制信息在寄存器和存储器之间的分配（在成本和速度间权衡）

微操作分组的原则：事件流动顺序恰当、避免冲突、时间尽可能少

CPU的基本元素：ALU、寄存器组、内部数据通路、控制器、外部数据通路

CPU需要完成的微操作：

在寄存器间传送数据

将数据从寄存器传送到外部接口（如总线）

将数据从外部接口传送到寄存器

将寄存器作为输入和输出，完成算术和逻辑运算

控制器的基本任务：定序、执行

控制器的输入：

指令寄存器：当前指令的寻址方式和操作码

标志：确定CPU的状态和前一个ALU操作的结果

时钟：控制器要在每个时钟脉冲完成一个或一组微操作

来着控制总线的控制信号：向控制器提供控制信号

控制器的输出：

CPU内的控制信号：用于寄存器间传输数据；用于启动特定ALU功能

到控制总线的控制信号：到存储器的控制信号；到I/O模块的控制信号

控制器的实现：

硬布线实现：

构成：指令寄存器、译码器、定时器、控制器

优点：速度较快

缺点：硬件设计较为复杂

微程序实现：

微指令的设计考虑：

微指令的大小：微指令越小控制器存储成本越低

地址生成时间：尽可能快地执行微指令

微指令类型：双地址字段；单地址字段；可变格式

构成：定序逻辑、控制地址寄存器、控制存储器、控制缓冲寄存器

优点：简化了控制器的设计任务；实现起来成本较低且出错机会减少

缺点：比硬布线控制器慢

（提问：控制器的实现属于计算机的组织还是结构？）

十七.输入输出

为什么外设不能直接连到系统总线上：

外设种类繁多，操作方法多种多样；

外设的数据传送速度一般比存储器和处理器慢得多；

某些外设的数据传送速度比存储器和处理器快；

外设使用的数据格式和字长通常和处理器不同

I/O模块的功能：

处理器通信：命令译码、传送数据、状态报告、地址识别

设备通信：包含命令、状态信息和数据

数据缓冲：解决速度差异问题

控制和定时

检错：设备报告的故障和传输过程中数据位的变化

I/O模块的外部接口：

并行接口：多根线连接，同时传多位数据

串行接口：一根线连接，每次只传一位数据

当传输速度和总线长度增加时，并行接口的总线时钟频率会被限制

常用接口：FireWire和USB，都是串行接口

I/O操作技术：

编程式I/O：

中断：无中断，需要100%占用CPU

I/O命令：控制命令、测试命令、读命令、写命令

I/O指令：编址方式：存储器映射式I/O、分离式I/O

中断驱动式I/O：

中断：有中断

中断时保存与恢复：PC、PSW、通用寄存器、掩码字

响应优先级和处理优先级：掩码字矩阵

设备识别方式：

轮询：轮询次序决定优先级

[菊花链](#菊花链)：模块连接顺序决定优先级

独立请求：中断控制器决定优先级

多条中断线：和前面三者结合，仅挑选优先级最高的中断线

与处理器连接的I/O方式的不足：

I/O速度受处理器测试和服务设备速度的限制；处理器负责管理I/O传送，对于每一次I/O传送，处理器必须执行很多指令

DMA：

组成：数据计数、数据寄存器、地址寄存器、控制逻辑

内存访问策略：

CPU停止法：

优点：控制简单

缺点：影响CPU，没有充分利用内存

适用：高速I/O设备的块传输

周期窃取法：

优点：充分利用CPU和内存，及时响应I/O请求

缺点：DMA每次都请求总线

适用：I/O周期大于存储周期

交替分时访问法：

优点：CPU未停止或等待，DMA不请求总线

缺点：CPU周期可能大于存储周期

配置机制：

单总线分离式：DMA使用[编程式I/O](#编程式IO)，所有模块共用总线，便宜但低效

单总线集合的DMA-I/O：DMA可能是I/O模块的一部分也可能是控制I/O 模块的单独模块；减少总线周期数

I/O总线式：多个I/O模块共享DMA，且易于扩展

I/O模块的演变：

CPU直接控制外设——编程式I/O——中断——DMA——I/O通道：I/O模块有内部的处理器和为I/O操作定制的指令集——I/O处理器：有一个局部存储器