计算机体系结构

机内代码及运算

数的进制

- 十进制
- 二进制
- 八进制
- 十六进制

进制转换

- 十进制转二进制：把被转换的十进制整数**反复地除以2**，**直到商为0**，所得的**余数(从末位读起)**就是这个数的二进制表示。简称“除2取余法” 。**注意商为0时，从末位读起**
- 二进制转十进制：二进制数按权展开求和

原码、反码、补码、移码

- 带符号数的表示：通常的做法是约定一个数的**最高位为符号位**，若该位为**0**，则**表示正数**；若该位为**1**，则**表示负数。**数值范围**-2^(n-1)~2^(n-1)-1**
- 原码：用最高位表示符号位，数值部分用二进制绝对值表示
- 反码：**正数的反码和其原码形式相同**，**负数**的反码是**除符号位**，其他各位**逐位取反(即0变1,1变为0)** 。
- 补码：**正数的补码和其原码形式相同**，**负数**的补码是原码**除符号位**以外**逐位取反(即0变1,1变为0)**，**最后在末尾加1**，既是反码+1。一个数经过两次补码，就会变成他的源码
- 移码（增码）：无论正数、负数，**在补码的基础上对符号位取反**，一般**用做浮点数的阶码**，引入的目的是为了保证浮点数的机器零为全0。

定点数和浮点数

- 计算机中，通常是用**定点数来表示整数和纯小数**，分别称为定点整数和定点小数。对于既有整数部分、又有小数部分的数，一般用浮点数表示。
- 定点数：
    1. 定点整数：小数点的位置固定在最低位的右边，不占位
    2. 定点小数：小数点的位置固定在符号位与最高数值位之间，表示一个纯小数
- 浮点数：N=M*R^e；M称为尾数，R称为基数，e为阶码（指数）；浮点数利用指数达到了浮动小数点的效果，从而灵活地表达更大范围的实数

校验码概述

- 编码体系：指一种编码方式中所有合法码字的集合
- 编码效率：合法码字占所有码字的比率就是编码效率
- 码距：一个编码系统中任意两个合法的编码之间的不同的二进制位的数目叫这两个码字的码距（如从A变到B那么ASCII码就需要更改1位，那么码距就是1，A变到C的码距就是2）
- 该编码系统的任意两个编码之间的距离的最小值
    1. 码距是衡量一种编码方式的抗错误能力的一个指标
    2. 码距越大抗错误能力越强

- 误码：数字信息在传输和存取的过程中，由于各种意外情况的发生，数据可能会发生错误

奇偶校验

- 奇偶校验较简单，**串口通信中使用**奇偶校验作为数据校验的方法
- **奇校验**：被传输的有效数据中“1”的个数是奇数个，校验位填“0”，否则填“1”**（采用奇校验，奇数个1，校验位为0，偶数个1，校验位为1）**
- **偶校验**：被传输的有效数据中“1”的个数是偶数个，校验位填“0”，否则填“1”**（采用偶校验，偶数个1，校验位为0，奇数个1，校验位为1）**
- 使用一位奇偶校验的方法能够检测出一位错误，但无法判断是哪一位出错**（可以检错，但是不可以纠错）**
- 当发生两位同时出错的情况时，奇偶校验也无法检测出来。所以奇偶校验常用于对少量数据的校验，如对一个字节的校验

海明码

- 海明码是奇偶校验的一种扩充，通过对奇偶校验的嵌套达到**纠错+检错**的目的
- 海明码采用多位校验码的方式，在这些多个校验位中的每一位都对不同的信息数据位进行奇偶校验，通过合理地安排每个校验位对原始数据进行的校验的位组合，可以达到发现错误、纠正错误的目的**(当出现两位错误时，海明码能够查错，但无法纠错)**
- 可查出多少位错误：可以发现“ ≤码距-1” 位的错误
- 可以纠正多少位错误：可以纠正“ <码距/2” 位的错误，因此如果要能够纠正n位错误，所需最小的码距应该是“ 2n+1"
- 海明码的原理：在数据中间加入几个校验码，码距均匀拉大，当某一位出错，会引起几个校验位的值发生变化
- 海明不等式：校验码个数为k，可以表示2^k个信息，1个信息用来表示“没有错误”，其余2^k -1个表示数据中存在错误，如果满足2^k - 1≥m+k （ m+k为编码后的数编总长度），则在理论上k个校验码就可以判断是哪一位（包括信息码和校验码）出现了问题
- **2^k - 1≥m+k （ m+k为编码后的数编总长度）**
- 海明码的编码规则：**校验位依次放在第2^i(i=0,1,2,3...)位**，其余位置为信息位。

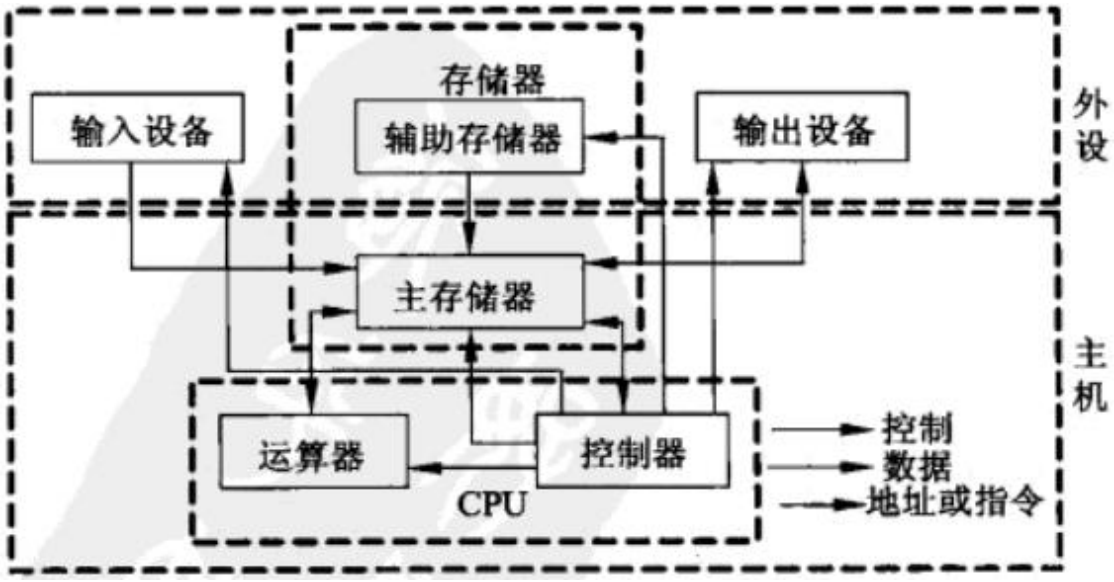
循环冗余校验码

- 广泛地在网络通信及磁盘存储时采用
- 生成多项式：在循环冗余校验(CRC)码中，无一例外地要提到多项式的概念。一个二进制数可以以一个多项式来表示。如1011表示为多项式X^3+X^1+X^0，如果把这里的x替换为2，这个多项式的值就是该数的值。从这个转换可以看出多项式最高幂次为n，则转换为二进制数有n+1位
- 编码组成：编码的组成是由K位信息码，加上R位的校验码（信息位后加校验位）
- 校验码的生成：
  1. 将K位数据C(x)左移R位，给校验位留下空间，得到移位后的多项式为C(x) × X^R
  2. 将这移位后的信息多项式除以生成多项式，得到R位的余数多项式
  3. 将余数作为校验码嵌入信息位左移后的空间
- 循环冗余校验码的纠错能力取决于K值和R值。在实践中，K值往往取得非常大，远远大于R的值，提高了编码效率。在这种情况下，循环冗余校验就只能检错不能纠错**（k值是信息位的个数；R值是校验位个数）**
- R位生成多项式可检测出所有双错、奇数位错和突发错位小于或等于R的突发错误

  什么是双错

中央处理器

计算机的组成

- 计算机硬件由五大部件构成：控制器、运算器、存储器、输入设备和输出设备
- 

- 运算器：也称算术逻辑单元(ALU)，对数据进行算术运算和逻辑运算
    1. 加法器（累加器）：专门**存放**算术或逻辑运算的**操作数**和**运算结果**的寄存器
    2. 程序状态寄存器（PSW）：用来存放两类信息：一类是**体现当前指令执行结果的各种状态信息**，如有无进位（CY位），有无溢出（OV位），结果正负（SF位），结果是否为零（ZF位），奇偶标志位（P位）等；另一类是**存放控制信息**，如允许中断(IF位)，跟踪标志（TF位）等
- 控制器：是分析和执行指令的部件
    1. 指令寄存器：保存现在执行的指令
    2. 指令译码器：分析操作码、指令的作用，完成怎么做的任务
    3. 程序计数器：存放下一条指令的地址
    4. 定时与控制电路
    5. 堆栈和堆栈指针

计算机的分类

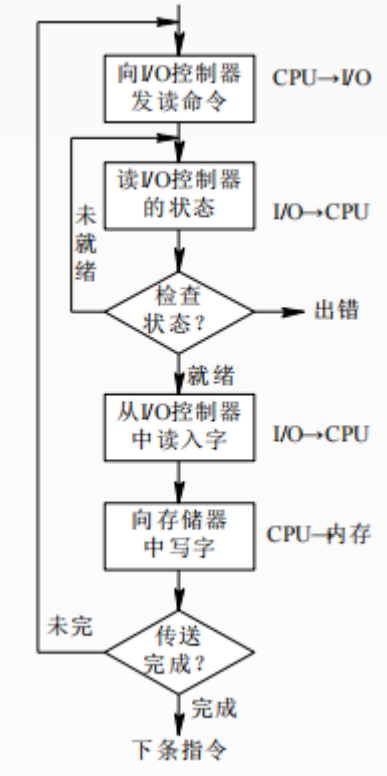
- 计算机体系结构是计算机系统的概念性结构和功能特性；常见的分类法包括**Flynn**、**冯氏分类法**两种
- Flynn分类法：根据指令流、数据流和多倍性三方面进行分类
    - 单指令流单数据流
        - 控制器：一个
        - 处理器：一个
        - 主存模块：一个
    - **单指令流多数据流**：异步执行
        - 控制器：一个
        - 处理器：多个
        - 主存模块：多个
    - 多指令流单数据流：不可能实现
        - 控制器：多个
        - 处理器：一个
        - 主存储模块：多个
    - 多指令流多数据流：并行执行
        - 控制器：多个
        - 处理器：多个

    - 主存模块多个
-

体系结构类型	结　构	关键特性	代　表
单指令流单数据流 SISD	控制部分：一个 处 理 器：一个 主存模块：一个		单处理器系统
单指令流多数据流 SIMD	控制部分：一个 处 理 器：多个 主存模块：多个	各处理器以异步的形式执行同一条指令	并行处理机 阵列处理机 超级向量处理机
多指令流单数据流 MISD	控制部分：多个 处 理 器：多个 主存模块：多个	被证明不可能，至少是不实际	目前没有，有文献称流水线计算机为此类
多指令流多数据流 MIMD	控制部分：多个 处 理 器：多个 主存模块：多个	能够实现作业、任务、指令等各级全面并行	多处理机系统 多计算机

输入/输出控制方式

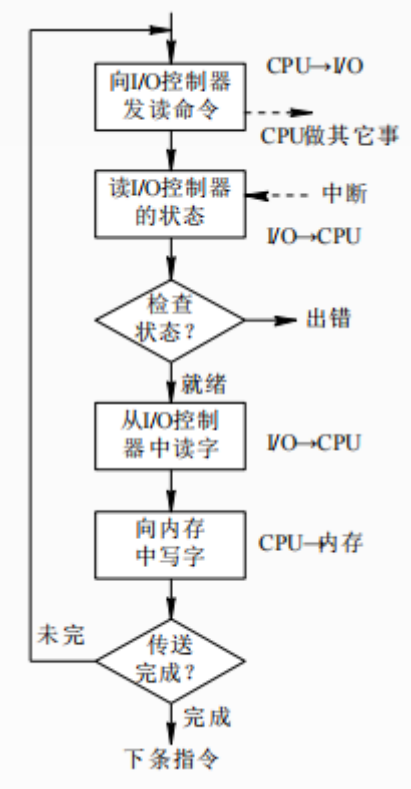
程序I/O方式：数据先放在控制器的数据存储寄存器中再由处理机将寄存器中的数据放入内存，一次只完成一个字的I/O操作

- 存在于**无中断机构**，处理机对I/O设备的控制采取程序I/O方式，或称为忙--等待方式，即在处理机向控制器发送一条I/O指令启动输入设备输入数据时，要同时把状态寄存器中的忙/闲标志置为1。然后便不断测试标志。当为1时，表示输入机尚未输完一个字，处理机应继续对该标志测试，直到它为0，表明**数据已输入到控制器的数据寄存器中**，于是**处理机将数据取出送入内存单元**，便完成了一个字的I/O 。
- 在程序I/O方式中，由于CPU高速,I/O设备低速致使CPU极大浪费
- 

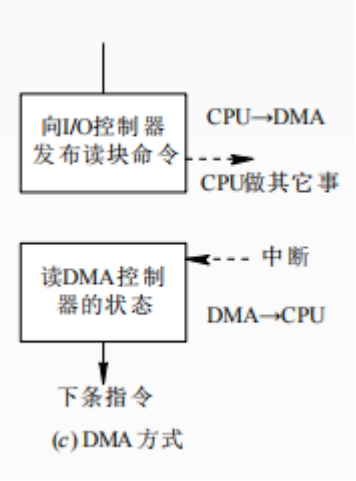
中断控股I/O方式：存在于慢速的设备上，如：打印机的控制方式；单次只能完成一个字的I/O操作；设备管理器与处理机是并行执行的

- 当某进程要启动某个I/O设备时，便由CPU向相应的设备控制器发出一条I/O命令，然后立即返回继续执行原来的任务。设备控制器于是按照命令的要求去控制指定I/O设备。这时CPU与I/O设备并行

操作

- 中断驱动方式在I/O设备输入数据的过程中，无需CPU干预，而是当I/O设备准备就绪时“主动”通知CPU。才需 CPU花费极短的时间去进行中断处理。从而大大地提高了整个系统的资源利用率及吞吐量，特别是CPU的利用率。但每中断一次仅能传输一个字（节）
- 

直接存储器访问（DMA I/O）控制方式：单次进行一个块的I/O操作，处理机与存储控制器是并行执行的，存储控制器直接将外存数据直接存入内存中；现在电脑的控制方式；

- 虽然中断方式比程序I/O方式更有效，但它仍是以字（节）为单位进行I/O的，每当完成一个字（节）的I/O时，控制器便要请求一次中断。极其低效的。因此引入了直接存储器访问方式。该方式的特点是：数据传输的基本单位是数据块；所传送的数据是从设备直接送入内存的，或者相反；仅在传送一个或多个数据块的开始和结束时，才需CPU干预，整块数据的传送是在控制器的控制下完成的。可见DMA方式又是成百倍的减少了CPU对I/O的干预，进一步提高了CPU与I/O设备的并行操作程度
- 

I/O通道控制方式：在超级计算机中使用；为CPU提供协处理器

- I/O通道是一种特殊的处理机。它具有执行I/O指令的能力，并通过执行通道(I/O)程序来控制I/O操作。但I/O通道又与一般的处理机不同，一是其指令类型单一，只能执行I/O操作有关的指令；二是通道没有自己的内存，与CPU共享内存
- 三类通道：
    1. 字节多路通道（Byte Multiplexor Channel）
    2. 数组选择通道（Block Selector Channel）
    3. 数组多路通道（Block Multiplexor Channel）

流水线技术

- 流水线技术是指在程序执行时，多条指令重叠进行操作的一种任务分解技术。把一个任务分解为若干顺序执行的子任务，不同的子任务由不同的执行机构来负责执行，而这些执行机构可以同时并行工作

计算执行时间

- 假定有某种类型的任务，可分成N个子任务，每个子任务需要时间t，则完成该任务所需的时间为N×t
    1. 若以**传统的方式**，完成k个任务所需的时间是**kNt**
    2. **使用流水线**技术，花费的时间是**Nt+(k-1)t**
- 注意，如果每个子任务所需的时间不同，其**时间取决于执行顺序中最慢的那一个子任务**
- **流水线的吞吐率**：指在单位时间内流水线所完成的任务数量或输出的结果数量
    - **tp=n/tk：tp为吞吐率、n为任务数、tk为完成任务所用的时间**
- **加速比**：用来衡量并行系统或程序并行化的性能和效果
    - **加速比=不采用流水线的执行时间 / 采用流水线的执行时间**

影响流水线的主要因素

1. 转移指令：因为前面的转移指令还没有完成，流水线无法确定下一条指令的地址，因此也就无法向流水线中添加这条指令；**比如中断或者是goto语句**
2. 共享资源访问的冲突：后一条指令需要使用的数据，与前一条指令发生冲突，或者相邻的指令使用了相同的寄存器；比**如两个任务同时访问/使用同一个寄存器**
3. 响应中断：当有中断请求时，流水线也会停止。对于这种情况有两种响应方式
    - 精确断点法：**立即停止**，这种方法能够立即**响应中断**
    - 不精确断点法：流水线中的指令**继续执行**，**不再新增指令到流水线**

精简指令计算机

指令系统

- 指计算机所能执行的全部指令的集合，它描述了计算机内全部的控制信息和“逻辑判断”能力
- 指令包括：操作码、地址码

地址码代表的地址类型

- 立即寻址：操作数直接是值
- 直接寻址：操作数的值是存放值得地址
- 间接寻址：操作数的值是存放地址的地址，需要进行两次的寻址才能找到值
- 寄存器寻址：寄存器的内容就是
- 寄存器间接寻址：寄存器的内容是存放值得地址

RISC与CISC

- CISC：为提高操作系统的效率，人们最初选择向指令系统中添加更多、更复杂的指令来实现，导致指令集越来越大。这种类型的计算机，称为复杂指令集计算机(CISC)
    - 主要特点：
        1. **指令数量多**：指令系统拥有大量的指令，有100 -250条
        2. **指令使用频率相差悬殊**：最常使用的是一些比较简单的指令，80%的时候使用的是20%指令
        3. **支持很多种寻址方式**：通常为5-20种
        4. **变长的指令**：指令长度不是固定的，变长的指令增加指令**译码电路的复杂性**
        5. 指令可以对存储器单元中数据直接进行处理：典型的CISC处理器通常都有指令能够**直接对内存单元中的数据进行处理**，其执行速度较慢
- RISC：对指令数目和寻址方式做精简，指令的指令周期相同，采用流水线技术，指令并行执行程度好，这类是精简指令集计算机(RISC)
    - 主要特点：
        1. **指令数量少**：优先选取使用频率最高的一些简单指令以及一些常用指令，避免使用复杂指令
        2. **指令的寻址方式少**：通常**只支持寄存器寻址方式**、**立即数寻址**方式以及**相对寻址**方式
        3. **指令长度固定，指令格式种类少**：因为RISC指令数量少，格式相对简单，其指令长度固定，指令之间各字段的划分比较一致，译码相对容易
        4. **只提供了Load/Store指令访问存储器**：只提供了从存储器读数Load和把数据写入存储器Store两条指令，其余所有的操作都在CPU的寄存器间进行
        5. **以硬布线逻辑控制为主**：为了**提高操作的执行速**度，通常采用硬布线逻辑来构建控制器
        6. **单周期指令执行**：因为简化了指令系统，很**容易利用流水线技术**使得大部分指令在一个机器周期内完成
        7. **优化的编译器**：RISC精简指令集使编译工作简单化
- 表 1-1　CISC 和 RISC 的简单对比

	CISC	RISC
指令条数	多	只选取最常见的指令
指令复杂度	高	低
指令长度	变化	短、固定
指令执行周期	随指令变化大	大多在一个机器同期完成
指令格式	复杂	简单
寻址方式	多	极少
涉及访问主存指令	多	极小，大部分只有两条存指令
通用寄存器数量	一般	大量
译码方式	微程序控制	硬件电路
对编译系统要求	低	高