# 计算机组成原理

每个二进制数位为1bit。

## 计算机系统发展

**计算机系统=硬件（计算机的实体，如主机/外设等）+软件（具有各类特殊功能的程序组成）**

**计算机性能的好坏取决于“软”，“硬”件功能的总和。**

软件：1.系统软件：操作系统、DBMS、标准程序库、网络软件、语言处理程序、服务程序等，用来管理整个计算机系统

2.应用软件：按任务需要编制成的各种程序。

第一台电子数字计算机：ENIAC（1946）冯诺依曼 逻辑元件：电子管（1.8w） 速度5000次加法/s

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 发展阶段 | 时间 | 逻辑元件 | 速度（次/s） | 内存 | 外存 |  |
| 第一代 | 1946-1957 | 电子管 | 几千-几万 | 汞延迟线、磁鼓 | 穿孔卡片、纸带 |  |
| 第二代 | 1958-1964 | 晶体管（贝尔实验室） | 几万-几十万 | 磁芯存储器 | 磁带 | FORTRAN、  操作系统雏形 |
| 第三代 | 1964-1971 | 中小规模集成电路 | 几十万-几百万 | 半导体存储器 | 磁带、磁盘 | 用于科学计算、有了分时操作系统 |
| 第四代 | 1972-现在 | 大规模、超大规模集成电路 | 上千万-万亿 | 半导体存储器 | 擦盘、磁带、光盘、半导体存储器 | 微处理器、微型计算机 |

### 硬件发展

1947年，贝尔实验室，发明了晶体管。

1955，肖克利在硅谷创建肖克利实验室谷粉有限公司。

1957年，其他人（从肖克利实验室出来的）创立了仙童半导体公司。

1959年，仙童半导体公司发明了“集成电路”。

1968年，摩尔等人离开仙童，创立Intel。

1969年，仙童销售部负责人桑德斯离开仙童，创立AMD。

**摩尔定律：集成电路上可容纳的晶体管数目，约每隔18个月便会增加一倍，整体性能也将提升一倍**。

### 软件发展

机器语言

汇编语言

高级语言：FORTRAN、PASCAL、C++

DOS

Windows

IOS

两级分化：微型计算机向更微型化、网络化、高性能、多用途

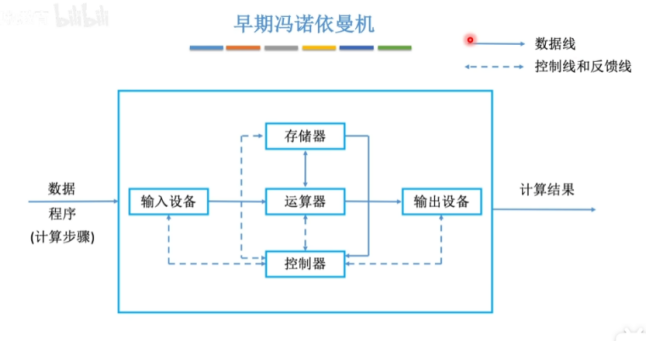
巨型机向更巨型化、超高速、并行处理、智能化

# 计算机硬件的基本组成

冯诺依曼结构：早期冯诺依曼机：ENIAC（手动接线控制计算）

冯诺依曼提出：**存储程序：将指令1以二进制代码的形式实现输入计算机的主存储器，然后按其在存储器中的收地址执行程序的第一条指令，以后就按该程序的规定顺序执行其他指令，直至程序执行结束。**

第一台采用冯诺依曼结构的计算机EDVAC（electronic discrete variable automatic Computer）



输入设备：将信息转换成机器能识别的形式

存储器：存放数据和程序

运算器：算术运算和逻辑运算

输出设备：将结果转换成人们熟悉的形式

控制器：指挥程序运行

**在计算机系统中，软件和硬件在逻辑上是等效的。**

**特点：1.计算机由五大部件组成。**

**2.指令和数据以同等地位存于存储器，可按地址寻访**

**3.指令和数据用二进制表示**

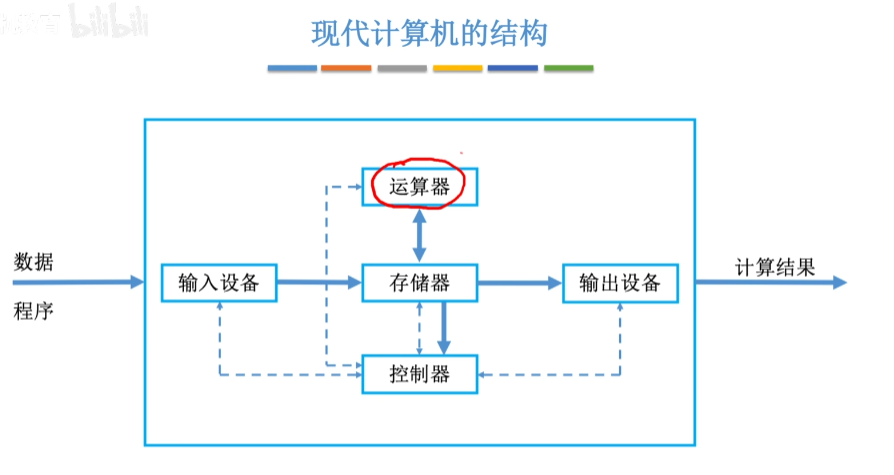
**4.指令由操作码和地址码组成**

**5.存储程序**

**6.以运算器为中心（输入输出设备与存储器之间的数据传送通过运算器完成）**

现代计算机结构：以存储器为中心

CPU=运算器+控制器

****

硬件：

主机：CPU 运算器、控制器

主存 存储器

IO设备：辅存 存储器

输入设备

输出设备

### 硬件工作原理

主存储器：存储体、MAR（Memory Address Register）、MDR（Memory Data Register）

数据在存储体中按地址存储，每个地址对应一个存储单元

存储单元：每个存储单元存放一串二进制代码

存储字：存储单元中二进制代码的组合

存储字长：存储单元中二进制代码的位数

存储元：即存储二进制的电子元件，每个存储元可存1bit

MAR位数反应存储单元的个数，MDR位数=存储字长

1个字（word）=16bit 一个字节（Byte）=8bit

运算器：ACC（累加器，用于存放操作数或运算结果）、MQ（乘商寄存器）、ALU（算术逻辑单元，复杂电路实现算术运算、逻辑运算）、X（操作数寄存器、通用寄存器）

控制器：CU（控制单元，分析指令，给出控制信号）、IR（指令寄存器，存放当前执行的指令）、PC（程序计数器，存放下一条指令地址，有自动加1功能）

（取指）取指令：PC

PC+1

（取指）分析指令：IR

执行指令：CU

CPU区分指令和数据的一句：指令周期的不同阶段。

### 计算机系统的多级层次结构

虚拟机器（高级语言机器）：用编译程序翻译成汇编语言程序

虚拟机器（汇编语言机器）：用汇编程序翻译成机器语言程序

虚拟机器（操作系统机器）：向上提供“广义指令”（系统调用）

传统机器（用机器语言的机器）：执行二进制机器指令

微程序机器（微程序指令系统）：由硬件直接执行微指令

三种级别的语言：高级语言：C/C++

汇编语言：助记符

机器语言：二进制代码

高级语言能够被解释程序（解释器）解释或直接被编译程序（编译器）编译

编译语言：C、C++

解释语言：Python

编译程序：将高级语言编写的源程序全部语句一次全部翻译成机器语言程序，而后在执行机器语言程序（只需翻译一次）

解释程序：将源程序的一条语句翻译成对应于机器语言的语句，并立即执行。紧接着在翻译下一句（每次执行都需要翻译）

### 计算机性能指标

存储器的性能指标

总容量=存储单元个数\*存储字长 bit

总容量=2^MAR位数\*MDR位数 bit =同/8 B

K=2^10

CPU的性能指标

CPU主频：CPU内数字脉冲信号振荡的频率

时钟周期=1/CPU主频

CPI：执行一条指令所需要的时钟周期

执行一条指令的耗时=CPI\*CPU时钟周期

IPS（Instructions Per Second）：每秒执行多少条指令 =主频/平均CPI

FLOPS（Floating-point Operations Per Second）：每秒执行多少次浮点运算

K=10^3

系统整体的性能指标

数据通路带宽：数据总线一次所能并行传送信息的位数

吞吐量：单位时间内处理请求的数量（主要取决于主存的存取周期）

响应时间：从用户向计算机发送一个请求，到系统对该请求做出响应并获得它所需要的结果的等待时间。

基准程序：用来测量计算机处理速度的一种使用程序，以便于被测量的计算机性能可以与运行相同程序的其他计算机性能进行比较。

CPU主频、CPI、指令系统等都会影响CPU性能。

基准程序也只是在特定情况下性能表现优异。

## 数据的表示和运算

### 进位计数制

基数：每个数码位所用到的不同符号的个数，r进制的基数为r

二进制B、八进制、16进制H、十进制D转换

整数部分/基数 取余

小数部分\*基数 取余

真值：符合人类习惯的数字

机器数：数字实际存到机器里的形式，正负号需要被“数字化”

### BCD码

用4位2进制表示1位十进制数

8421码

正常对应

Ex：5+8=13

0101+1000=？

对应位相加=1101落在映射外+6即+0110=10011化为4位形式=0001 0110=1 3

余3码

8421码+（0011）B

2421码

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0000 | 0001 | 0010 | 0011 | 0100 | 1011 | 1100 | 1101 | 1110 | 1111 |

### 无符号整数的表示和运算

无符号整数：全部二进制位都是数值位，没有符号位，第i位的位权是2^i-1

n bit的无符号整数表示范围为0-2^n-1，超出则溢出

可以表示的最小的数全0，可以表示的最大的数全1

加法运算：从最低位开始，按位相加，并往更高位进位

减法运算：被减数不变，减数全部位取反、末位+1

从最低位开始，按位相加，并往更高位进位

### 带符号整数的表示和运算（原/反/补）

原码表示：符号位：0正1负

原码缺点：符号位不能参与运算，需要设计复杂的硬件电路才能处理。

补码表示真值：符号位可以参加运算。

反码：正数：反码不变，补码不变

负数：反码：符号位不变，数值位取反；补码：末位+1

补码<->真值：从右往左找到第一个1，这个1左边的所有数值位按位取反

补码加法：按位相加（符号位参与运算）

补码减法：相反数补码：全部位按位取反、末位+1；然后按照加法操作

### （原/反/补）码特性



### 带符号整数移码表示

移码：对补码符号位取反（只能用于表示整数）

### 定点小数表示和运算

正反补和定点整数一摸一样。

区别：位数扩展时，扩展位置不同



### 奇偶校验码

奇校验码：1的个数为奇数

偶校验码：1的个数为偶数

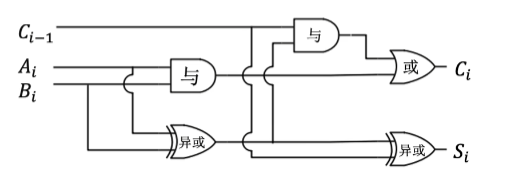
保证有奇数/偶数个1

偶校验硬件实现：使用异或（模2+）运算获得校验位

### 算术逻辑单元

ALU用于算术运算、逻辑运算、辅助功能（移位、求补）

一位全加器



串行加法器：只有一个一位全加器

串行进位的并行加法器：串行进位又称为行波进位

### 并行进位加法器

并行进位的并行加法器：各级进位信号同时形成，又称为先行进位、同时进位

### 补码加减运算器

补码减法：相反数补码：全部位按位取反、末位+1；然后按照加法操作

最高位进位后会发生溢出

无符号数加减运算与补码相同

### 标志位生成

OF溢出标志：有符号数才有意义，OF=1说明发生溢出，OF=最高位产生进位异或此高位产生进位

SF符号标志：SF=最高位的本位和

ZF零标志：ZF=1，运算结果为0

CF进位/借位标志：CF=最高位产生的进位异或SUB（加减法标志），只对无符号数有意义，CF=1时表示发生了无符号数加减法溢出

### 定点数移位运算

算术移位：改变各数码位的位权

右移：/ ，低位舍弃，若舍弃的位！=0则会丢失精度。

左移：\*，高位舍弃，若舍弃的位！=0则会出现严重误差。

原码的算术移位：符号位保持不变，仅对数值位进行移位，补0

反码的算术移位：正数相同，反码的负数需要补1

补码的算术移位：正数相同，负数补码：右移：高位补1，低位舍弃；左移：低位补0，高位舍弃

逻辑右移：补0舍弃

循环移位、带进位的循环左移

### 原码乘法

符号位异或，数值为取绝对值进行乘法计算，利用加法和移位

移位：逻辑右移

ACC高位积，MQ低位积和乘数

### 补码乘法

符号位参与运算

辅助位MQ的右1位：

辅助位-MQ中最低位=1:ACC+【x】补

辅助位-MQ中最低位=0:ACC+0

辅助位-MQ中最低位=-1:ACC+【-x】补

右移：算术右移

最后还要进行一次加法，不进行移位

### 原码除法

正常除数，5位字长即可，符号位单独计算，取数值位的绝对值计算，逻辑左移

恢复余数法：负数则恢复（加上除数即可）

加减交替法：若余数为负，商0，左移1位加上除数，得到下一个新余数，若正则商为1，若负则商0

余数正负性和商相同，若剩下余数为负需商0，并加上除数，得到正确余数

### 补码除法

符号位参与运算，双符号位被除数和除数同号，则减法，若异号则加法

余数和除数同号，商1，执行减法；余数和除数异号，商0，执行加法，末位商恒置1（精度不超过2-n）

### C语言中强制类型转换

定点整数使用“补码”存储的

长整型变短整数，高位截断保留低位

短整型变长整型，符号扩展

### 数据的存储和排列

大小端模式

大端：高地址存储低位

小端：低地址存储低位

边界对齐：一个字一个字存储，不分割数据

### 浮点数表示

阶码：阶符+阶码数值；反映浮点数的表示范围及小数点实际位置

尾数：数符+尾数数值；反映浮点数的精度

尾数的最高位时无效值会丧失精度

规格化浮点数：规定尾数的最高数值位必须是一个有效值

左规：将尾数算数左移一位，阶码-1

右规：将尾数算数右移一位（双符号位为01或10），阶码+1

规格化浮点数的原码范围（最高位必须为1）正数：1/2<=M<=1-2-n

负数：2-n-1<=M<=-1/2

补码正数：相同

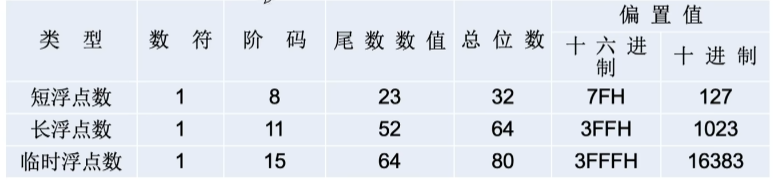
负数：最高数值位为0，-1<=M<=-（1/2+2-n）

### IEEE754

移码=真值+偏置值

尾数使用原码表示，隐藏表示最高位为1

偏置值为127D



阶码全0，全1用作特殊处理，因此-126<=阶码<=127

最小绝对值1.0B\*2-126

最大绝对值1.111···B\*2127

阶码全0，尾数不全为0时，表示非规格化小数+（0.XXXXX）B\*2-126

阶码全0，尾数全为0时，表示+0

阶码全1，尾数全为0时，表示无穷大

阶码全1，尾数不全为0时，表示非数值“NAN”（NOT A NUMBER）

### 浮点数运算

步骤：对阶、尾数加减、规格化、舍入、判断溢出

对阶：小阶向大阶靠齐

溢出：阶码超出

舍入：“0”舍“1”入、恒置“1”

强制类型转换

损失精度或溢出，自行判断