处理器+内存=计算机

计算机系统=硬件+软件

硬件：运算器+控制器=CPU，存储器（高速缓存 主存储器 虚拟存储器），输入输出设备 用总线和接口接起来就是计算机

运算器ALU：+-\*/等算术运算，逻辑运算

字长：运算器算一次最多能算的二进制位数

存储器：存信息=程序（由指令构成）+数据，二进制代码存储

存储元：保存一位二进制数0或者1的触发器

存储单元：若干个存储元构成一个存储单元

地址：存储单元的编号

存储容量：~~存储器所有储存单元的总数。要指明~~几个存储单元，每个存储单元有多少位二进制数

1B=8b

容量K=210  速度 K=103

M=220 M=106

G=230 G=109

例子：64KB=64\*210\*8位

控制器：（电脑的大脑）控制器是发号施令的部件

1. 取指令->分析指令->执行指令（循环执行）
2. 保证指令按规定序列自动连续的执行
3. 对各种异常情况和请求及时响应和处理

指令=操作码（干嘛）+地址码（从哪找）

控制器：取指令\_取指周期，执行指令\_执行周期

指令计数器(PC),一般+1，可以+多个

输入输出设备统称为外部设备，通过适配器与主机相连

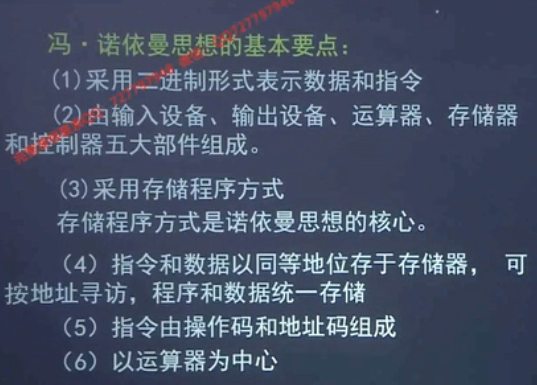
软件=系统软件+应用软件

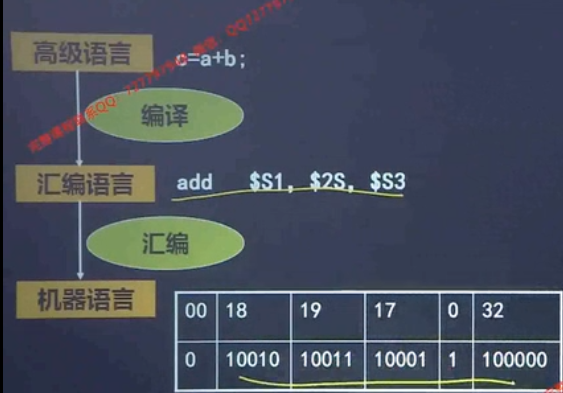
目的程序是机器语言书写的程序

汇编程序

源程序，与具体机器无关（如C语言）

冯.诺依曼核心思想：存储程序控制





计算机性能指标：

机器字长：一次二进制运算的位数。字长长了精度高

吞吐量：一台计算机在某一时间间隔能够处理的信息量，单位字节/s

响应时间：输入有效到系统产生相应之间的时间度量微秒（10-6s）纳秒（10-9）

利用率：实际使用时间/给定时间 的百分比

总线带宽：运算器与存储器之间互连的内部总线的二进制位数

存储器带宽：字节数/s

主频/时钟周期：主时钟不断产生固定频率的时钟，主时钟的频率f叫CPU的主存

主频的倒数称为CPU的时钟周期T Tf=1

CPU的执行时间：CPU的时钟周期数\*CPU的时钟周期长

CPI平均执行一条指令所需要的平均时钟周期数

CPI=执行某段程序所需的CPU时钟周期数/该程序包含的指令条数 （1/CPI是条/时间）

MISP：百万条指令/s MISP=指令条数/（执行时间\*106）

MFLOPS：百万次浮点操作/s MFLOPS=程序中的浮点操作次数/（程序执行时间\*106）

TFLOPS：亿万次浮点操作数/s T：1012

进制转换

定点数据的表示吗：原反补移

计算机内部信息=程序+数据

控制信息=指令+控制字

数据信息=数值型（定点数、浮点数）+非数值型（数字串、字符与字符串、汉字与汉字串、图像、音频与视频） ~~~全都是二进制的

常用的数据表示格式：定点格式和浮点格式

定点数值范围小，要求硬件简单

浮点数值范围大，要求硬件复杂

数据范围：最大值和最小值

数据精度：实数表示的有效数字的位数

N位无符号数，即N位数值位 0~2N-1

N位有符号数，即N-1位数值位；符号位1是负数，0是正数

纯小数：-(1-2-(N-1))~(1-2-(N-1)) 因为0.11...1=1-0.00...1

纯整数：-（2(N-1)-1）~（2(N-1)-1） 因为11...1=100...0-1但是是从20开始

BCD码就是光要0-9的二进制，也就是十进制加法器

原码就是数值的绝对值前面加上一个符号位【原码0占了俩编码，有+0和-0】

正数和负数转换，把第一位符号位取反

0的表示不唯一

原码字长n，有2n个编码，但对应的真值为2n-1

负数原码形式上看着比正数大

原码移位规则：符号位不变，左移相当于\*2，空位补0；右移相当于/2（十进制也是这样,左移\*10，右移/10） 应用：A\*7=A\*8-1也就是左移3位后减去原来a【注意移位不动符号位】

转二进制大法：【因为计算开，这样最快】

... 8 4 2 1 1/2 1/4 1/8 ...

阶码：值指明了[小数点](https://baike.baidu.com/item/%E5%B0%8F%E6%95%B0%E7%82%B9" \t "https://baike.baidu.com/item/%E9%98%B6%E7%A0%81/_blank)在数据中的位置

原码简单易懂，与真值转换容易，实现乘除法方便（数值位就是绝对值相乘除，符号位异或即可）

不足：0的两种表示；加减法复杂

补码（和时钟似的顺or逆->负数用补码表示，就把减法转化为加法）

表的时针 [-3]补=9=12+(-3)(mod 12)=>6-3=6+9

表的秒针 [-20]补=40=60+（-20）（mod 60）=>50-20=50+40

也就是-X相当于+[-X]补

[X]补=M+X（mod M）知道这个理了，正数的补码就是本身，负数的补码是模+它也是模-（它的绝对值）【所以，补码就是模加真值（真值：保留正负的原数）】

小数模是2，整数模是2n+1

补码0的表示唯一

纯小数中：补码能表示-1是借用原码那个-0来表示的，原码不能表示-1

整数中：补码能表示-2n是借用-0来表示的，原码不能表示-2n

整数原码样子和反码一样

负数符号位不变，各位取反，末尾+1

正的补码求负数的补码：含符号位各位取反+1，反之亦然

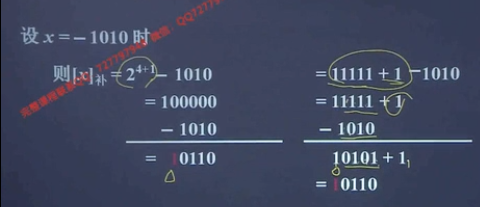
左移补0，右移补符号位

负数补码形式上大于正数的补码

补码拓展：

定点小数：最低位用0

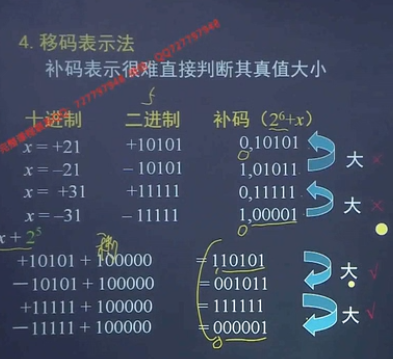
定点整数：最高位用符号位



1. 除符号各位取反后+1
2. 后往前，找第一个1，然后前面按位取反

反码：除符号位各位求反就是反码【补-1】

移码：正数用1，负数用0【补的符号位换了，也可以看成是从模指数少1】



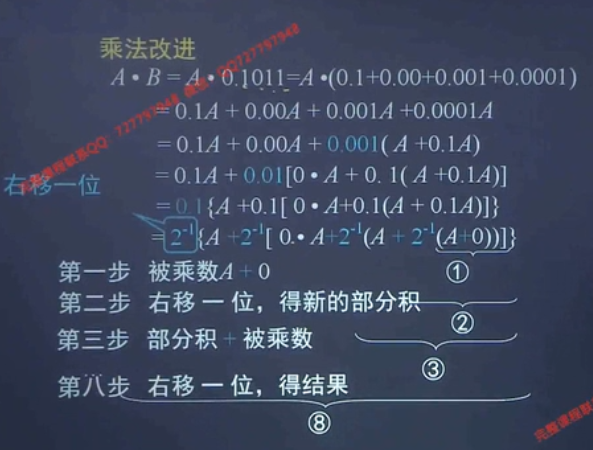
移码的实质把真值映像到一个正数域

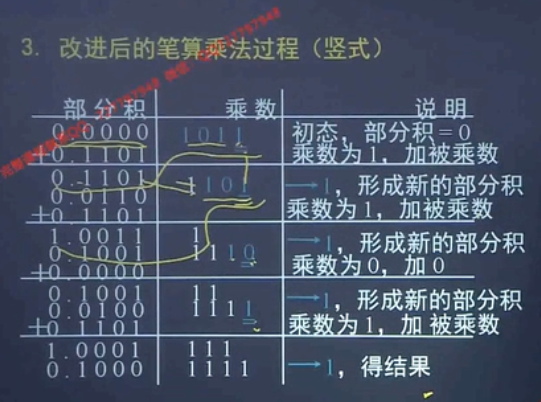
【补码加法进位就丢了，因为模呀！】

1. 同号相加变了号就是溢出了
2. 符号位双倍，符号位相加，不同则溢出，最开头是应该的
3. 符号位进位和数相加进位相同则不溢出

乘积的位数扩大一倍

加一次移一次【0就+0；1就+本身】





符号位是符号位异或；不想增加硬件负担：移位放到乘数的高位