

《程序是怎样跑起来的》（上）



c747190cc2f5 (/u/c747190cc2f5)

2019.05.01 21:49* 字数 2114 阅读 1 评论 0 喜欢 0
(/u/c747190cc2f5)

学习笔记

此书前言

无论任何事情，了解其本质非常重要。只有了解了本质才能提高利用效率。这样一来，即使有新技术出现，也能很容易的理解并掌握。

第1章 对程序员来说CPU是什么

本章提问

1. 程序是什么？
2. 程序是由什么组成的？
3. 什么是机器语言？
4. 正在运行的程序存储在什么位置？
5. 什么是内存地址？
6. 计算机的构成元件中，负责程序的解释和运行的是哪一个？

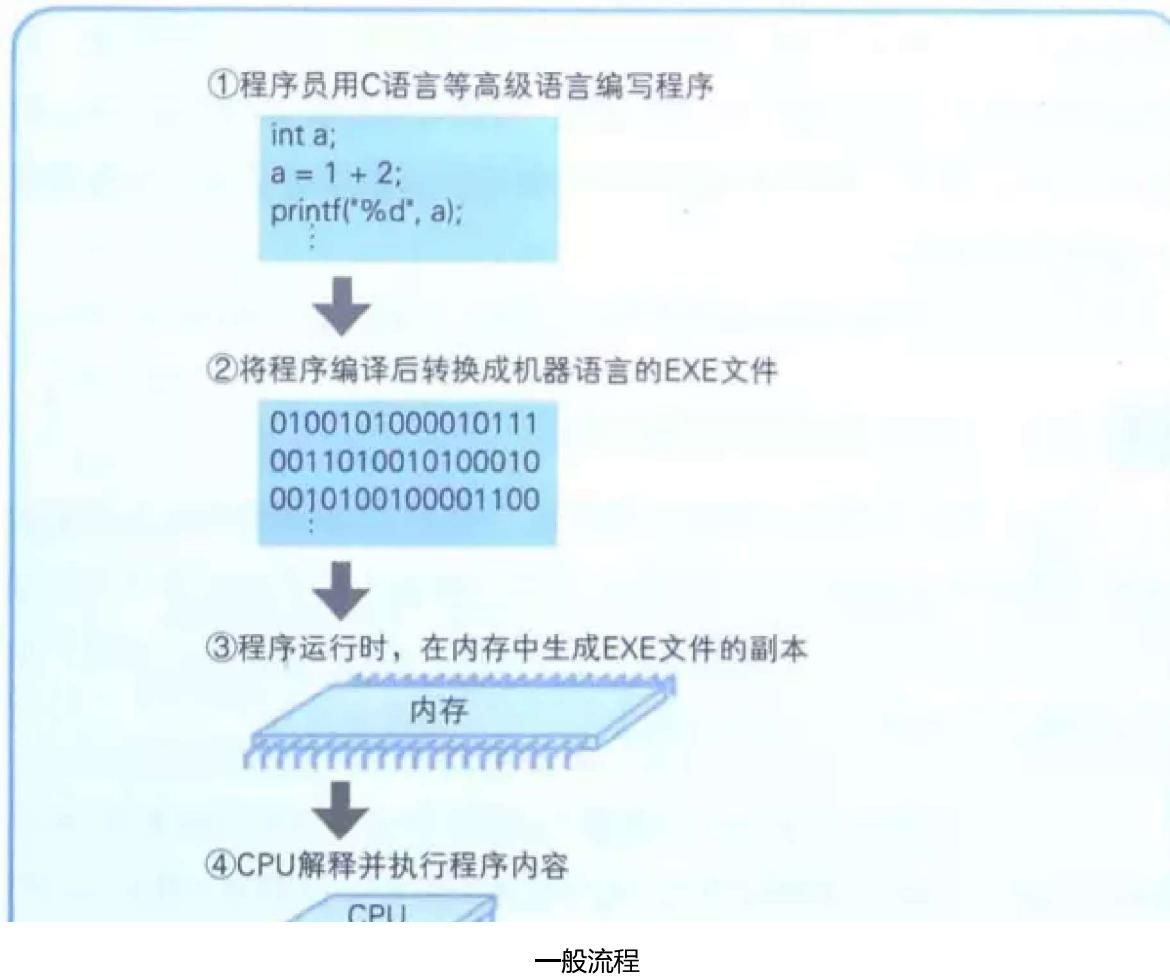
本章重点

重点要理解寄存器

1.1 CPU内部结构解析

一、先看程序运行的一般流程：





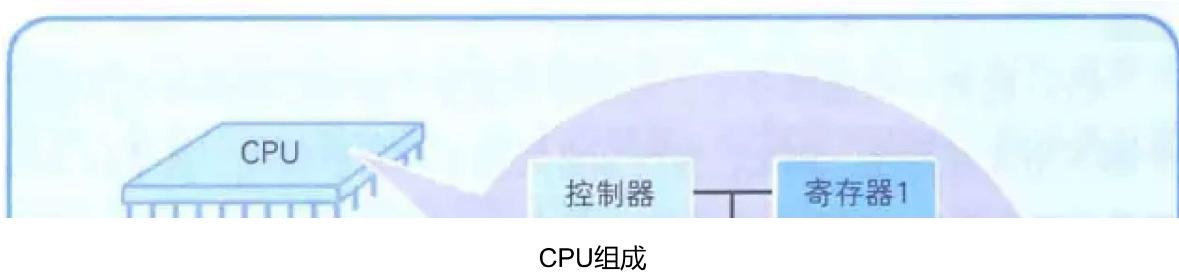
这个图可能只是在说编译型语言吧。

在程序运行过程中，CPU负责解释和运行最终转换成机器语言的程序内容。

二、CPU的组成

- CPU是由寄存器、控制器、运算器、时钟组成的；
- 寄存器用来暂时存储指令、数据等对象，一个CPU可能会有20~100个寄存器；
- 控制器负责把内存上的指令和数据读入寄存器，并根据运算结果来控制计算机；
- 运算器负责运算被读入寄存器的内容；
- 时钟负责发出CPU开始计时的信号。





三、内存

通常说的内存指计算机的主存储器，其中的数据会随关机而清除。

1.2 CPU是寄存器的集合体

程序员需要重点关注寄存器，其它的了解即可，因为 程序是把寄存器作为对象来描述的。

使用高级语言编译后会转化成机器语言，然后再通过CPU内部的寄存器来处理。

通过阅读汇编语言的代码，可以了解转化成机器语言的程序运行情况。

汇编语言为每一个机器语言指令添加一个助记符，汇编语言与机器语言是一一对应的。

将汇编语言转化为机器语言称为汇编，反之称为反汇编。

下面是一个汇编语言示例：

```
mov  eax, dword ptr [ebp-8]    …把数值从内存复制到 eax
add  eax, dword ptr [ebp-0Ch]  …eax 的数值和内存的数值相加
mov  dword ptr [ebp-4], eax   …把 eax 的数值（上一步的相加结果）存储在内存中
```

汇编语言示例

其中，add（相加）与mov（存储，move）都是助记符，eax和ebp都是寄存器。

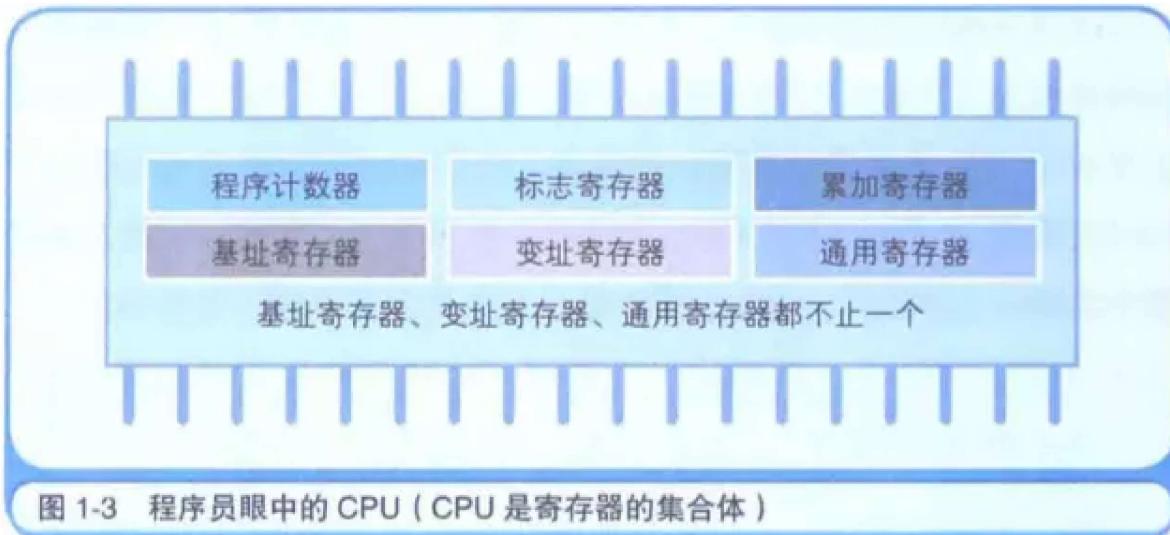
从示例中可以看出，机器语言级别的程序是通过 寄存器 来处理的。

我们可以将寄存器大致分为8类，用于运算的数值放在累加寄存器中存储，表示内存的数值放在基址寄存器和变址寄存器中存储，上面的eax和ebp分别是累加寄存器和基址寄存器。



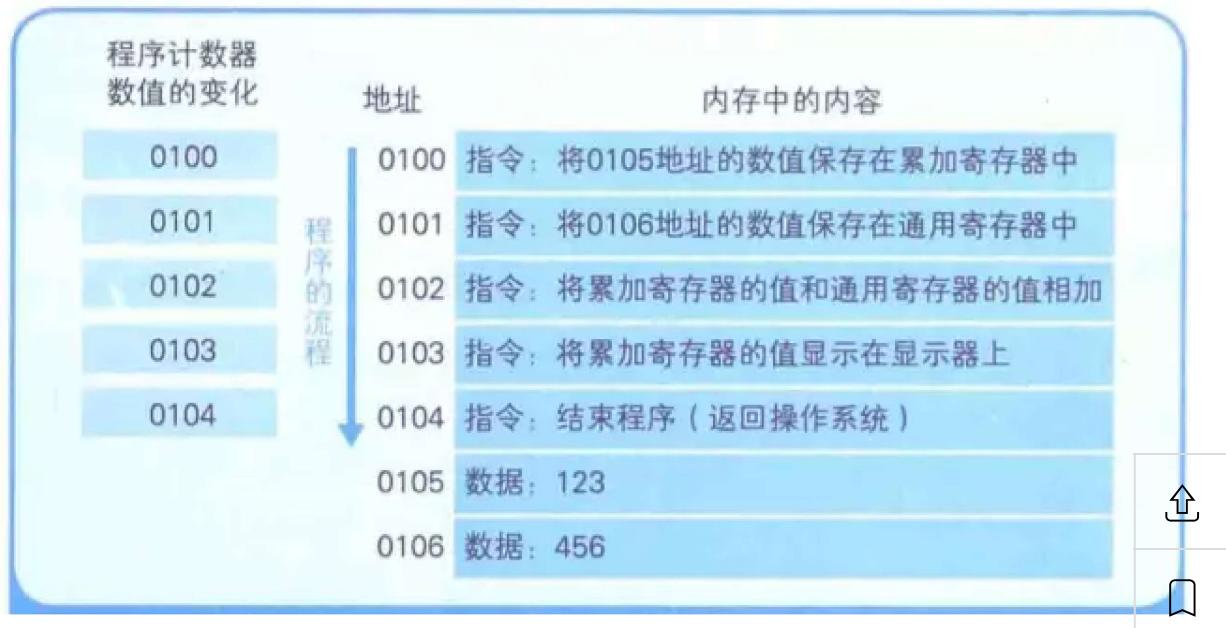
寄存器大致分类

对程序员来说，CPU就是具有各种功能的寄存器的集合体。



1.3 决定程序流程的程序计数器

下图中的流程计算 $123+456$:



一个命令或一个数据通常被存储在多个地址上，为了便于说明，图中将其都分配到了一个地址中。

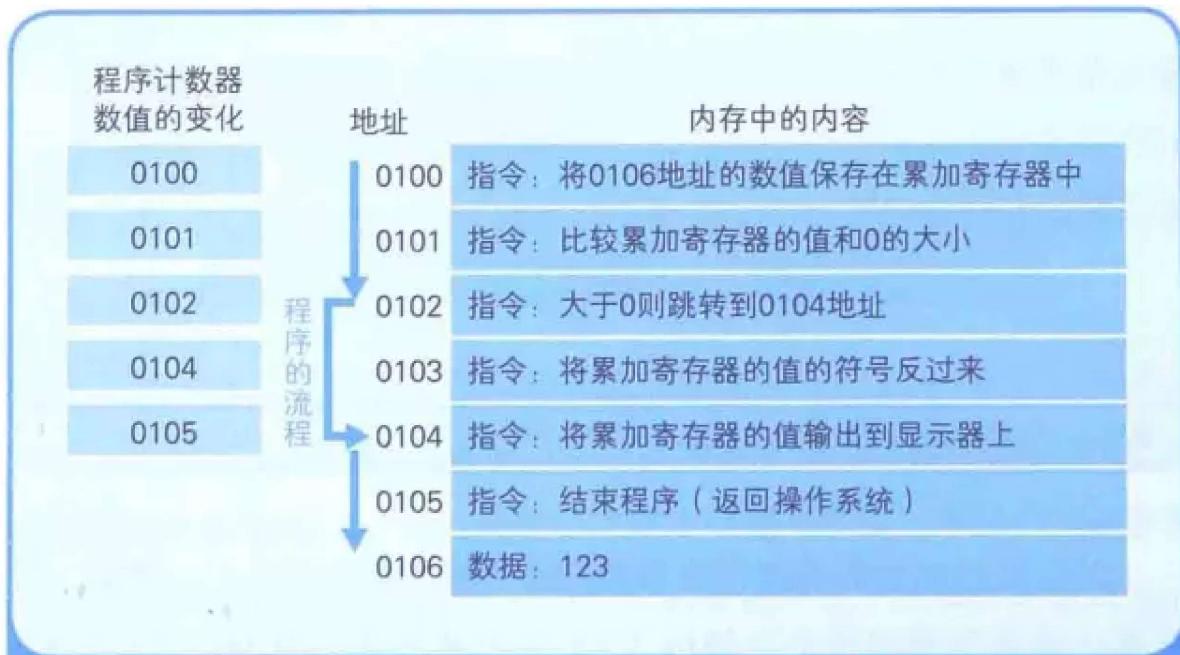
1. 操作系统把程序复制到内存；
2. 程序运行，没运行一步，程序计数器加一；
3. 控制器根据程序计数器的数值，从内存读取命令并执行。

总而言之，程序计数器决定程序的流程。

1.4 条件分支和循环机制

程序的流程分为 顺序、条件分支、循环 三种。

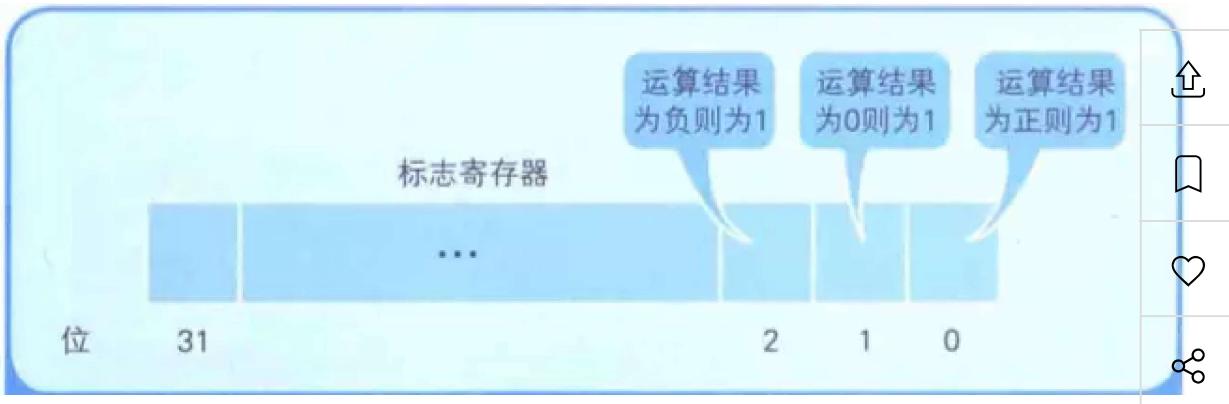
下面的图很清楚的讲述了条件分支时的程序计数器依靠跳转指令(jump)运作：



条件分支示例

关于标志寄存器：

标志寄存器负责保存运算结果的正、负、零，还有溢出、奇偶校验的结果。正、负、零三种状态保存寄存器的三个位，根据数据状态把相应位置为1。



标志寄存器的三个位

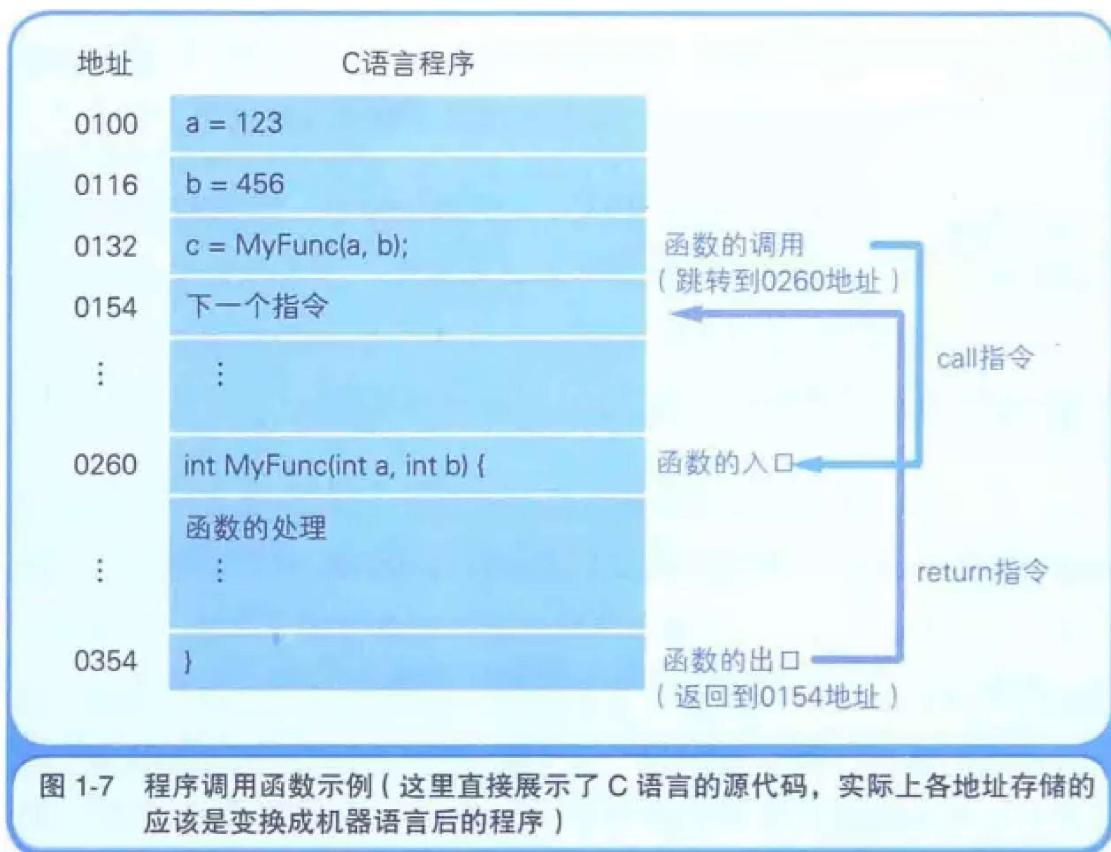
关于比较运算：

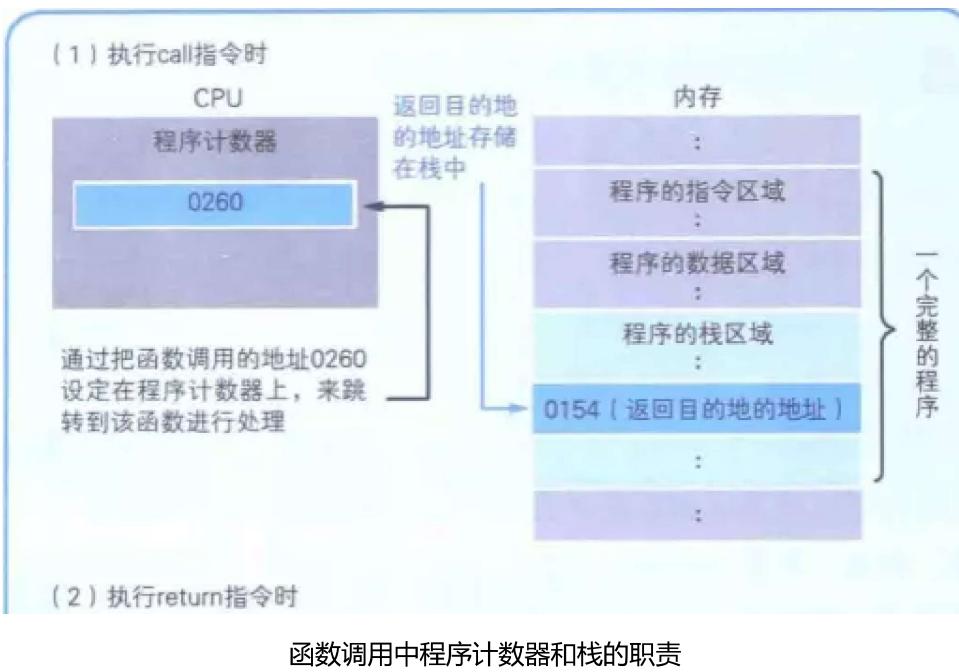
实际上是把要比较的两个数相减后把结果的状态保存到标志寄存器中。

1.5 函数的调用机制

依靠call、return指令和栈实现调用函数的下一行指令的返回。

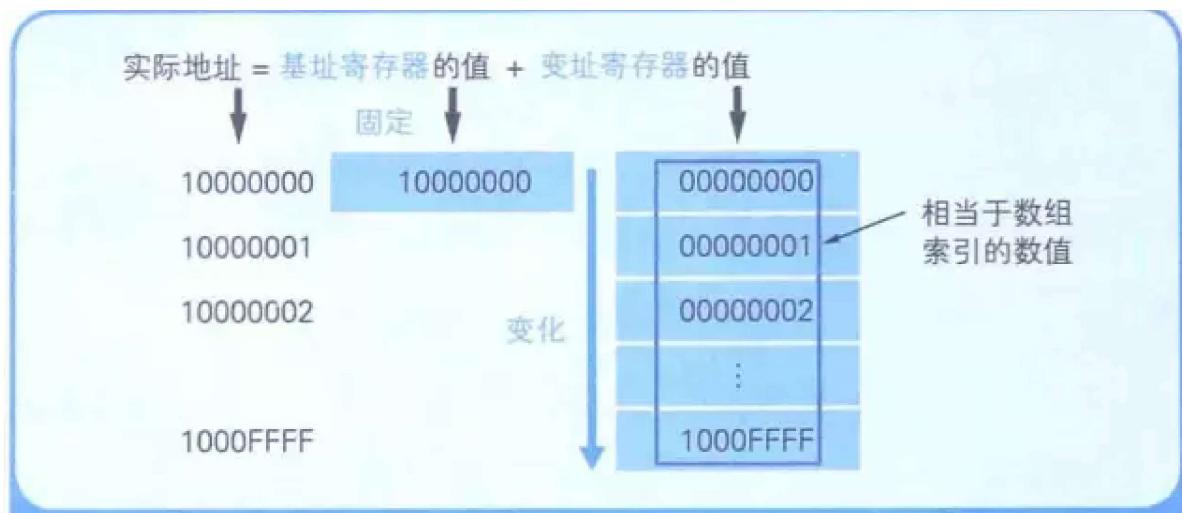
1. 调用函数时，call会把调用函数后要执行的指令存储在名为栈的主存内；
2. 函数执行完毕后，再用return把栈中的地址设定到程序计数器中实现返回。





1.6 通过地址和索引实现数组

实现数组要依靠基址寄存器和变址寄存器。基址寄存器不变，改变变址寄存器，变址寄存器的值就相当于数组的索引。



地址与索引

1.7 CPU的处理其实很简单

虽然程序很复杂，但是CPU执行的机器语言指令种类是比较少的，下图对指令进行了大体分类：



类 型	功 能
数据转送指令	寄存器和内存、内存和内存、寄存器和外围设备 ^① 之间的数据读写操作
运算指令	用累加寄存器执行算术运算、逻辑运算、比较运算和移位运算
跳转指令	实现条件分支、循环、强制跳转等
call/return 指令	函数的调用 / 返回调用前的地址

大体分类

问题答案：

1. 一般所说的程序，指的是行事的先后顺序，计算机程序与之类似，指的是指示计算机每一步动作的一组指令；
2. 指令和数据。程序是指令和数据的组合体。如print("Hello")，print是指令，Hello是数据。
3. CPU可以直接识别并使用的语言；
4. 内存，硬盘等媒介保存的程序需要复制到内存后才能运行；
5. 内存中，用来表示指令和数据存储位置的数值；
6. CPU.

第2章 数据是用二进制数表示的

本章提问

1. 32位是几个字节？
2. 二进制数01011100转换成十进制数是多少？
3. 二进制数00001111左移两位后，会变成原数的几倍？
4. 补码形式表示的8位二进制数11111111，用十进制数表示的话是多少？
5. 补码形式表示的8位二进制数10101010，用16位的二进制数表示的话是多少？
6. 反转部分图形模式时，使用的是什么逻辑运算？

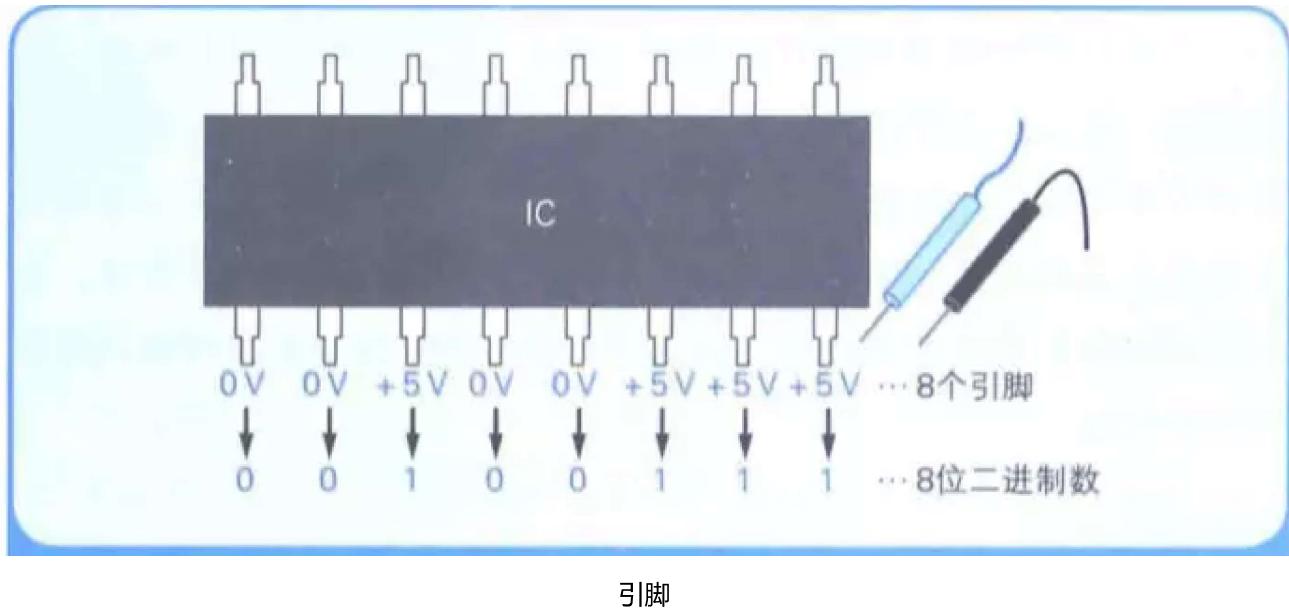


本章重点

二进制及其运算

2.1 用二进制数表示计算机信息的原因

计算机内部的电子元件是由IC (Integrated Circuit, 集成电路) 构成的, CPU也不例外。IC的引脚有引脚, 有电流通过时, 表示1, 反之表示0。



引脚

其中一个引脚表示一位(bit, 即binary digit), 8位表示一个字节, 内存和磁盘都是用字节单位储存和读写数据, 因此, 字节是信息的基本单位。32位处理器, 即表示一次可以处理32位(4字节)的二进制信息。

对于用二进制表示的信息, 计算机不会区分它是数值、文字, 还是某种图片的模式等, 而是根据编写程序的各位对计算机发出的指示来进行信息的处理。

例如 00100111 这样的二进制数, 既可以视为纯粹的数值作加法运算, 也可以视为“‘’(单引号, single quotation) 文字而显示在显示器上, 或者视为■■□■■□□□这一图形模式印刷出来。具体进行何种处理, 取决于程序的编写方式。

仓

贝

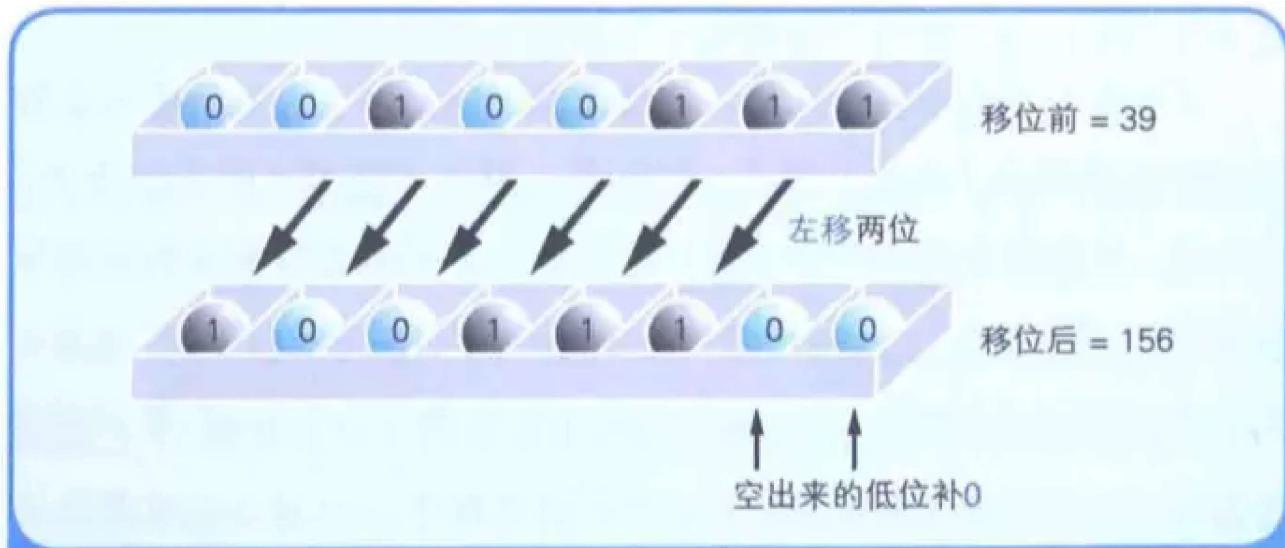
心

链

2.2 什么是二进制数

各种进制的转换。

2.3 移位运算和乘除运算的关系



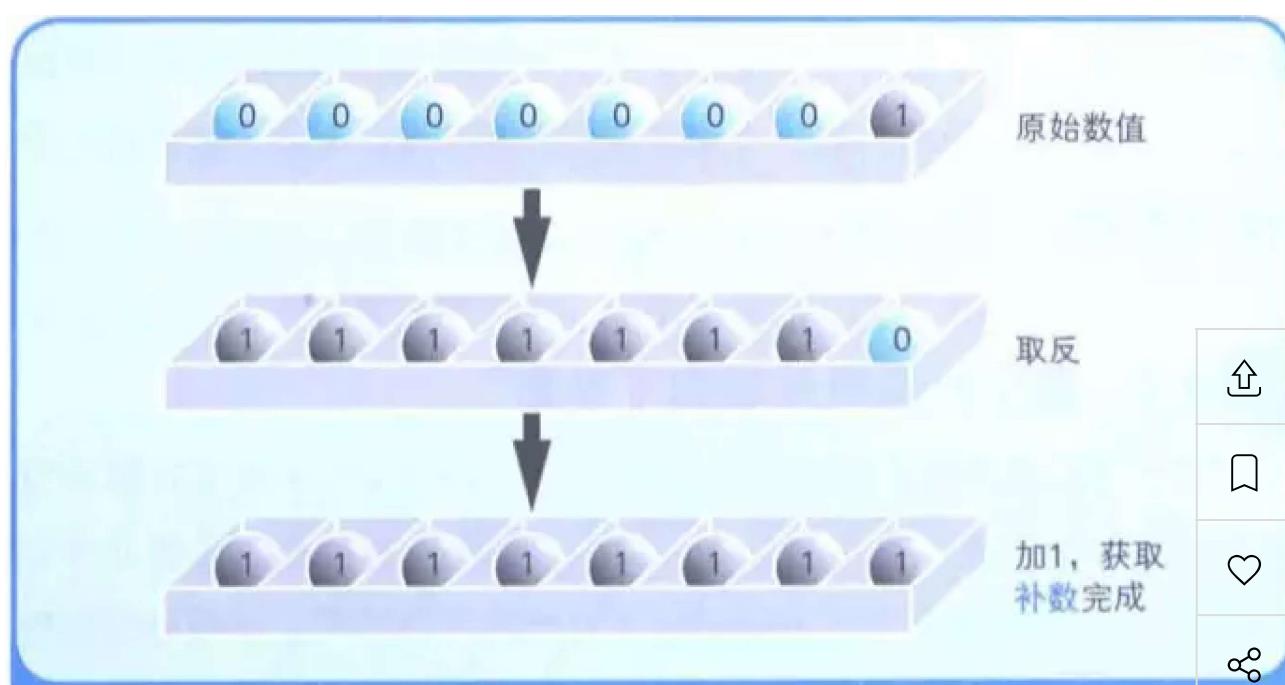
左移两位的运算

左移运算需要在空出来的低位补0，右移运算在后面说明。另外，移位导致溢出的数字，直接丢弃即可。

可以依靠移位运算来进行乘除运算，二进制左移后会变为原来的二倍、四倍等等，右移会变为二分之一、四分之一等等。

2.4 便于计算机处理的补数

将所有的位取反再加一就得到补数，数及其补数互为相反数。



得到补数



为什么数与其补数互为相反数？

例如，00000001 和取反后的 11111110 相加，结果为 11111111，全部数位均为 1。因此，比 11111110 大 1 的数加上 00000001 后，11111111 变为 9 位的 100000000。由于在 8 位的范围内运算时第 9 位会被计算机忽略，因此结果就变成了 00000000。

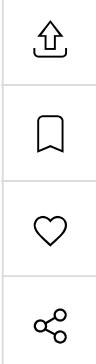
计算机用加法实现减法运算，减某数就等于加上某数的补数。

$$\begin{array}{r} 00000011 \dots 3 \\ + 11111011 \dots \text{用补数表示的}-5 \\ \hline 11111110 \dots \text{用补数表示的运算结果}-2 \end{array}$$

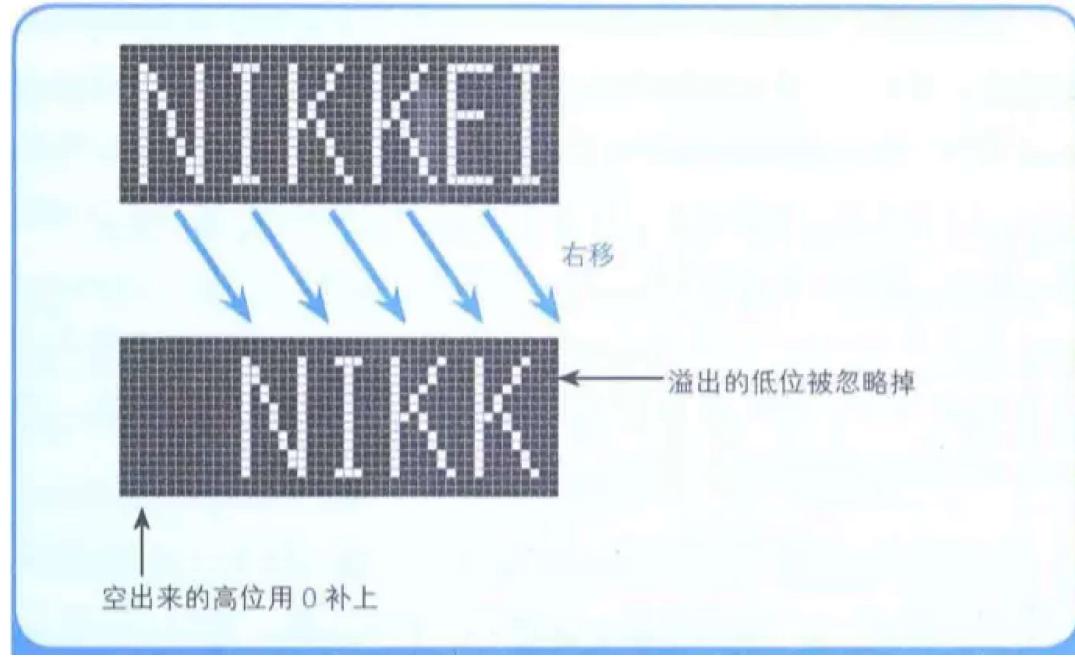
3-5的运算结果

2.5 逻辑右移和算术右移的区别

什么是逻辑右移？



当二进制数的值表示图形模式而非数值时，移位后需要在最高位补0.类似于霓虹灯往右滚动的效果。



什么是算术右移？

当二进制数作为带符号的数值进行运算时，移位后要在最高位填充移位前符号位的值(0或1)，这就叫算术右移。

只有在右移时才必须区分逻辑右移和算术右移的区别。左移时，只需要在空出来的低位补0即可。

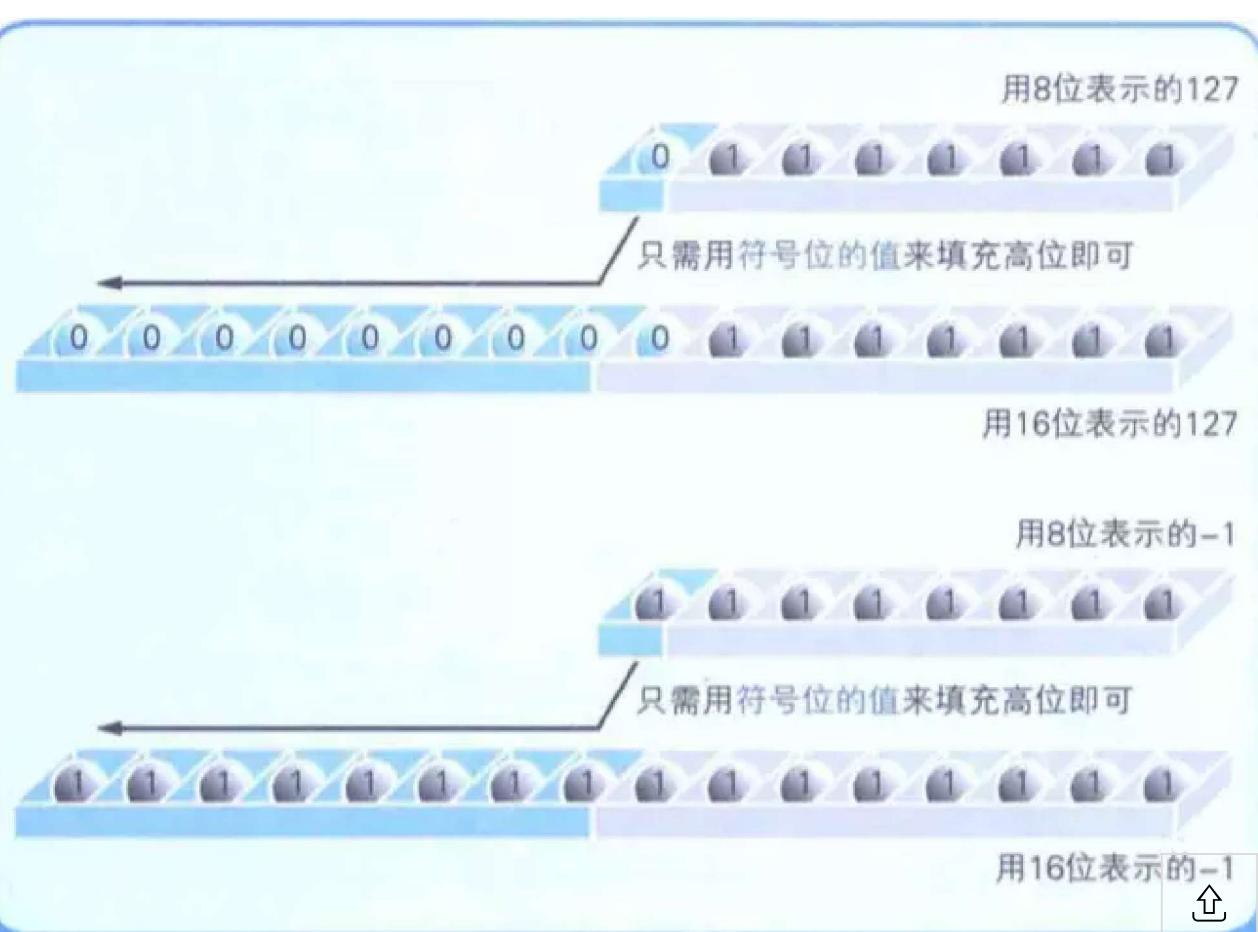


● 逻辑右移



逻辑右移与算术右移的区别

符号填充：



符号填充

2.6 掌握逻辑运算的窍门

4种类型：逻辑非、逻辑或、逻辑与、逻辑异或。

逻辑非运算时，全部取反

$$\text{NOT } \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array}$$

逻辑与运算时，表示 1 的部分不变，表示 0 的部分变成 0

$$\begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} \text{ AND } \begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 0 & 0 \\ \hline \end{array}$$

逻辑或运算时，表示 0 的部分不变，表示 1 的部分变成 1

$$\begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} \text{ OR } \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 1 \\ \hline \end{array}$$

逻辑异或运算时，表示 0 的部分不变，表示 1 的部分取反

$$\begin{array}{|c|c|} \hline 1 & 0 \\ \hline \end{array} \text{ XOR } \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 0 & 1 \\ \hline \end{array}$$

答案：

1. 4
2. 92
3. 4倍
4. -1
5. 111111110101010
6. 异或



第3章 计算机进行小数运算时出错的原因

本章提问

1. 二进制数 0.1，用十进制数表示的话是多少？
2. 用小数点后有 3 位的二进制数，能表示十进制数 0.625 吗？
3. 将小数分为符号、尾数、基数、指数 4 部分进行表现的形式称为什么？
4. 二进制数的基数是多少？
5. 通过把 0 作为数值范围的中间值，从而在不使用符号位的情况下表示负数的表示方法称为什么？
6. 10101100.01010011 这个二进制数，用十六进制数表示的话是多少？

问题

本章重点

计算机进行小数处理的机制

3.1 将0.1累加100次也得不到10

```
#include <stdio.h>

void main() {
    float sum;
    int i;

    // 将保存总和的变量清 0
    sum = 0;

    // 0.1相加 100 次
    for (i = 1; i <= 100; i++) {
        sum += 0.1;
    }

    // 显示结果
    printf("%f\n", sum);
}
```



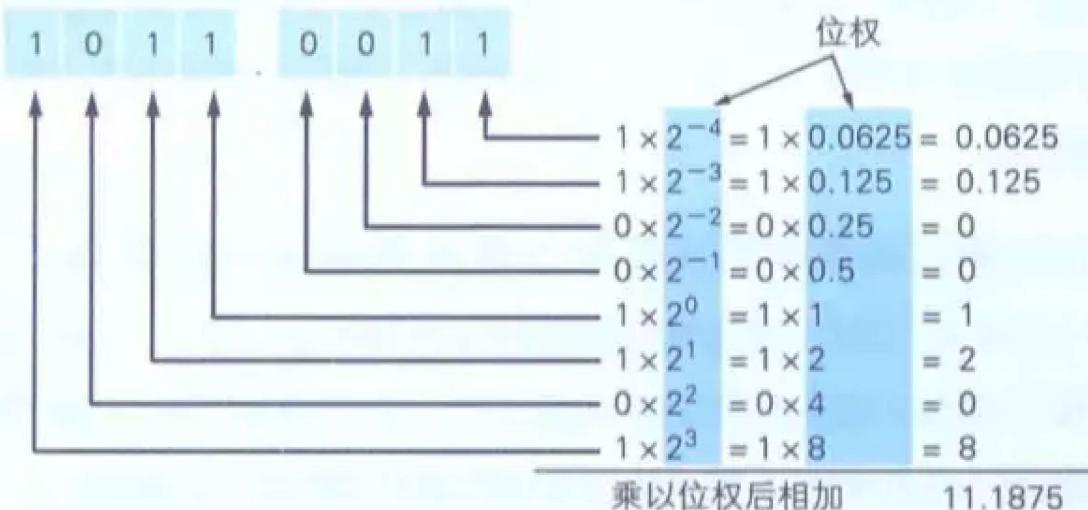
将0.1累加100次（C语言）



结果

3.2 用二进制数表示小数

就是用负数来表示相应位数的指数



二进制小数转十进制

3.3 计算机运算出错的原因

如图：



表 3-1 小数点后 4 位能够用二进制数表示的数值
二进制数是连续的，十进制数是非连贯的

二进制数	对应的十进制数
0.0000	0
0.0001	0.0625
0.0010	0.125
0.0011	0.1875
0.0100	0.25
0.0101	0.3125
0.0110	0.375
0.0111	0.4375
0.1000	0.5
0.1001	0.5625
0.1010	0.625

二进制小数无法连贯的表示十进制小数

3.4 什么是浮点数

浮点数的形式：

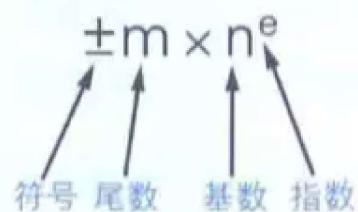


图 3-3 浮点数的表现形式。由符号、尾数、基数、指数四部分构成

浮点数由 符号、尾数、基数、指数 四部分构成。

浮点数的构造 (IEEE的规定)：



● 双精度浮点数（共64位）



其中，符号部分非0即1；尾数部分用的是“将小数点的前面的值固定为1的正则表达式”，指数部分用“EXCESS系统表现”。简单的来说，小数就是尾数部分 $\times 2$ 的指数部分次幂。

PS，正则表达式是按照特定的规则来表示数据的形式。

3.5 正则表达式和EXCESS系统

尾数部分是如何用正则表达式来表示的呢？

先看十进制。比如十进制数0.75，可以用很多种方法来表示它，正如下图所示。由于方法太多，需要制定一个统一的规则方便计算机处理，比如可以规定十进制小数“小数点前面是0，小数点后面第一位不能是0”。

$$0.75 = 0.75 \times 10^0$$

$$0.75 = 75 \times 10^{-2}$$

$$0.75 = 0.075 \times 10^1$$

数值的多种表现形式

同样的，我们也可以为二进制指定规则。在二进制中，我们使用的是“将小数点前面固定为1”的正则表达式（个人理解：由于二进制只由0、1组成，如此做就相当于在小数点前保留1位有效数字）。这样的话，第一位中的1可以不去保存，因为怎么样都是1，这样就节省了一个数据位，也就可以表示更大范围的数据。



1011.0011

... 原始数值

如果第1位始终是1，那么0怎么表示呢？查了资料 (<https://links.jianshu.com/go?to=http%3A%2F%2Fwww.cnblogs.com%2Fgerman-iris%2Fp%2F5759557.html>)，当尾数和指数均为0时，表示0，那1又怎么表示呢.....暂时还没有找到资料，知道的大佬可以告诉我下。

指数部分与EXCESS系统

就是把中间的数当成0来看。

实际的值(二进制数)	实际的值(十进制数)	EXCESS系统表现(十进制数)
11111111	255	128 (= 255 - 127)
11111110	254	127 (= 254 - 127)
:	:	:
01111111	127	0 (= 127 - 127)
01111110	126	-1 (= 126 - 127)
:	:	:
00000001	1	-126 (= 1 - 127)
00000000	0	-127 (= 0 - 127)

EXCESS

3.6 在实际的程序中进行确认

```
#include <stdio.h>
#include <string.h>

void main() {
    float data;
    unsigned long buff;
```

代码第一部分



```

int i;
char s[34];

// 将 0.75 以单精度浮点数的形式存储在变量 data 中。
data = (float)0.75;

// 把数据复制到 4 字节长度的整数变量 buff 中以逐个提取出每一位。
memcpy(&buff, &data, 4);

// 逐一提取出每一位
for (i = 33; i >= 0; i--) {
    if(i == 1 || i == 10) {
        // 加入破折号来区分符号部分、指数部分和尾数部分。
        s[i] = '-';
    } else {
        // 为各个字节赋值 '0' 或者 '1'。
        if (buff % 2 == 1) {
            s[i] = '1';
        } else {
            s[i] = '0';
        }
        buff /= 2;
    }
}

```

代码第二部分

说下我的理解，`memcpy`就相当于把`data`内存中由0、1组成的序列复制到`buff`中去，但是程序中`buff`的值是用十进制显式表示的，所以需要不停对2取余获得真实的二进制序列。
程序结果及解释：



该程序执行后，十进制数 0.75 用单精度浮点数来表示就变成了 0-01111110-10000000000000000000000000000000 (图 3-7)。加入破折号 (-) 是为了区分符号部分、指数部分、尾数部分。这里，符号部分为 0，指数部分为 01111110，尾数部分为 10000000000000000000000000000000。因为 0.75 是正数，所以符号位是 0。指数部分的 01111110 是十进制数 126，用 EXCESS 系统表现就是 -1 ($126 - 127 = -1$)。根据正则表达式的规则，小数点前面的第 1 位是 1，因此尾数部分 10000000000000000000000000000000 实际上表示的是 1.1000000000000000000000000000000 这个二进制数。将尾数部分

3.7 如何避免计算机计算出错

- 回避策略，即无视错误，一点微小的偏差不会造成什么严重后果；
- 扩大相应的倍数用整数进行运算；
- 采用BCD方法。(暂时不太懂)

3.8 二进制数和十六进制数

二进制的4位等于十六进制的1位。

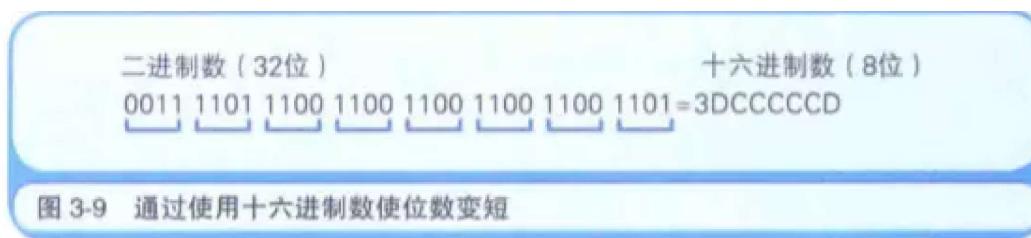


图 3-9 通过使用十六进制数使位数变短

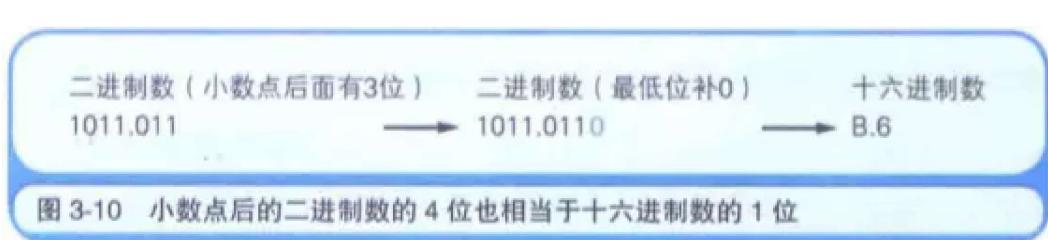


图 3-10 小数点后的二进制数的 4 位也相当于十六进制数的 1 位

问题答案：

1. 0.5
2. 能表示
3. 浮点数（浮点数形式）
4. 2

《程序是怎样跑起来的》（中）

(<https://www.jianshu.com/p/7212a15d26d8>)

小礼物走一走，来简书关注我

赞赏支持

程序是怎样跑起来的 (/nb/36570283)

© 著作权归作者所有



c747190cc2f5 (/u/c747190cc2f5)

写了 25958 字，被 1 人关注，获得了 3 个喜欢
(/u/c747190cc2f5)

一个菜鸡的挣扎...

喜欢



更多分享



写下你的评论...



评论 关闭评论

智慧如你，不想发表一点想法咩~

