高等学校试用教材

汇编语言程序设计

PDP-11

PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11
PDP-11

罗昌隆 编

汇编语言程序设计
罗昌隆 编

高等学校试用教材

汇编语言程序设计

(PDP-11)

罗昌隆 编

高等教育出版社

内 容 提 要

本书是按照教育部部属高等学校(工科、综合大学)计算机软件专业汇编语言程序设计教学大纲编写的．本书以 PDP-11 机的 MACRO-11 宏汇编语言为例，介绍了汇编语言程序设计的基本概念、方法和技巧．全书共分十章，包括 PDP-11 计算机的一般介绍、MACRO-11 汇编语言和汇编语言程序的基本结构和程序设计基本方法——分支程序、循环程序和子程序，以及采用模块化程序设计时，各模块程序的联接与浮动等内容．

本书可作为计算机软件及有关专业的教材也可供有关技术人员参考．

高等学校试用教材

汇编语言程序设计

(PDP-11)

罗昌隆 编

*

高等教育出版社出版

新华书店北京发行所发行

人民教育出版社印刷厂印装

*

开本 787×1092 1/16 印张 14.25 字数 320,000

1984 年 9 月第 1 版 1985 年 5 月第 1 次印刷

印数 000,001—23,800

书号 13010·01046 定价 2.40 元

前　言

本书是按照教育部部属高等学校(工科、综合大学)计算机软件专业汇编语言程序设计教学大纲编写的，可作为计算机软件和工程专业的教材.

本书以PDP-11(DJS-2000)计算机的MACRO-11宏汇编语言为例，介绍汇编语言程序设计的基本概念、方法和技巧. 通过本课程的教学，培养学生编写、调试、阅读和分析汇编语言程序的能力.

第一章介绍程序设计的一般概念.

第二章介绍计算机中数和字符的表示. 若本章内容已在其它先行课讲授，在这里可以不讲，或作为自学内容.

第三章是对PDP-11(DJS-2000)计算机的一般介绍. 着重介绍指令格式和寻址方式.

第四章介绍MACRO-11汇编语言，包括汇编语言的基本概念、各类汇编语言以及汇编过程. 为了使学生较早的上机实习，我们把输入/输出系统宏指令和库子程序调用的使用提到本章介绍. 这样，从本章起，以后各章都可以边学习，边编写程序，边上机实习，做到真正掌握教材中的内容.

第五章至第七章介绍汇编语言程序的基本结构和程序设计基本方法——分支程序设计、循环程序设计和子程序设计，以及采用模块化程序设计时，各模块程序的联结与浮动.

第八章介绍高级汇编语言技术，包括宏指令、条件汇编命令和重复块命令.

第九章介绍直接控制外部设备的输入/输出程序设计方法和中断处理.

第十章介绍浮点处理，本章内容作为选学.

在附录A中介绍了在PDP-11 RSX-11M(或RT-11)操作系统环境中建立和运行一个程序的过程以及编辑程序、任务建立程序(连接程序)、调试程序以及外围交换程序(PIP)，这些内容可在学生第一次上机之前介绍. 附录B总结了PDP-11机指令系统. 附录C给出了MACRO-11汇编语言程序的语法错误信息.

每章末都有一定数量的习题供读者练习.

书中主要例题都在PDP-11/23机上(在RSX-11M下)运行通过，正确无误.

本书由高等学校计算机软件教材编审委员会所组织的审稿会进行审查. 主审人是北京大学朱慧真老师，参加审稿会的有北京大学、武汉大学、清华大学、山东大学、西安交通大学、哈尔滨工业大学的同志，与会同志提出了许多宝贵意见. 在编写过程中，华侨大学吕松林和西安交通大学刘海岩同志也提出了许多宝贵的意见和建议，我院的甘圣予和金益民等同志给予了很大帮助，在此，表示衷心的感谢.

限于编者的水平，书中一定存在不少错误和不足之处，敬请读者批评指出.

罗　昌　隆

1984年2月于西北电讯工程学院

目　录

第一章　程序设计的一般概念

§1.1　电子数字计算机的基本结构和组成

现代的电子数字计算机是一种能自动地高速地处理数据的工具，它每秒钟能进行成千上万次各种不同的基本操作，有的可达每秒几百万至上亿次．大多数通用的电子数字计算机由下面五大部件组成：存贮器、运算器、控制器(运算器和控制器统称为中央处理器记为CPU)、输入和输出设备(称为I/O系统)．这些部件之间通过一个或几个总线相互传送数据．图1-1表示了这些部件之间的关系．

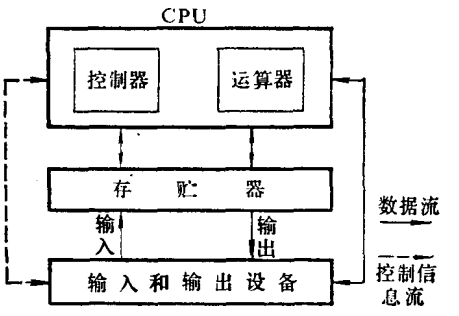


图1-1　电子计算机的简单框图

一、存贮器

存贮器是存放指令和数据的仓库．

存贮器通常由磁芯或半导体元件构成．一个磁芯或半导体元件可以表示一个二进制数字：0或1，称为一位(bit)．

存贮器中所有位被排列成存贮单元阵列，如图1-2所示．

每个存贮单元由固定个数的位(例如p位)组成，即每个存贮单元的大小相同．为了区分存贮器的不同单元，把存贮器全部单元按照一定顺序编号，每个单元的编号称为该单元的**地址**．不同的存贮单元具有不同的地址，用这个地址可以唯一地存取该单元的内容．于是一个存贮单元有两个特征：

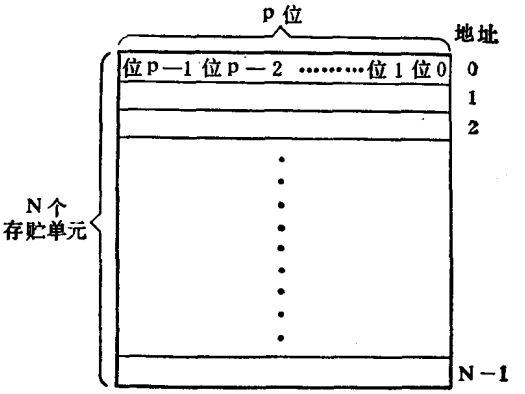


图1-2　存贮器组织

1. 地址——该单元在存贮器中相对的位置．

2. 内容——存贮在该单元中的代码．通常，我们使用一个符号，例如A，表示某个单元的地址，则记号(A)表示单元A中的内容．类似地，((A))表示(A)的内容(即以单元A中内容为地址的存贮单元的内容)．反之，若代码a存放在地址为A的单元中，则可记为：

$$\langle a\rangle = A$$

当我们要把一个代码送到某个存贮单元(简称写入)或从某个存贮单元取出(简称读出)时，首先要告诉机器相应的存贮单元的地址，即先把地址送往存贮器，然后由存贮器按地址查到对应

的单元，查到以后才把新的代码放进去或把该单元的代码取出来．这里要强调的是：在通常情况下，从存贮器某个单元读出代码后，这个单元的原内容仍保持不变，以便今后继续使用．反之，若往某个单元写入新的代码，那么原来的代码被“挤掉”，新的代码就自动地取代了原来的代码．

一个存贮器的单元总数称为存贮器的“容量”．

二、中央处理器(CPU)

中央处理器是由运算器和控制器组成，它是计算机的头脑．

运算器是对数据进行各种算术运算和逻辑运算等的部件．在运算过程中，运算器不断地得到由存贮器提供的数据，并能把求得的结果(包括中间结果)送回存贮器或暂时保存在CPU内部的寄存器中，它的整个运算是在控制器统一指挥下有规律地进行的．

控制器是计算机中的管理部件，它统一指挥和控制计算机各个部件．

控制器是按照指令向各部件发布命令，使它们完成指令所规定的操作．**指令**是指示机器执行操作的命令．指令按功能通常可分为如下几类：

1．数据传送指令　将数据从内存某个单元或寄存器传送到另一个单元或寄存器，或在内存和I/O设备之间传送数据．

2．算术指令　对一个或几个数执行算术运算，如两个单元的内容相加，一个单元的内容加1等．

3．逻辑指令　执行逻辑操作，如逻辑与、逻辑或等．

4．比较指令　考察一个单元或寄存器的内容或比较两个单元(或寄存器)的内容．

5．转移指令　改变指令序列的动态执行，无条件地或根据比较、算术、逻辑指令的结果有条件地转移到指定的指令执行．

6．控制性指令　控制和改变主机状态的指令，如停机指令等．

指令应指出：(1) 进行何种操作，(2) 操作数的地址及存放结果的地址．因此，指令一般具有下面格式：

操作码	地址码

其中操作码指出操作种类，如加、减、传送、比较、转移等；地址码指出参与操作的数据的位置或操作对象，或操作结果的存放位置．它可以是内存单元地址，可以是寄存器号，或设备寄存器号；地址码有时也可以是下一条要执行的指令的存放位置的信息．

有的指令地址码部分可以没有，有的可以有一个位置的信息，有的可以有多个位置的信息．

指令中的操作码，地址码均用二进制码表示，把它们的二进制编码排在一起，就组成一条指令的二进制编码．

指令和它们的操作数据一起存于存贮器．控制器从内存依次取出每条指令，分析被执行的指令，并命令计算机中各部件执行指令所规定的操作．下一条被执行的指令的地址(存贮器地址)保存在CPU内部的一个特定的寄存器中，此寄存器称为**程序计数器**(PC)．当程序装入存贮器

时，PC 装入第一条指令的地址，CPU 使用 PC 取出这条指令，然后取操作数，执行此条指令的功能，产生的结果送回存贮器．图 1-3 说明取、分析和执行一条指令的全周期．

注意每次取一条指令时，修改 PC 使它指向下一条指令．于是，处理器一条接着一条地执行指令．某些指令，如转移，明显地改变 PC 的内容，使得下一条执行的指令不是在下一个单元，而是控制转向一个新的单元，改变了指令序列的执行顺序．

CPU 内部还有若干个寄存器，这些寄存器可以保存数据和指令．某些计算机有几个执行不同功能的寄存器，例如有的保存浮点数．此外，还有几个**通用寄存器**，可作一般目的使用．

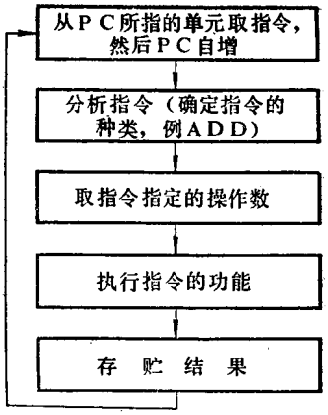


图 1-3　指令执行的周期

三、输入/输出系统

输入/输出系统是处理器和人及其它处理器进行通讯的部件．它包含各种类型的输入和输出设备以及输入和输出的控制部件．

计算机把解题所需的数据和程序，以及告诉计算机应该做什么的命令通过输入设备送到计算机里去．在计算执行过程完毕后，还要把计算的结果和信息(需要时，也可以是中间结果)保存在输出设备上或送出来．

最常用的输入/输出设备是终端、行式打印机、磁带和磁盘．终端是人和计算机之间方便的通讯工具，它由两部分组成：输入部分是键盘；输出部分有的是显示器(CRT)，有的是打印机．行式打印机是快速打印设备．磁带和磁盘能以较快速度成组传送信息，它是存放程序和数据的大容量存贮设备．

四、软件

现代的计算机除了上述的五大部件(统称为硬件)之外，还配备丰富的软件．所谓**计算机软件**就是为提高计算机的使用效率，扩大计算机的功能所设计的程序、数据和有关资料的总和．例如，汇编程序、编译程序、解释程序都是计算机软件．除此之外，操作系统、数据库管理系统等都是软件．电子计算机发展到现阶段不可能没有软件．

§1.2　程序设计的一般概念

近年来，计算机应用的领域迅速扩大．到目前为止，它不仅可以应用于科学设计与计算，而且还应用于商业、学校和政府部门的事务处理，应用于厂矿企业生产过程的自动控制，应用于国防建设，此外，它还可以帮助医生为患者诊断和治疗疾病，帮助人们学习等等．

总之，现代数字计算机能够存贮信息，高速地执行运算，并能根据结果作出判断，最后获得给定问题的解．

一、程序的概念

虽然，计算机具有广泛的应用、强大的功能，但计算机所执行的每一步工作，必须预先由人安排好．也就是说，人们对所要解决的问题事先要有一个明确的计算步骤或操作过程，并且要把它

用计算机能识别的语言表达出来．即是把解法写成计算机能完成的一系列指令．每一条指令指示计算机执行一项基本操作．为使计算机获得给定问题的解，所需的指令序列和数据称为**程序**．因此，人们对所求的问题要写出一个程序，编写程序的过程称为**程序设计**．程序是依据解题的某个算法而编写的，存放在计算机存贮器里，计算机执行它，就能解决所求的问题．

当你在设计和实现一个新的程序时，应考虑下面这些目标：

1. 这个程序应能正确地解决所求的问题，即程序的正确性．

2. 这个程序应是容易理解和阅读的，即程序的可读性．这样要求你采用好的程序结构，并且尽可能地把解题的算法和步骤告诉给读者．

3. 这个程序是容易修改和维护的，即程序的可维护性．

要达到上述要求，就要求你在学习过程中养成好的程序设计方法、技巧、风格和素养，并在实践中不断总结和提高．

二、程序设计的几个阶段

为了使计算机成功地解决一个问题，可能要经过下面几个阶段：

1. 明确求解的意义．
2. 确定算法．选取最合适的解题方法，确定输入、输出数据的格式．
3. 分析算法，或画出程序框图．
4. 用程序设计语言编写程序．
5. 查错和调试程序．
6. 写出有关资料．

问题的确定　在计算机解决所求问题之前，必须将问题的精确意义搞清楚，如果问题没有弄得相当明确，那么大量时间和精力可能被浪费．同时，要把计算或处理对象的物理过程和工作状态，用数学公式或用文字描述出来，建立数学模型．

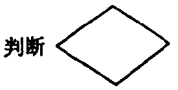
确定算法　由于计算机只会做加、减、乘、除等这些最基本的算术运算及某些简单的逻辑运算，因此，要将上阶段归结的数学模型转化为一系列计算机能完成的基本操作．算法是解决问题的步骤和方法．为解决一个问题可能存在许多种方法．选择何种方法依赖于使用的是什么样的计算机系统，以及依赖于所求问题的精度和速度等等要求．根据问题的要求应明确输入、输出些什么数据，它们的格式是怎样的．如果数据有很好的结构，就会简化计算机的处理．

分析算法，画出框图　当计算机解决一些比较复杂的问题时，算法可能比较复杂，编制程序就比较困难．为此可将算法按照逻辑关系划分成较小的可以执行的步骤，然后用某种描述语言或用图形把它表达出来，这种图形通常称为**框图**．它指示了计算机应该执行操作的逻辑次序．程序的框图对于程序员确定解题的方法以及编写程序指令会有所帮助．另外，当检查程序错误时，也是有用的．

框图基本上是框和有向线的集合体，框指示该做什么，而有向线指明框的顺序．框具有各种形状，表示程序中要执行的动作．图 1-4 描述了各种框符号，这是由 ANSI/ISO(国际标准化组织)规定的符号．

椭圆符号表示程序中某个终端点．使用它指明程序的开始，停止或中断．把适当的字记入这个符号中．

矩形符号表示某个过程的功能．这过程可能是为执行一个给定的任务所需的一条指令或一组指令．所要执行的任务的简短叙述记入这符号中．



菱形符号用于表示程序在这一点要进行比较、判断，决定程序的流向．检验条件记入符号中，并使用检验的各种可能结果标出在这一点程序的各种流向．

一个小圆内标以一个标识符．使用它把具有相同标识符的两个小圆，一个终止程序的流向，另一个开始程序的流向的圆连接起来．于是，允许框图画在不同页面上，通过它连接起来，也允许消除流线的交叉．

图 1-4　框图符号

例如，把存贮在单元 A, B, C 中的三个数按递增次序排列，运算的框图如图 1-5.

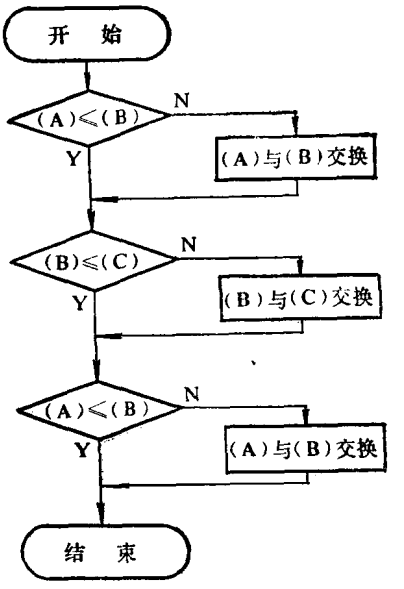


图 1-5　将数(A), (B), (C)按递增顺序排列

用程序设计语言编写程序　根据算法或框图，就可用程序设计语言编写程序．这里所说的程序设计语言可以是某种具体计算机能识别的机器语言；也可以是用符号表示每一条指令的汇编语言；以及高级程序设计语言，例如 BASIC、FORTRAN、ALGOL、COBOL、PASCAL等．

程序查错和调试　开始编写的程序往往存在错误，要求我们找出程序中可能存在的任何错误．如果遗漏掉某些必要的指令或编码不正确，结果就可能出乎意外，这些错误必须予以发现并加以纠正．有些错误(主要是语法错误)编译程序或汇编程序就能发现；有些错误在程序执行时才能发现．查错和调试有时可能重复多次，直至程序正确运行为止．

资料整理　资料是程序设计的重要部分，好的资料有助于设计者进行调试和排错，而且对程序的使用、维护和扩充都是必不可少的．资料通常包括对程序的说明、框图、程序清单、内存分配图和程序库等．

习　题

1.1　电子数字计算机由哪几部分组成？各部分的功能是什么？

1.2　什么叫指令？什么叫程序？什么叫程序设计？

1.3　画出求五个数中所有正数之和的框图(结果存于单元 D中)．

1.4　将三个数按递减的次序排列，画出它的框图．

1.5　什么是计算机的软件？

第二章　计算机中数的表示

§2.1　各种计数制及数制间的相互转换

电子计算机要进行大量的数据运算，采用什么样的数制来表示数，对机器的性能和程序设计方便性都有着很大的影响．在电子计算机中使用的数制有二进制、八进制、十六进制等，以二进制和八进制为最常用．本节将讲述它们的表示和彼此之间的转换．

2.1.1　十进制数的表示

在日常工作和生产劳动中，我们最常用最熟悉的就是十进制数．数是由一个或几个数字组成，它表示一个数目．例如，35 是一个数，它是由数字 3 和 5 组成．十进制数是用十个不同的数字 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 来表示．数字处于不同的位置代表的意义是不同的．例如，4937.16 中小数点左边的第一位 7 是个位，表示它本身的数值；左边第 2 位是十位，表示 3×10；左边第三位是百位，表示 9×100；左边第四位是千位，表示 4×1000；而小数点右边的第一位是十分之一位，表示$1\times\frac{1}{10}$；右边第二位是百分之一位，表示$6\times\frac{1}{100}$．因而这个数可以写为：

$$4937.16=4\times1000+9\times100+3\times10+7+1\times10^{-1}+6\times10^{-2}$$

一般情形，任意一个十进制数

$$N=\pm k_n k_{n-1}\cdots\cdots k_1 k_0.k_{-1}k_{-2}\cdots\cdots k_{-m}$$

可表示为：
$$\begin{aligned}N=\pm[&k_n(10)^n+k_{n-1}(10)^{n-1}+\cdots+k_1(10)^1+k_0(10)^0\\&+k_{-1}(10)^{-1}+k_{-2}(10)^{-2}+\cdots+k_{-m}(10)^{-m}]\end{aligned}\tag{2.1}$$

其中 $k_j\in(0, 1, 2, 3, 4, \cdots, 9)$.

上式中 m, n 为正整数，m 表示数 N 的小数位数，n 表示整数位数减一，括号内的 10 称为十进制计数制的基数．所谓某计数制的基数，就是在该计数制中用到的数字个数．由于这里用到 10 个数字，所以基数是 10．当基数是 10 时，每位计满十向高位进一，即“逢十进一”．

在日常生活和生产中除十进制计数外，还经常用到十二进制(如铅笔 12 支为 1 打)，十六进制(如老秤 16 两为 1 斤)和六十进制(如时间 60 秒为 1 分，60 分为 1 小时)等等．因此，计数制的基数不一定是 10，而可以是任意的正整数 R．这样，任意一个数 N 都可以表示为：

$$\begin{aligned}N=\pm[&k_nR^n+k_{n-1}R^{n-1}+\cdots+k_1R^1+k_0R^0\\&+k_{-1}R^{-1}+k_{-2}R^{-2}+\cdots+k_{-m}R^{-m}]\end{aligned}\tag{2.2}$$

式中 m, n 为正整数，$k_i\in(0, 1, \cdots, R-1)$.

若取基数 $R=2$，这就是二进制，它是电子计算机中广泛采用的计数制．

2.1.2 二进制数的表示

二进制数的基数是2，每个数位只能取两个不同的数字“0”和“1”，而且是“逢二进一”。为了熟悉二进制数的表示，我们先来看一看十进制中十个不同的数字如何用二进制来表示。

$$0_{10}=0_2 \qquad 1_{10}=1_2$$
$$2_{10}=10_2 \qquad 3_{10}=11_2$$
$$4_{10}=100_2 \qquad 5_{10}=101_2$$
$$6_{10}=110_2 \qquad 7_{10}=111_2$$
$$8_{10}=1000_2 \qquad 9_{10}=1001_2$$

为了表示不同计数制的数，我们在数的右下角标出每个数所在计数制的基数。例如

$$9_{10}=1001_2 \qquad 0.125_{10}=0.001_2$$

由于二进制数的基数是 2，于是任意二进制数可写成形如公式(2.2)的展开式，例如

$$11011.01_2=1\times2^4+1\times2^3+0\times2^2+1\times1^1$$
$$+1\times2^0+0\times2^{-1}+1\times2^{-2}$$

一般地，任意一个二进制数

$$N=k_nk_{n-1}\cdots k_0.k_{-1}\cdots k_{-m}$$

可展开成

$$N=k_n2^n+k_{n-1}2^{n-1}+\cdots+k_02^0+k_{-1}2^{-1}+\cdots+k_{-m}2^{-m}$$

其中 $k_i\in(0,1)$，m，n 为正整数。

我们已经知道，现代电子数字计算机大多数采用二进制来表示数，这是为什么呢？

首先，由于二进制只使用两个不同的数字符号，所以任何具有两个不同稳定状态的元件都可用来表示数的每一位。而制造具有两个稳定状态的元件要比制造多稳定状态(如十个稳定状态)的元件容易得多。事实上，人们已找到很多种元件具有两个稳定的状态，例如，继电器触点的关闭和释放，电子管或晶体管的导通和截止，电容器的充电和放电等等。这些元件都是按照极简单的操作“开”或“关”，“是”或“不是”、“有”或“没有”的原理工作的，因此工作可靠，而且抗干扰性能好。同样，如采用二进制，电子计算机中数的存贮和传送也可采用简单而可靠的表示法，如脉冲的有无，脉冲极性的正负，电位的高低等。

并且，采用二进制数，对电子计算机的制造来说，可以节省元件。

其次，从运算的简便性来看，二进制运算是最简单的。

由于二进制数的运算原则是“逢二进一，借一当二”，所以四则运算相当简便。

加法 它的法则是

$$0+0=0;\ 0+1=1+0=1;\ 1+1=10$$

例如求

$$11011.01+101011.11=?$$

算式如下：

$$\begin{array}{r} 11011.01 \\ +)\,101011.11 \\ \hline 1000111.00 \end{array}$$

即

$$11011.01+101011.11=1000111.00$$

减法 它的法则是

$$0-0=0;\ \ 1-0=1;\ \ 1-1=0;\ \ 10-1=1$$

例如求 $11101.01-1100.10=?$

算式如下：

$$\begin{array}{r} 11101.01 \\ -)\ \ 1100.10 \\ \hline 10000.11 \end{array}$$

即

$$11101.01-1100.10=10000.11$$

对于加法和减法运算，关键是把小数点对齐，然后根据它们相应的法则进行运算。

乘法 它的法则是

$$0\times0=0;\ \ 1\times0=0\times1=0;\ \ 1\times1=1$$

例如求 $1101.01\times110.11=?$

算式如下：

$$\begin{array}{r} 1101.01 \\ \times)\ \ 110.11 \\ \hline 1101\ 01 \\ 11010\ 1 \\ 000000 \\ 110101 \\ 110101 \\ \hline 1011001.01\ 11 \end{array}$$

即

$$1101.01\times110.11=1011001.0111$$

除法 它的法则是

$$0\div1=0;\ \ 1\div1=1$$

例如求 $1101.1\div110=?$

算式如下：

$$\begin{array}{r|l} & 10.01 \\ \hline 110 & 1101.1 \\ & 110 \\ \hline & 1\ \ 10 \\ & 1\ \ 10 \\ \hline & 0 \end{array}$$

即

$$1101.1 \div 110 = 10.01$$

对于乘法和除法运算，就象我们平常对于十进制数的乘法和除法一样进行，算完之后再确定小数点的位置．从上面算例可以看出，二进制数的乘、除法运算实际上就是二进制的加、减法运算．由于四则运算简单，从而使电子计算机中实现二进制运算的线路比较容易设计．

因为上述优点，所以二进制在电子数字计算机中获得了广泛的应用．

2.1.3 八进制数及它与二进制数的互相转换

二进制数在计算机中被广泛使用，然而，二进制写起来很长，读起来也比较麻烦．人们通过工作实践发现八进制比二进制书写要简短，而且八进制与二进制之间转换也很方便，因此，计算机工作者乐于采用八进制．

八进制的基数 R＝8，每位可能取八个不同的数字符号 0，1，2，3，4，5，6，7 中的一个，运算规律是逢八进一．例如：

$$\begin{aligned} 742.5_8 &= 7\times8^2+4\times8^1+2\times8^0+5\times8^{-1} \\ &= 7\times64+32+2+5\times0.125 = 482.625_{10} \end{aligned}$$

因为二进制与八进制的基数成整数幂($R=8=2^3$)，所以它们之间的转换是很方便的．

首先讨论整数情况，设某整数在二进制中写成

$$\begin{aligned} N &= k_n k_{n-1}\cdots k_1 k_0 \\ &= k_n 2^n + k_{n-1}2^{n-1} + \cdots + k_1 2^1 + k_0 \end{aligned} \tag{2.3}$$

在八进制中写成

$$\begin{aligned} N &= l_m l_{m-1}\cdots l_1 l_0 \\ &= l_m 8^m + l_{m-1}8^{m-1} + \cdots + l_1 8 + l_0 \end{aligned} \tag{2.4}$$

将两式均除以 8（即式(2.3)用 2^3 去除，式(2.4)用 8 去除），应得到相等的商数和余数．

商数为

$$l_m 8^{m-1} + l_{m-1}8^{m-2} + \cdots + l_2 8 + l_1 = k_n 2^{n-3} + k_{n-1}2^{n-4} + \cdots + k_4 2 + k_3$$

余数为

$$l_0 = k_2 2^2 + k_1 2 + k_0 = (k_2 k_1 k_0)_2$$

依次用 8 去除上次的商数，得到

$$l_1 = (k_5 k_4 k_3)_2, \quad l_2 = (k_8 k_7 k_6)_2, \cdots$$

等等相似的结果．

由此得到的结论是：从八进制整数转换成二进制整数时，只需将每位八进制数用三位二进制数表示即可．从二进制整数转换成八进制整数时只需从右开始每三位二进制数（如果位数不足三位时，左边可以补 0）用一个八进制数表示．

现在讨论小数的情况．设某个小数在二进制中写成

$$x = 0.\, k_{-1}k_{-2}\cdots k_{-m}$$

$$=k_{-1}2^{-1}+k_{-2}2^{-2}+\cdots+k_{-m}2^{-m} \tag{2.5}$$

在八进制写成

$$x=0.l_{-1}l_{-2}\cdots l_{-v}$$
$$=l_{-1}8^{-1}+l_{-2}8^{-2}+\cdots+l_{-v}8^{-v} \tag{2.6}$$

将式(2.5)和式(2.6)均乘以 8，则所得整数部分和小数部分应相等.

整数部分为

$$l_{-1}=k_{-1}2^2+k_{-2}2^1+k_{-3}2^0=(k_{-1}k_{-2}k_{-3})_2$$

小数部分为

$$l_{-2}8^{-1}+l_{-3}8^{-2}+\cdots+l_{-v}8^{-v+1}$$
$$=k_{-4}2^{-1}+k_{-5}2^{-2}+\cdots+k_{-m}2^{-m+3}$$

依次用 8 去乘上次的小数部分，得到

$$l_{-2}=(k_{-4}k_{-5}k_{-6})_2,\ l_{-3}=(k_{-7}k_{-8}k_{-9})_2,\ \cdots$$

于是得到的结论是：从八进制小数转换成二进制数时，只需将每位八进制数用三位二进制数代替. 从二进制小数转换成八进制数时，只需从小数点开始向右每三位二进制数(如果位数不足三位时，右边可以补 0)用一个八进制数代替即可.

如果一个数既有整数部分，又有小数部分，则可将整数部分与小数部分分别进行转换，然后合并就可得到结果.

例 1 把八进制数 70.521 表示成二进制数.

7	0	.	5	2	1
↓	↓		↓	↓	↓
111	000	.	101	010	001

所以

$$70.521_8=111000.101010001_2$$

例 2 把二进制数 1011011.00101011 表示成八进制数.

1,011, 011 . 001, 010, 11

0 0 1	0 1 1	0 1 1	.	0 0 1	0 1 0	1 1 0
↓	↓	↓		↓	↓	
1	3	3	.	1	2	6

所以

$$1011011.00101011_2=133.126_8$$

2.1.4 十进制数转换成八进制数

日常生活中我们经常碰到的数是十进制的，而计算机工作者常用的是八进制，于是，经常需要把十进制数转换成八进制数，以及把八进制数转换成十进制数. 在这一段我们先讨论如何把十进制数转换成八进制数，而在下一段将讨论八进制数转换成十进制数的问题.

先讨论整数的情况. 假设已知十进制正整数 N_{10}(对于负整数同样处理, 只要在最后加上负号就行了), 求八进制整数 N_8.

设 $$N_8=(k_nk_{n-1}\cdots k_1k_0)_8$$

现在的问题是要确定 $k_n, k_{n-1}, \cdots, k_1, k_0$ 的值.

根据八进制的定义, 可展开成

$$N_8=k_n8^n+k_{n-1}8^{n-1}+\cdots+k_18+k_0$$

因为 $$N_{10}=N_8$$

所以有 $$N_{10}=k_n8^n+k_{n-1}8^{n-1}+\cdots+k_18+k_0$$

把上面等式的左右两边用 8 去除, 即得

$$\frac{N_{10}}{8}=(k_n8^{n-1}+k_{n-1}8^{n-2}+\cdots+k_1)+\frac{k_0}{8}$$

此式表明 k_0 是 N_{10} 除以 8 的余数, 而 $k_n8^{n-1}+k_{n-1}8^{n-2}+\cdots+k_1$ 是它的商数, 再把这个商数除以 8 时, 又得到第二个商和第二个余数, 这第二个余数就是所求的 k_1, 如此继续下去, 可以得到 $k_2, k_3, \cdots$, 直到最后一个商为 0 时, 对应的余数就是 k_n, 也就是八进制的最高位.

将上述过程写成公式如下:

$$\left.\begin{aligned}
N_{10}&=8\times q_1+k_0\\
q_1&=8\times q_2+k_1\\
q_2&=8\times q_3+k_2\\
&\vdots\\
q_{n-1}&=8\times q_n+k_{n-1}\\
q_n&=8\times q_{n+1}+k_n
\end{aligned}\right\} \qquad (2.7)$$

此时 $q_{n+1}=0$. 所得八进制数为

$$N_8=k_nk_{n-1}\cdots k_1k_0$$

例 3 将十进制数 279 化为八进制数. 由(2.7)式可得

		余数
8 | 2 7 9		7……k_0
8 | 3 4		2……k_1
8 | 4		4……k_2
0		

这里除得的余数依次是 7, 2 和 4, 所以所求八进制数为 427 即

$$279_{10}=427_8$$

在此提醒读者注意, 对于各种计数制中的数, 在读的时候不能一律按十进制数的习惯读法. 在十进制中, 279 可读作二百七十九. 然而在八进制中, 427 不可读作四百二十七, 只能按每位数字的名称读作四二七即可.

现在讨论小数的情况. 仍假设已知正的十进制小数 N_{10}, 求八进制小数 N_8.

设 $$N_8=(k_{-1}k_{-2}\cdots\cdots k_{-m})_8$$

目的是要确定 $k_{-1}, k_{-2}, \cdots, k_{-m}$. 按八进制的定义展开成