程序设计语言

FORTRAN IV

南京电子计算站

西门子7.000系列

操作系统 BS1000

FORTRAN IV

前　言

这个本子是在一年前编写的《FORTRAN Ⅳ 语言基础讲义》的基础上，考虑了南京电子计算站举办各次FORTRAN语言推广训练班的经验，以及在SIEMENS7.730电子计算机上实践的经验，修改增补而来的。内容全面而较切实际。编写时想照顾两个有些矛盾的要求：使得初学者易阅读及使得SIEMENS FORTRAN Ⅳ的用户易查核，做了一番努力，未必奏效。

第〇章是入门。对于没有接触过先进操作系统上的程序设计语言的人，是必读的。

第一至八章，由浅入深地介绍SIEMENS FORTRAN Ⅳ语言的全部内容。该语言包括国际标准化组织（ISO）1972年建议的完全级FORTRAN标准文本全部内容。而且有很强大的扩充功能，特别是提供了使用高速大容量磁盘存贮器的功能。

第九章是如何编译、连接编辑及运行一个FORTRAN程序，如何阅读出错报告。

第十章是如何使用SIEMENS FORTRAN Ⅳ的输入输出文件，有关使用磁带文件和磁盘文件的详尽说明。

在附录中有一个SIEMENS FORTRAN Ⅳ的简表，简明地图解它的基本内容。

这个本子的目录可权充粗略的索引。

这个本子仍是很不成熟的，谬误之处，可能很多。敬请读者批评指教为盼！

这个本子由我站冯世烽同志编写，并得到本站软件组邓仁南、马锁生、陈德钊、程乃毅、赵志良等同志的协助，参考及引用了他们的译稿或研究结果。邓仁南、刘薇青、袁红玲同志帮助校阅了全文。这个本子的插图及附录中的简表都是宋锡琴同志绘制的。

南京电子计算站　1979年1月1日

SIEMENS BS1000操作系统

程序设计语言 FORTRAN Ⅳ

前　言

目　录

第六章 输入输出格式

第七章 子程序

参考资料

第〇章　在电子计算机上算题

0.1. 手算和电算

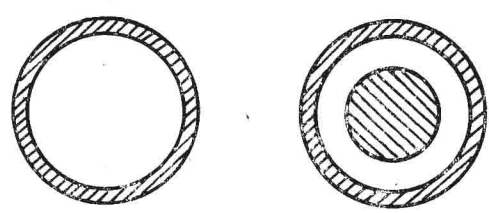
电子计算机是一种运算速度快、存贮容量大、计算精度高、功能广泛的计算工具。实际上，今日电子计算机的应用范围已经远远不是数值计算，它在科学研究、生产控制、经济管理、情报资料、国防军事等等方面无孔不入，甚至它已经在进入人们的日常生活。但是科学技术方面的数值计算，仍然是电子计算机应用的重要方面之一。实现四个现代化，科学技术工作者必须掌握电子计算机的使用。

但是，计算机是怎样计算的？我们又怎样在计算机上算题？

前一个问题随着计算机系统的不断完善，使用计算机的一般用户已经可以不必很关心了。但后一个问题，在一个相当的时期里，用户（我们把所有使用计算机的人都称为用户），特别是用计算机来算自己特定课题的科技工作者，是应该知道的。首先，他们应该对计算机有所了解；其次，他们应该会用程序设计语言设计算题的程序；第三，他们应该知道怎样在计算机上执行自己的程序；等等。

我们先来看看手算和电算有什么不同。举一个简单的例子：

有一批口径不同，厚薄不同，有芯管或无芯管（图 0.1）的管道。设已知其外直径do、壁厚h以及芯管的直径dco，要计算它们各自的横断面积。



甲：无芯管的管道　　乙：有芯管的管道

图 0.1

我们不妨只考虑有芯管的管道。问题的算法很简单。先求管道内直径

$$di = do - 2 * h \qquad (注)$$

而横断面积：$A = (\pi * di^2 - \pi * dco^2) / 4$；

化　　简：$A = \pi * (di^2 - dco^2) / 4$。

我们可以按下列步骤计算：

注：此表达式中的 * 号即为乘×号，见3.1.1.2.节，今后，我们以 * 作为乘号。

第1步。把要计算的第一根管道的外直径、壁厚、芯管直径登记在下表第一行do、h、dco项下；

序号	do	h	dco	di	di^2	dco^2	$di^2—dco^2$	A
1	610·0	10·0	200·0					

第2步。计算管道的内直径di，并把结果记在上表第一行的di项下；

等3步。计算di^2，记在第一行的di^2项下；

等4步。计算dco^2，记在第一行的dco^2项下；

第5步。计算（di^2-dco^2），记在第一行的（di^2-dco^2）项下。

第6步。计算$\pi*(di^2-dco^2)/4$，记在第一行的A项下。

然后，再对第二根管道重复上述6步。依次第三根、第四根……等等。最后，为了使用这些结果，必要时还要把它们腾抄下来。

当然，具体的步骤可以有多种多样，但对于复杂的计算，对多次重复的计算，总要做这些工作：造一张记载已知数据、中间结果以及最终计算结果的表，并在一开始和每一步计算后在其上登记这些数据；进行各步计算，可以笔算，也可以用计算尺、算盘、查表、以及各种手摇或电子计算器等等；最后，腾抄结果去使用。最重要的，是要由人先设想好计算步骤，再按步骤逐步去做。

这就是手算。对于有成千上万个数据，要进行成千上万次运算的题目，显然手算很难胜任。在电子计算机发明以前，人们采取了很多办法来提高计算速度。电动和电子计算器的速度比较高，但是每步计算仍要由人把数据和操作命令（就是怎么算：加减乘除等等）从键盘上打进去。计算器不管算得多快，这个由人手操作的部分总是拖后腿的。

电子计算机的发明，就是解决这个矛盾。电子计算机主要有这样几部分：和手算时有一个记载各种数据的表一样，计算机有一个存贮器（称内存贮器，简称内存），但是容量比一张表大得多；和手算时有运算器具一样，它有一个高速多功能的运算器；它还有一个负责指挥运算器等按步骤工作的控制器。（图0.2）和以前的所有计算器不同，在电子计算机上采取了一个革命性的措施。这就是把要计算机“怎么做”的一系列操作命令（我们称为指令序列——又称程序），全部存放在计算机内部的存贮器里。控制器将指令一条一条依次取来，然后分析执行。这就比人工在计算机外操作快多了。

把计算用的数据和计算程序放进内存贮器，称作“输入”。输入是通过输入设备（例如卡片阅读机、纸带输入机等）高速进行的。而把从计算机取得计算结果或中间数据，称作“输出”。输出是通过输出设备（例如行式打印机等）进行的。有的设备，如磁盘机、磁带机、

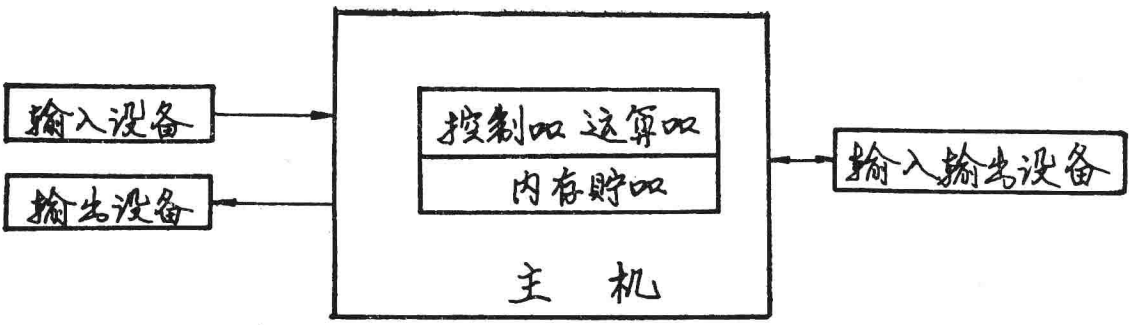


图 0.2

控制台打字机、远距离终端设备等，既可以用作输入，也可以用作输出，被称为输入输出设备。（磁盘和磁带又称作外存贮器。）这些设备统称外部设备或外围设备。

这样，当我们用电子计算机计算前述管道横断面积的问题时，只要做：

第一，把关于这个问题怎么样计算的指令序列——程序设计好，并输入计算机；

第二，把计算中要用的数据准备好，并按程序规定输入计算机。可以一次全输入，也可以分批；

第三，命令计算机开始按程序规定工作。

当然，不要忘记在程序里写上输出计算结果的命令。

当问题计算比较简单、重复计算次数很少，精度要求较低的情况下，可能看不出电算有什么好处，甚至会觉得麻烦。但是，在相反的情况下，我们将看到，电算显然比手算优越。有的时候，不用电算是不能解决问题的。

在电子计算机上解算复杂问题，一般应该有以下的步骤：

（1）对问题分析归纳，提出数学模型，就是提出能描述问题的一系列数学式子（方程式或公式）。

（2）根据数学模型，确定恰当的计算方法。

（3）根据计算方法的要求，设计程序的框图。所谓框图就是一组其内写有说明文字的方框或其他图形，之间用有方向的箭头衔接，可以描述程序执行的流程。常用的框图符号说明如图0.3所示。

（4）用程序设计语言编写程序，这种程序称作源程序。

（5）把源程序转变成计算机可以执行的形式。

（6）用事先设计好的一套数据来试算，以考验程序的正确性。这也称为调试程序。

（7）实际计算。

其中，（1）（2）两步是非常重要的。模型正确，整个计算才有意义。好的计算方法，不仅可以使计算迅速，而且使结果可靠。但这不是本书的范围，不拟多论。（3）（4）（5）三步也很重要，一些概念在下几节里介绍一下、而大部分内容，将在各章中结合FORTRAN语言来讨论，但也只能讲个入门。

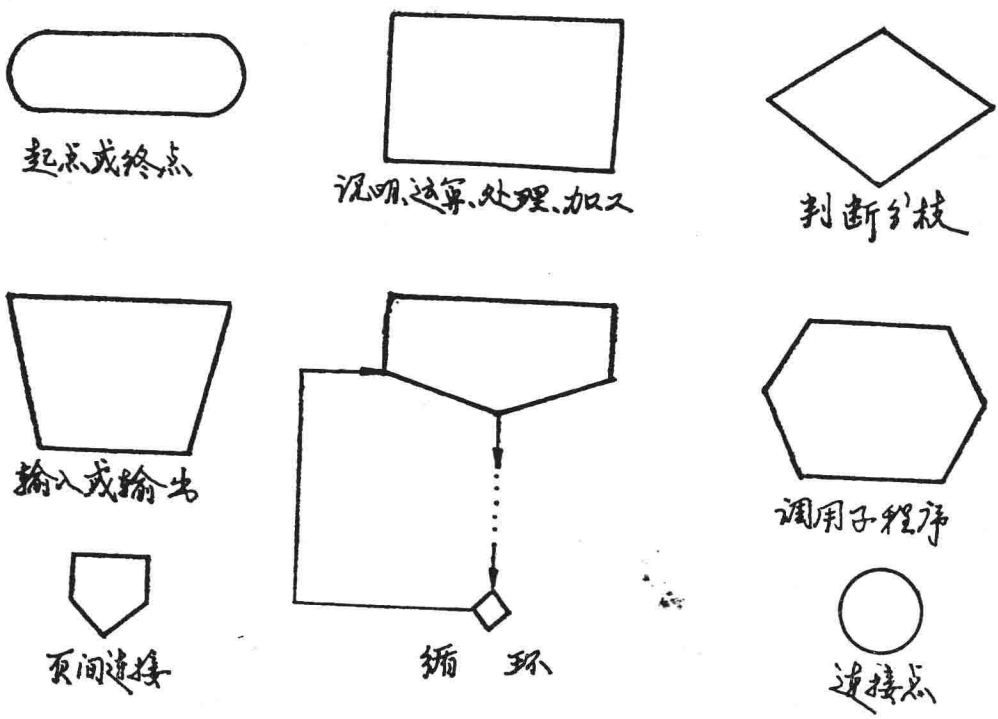


图 0.3 框图符号说明

0.2.程序设计语言

我们知道，程序就是一系列指令。每种电子计算机都有自己的一套机器指令，用这些指令就可以给计算机下命令。用机器“懂得”的这种指令编制的程序，称作是用“机器语言”编制的程序，也称“手编程序”。

每条机器指令都是一串数码，称为“代码”，放在存贮器里或写在纸上时，和其他数据并无二致。要用计算机算题，就得记住这一串串数码作为指令的意义，并且把它们编排好。在编排中，又还要安排和记住你用的每条指令和每个数据在内存贮器中的位置。随着计算的进行，还要清楚这些存贮内容发生的变化。对于算题的人，这是非常繁琐，非常费时费力的事。除了计算机的专业人员，旁人就很难应用计算机。于是，很快就出现了以符号代表指令数码和存贮地址的“汇编语言”。汇编语言和机器语言原则上是一一对应的。所以每一种计算机，有自己特有的一套汇编语言。在这种机器上用汇编语言编制的程序，在另一种机器上就不能用。而且仍然很不直观，不易编写程序，也不易交流。

五十年代中期开始，人们开始发展高级程序设计语言。FORTRAN、ALGOL、COBOL、PL1 等等相继出现。这些语言有一个共同的特点，就是都是“面向过程的”，而不象机器语言和汇编语言是“面向机器的”。它们比较接近数学语言和人们的自然语言，对计算过程的刻划比较清楚直观。例如要把变量X的值和变量Y的值相加，而把两者之和放到变量Z所占

的内存单元里去。用ALGOL语言，只须用语句

Z：=X+Y;

而用FORTRAN语言，则用语句

Z =X+Y

这与计算公式几乎一样。对计算过程的控制语句也是直观的。例如要计算

$$y=\begin{cases}-x & \text{当}x\leqslant 0\text{时},\\ x & \text{当}x>0\text{时},\end{cases}$$

ALGOL 语言用语句

'IF' X≤0 'THEN' Y：=-X'ELSE' Y：=X;

而FORTRAN语言用两个语句

Y=-X

IF (X·GT·0) Y=X

用这些程序设计语言编制程序，对算题的人当然是方便多了。但是，用这种程序去指挥计算机是不行的。因为计算机不“懂”这些语言，它只“懂”机器语言。不过这问题是可以解决的。既然语言不通，那为每种语言配一个“翻译”就是了。

对于一种高级程序设计语言，计算机专业人员为每种计算机研制了一套称为“编译程序”的程序。编译程序本身是用机器语言编制的。它的作用主要是把高级语言的每一个“语句”，“翻译”成机器语言的若干条指令。它不仅“译”，而且“编”。它可以根据高级语言所写的程序里的说明部分，为程序分配内存区域，编制表格。甚至可以为程序里局部的计算寻求最好的算法，如对表达式计算的优化等。经过编译，程序变成计算机“懂得”的可以执行的程序。我们称为“目标程序”。而把编译前用高级语言写的程序称为“源程序”。源程序是编译程序的“原始数据”，加工的对象。而目标程序是编译程序的“结果数据”，加工的产物。有了编译程序，人们就可以方便地通过高级程序设计语言来在计算机上算题了。

高级语言是通用的，也就是它们不依赖于机器。原则上，用某种高级语言设计的程序，在任何计算机上都可以执行，只要这台计算机配有这种语言的编译程序。这就使得一些成熟的、有效的好程序，可以供所有需要者使用，而不必重新设计。从而节约了大量的人力、物力和时间。

目前普遍使用的 FORTRAN 语言和 ALGOL 语言，适用于科学技术方面的数值计算。COBOL语言适用于各种数据处理。而PL 1语言则综合有这几种语言的广泛功能。

随着电子技术的发展，计算机的运算速度越来越快，存贮容量越来越大。这样再由人工来操纵管理，就无法发挥计算机的快、大的特点。最好计算机本身的管理也自动化。人们又研制了一大批各种各样的程序，由它们来管理分配内存和外部设备，使几个用户的程序能合理的使用这些“资源”；由它们来调度组织各个用户程序的执行；由它们来向用户提供调试追踪、错误诊断及数据和程序的保存等服务功能。这一大批程序，包括前述的各种高级语言的编译程序，就是计算机的“软件”或“软设备”。其中最重要的一套程序，就是“操作系统”。在操作系统的控制下，现代计算机是自动化运行的。只有如此，计算机才可能效率高功能广。

软件通常是放在磁盘或磁带上，需要的时候，调入内存执行其功能。

软件这个称呼是相对于计算机“硬件”或“硬设备”而来的。硬件是指前节我们提到过的主机、外围设备以及电源、空气调节装置等机器。硬件和软件合在一起，称为一个“计算机系统”。硬件是计算机的物质基础，软件使得这个基础的效能发挥出来。现代计算机没有软件是不能运行的。

0.3. 源程序的再加工

把源程序转变为计算机能执行的目标程序，即源程序的再加工，在不同的计算机系统上是不一样的。大致有以下几种：

第一种。先把编译程序装入内存，启动编译程序。编译程序把源程序作为输入数据加工，而产生的目标程序也在内存中。然后执行目标程序。下次计算，一切从头来。这种办法的好处是实现起来简单。缺点主要有二：首先是每次计算都要重新编译，在算题程序调试成功以后，这显然是不必要的，浪费了宝贵的机器时间。其次，编译程序在完成编译任务以后，仍然在内存里占有一片往往是很大的内存区域，不能再作其他用途。浪费了宝贵的机器内存空间。

第二种。与第一种不同是，把编译得到的目标程序输出到磁盘或磁带上，要执行它的时候，再装入内存。这就克服了第一种办法的缺点。

但是，这种办法仍然限制着程序的灵活性。在一个程序里，有一些程序部分是可以和其他程序通用的。例如，求一个一元函数在一段区间上的定积分，求一批数值的平方和，解一个N阶的线性方程组，求一个代数方程的根，对矩阵的各种运算，进行一次富氏变换，等等。按说，这些部分最好不用每次写源程序及编译调试。另一方面，在一个程序的调试过程中，某一局部有一点错误，因而把整个程序完全重新编译也是不合理的。特别是，对象FORTRAN这样的语言，他们有可以象搭积木一样组合起来的程序结构，就发挥不出这种结构的优点。于是，又有第三种办法。

第三种。编译程序把源程序上各特定的程序部分编译成各自独立的“目标模块”（MODULE）。它们是机器语言形式的，但还不是可以放进内存执行的。把这些目标模块存放在磁盘上的一个区域——“模块库”里。然后，根据用户提出的要求，由一个专门负责装配组合的系统程序（有的系统上叫“连接编辑程序”），把有关模块装配成一个可以执行的目标程序（有的系统上叫“相”，PHASE），并送到磁盘上另一个区域——“相库”存起来。在需要时，装入内存执行。这种办法就可以实现上述灵活性要求。

为了更有效地使用计算机系统，在目标程序的结构组成上还可以有其他种种改进。这里就不一一介绍了。

我们从西德西门子公司引进的SIEMENS 7.000系列的BS 1000操作系统，在形成可执行的目标程序时采用了上述的第三种办法，其过程如图0.4所示。

图0.4所示的三个阶段，都可以独立实行，也可连续实行两个阶段，或三个阶段一次完成。

对于SIEMENS 7.000系列的计算机，一切源程序的原始输入，大多采用穿孔卡片。用卡片的方法，在调试修改源程序时有巨大的灵活性。因此，源程序在程序纸上按规定写好以后，还应该交给穿孔员穿孔成卡片。并由用户仔细校对正确。

SIEMENS 7.000系列的计算机系统，和其他具有强功能的操作系统的计算机系统一样，是实行操作员制度的。就是用户不必亲自在机器上操作，也不进机房。要计算机完成的工作，全部按规定格式穿孔在控制卡片上。算题时，只要把控制卡片叠连同所需的源程序卡片叠和数据卡片叠或者这两种卡片叠之一交给操作员，并口头或书面向操作员说明作业内容就可以了。关于控制卡片的格式和用法，我们将在第九章介绍。

0.4.存贮信息的单位

存贮信息的单位及信息的内部表示这两个问题，都是因机器不同而异的。虽然高级程序设计语言是不依赖于机器的，但是高级语言写的程序，最终还是在一台特定的机器上执行。那这台机器内存放不放得下你的程序？这台机器可表示的数值范围有多大？以及在这台机器上怎样安排输入输出？等问题，都要求用户对这两个问题有所了解。

一切数值、字母、符号，以及它们的各种组合，都是计算机处理的对象，称为信息。信息在计算机系统内部表示成为一串串的数码。但电子计算机内部并不用常见的十进制数，而是用二进制数。就是每一数位只能取数字0或1，两个1相加就要进一位为10。这是因为电子计算机是由电子元件构成的，而用电子元件的两个稳定状态来表示数字是最好的。这两个稳定状态可以是，电位的高和低，脉冲的有和无，磁性元件的两种相反的磁化等。存贮信息的最小单位就是表示一个二进制数位的电子元件。抽象地说，就是一个二进制位（BIT）。

计算机的存贮器就象一长列房间，里面存放信息。为了从中取出信息或向里面存入信息，就要把房间编号。计算机的术语叫做“编址”。如果认为一个二进制位就是一个房间，从而编一个号。那一台中等容量的计算机的内存贮器就有几百万个二进制位，编号量很大，查找地址比较费时间。特别是，很多有实际意义的信息，如数值、字符、机器指令等，都不是用一个二进制位能表示的，往往要好多个二进制位组合起来才能表示。用二进制位作为编址的最小单位显然是不必要的，也不太好。

不少计算机系统上用“计算机字”，简称“字”（WORD），作为编址的单位。一个字有若干个二进制位。二进制位的个数称为“字长”，例有字长16位、32位、48位、60位等等之分别。但是，一些在国际上有影响的计算机系列，例IBM系列等，并不是以字为编址的基本单位，而是以比字短的字节(BYTE)作为编址基本单位。同时保留字的概念。SIEMENS7.000系列也是如此。

每一个字节是由9个连续的二进制位组成。由于其中有一位是用于硬件校验的，所以从逻辑级别上讲，一个字节由8个连续的信息位组成。在存贮器里，每个字节都编以地址号码，第一个字节地址为0，第二个字节地址为1，第三个为2……等等。每个字节最多可以表示256种不同的信息状态。重要的是，我们可以根据地址向任意一个字节存取信息。

在一个字节内的二进制位，也是有顺序的：第1位，第2位，……，第8位，或称2^7位，2^6位，……，2^0位。这样，即使有的有实际意义的信息只用一个二进制位表示，也可以兼顾。

在SIEMENS 7.000系列上，每个字由连续4个字节组成，也称整字或全字。字的地址就是用它的第一个字节的地址。为了寻址方便，规定字的地址数应该能被4整除。

相对于整字，还有半字和双字的概念。半字和双字是分别由连续2个和8个字节组成的存贮单位，它们的地址也用它们各自的第一个字节的地址。这个地址应该分别能被2或8整除。

常用的一个说明存贮容量的单位是千字节(KB)。1KB等于1024字节。例如说SIEMENS 7.730主机内存贮器的容量是512KB，就是指它有524288字节，相当于131072字（字长32位）。

0.5. 二进制数和十六进制数简介

各种信息在机器内都用一串二进制数表示，即表示为0和1组成的序列。机器内部的编址、计数也是这样。往往一个信息要用几十位二进制的0和1表示，这在书面上写时很不方便，看时眼花瞭乱。由于二进制数与十六进制数之间有很直观的对应关系，所以常用十六进制数作显示内部的外部书写。

用高级语言编写程序，一般只用十进制表示数，但有时也要用到信息的内部表示。而且在编译、连接、调试程序时，计算机将打印出很多资料和错误信息，里面也用到十六进制数。但由于上述两方面用途主要都不涉及小数，因此这里介绍二进制和十六进制数时只谈正整数。

二进制是逢二进一的数制。它们基数是0和1，也就是每个数位只能取0和1这两个数字之一。

而十六进制是逢十六进一的数制。它的基数有十六个：0，1，2，3，4，5，6，7，8，9，A（十），B（十一），C(十二)，D（十三），E（十四），F（十五）。每个数字可以表示十六个数值。

十进制，二进制，十六进制这三种数制里表示同一数值时的对照举例如下：

二进制数	十进制数	十六进制数
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E

1 1 1 1	1 5	F
1 0 0 0 0	1 6	1 0
1 0 1 0 0	2 0	1 4
1 0 1 0 1 1	4 3	2 B

这三种数制间的转换方法如下：

（1）十六转十或二转十。

设以二进制（或十六进制）表示的数为

$a_n a_{n-1} \cdots a_1 a_0$

则其等值的十进制数按下列公式计算：

$$N = \sum_{k=0}^{n} a_k b^k$$

其中，b——在二进制时为2，在十六进制时为16；

a_k——基数，在二进制时为0或1，在十六进制时为写成十进制数的0—15这十六个数之一；

n——二进制或十六进制正整数的位数减1。

例.

化 $(1\,000\,1\,0)_2$为十进制数

$$N = 1*2^5 + 0*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 0*2^0$$
$$= 32 + 0 + 0 + 0 + 2 + 0$$
$$= 34$$

即 $(100010)_2 = (34)_{10}$

例.

化 $(7\,FFF)_{16}$为十进制数。

$$N = 7*16^3 + 15*16^2 + 15*16^1 + 15*16^0$$
$$= 28612 + 3840 + 240 + 15$$
$$= 32767$$

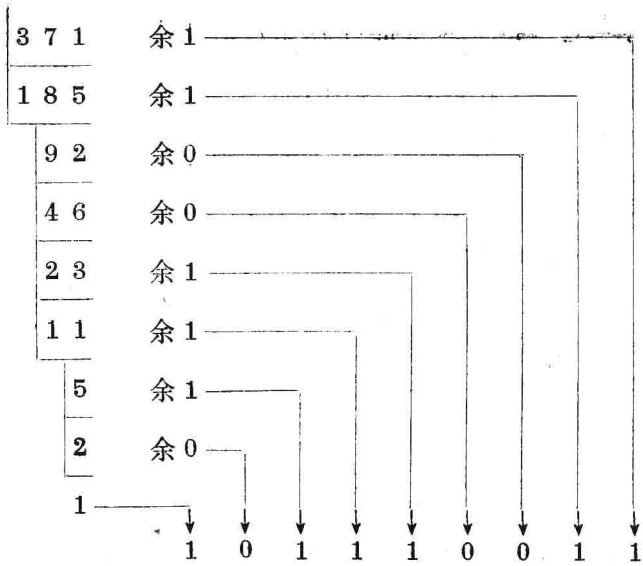
即 $(7\,FFF)_{16} = (32767)_{10}$

（2）十转二。

将该十进制数逐次除以2，而将每次的余数（0或1）依次从右向左记下，所得就是同值的二进制数。

例.

化 $(371)_{10}$为二进制数。



即 $(371)_{10}=(1\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1)_2$

（3）二转十六。

注意每四位二进制数对应一位十六进制数。把该二进制数从个位向左每四位分作一组，把每组的值用十六进制数字写出即可。

例.

化 $(1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1)_2$为十六进制数。

1 0100 1010 1101
↓ ↓ ↓ ↓
1 4 A D

即 $(1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1)_2=(14AD)_{16}$

（4）十六转二。

把（3）的过程倒过来进行。

例.

化 $(3\ E\ 9\ B\ 7)_{16}$为二进制数。

3 E 9 B 7
↓ ↓ ↓ ↓ ↓
0011 1110 1001 1011 0111

即 $(3\ E\ 9\ B\ 7)_{16}=(11111\ 0\ 100110110111)_2$

（5）十转十六。

先做十转二，再做二转十六。

例.

化 $(4126)_{10}$为十六进制数