程序设计自动化系统

〔苏联〕 Н. П. 脫利峰諾夫 М. Р. 舒拉-布拉 编

徐家福 周志明 譯

上海科学技术出版社

內 容 提 要

本书闡述了苏联国立莫斯科大学計算中心所研制的程序設計自动化系統.本系統是以利用汇編程序、編譯程序以及一系列的檢查程序为基础的. 全书共三部分,十三篇文章,依次对本系統作了連貫的叙述,可供熟悉程序設計基础并在这一領域內工作的讀者参考.

СИСТЕМА
АВТОМАТИЗАЦИИ
ПРОГРАММИРОВАНИЯ
Н. П. Трифонов и М. Р. Шура-Бура
Физматгиз, 1961

程序設計自动化系統

徐家福　周志明　譯

上海科学技术出版社出版（上海瑞金二路450号）
上海市书刊出版业营业許可証出093号

上海新华印刷厂印刷　新华书店上海发行所发行

开本 850×1156 1/32　印張 4 24/32　排版字数 124,000
1965年1月第1版　1965年1月第1次印刷
印数 1—4,300

統一书号 13119·637　定价(科六) 0.75 元

序　言

本汇編中陈述了在建立程序設計自动化系統方面的一些工作成果. 这些工作都是 1956～1958 这几年内在以 М. В. Ломоносов 为名的国立莫斯科大学的計算中心在"箭牌"机器上完成的. 本汇編涉及到程序設計自动化問題中相当广泛的課題,如程序的編制,所編程序正确性的檢查,以及在計算过程中机器工作的正确性的檢查等.

所述的程序設計自动化系統的基础是編譯性方法,这是以前在国立莫斯科大学的計算中心就 М-2 机器所研制的标准子程序系統❶的进一步的发展.

本汇編中所援引的一列程序 (60СП, ССП-2, 77СП,計算檢查程序)业已經过长时期的使用,实际工作确实表明了它們是有成效的,并且使用起来很方便. 至于調試程序的程序以及編譯程序,虽然还缺乏足够的实际使用經驗,但是可以期望,它們也都是很有成效的.

所有納入本汇編中的工作全都是在一个統一的程序設計自动化系統的范圍內,因此,本汇編实际上是一本完整的书,其中給出了这一系統的依次而連貫的陈述.

本汇編可供熟悉程序設計基础的人員使用,在所引的程序的邏輯图中应用了熟知的記号❷. 在叙述过程中,必要时仍对所用記号的具体內容作了說明.

对标准程序庫 (НСП) 所編的一些程序都被完整地引入了. 为

---

❶ 参見 М. Р. Шура-Бура 主編的 «Система стандартных подпрограмм» 一书, Физматгиз, 莫斯科, 1958.

❷ 参見 «Проблемы кибернетики» 汇編第一期上 А. А. Ляпунов 的文章 «О логических схемах программ», Физматгиз, 莫斯科, 1958.

了分析它們，讀者需要了解“Стрела-4”(箭牌-4)机器的指令碼❶，以及НСП的工作原理．仅对程序工作的算法及其邏輯图感兴趣的讀者可以掠过对具体程序的分析，这样并无害于理解后面的內容．

在所有程序的描述中，为了指出各块的长度，将利用八进制的計数系統．

Н. П. Трифонов, М. Р. Шура-Бура

❶ 关于“箭牌”型的机器的必要知識可在 А. И. Китов 及 Н. А. Криницкий 合著的《Электронные цифровые машины и программирование》(Физматгиз, 莫斯科, 1958)一书中找到．“箭牌-4”机器和这一类型的别的机器不同，它具有运算碼为 23 的运算，指令

a b c 0 23

表示：把单元 a 的內容送往单元 c，并且按地址 b 无条件控制轉移．

目　录

此为试读,需要完整PDF请访问: www.ertongbook.com

引　論

Н. П. Трифонов, М. Р. Шура-Бура

如所熟知，程序乃是用机器的基本运算的术語对問題解法的广义算法的描述，而程序設計中的主要困难和下述事实有关，即：一些复杂的运算不可能列入具有一定限度的基本运算組中，否則便要使机器的結构大为复杂；使得程序設計复杂化的另一事实可能是数的表达形式，因为就定点机器进行程序設計时，在选取比例因子中要产生很大困难.

为了克服上述困难，存在有程序設計自动化的各种方法，它們之間的差异在于：用来描述算法的語言的选法不同以及在机器上对于用該种語言所描述的算法的实現方法不同. 在程序設計自动化的各种不同方法中可以提出两种流行最广的方法.

程序設計自动化的第一种方法是：描述算法的語言最大限度地接近通常的数学符号，尽可能与机器无关. 例如，我們可以把問題解法的算法描述表示成用算子术語写成的邏輯图的形式. 这种情形，称为編譯程序的专用程序. 便根据邏輯图中所給出的信息，編出写成眞机器碼的程序.

这些方法中的第二种方法以使用标准子程序为基础，其实质如下：由于机器是通用的，所以借助已有的基本运算組可以实現任一个我們所希望的运算——为此，只須編制相应的子程序. 在該子程序中，可以备有一些数的运算，其形式和所用机器上数的表达形式不同. (例如，在定点机器上备有数的浮点运算.)这样，用程序的方法便可在机器上引入任何我們所需要的基本运算組以及数的表达形式.

这时，为了給出关于在所考虑时刻需要执行什么运算 (亦卽，什么子程序)，需要对那些量进行运算等等的信息，須使用某一种

約定的語言（条件語言）。特别是，这种語言在給出信息的形式上可体現出是某一台假想机器的通常指令，而这台假想机器的基本运算組，数的表达形式以及地址等都是用程序的方法来引入的。

这样約定的給出信息的形式称为伪指令，因为这种指令（指令系統）并不属于某一台現实机器，而是属于某一台假想的机器的。

为了使現实机器能够执行所給出的伪程序（亦即，用伪指令所編成的程序），需要有一个翻譯程序，后者应当把每一伪指令"翻譯"成現实机器的語言，亦即，根据在伪指令中所給出的信息，作出一串机器指令，借以实現伪指令中所給出的运算。照例，这种翻譯工作总是借助标准子程序来实現的——对每一伪指令，翻譯程序把相应的标准子程序調到机器的运算存儲器中，并把該子程序"調整"成适合于对所給出的一些量来进行工作。

由翻譯程序所調用的子程序的执行可以用下列两种方法之一来实現。第一种方法是：在譯好每一伪指令之后，紧接着便执行相应的标准子程序。这样的翻譯程序，其工作在于用机器运算来"解釋"伪指令，称之为解釋性程序。

这种方法的主要优点是：由于解釋与执行每一伪指令和其他伪指令无关，故解釋工作簡单。这种方法的另一优点又在于：为了进行解釋工作，需要运算存儲器的容量不大。

另一方面，由于当每一伪指令重复利用时，每用一次都得重新解釋一次。这样，利用解釋性程序便带来了机器时間的大量損耗。由此显然可知，解釋性程序基本上是用于运算存儲器容量小而速度却相当快的机器。在运算存儲器容量相当大的机器上利用这种方法是不合理的。

翻譯程序的另一种可能的工作方法是：把伪指令換成写成机器語言的相应子程序，然后，再把这些子程序联成一个工作程序。在整个伪程序都翻譯成眞机器碼以后，所得到的程序才能执行。这样，程序的翻譯和执行就明显地划分为两个时間阶段，这一点却是解釋性程序所沒有的。在这种情形下，翻譯程序的作用在于：譯伪指令并把各别的子程序联成（編譯成）一个单一的工作程序，这

类翻譯程序称为編譯性程序.

如果机器是对浮点系統中的数工作的，并且指令系統足够完备，那末，在这样的机器上利用伪指令来实現程序設計自动化，看来是不会有多大成效的.（例如，我們指出，БЭСМ-1 及箭牌等机器，它們具有計算初等函数的专門設备.）在这样的机器上，不用伪指令的簡化的編譯性程序得到了广泛的使用. 这些程序的基本职能是：由备好的程序的各别部分（特别是，由标准子程序）汇編成工作程序. 类似的編譯性程序称为汇編程序.

对箭牌机器，程序設計自动化的研究工作基本上是按照建立有效的編譯程序（ПП）及汇編程序的途徑来进行的. 这两种方法中的每一种各有其优缺点.

編譯程序的方法比較更为一般些. 但是在 ПП 的实际应用中却遇到一些困难. 基本的困难是：在用 ПП 时，問題的准备工作是按照一定的規則进行的，这些規則和具体 ПП 的特性有关，因而，为了掌握这些規則，就需要相当时間以及实际的工作經驗.

借用 ПП 所得到的程序的調試工作也和手編程序的調試工作不同. 所有这一切都归結为：为了用 ПП 工作，需要一些較高质量的专家.

为了利用 ПП，問題的准备工作通过若干阶段：編邏輯图，譯碼，穿孔，并且为了加速問題的进程并减少錯誤起見，在問題的准备过程中，总要求在每一阶段，工作都应由对該阶段工作受过特殊訓练的人員来完成. 由于这一点，便需要很复杂的同时又是很明确的工作組織.

此外，編譯程序所編出的程序，其质量可能低于由熟练的程序設計員所編出的程序. 所以，在一系列情形中利用 ПП 就显得并不合适.

汇編程序的方法可以大大减輕新程序的編制及調試工作，如果在所考虑的程序中可以利用前已編好、并調試好的子程序，那末，程序設計員所应重新編制的只是程序的那些不足的部分，然后再把所有各部分联成一个单一的工作程序.

这里的一个严重缺点是：程序的不足部分必须手編，因此，程序設計自动化的程度实质上就和待解問題的种类、子程序庫的組成以及把各别部分联成一个单一的程序的方法等事有关.

当在机器上待解問題的类型十分广泛时（例如，在МГУ❶的計算中心所出現的情形），最适宜的程序設計自动化系統可能是这样的：它可以容許利用上述各种方法中每一种方法的优点. 在МГУ的計算中心研制程序設計自动化系統时，在某种程度上就提出了要滿足前述要求的課題.

作为本程序汇編中所陈述的系統的基础便是編譯性方法，該方法借用标准汇編程序（ССП-2）得以很簡单地由各别的标准块（亦卽，备好的程序部分，它們是按照某些标准規則所編成的条件地址的程序部分）得出工作程序. 程序庫中的子程序以及不在程序庫中的某些已准备好的程序部分都可以作为标准块.

标准块庫和ССП-2的組織乃是以前在МГУ計算中心为М-2机器所研制的标准子程序系統的进一步发展.

本編譯程序（ПП）是程序設計自动化系統的一个組成部分，在它的研制过程中曾經参考了以往所編制的若干編譯程序的經驗. ПП的基本职能在于：使那些不能用現成的标准块来代替的程序部分的編制工作自动化. ПП的工作結果也是标准块，它們在往后为了借助ССП-2得出工作程序，可以和其他标准块同等地使用. ПП在使用方面的这一特性簡化了其实际应用. 但是，在其普遍性及通用性方面并未受到任何限制，如果我們愿意的話，也可以独立使用ПП.

此外，以試驗的形式編出了这样的程序，它解决了ПП的逆問題，卽把由ПП所編出的程序的邏輯图綜合出来.

在机器的运用中不仅程序編制过程的自动化具有重大意义，而且所編程序的調試过程的自动化以及在解題过程中机器工作正确性的檢驗工作的自动化也都具有重大意义.

为了使得程序調試过程自动化，在МГУ計算中心研制了一种

❶ МГУ系"国立莫斯科大学"的縮写.——譯者注

非常通用的檢驗程序以及它的簡化的变形，后者在实际使用中往往更方便一些.

为了檢驗解題过程中机器工作的正确性，編制了檢驗計算的标准程序. 借助这一程序，把問題的解法过程分为若干各别的步驟，其中每一步都进行两次(三次)复算，以比較存儲器中的状态是否相同. 借助 CCII-2 可以很容易地把这一檢驗程序嵌入任一程序中.

在本程序汇編中給出了所有前述程序的描述以及相应系統的总的描述.

第一部分　利用子程序庫的程序設計系統

1. 利用子程序庫的程序設計系統

Е. А. Жоголев

§1　利用汇編程序的程序設計系統

求解各种問題的程序編制工作，如果在所給問題的程序設計中有可能利用先前已經編好的程序，而不需預先加以手工改造的話，那末，在一定程度上就可以減輕負担，而且在相当大的程度上簡化了这些程序的調試工作．可是，当我們在編制一个較复杂的程序而要用到另一程序时，由于在編制各种不同的程序中要把同一个程序放到存儲器的不同位置上，于是便引起了一系列的困难．为了更好地了解这些困难，以及拟訂克服这些困难的途徑，有必要先对程序的結构进行一些分析．

在求解每一問題时，我們总是把机器上的存儲器分为若干块，在这些块中既可以放程序的各别部分，也可以放各种量(变量或常量)的值．在解題时，一般說，各块間都有一定的相互联系．例如，由一块到另一块(如果这些都是程序块)的控制轉移，在某一块(程序块)工作时，要利用另一数块的数值或者把結果記入另一数块．

如果要改变这些块中的某一块在存儲器中的位置，一般說来，便需要把一些程序块进行不同程度的加工改造．我們可以借助于专門的程序来使各块的改造工作由机器本身自动地进行．但是，在

此为试读，需要完整PDF请访问：www.ertongbook.com

这里必须使这些程序块都满足一定的要求，这些要求都是由加工程序决定的．满足这些要求的程序块，亦即，在求解各种問題时可以存放在內存儲器的各种不同位置的程序块称为标准块．不满足这些要求的程序块，亦即，仅当这些块以及(或者)与前者相关的各块存放在內存儲器中固定位置时才能正确工作的程序块称为确定块(或固定块)．往后，将专門考察标准块，虽然有时和标准块一起，也利用一些确定块．

首先必須抽出一类簡单的标准块，即所謂不变块．这是指：在求解該問題中改变所利用的任何块在內存儲器中的位置，而其状态并不会因之而改变的块．例如，如果把数块看成程序块的特殊情形，那末，所有的数块便都属于不变块这一类．如果某一块的状态依賴于它在內存儲器中的位置，而与参与解題的其他各块在內存儲器中的相互位置无关，那末这样的块便称为閉块．其状态不仅和它本身在內存儲器中的位置有关，而且也和参与解題的其他各块在內存儲器中的位置有关的块則称为开块．

在求解某一問題时，如果要改变各块的位置，照例，只須把属于各块的代碼(指令或常数)的地址部分加以相应改变即可．而重要的是要知道，在某一指令中，或者一般說，在一代碼中这一或那一地址是如何依賴于各块在內存儲器中的位置的．在这里，我們把地址了解成不仅是某一存儲单元的地址，而且也是指写在指令的相应各位的任一代碼．运算碼依賴于各块在內存儲器中的位置的情形，我們将不予考慮．

当任意改变各块在內存儲器中的位置时，这一或那一代碼的某些地址并不因之而改变，这样的地址称为不变地址．在指令中起輔助作用的任一地址就都是不变地址．例如，为了对某一成組运算的重复次数編碼而用的地址便是不变地址．在这种意义下，任何数的整个地址部分都是不变地址．此外，在一系列的机器上固定有一列存儲单元作为标准工作单元，这时，属于这些单元的地址也都是不变地址．

如果这一或那一指令中(或者一般說，代碼中)的地址仅依賴

于包含該代碼的那一块在內存儲器中的位置，这样的地址便称为內系地址. 下述地址便是內系地址的一个例子，即由該块(含有所論地址的块) 的一条指令按該地址控制轉移到同一块的另一条指令的地址.

如果一块的这一或那一行中的地址依賴于参与解題的其他各块在內存儲器中的相互位置，或者依賴于程序的某一参数，便称該地址为外系地址. 下述地址可以作为外系地址的例子，即把所考虑的块的某一条指令的执行結果按該地址記入属于另一块的存儲单元中的地址.

这样，不变块仅含不变地址；閉块，除了內系地址以外，还可以包含不变地址，而开块，除了外系地址以外，也可以包含不变地址和內系地址. 現在，对于在內存儲器中具有各种不同相互位置的各块的利用問題便可以归結为內系地址及外系地址改造工作的自动化問題了.

內系地址的改造工作只需要唯一的一个信息，即其所属的块在內存儲器中的位置(这里假定，据以区别內系地址和所有其他地址的法則已經知道). 这样的改造工作宜于在把該块輸入到內存儲器的某一位置时进行，因为在輸入指令中就包含了改造內系地址的必要信息. 对于 M-2 机器，这一点是在标准子程序"輸入程序"中实現的❶. 稍后，关于箭牌-4 机器，这一点是在 60CII 中实現的(参見下面关于 60CII 的文章).

外系地址的改造問題就要更复杂一些. 为了改造外系地址，需要关于求解所論問題中用到的所有各块在內存儲器中的相互位置的信息，关于程序中的参数值的信息，以及关于这一或那一外系地址属于那一块中的那一行或者属于程序的那一参数的信息 (这里仍假定，据以区别外系地址和所有其他各类地址的法則是已經知道的).

为了給出这样的信息，必須对每一块賦与某一个碼，例如，它

❶ 参見 M. P. Шура-Бура 主編的《Система стандартных подпрограмм》一书，Физматгиз, 莫斯科，1958.

的号数．这时，关于各块在內存儲器中相互位置的信息可以用內存分配表(TPII)的形式給出．內存分配表也可以用来把必要的块輸入到存儲器中．

我們必須对每一个具体問題具体地来分配內存．在这里需要有关于所用各块的长短的信息，以及关于在求解具体問題过程中利用它們的特征的信息．在特例，常常需要把一系列的块(特别是工作单元块)放到內存儲器的相同位置，如果这样的存放并不引起程序工作錯誤的話．如果所有这些信息都依某种方式輸入到机器中，那末，根据每一块在內存儲器中位置的計算，亦卽，根据 TPII 的形成，便可以把全部(或一部分)机械工作委托給机器来做．为此目的，宜于把所有各块按照在解題时利用它們的特征分成三組：

属于第一組的是一些这样的块，卽它們在內存儲器中可以具有任意的初始状态，后者对解題是无影响的（例如，工作单元块就是如此）．因而这样的块并无必要輸入到机器中．这样的块称为消极块．

属于第二組的是一些这样的块，卽在解題以前它們并不能有任意状态的块(例如程序块)，因而，必須預先把它們輸入到机器中并且要接受相应的改造，这样的块称为积极块．

此外，我們将分出一些这样的块，卽在分配內存时它們和一些其他的块，或者和一些与之有关的部分放在一处，这样的块称为联通块．把联通块輸入到机器中，并且加以相应的改造是可能的．

在自动編制 TPII 时，块的位置，一般說，将是根据不同的算法来計算的．这些算法依賴于这一块属于那一組．例如，任一联通块在內存中的位置，只有当和它放在同一处的联通块的位置确定以后，才能确定．

除了輸入参与解題的块在內存中位置的信息外，还宜于輸入关于程序的各参数值的信息．对于每一个这样的参数，可以賦与某一个块的号数，并且对于和任一参数相联系的外系地址，也按照“改造和任一块相联系的外系地址”的同样法則来加以改造．凡是其号数我們約定作为表示程序的参数的块則称为虛拟块．在解題

时这些块在內存中是不占位置的，因此它們可以属于前述三組中的任一組．为了給出某一参数的值，只須在 TPII 中給出相应的虛拟块在內存中的开始地址，該地址乃等于所論参数的值．今后，我們不打算专門来討論和程序的参数相联系的外系地址的改造問題，而假定，这样的改造工作是按照前述一般法則来进行的，即考虑到前述的程序的参数借助于虛拟块的解釋方法．

关于改造外系地址所必要的信息的第二部分，亦即关于某一外系地址属于那一块中的那一行或者属于程序中的那一个参数的信息，可以用各种不同的方法給出来．例如，可以把整个这一信息在这外系地址中指明出来．为此，只須把这一外系地址分为两部分，在一部分中指出块的号数，而在另一部分中指出該块中行的号数．

我們认为下述的另一种方法更要方便一些．在这种方法中，我們对每一个开块，附以一个补充表，即所謂外系地址表(TBA)，在这个表中給出該块中的所有外系地址的全部必要的信息．和前一种方法比較起来，这种方法有如下的优点：首先，在第一种方法中对块中所容許的行的最大数目以及对标准块庫中所容許的块的最大数目都加上了較强的限制．其次，当对外系地址的所指信息予以某种改变时，例如，当改变某些外系地址所属的块的号数时，第一种方法，事实上必須查看对它說来是改变外系地址的信息的整个一块；而第二种方法則只須查看相应的外系地址表．

如果对庫中有的每一标准块經常固定一个确定的号数，那末，对于每一块(当然是开块——譯者注)的 TBA 便也是标准的了，因而当編制每一个新程序时，只須編制标准块庫中所缺的块，以及編制为了求解該問題所需的 TPII 或者給出自动編制 TPII 所必須的信息．

也可以借助編譯程序来編制新块，这里只須要求編譯程序給出它所編的各块是标准的．本程序自动化系統可以有成效地用于各种自动計算机上．下面将对箭牌-4 机器詳細說明这一系統．該系統是以利用标准汇編程序 CCII-2 以及 60 CII 为基础的．

§2 标准汇編程序的职能及其利用

在討論标准汇編程序 CCII-2 的特性以前，必須說明該程序对标准块所加的要求，亦卽，必須回答这样的問題：应該如何来編制标准块，应該按照那些法則来区别不变地址，內系地址以及外系地址.

标准块的編制

任一个閉块或开块都应該从地址 2000 編起，亦卽，該块的第零行（这里假定，每一块的行都是自第零号起編号）将具有地址 2000，第一行将具有地址 2001，依此类推. 这时，所有內系地址 s_i 将用区間 $2000 \leqslant s_i \leqslant 3777$ 中的号数表示出来. 只有成組运算的第二地址（它們是用来对成組运算的重复次数进行編碼的）以及該块的末端存放数的行中某一个数的全部地址（在这些地址中常常必須存放具有各种不同地址部分的碼子）为例外.

按这种方式所給出的內系地址，其改造工作是借助 60CII（参見下面关于 60CII 的文章）来进行的，如果利用下面的指令来轉向 60CII 的話：

N	n	a_H	1	60
0000	n_1	0000	0	00

这里，当块是由穿孔卡片輸入时，$N=0000$；如果块是由磁带輸入的，N 便表示磁带的号数；当由內存到內存的傳送 H→H 时，N 便表示存儲单元的地址；n 是該块中的行数（不計檢查和那一行），亦卽，n 等于檢查和行的地址与块首地址之差；a_H 为块首地址；n_1 表示块中起首的需要改造內系地址的行数. 这里在具有号数 $2000 \div 2000+n_1-1$ 的諸指令中，所有形如

$$s_i = 2000 + k_i$$

的內系地址都将換成下面的眞地址：

$$a_i = a_H + k_i.$$

对每一个标准块必須賦与一个确定的号数 M_k，这里 M_k 属于

下述区間：

$$0001 \leqslant M_k \leqslant 7777.$$

在編制开块时，外系地址应当按下列方式来編碼，即对于一个外系地址取区間 $6000 \leqslant e_i \leqslant 7377$ 中的某一条件数 e_i 与之对应，只要这个 e_i 在該块中并不用来表示其他的外系地址即可．与此同时，在該块的 TBA 中填上相应的一行：

$$e_i \quad M_i \quad \Delta_i \quad 0 \quad 00,$$

其中 e_i 为外系地址的条件表示，M_i 为該外系地址所属的块的号数，Δ_i 为外系地址在該块中所对应的行的号数．

有时也可以把标准常数存儲器(HK)中的地址作为外系地址，如果我們需要对它們进行某一改变的話（例如，当重新分配 HK 时）．

这样，外系地址便了解成区間

$$6000 \leqslant e_i \leqslant 7777$$

中的地址 e_i，我們把它补充地固定在該块的 TBA 中．和內系地址的情形一样，在該块的末端用来可存放某些任意代碼的各行中的某一个数的全部地址为例外．

为了在 TBA 中书写信息簡便起見，可以成組地給出信息．在这种情形下，行序列

$$\begin{array}{ccccc} e_i & M_i & \Delta_i & 0 & 00 \\ e_{i+1} & M_{i+1} & \Delta_{i+1} & 0 & 00 \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ e_{i+m_i} & M_{i+m_i} & \Delta_{i+m_i} & 0 & 00 \end{array}$$

在 TBA 中換成两行：

$$\begin{array}{ccccc} 0000 & m_i & 0000 & 0 & \theta_i \\ e_i & M_i & \Delta_i & 0 & 00, \end{array}$$

如果

$$e_{i+k}=e_i+\varepsilon_1 k, \quad M_{i+k}=M_i+\varepsilon_2 k$$

并且

此为试读，需要完整PDF请访问：www.ertongbook.com