中山大学学报
1986年第3期 ACTA SCIENTIARUM NATURALIUM UNIVERSITATIS SUNYATSENI No. 3, 1986

关于P代码的翻译

肖金声
(计算中心)

摘 要

本文介绍P-code的一种翻译方案。它能彻底抹掉栈式机器的痕迹，较好地利用单运算器的特点，从而产生质量较好的目标代码。

PASCAL语言之所以得到广泛的承认和普及的原因之一，是它有一个可移植的编译程序Pascal P4。

Pascal P4处理PASCAL的一个子集，产生的目标是假想栈式计算机(SC)上的汇编语言，即所谓的P-code。关于Pascal P4，清华大学的杨德元已在“PASCAL可移植编译程序及其分析说明”一书中作了很好的介绍。事实证明，国外确有一些计算机上的PASCAL编译程序是通过Pascal P4进行移植而得到的，其优点是代价小、见效快。

诚然，Pascal P4所实现的并非完整的PASCAL，而是它的一个子集。其限制是：

1．过程和函数不能用作参数；

2．不能用GOTO语句转出过程体或函数体；

3．只允许预定义的text类型的文件，而不允许任何其他文件；

4．无紧缩(PACKING)处理。

另外一个改变，就是用标准过程MARK和RELEASE代替PASCAL修改报告中的DISPOSE过程。

由此可见，所实现的是一个并不算小的子集，对于一般的应用已相当够了(Pascal P4本身就是用这子集写出的)。更何况实现之后，人们还可以根据各自的需要加以自展扩充。

尽快在国产机上移植Pascal P4，是摆在我们面前的一个很自然的课题。

一、在国产机上移植的困难

为迅速而广泛地移植PASCAL编译程序，Pascal P4的作者U．Ammann和K．Je-

本文1985年10月收到

uson早就准备好了一组完整的文件资料：

1．PASCAL源语言形式的P编译程序（即Pascal P4）；

2．P-code形式的P编译程序（前者的译本）；

3．PASCAL源语言形式的P-code的汇编、解释程序，它确定P-code的语义；

4．PASCAL语言报告(Jensen and wirth 1974)；

5．有关SC的文件，它提供了P-code程序的语法和其它有关资料。

整个移植工作可根据这组文件资料来进行。特别值得一提的是，上述文件1是参数化的，允许移植者根据目标机器字长和编址情况指定表示各类型数据所需单元数，Pascal P4便能据此进行单元分配。由此而得出的文件2相应地也会有所不同。

此外，他们还就移植策略提出了三个不同的建议方案：

A．为目标机器写一个P-code的汇编、解释程序

即把文件3作一次笔译。实现这一方案，所需存贮在100K字节以上。采用此方案时，编译和运行都是解释执行的，因此效率相当低。

B．建立P-code的宏处理程序

通过宏处理而得出的编译程序必须要占据大量的存贮空间；同时，如果目标机器不是栈式机器，编译程序便要模拟栈式机器来进行工作，而这也是低效的。

C．利用“线索码”技术

如果说方案A是解释性的、方案B是翻译性的，那么现在的方案C则是一种折衷，即翻译和解释的混合。其做法是，首先根据目标机器的特点设计一种中间语言；其次，为该中间语言建立一个运行程序包；然后写一个汇编、解释程序来执行P-code程序。此方案与前述资料3的不同之处在于中间代码的级别和线索码的特定执行方式，

此外，他们还提出了一个不经P-code的第四方案。如果实现者手头上除了目标机器之外，还可以使用一台能运行PASCAL的宿主机，而且大得足以运行Pascal P4，那么实现者只需把文件1的代码生成部分修改成产生目标机器的代码，然后让它自编译一次，就可得到高效的编译程序。具备这种条件的实现者当然不多，为了不失普遍性，我们限于讨论仅用目标机器而又不太困难就能获得高效编译程序的方案。

参考前面三种方案，考虑在国产机上进行移植应采取哪一种策略，就有必要看看国产机的共性。为了使适应面广，我们着眼于广泛使用中的各种小型机。它们与移植工作有关的共同点可归结为：

（1）内、外存都比较小；

（2）速度不高，一般在每秒数万到数十万次之间；

（3）大都按字编址而不是按字节编址，这加剧了存贮的紧张程度；

（4）单运算器；

（5）大都不是栈式机器。

这种状况迫使实现者在确定移植方案时既要少用空间，又要效率高。而以上推荐的三种方案中，虽然效率和存贮量各异，但就空间而言，需求量最小的第一方案也不能少于100K字节；就效率而论，无论是解释、翻译，还是两者混合，在进一步自展之前，都要模拟栈式计算机的动作，效率损失严重。

因此，在国产机上进行移植，有必要另寻善策。

二、翻译和移植

我们的移植方案是打破模拟栈式计算机的框框，充分利用单运算器的特点进行一次有“窥孔优化”的翻译，产生常规的合理的目标程序。

我们把这个翻译程序命名为P-code汇编程序，简称P汇编。但不同的是，在通常的汇编或宏处理程序中总是把一条汇编指令对应于固定的一条或数条目标指令，而P汇编却不是这样。在处理一条P-code指令时，要根据它前后文的情况来决定必要的目标指令，因而一条P-code指令所对应的目标指令组是不固定的。实际上，P汇编更象是一个编译程序的代码生成部分，只是没有任何表格为之提供辅助信息而已。

由于试验是以121机为目标机器来做的，所以下面的讨论会涉及121机的一些具体情况。相信这不会失去一般性，因为121是一种功能一般的机型，在这种机型上能做到的事，在其他性能较好的机型上就更能做到。

121机P汇编的目标语言是121机的基本汇编语言，它的指令同机器指令基本上是一对一的。

在设计P汇编时，我们的出发点是时间和空间的节省兼顾，有矛盾时以节省空间为主。基本措施为：

1．把栈式计算机的指令改造成常规合理的指令

在目标程序中设法抹掉栈式计算机的痕迹是节省时、空的关键。做法有二：首先，汇编时用一数据栈来对应栈式计算机的运行栈。对于P-code的进栈指令，只是把进栈对象暂时记进数据栈，而不产生目标指令。只是遇到使用这些数据的指令时，才根据当时的情况产生适当的指令；其二，为了利用单运算器中的寄存器产生合理的指令，我们把这寄存器看作汇编数据栈的活动栈顶。下面用例子来说明这种想法。

若A和B是实型变量，J是整型变量，则语句A＝B＋J所对应的P-code指令组为

```
LDOR    13
LDOI    10
FLT
ADR
SROR    14
```

这里A、B和J都是基本层的变量，相对地址分别是14、13和10。汇编过程见表1。这样的目标代码当然可以令人满意了。

2．使用不同的寻址方式

关于调用过程或函数时数据段的处理，我们采用CDC 6000系列中对PASCAL系统所用的运行方式，即保留一个指示表，运行的每时每刻在其中存放着当时可引用数据段的原点地址。这样一来，除常量以外，对所有数据的引用均需变址。

表1

处理的指令	处　理　动　作	编出的指令
LDOR 13	基本层地址进数据栈	—
LDOI 10	基本层地址进数据栈	—
FLT	编出将栈顶元素浮点化指令；退栈；记寄存器为新栈顶	002 190A 006 0000
ADR	编浮点加指令，退数据栈顶；记寄存器为栈顶	008 190D
SROR 14	编传送指令；记寄存器已空	004 190E

121机的指令有两种变址方式：零变址要增加8微秒变址时间，长变址则要另加一条变址指令，执行时间是24微秒，为零变址的三倍。从运行效率来考虑，最好不用变址，但这是不可能的。现实的办法是多用零变址、少用长变址。可惜121机中能用作零变址寄存器的只有一个单元。在这种条件下，根据过程(或函数)体中使用的量多是参数和局部量的实际情况，我们决定当前层的量使用零变址，在最外层（即基本层）说明的量用绝对地址，其他层的量用才用长变址。这样一来，一般程序中实际使用长变址的机会就不多了。汇编前面那个例中的语句时，由于都是基本层的量，所以没有出现变址指令。

3．充分利用运行程序包

标准过程以及一些经常出现的动作，如进、出过程，退栈、双字符运算等。一律用运行子程序来处理，以求最大限度地缩短目标程序的长度。

4．目标程序分段运行

由于内存小，又要留出足够的数据空间，因而大程序必须分段运行，特别是P编译程序本身就只能这样运行。

这件事对于没有分页操作系统支持的机器来说，只能在运行程序包中加进这一功能。至于分多少页，一页多大，则应视具体机器的情况而定。

在我们的具体方案中，是通过改进原有的121机基本汇编程序来完成这一任务的。现在121基本汇编程序具有这样的功能：允许用户指定一个固定大小的目标区，汇编程序就以此为单位来分段（块）汇编用户提交的源程序。遇到段与段之间的转移时，就辅以少量的转移信息。运行时，有关的运行子程序便可按转移信息的要求调进相应的程序段，并转到相应的指令去执行。

以上便是P汇编所采取的主要措施，其他细节处理从略。当然，实现时还要克服一些具体困难，譬如双字符运算和间接访问等。

表2列出了13个实例的两种指令条数的对比。统计时删去了标号等不占存贮的指令。这13个例子使用了PASCAL的所有类型和语句，因而有一定的代表性。其中例3增加比例最小，仅为13.5%；例5和例8增加最多，为95.5%。合计增加40.9%。

表2

	1	2	3	4	5	6	7	8	9	10	11	12	13	合计
P	25	47	37	93	22	60	206	44	49	107	28	36	18	772
121	35	54	42	120	43	87	272	86	94	135	36	60	24	1088

完成P 汇编之后，就可把文件2 加工成能在目标机器上运行的程序。然后，两者联合加工就能运行PASCAL程序。由于P汇编所产生的目标程序质量尚好，就是不再进行自展也能有相当可以的运行效率。

三、结束语

我们认为，对于一个熟悉目标机器的人，写一个P汇编并不比文件3 难太多，大约有半个人年就够了，再加上运行系统，也不会超过一个人年的工作量。因此，这一方案有实践意义。这一方案的实现，不仅可以获得可运行的PascalP4，同时也完成了所有用P-code表达的算法的移植。

目前，我们已实际获得了121基本汇编语言形式的P编译程序。这项工作得到了几位同行的合作：李师贤完成了121基本汇编程序的改造；方文振在调试程序时付出了辛勤的劳动；广东省计算中心在程序调试过程中给予了支持。

参考文献

〔1〕K. V. Nori, U. Ammann, K. Jensen, H. H. Nagele and Ch. Jacobi〔1976〕, The PASCAL<P> Compiler: Implementation Notes, Institut fur Infomatik.

〔2〕K. Jensen and Wirth 〔1975〕 Pascal User Munual and Report, Springer-Verlag, New York.

〔3〕N. Wirth〔1971〕, The Design of a Pascal Compiler, Software exp & Prac., Vol. 1, 309-373, 中译本，俞嘉惠译，程序设计语言PASCAL，科学出版社，1983.

〔4〕杨德元，Pascal可移植编译程序及其分析说明，清华大学出版社，1982.

On the Translation of P-code

Xiao Jinsheng

Abstract

This paper introduced a scheme of the translation of P-code. It can eliminate thorough the trace of stack compnter, and use more effectively characteristic of single arithmetic unit, thus produce the object code that has better quality.