南京航空学院

研究生硕士学位论文

研究生姓名 辛欣

专　　业 计算机应用

研究方向 多机系统

指导教师 邱百光付教授

一九八五 年 一 月

通用的机口码仪息生成软件

摘要

GEMCIGS是"通用机口码程序分析系统"的前半部分，它的任务是针对各种不同的计林机指令系统，生成处理它们的机口指令的规则，并利用这些规则，处理它们的机口码目标程序，提供有关的仪息，给"通用机口码程序分析系统"的后半部分——程序分析系统。实验表明，本系统对于用户是实用的、方便的、灵活的。

GENERAL MACHINE CODE INFORMATION GENERATION SOFTWARE (GEMCIGS)

Abstract

GEMCIGS is the first half of General Machine Code Program System (GMCPS). In accordance with various instruction systems, it can generate the rules which process the machine instructions of these system and treat their machine code goal programs, by means of these rules. Then it provides the second half of GMCPS---Program Analysis System with the information concerned. The experiment shows that the system is practical, convenient, and flexible for users.

目　录

一. 引言

当前，我国进口了大量各种各样的计算机，配在这些计算机上的软件也相当丰富。为了提高我国对这些计算机的利用率，一方面要在它们上花大功夫进行软件研制、开发工作，另一方面，要对这些计算机上现有的进口软件进行解剖和消化。在某种意义上说，后者更为经济和有效。

在进口的软件中，有些软件的文档资料齐全，还有些没有提供齐全的文档资料，甚至只有机口码目标程序。对它们进行解剖最困难。而且，现在我国进口的计算机种类繁多，不同的计算机型号，其机口码的形式往往也不相同。针对上述情况，有必要搞一个通用的软件工具，协助人们对这些机口码程序进行分析、解剖和打开。

基于以上情况，我们研制了一个通用机口码程序分析系统"

本课题所构造的系统——GEMCIGS（通用的机口码信息生成软件）是上述系统的前半部分，它主要包括两大部分，一部分是GIPRG，它的任务是针对不同的计算机指令系统生成处理它们的机口码指令的规则；另一部分是GRP，它的任务是利用上面的规则，对相应的指令系统的机口码目标程序进行处理，提供统一的中间代码给"通用机口码程序分析系统"的后半部分——程序分析系统。

从已有的文献来看，有关这方面的论文比较少。华东水力学院的同志搞过一个通用反汇编系统[1]。在解决对各种指令系统通用的问题上，他们自己搞了一种形式语言

用这种语言对要处理的指令系统加以描述，然后送到一个生成中，将它变为一个反汇编程序，此程序来实现反汇编。

首先，上述方法对于用户来说是不方便的，用户必须要具备形式语言的知识，而且要花一定时间去懂一个比较复杂的语言，并用它编一个较冗长的程序来完成用户的工作；其次，上述系统只是为反汇编服务的，而本系统不光是反汇编，而且还要对机内码程序进行分析，因此，需要的信息量比一个反汇编程序所能指供的要多。

本系统在实现通用性的基础上，功能更强，使用上更为实用、灵活、方便、简单。

大家知道，"通用性"问题是一个带普遍性的问题，比如对情报检索系统、工厂管理系统等来说，"通用性"往往意味着减少重复劳动、提高工作效率、提高应变能力。因此，本课题在设计思想上，想如何用计算机来模拟人的"通用性"思维，以便使其具有更大的推广价值。

以下将介绍这个系统的实现及功能。

二. 系统的设计思想。

本节讲一下GEMCIGS的设计思想。

·当前，为了搞清人的智能的本质，加快发展人工智能，人们提出了各种各样的思维模型、大脑模型。其中有一种提法认为，人的思维是有层次的，并且是宝塔形的，高层次的思维指导、支配低层次的思维[2]、[3]，思维就是一个过程，在这个过程中，使思维的对象即信息不断地进行转化，所以，思维就是对信息进行处理的过程。

从方法论的角度讲，解决任何问题都要有相应的方法，方法是在任何一个领域中的行为方式，是用来达到某种目的手段、过程的总和。有人还提出，逻辑思维可以分为求知性思维和实际性思维，前者以认识、反映客观事物为目的，后者以求知性思维的结果为基础，来说明、执行和运用它们。方法就属于后者[4]、[5]。显而易见，方法也是一种信息处理过程，方法之间具有层次关系，并且是宝塔形的。

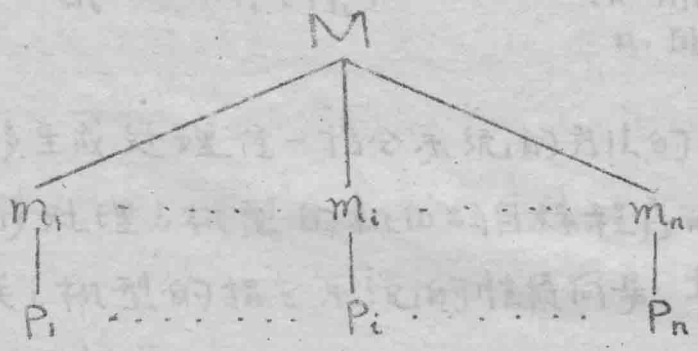
这里仅就两层方法之间的关系进行讨论。

由上面可知，解决任一问题，从而达到某预定目的方法的内容不是任意的，它是以已获得的有关这一问题的知识为依据的，是在把问题的已被认识到的因果的和其它重要的联系加以整理后的基础上建立起来的。

如果，把上述的解决具体问题的方法看做低层方法，则产生它们的方法就是高层方法。这一方法是一个学习、分析、综合的过程，通过它，获取有关问题的知识，对其进行分析，找

出其中具有规律性的关系、联系，并将它们转化成解决那个问题的方法。

我们用M表示上层方法，用m_i表示下层方法，用P_i表示具体问题，则可以用图1形象地表示它们之间的关系：



（图一）

本课题的目标是能够处理任一计算机型号（在冯氏体构范围内的）的指令系统的机口码目标程序，以中间代码形式给出程序中每一条指令的有关信息，供PAS（通用机口码分析系统的另一子系统）分析时使用。我们记这一目标为G。

大家知道，各种不同的机型，其指令系统都是不同的，因而，它们的机口码目标程序当然也不同。我们把G_i表示为能处理所有i机型的机口码目标程序，则

$$G \Longleftrightarrow G_1 \wedge G_2 \wedge G_3 \wedge \cdots \wedge G_n \qquad (n\text{为自然数})$$

为了实现这一目标，如果把依据上述的高层、低层方法分析、解决问题的过程用数学模型加以描述，则下式必须成立：

$$m_i = M(I_i)$$

$$I_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ik} \end{pmatrix} \in \{I_1, I_2, \cdots I_n\}$$

$$G_i = m_i(P_{I_i})$$

(k: 有限大的自然数,
n: 同上式)

M: 表示能够生成处理任一指令系统的方法的上层方法。

m_i: 表示能够处理 i 机型的机口码目标程序的下层方法。

I_i: 表示有关 i 机型的指令系统的性质向量，其中，x_{ij} (1≤j 表示其一项性质。

P_{I_i}: 表示具有 I_i 性质的指令系统的机口码目标程序。

由方法的层次性可知，任何方法的处理对象都是有一定的限制的，对象的性质都要在一定的范围之内，绝不可有一种方法能够处理任何问题[6]。

所以，如果 M 要能够处理 $\{I_1, \cdots I_n\}$ 集合中任一个 I_i (1≤i≤n 就必须能够找到一个向量 $\begin{pmatrix} x_1 \\ \vdots \\ x_K \end{pmatrix}$ 和一个函数 E，使 $E(I_1)$ $E(I_2), \cdots E(I_n)$ 都是向量 $\begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$ 的子向量。($k \le K$)

上面的意思就是要在 I_i 之间找共性，若它们有共性，尽 I_i 的个数可以很多，但在性质上，它们是在一个有限的范围之内的；若没有共性，则在性质上，它们不在一个范围之内，由上可知，M 就不能确定了。而林法不过是受到某些限制的方法[7]，因而上述模型也就是不可计算的了。

下面就来看一看，是否任何指令系统之间有其共性，从而存在一个M，可以达到通用的目的。

任何冯氏结构的通用数字计算机指令系统的每条指令一般由两部分组成，一部分是表示执行什么操作的操作码，另一部分是操作的对象——操作数，而且，每条指令在机内内部都用一串可变长的二进制数字表示。因此，操作码部分和操作数部分全部可由这一串数字表现出来，不管是什么机内的指令，我们都可以通过确定这两部分的位置来达到识别它们的目的。

就前一部分来说，尽管各种不同的计算机的指令的长度不同，操作码位置不同，有的指令的操作码甚至还不是集中在一个位置上，而是分散在指令的几处，但是它们都是对应着本机内的最基本的操作。不同机内的指令操作码长短有别，一般讲，操作码越长，此指令系统所含的指令越多，否则越少。但是，不管一机内指令系统中指令个数是多还是少，它的操作种类必须是完备的，即对一切可计算的问题，都可以用这指令系统中的指令编程序，而且能实现自动计算并得到正确的结果。若指令系统包含<1>运算型指令（算术的和逻辑的），<2>传送型指令，<3>移位型指令，<4>转移型指令（条件的和无条件的），<5>停机指令，<6>输入、输出型指令，则此指令系统就是完备的[8]。在下面的PAS中，分析目标程序，给出的指令模型只有四种类型：<1>条件转移，<2>无条件转移，<3>顺序，<4>停机。所以，任

何指令系统的指令都可以用这种模型表示，对任何指令系统的指令类型，我们能够找到它们的共性。

后一部分说明参与操作的数是什么，它们从何而来，操作的结果到何处去。

决定操作数来去的方式叫寻址方式，不知道寻址方式，就不能得到操作数，下面的PAS就不可能把指令和数据分开，亦就不能找出程序的流图。计算机的寻址方式多种多样，但尽管如此，它们还是有共性的。这是因为冯氏狭道结构决定了任何程序完成它的任务，必须通过在这个狭道上往返单个字来完成，不管是操作对象本身，还是操作对象的地址都必须以单字形式往返于CPU的各有关寄存器和内存储器之间[9]、[10]。再加上CPU中的寄存器不管类型还是个数都是有限的，这样，就决定了任何计算机（冯氏结构范围之内的）指令操作数的寻址方式只能在一定范围内变化，是可以统一处理的。

同样，由于上述原因，任何机器的指令操作数也仅包括直接操作数、立即操作数、偏移量、各种寄存器的内容，或它们的内容的内容等等。而且，寄存器的结构都是围绕着单字形式而定的，因此，也可以对任何计算机指令操作数进行统一处理。

这样，我们就可以依据上面讲到的所有共性来构造一个通用的M，它根据获取的某一指令系统的指令格式、寄存器情况、指令类型和寻址方式等知识，来总结、归纳为处理该指令系统的机器码目标程序的方法的知识，即 m_i。m_i 的执行就能够处理相应的目标程序，提供以中间代码

为表示形式的有关信息，给PAS。

以下，讲一下上述思想是如何具体实现的。

三. GEMCIGS的系统结构.

实现上述思想，可以有两种途径。

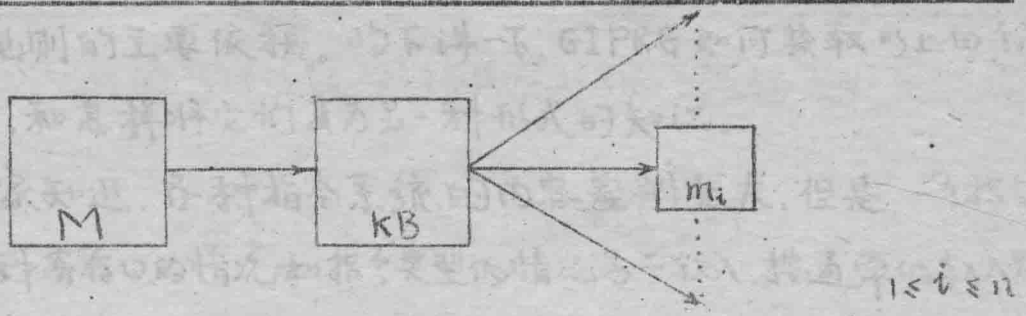
一种是利用人工智能语言Prolog来构造系统。因为这种语言的语句形式和数据形式是统一的，即由某程序处理、产生的数据，可以是完成某功能的程序，并且它同样可在同一计算机上运行[11]。这样，我们就可以用Prolog语言构造一个M，根据输入I_i，M就可以产生出数据m_i，m_i的执行就能够实现G_i。但是，在现阶段，这种语言不管是在时间上还是在空间上，它所编出程序的效率都较低，尤其在程序模块化方面，不如传统语言[12]。这样，就限制了应用它编写大的程序。

第二种途径是利用产生式系统来实现本系统[13]。这是因为产生式系统强调问题求解的过程。

上面已经说过，m_i正是一种信息处理过程，因此，可以用代表这些过程的产生式来表示它们。产生式规则的前件是满足不满足某一种指令格式的条件，它的后件是如何处理这种指令格式的指令的步骤。这是因为，任何指令系统的指令都分别属于这系统的几种指令格式，正是这些指令格式来指出属于它的指令的各种情况，所以要处理一条指令，体现要知道它属于什么指令格式。

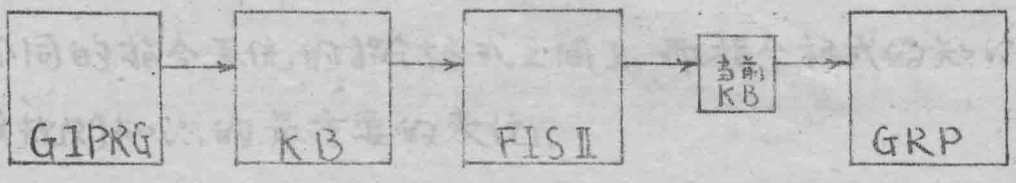
显然，这些规则并不象通常的产生式系统那样，一开始就存在[14]，而是由M通过获取的关于某指令系统的知识而产生的，而且它们具有方法的性质。

图2是本个系统的形象示心图。



（图 2）

本系统的结构如图3所示：



（图 3）

从上图可以看出，本系统主要由两部分组成。概括地讲，前一部分的作用是学习、消化知识，后一部分的作用是利用已有的知识。不管是已学到的知识，还是要利用的知识，都放在知识库(KB)中。

GIPRG的作用就如同上面的M。根据上节的分析，我们还可以归纳出各指令系统的几点共性：

<1> 每一指令系统都有若干种寄存器。

<2> 每一指令系统都有若干种寻址方式。

<3> 每一指令系统都有若干种指令格式。

<4> 每一指令系统的指令类型都是有限的。

如果对一指令系统，我们掌握了它的以上四点情况，就能够处理它的机器码目标程序。而且，<3>和<4>还是生成处理机器

指令的规则的主要依据。以下讲一下，GIPRG如何获取以上四方面的知识，和怎样将它们表为另一种形式的知识。

大家知道，各种指令系统的内容差别很大，但是，一个指令系统的各种寄存的情况和指令类型的情况易于输入，按通常的输入方式就能将它的输入。下一节，将详细说明如何输入它们。而对于指令系统来说，最为灵活多变的就是指令格式和寻址方式了，尤其是指令格式，每一种指令系统的指令格式之间就很不一样，更不用说不同的指令系统的指令格式之间了。而指令格式的知识，又是形成规则知识的最重要的来说。

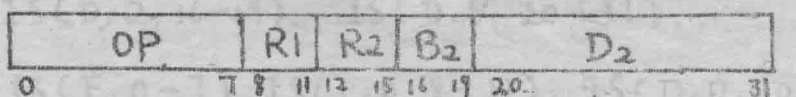
GIPRG所获取的关于寻址方式和指令格式的知识，主要是用谓词 simple、Before、Is 来表示的。因为，第一，用谓词表示，对于传授知识的用户来说是直观的，谓词表示与人的自然语言相接近；第二，谓词表示在当前来说，对于计算机科技运用，它比自然语言要容易处理得多；第三，用谓词表示来进行输入，与通常输入方式相比较，其最大优点在于灵活、多样。

下面就结合例子，讲一下指令格式的输入。寻址方式的输入同它类似，下一节将详细介绍。

例：

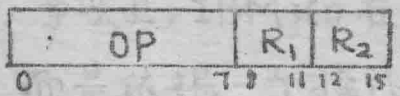
IBM360的RX和RR格式如下：

RX：



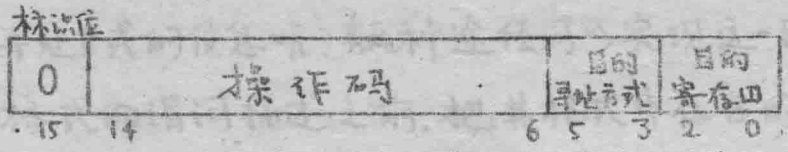
<1>

RR：

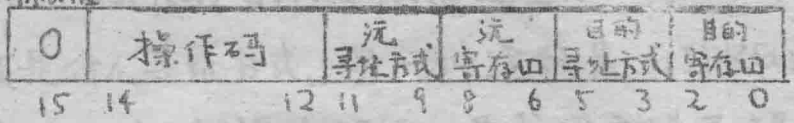


<2>

PDP-11的单操作数指令格式和双操作数格式如下：

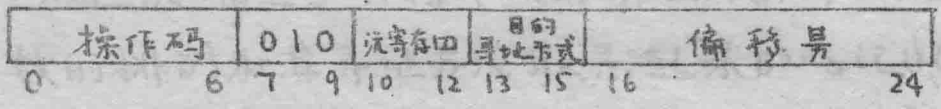


<3>

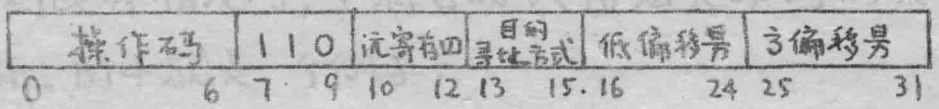


<4>

Intel 8086的两条指令格式如下：



<5>



<6>

从上面的例子中，我们可以发现，各指令格式之间的区别太大了，不管是指令的长度、操作码、操作数的位置，还是寻址方式、操作码的形式都有差别。

对于以上例子，以下讲一下如何用谓词IS来描述它们。谓词中的有号请参阅图16，下一节还要专门讲为什么使用简写符号。

<1> IS(F,0-7), IS(C:0,8-11), IS(D:0,12-15),
IS(D:2,16-19), IS(D:6,20-31).

<2> IS(F,0-7), IS(C:0,8-11), IS(D:0,12-15).

事实上，IBM360在寻址方式上不是利用专门的寻址方式位，而是依据一个操作数的各部分的排列来隐含表示寻址方式，象上述的"通用寄存器，基址寄存器，偏移量"

到就表示寻址方式是基址寻址方式。这种寻址方式的表示法不只是局限IBM360，它是有代表性的。GIPRG怎么会从这一描述得到它的寻址方式的信息呢？有两种途径可以实现这一目标。一种是，在输入指令格式的谓词描述之前，把其格式中存在的寻址方式的知识先输入给GIPRG，GIPRG依据这些知识来判别指令格式中的操作数是什么寻址方式；另一种途径是，GIPRG遇到一种操作数的排列后，主动提出疑问，要求用户给予解答。当用户回答之后，它不但能确定当前操作数的寻址方式，而且还将当前操作数的排列和其寻址方式联系起来加以记忆，当以后再遇到此种情况时，它将自动地用先头学到的知识加以判断。图4就是一个例子：

```
WHAT ADDRESSING MODE IS
* * * GER_REG  ?  * * *

PLEASE SEE CODE LIST ABOUT ADDRESSING MODE NAME!
DIRECT_ADDRESS=0.                    REGISTERDIRECT_ADDRESS=1.
INDEX=2.                             INDIRECT_ADDRESS=3.
REGISTER_INDIRECT=4.                 INDEX_INDIRECT=5.
IMEDIATE_ADDRESS=6.                  RELATIVE_ADDRESS=7.
SEG_ADDRESS=8.                       OTHER=9.
PLEASE INPUT(0_9):
0
```

（图 4）

由于不同种的指令系统，可能对于同一种操作数的排列，规定为不同的寻址方式，这时，GIPRG会自动发现这种情况，并向用户提出来，在用户回答后，GIPRG就按照它来确定本指令系统的这种操作数排列的寻址方式。请看下例（图5所示）：