南京航空学院

研究生硕士学位论文

研究生姓名 王海鹰

专　　业 计算机应用

研究方向 多机系统

指导教师 邱日光付教授

一九八五年一月

软件工具GMPAS的设计和实现

摘　要

本文报告了我们研制的软件工具GMPAS（General-purpose Machine code Program Analysis System）。文章讨论了机器代码程序的分析进程，叙述了GMPAS的系统结构和功能，并重点介绍了用于分析的指令模型，系统的扫描模块，分析标法及用户接口。该系统可以对各种机器指令系统的机器代码程序提供较强的综合分析能力，并对分析结果提供多种形式的输出报告。

THE DESIGN AND IMPLEMENTATION OF A SOFTWARE TOOL: GMPAS

Wang Hai-Ying

ABSTRACT

This paper outlines the design and implementation of the GMPAS(General-purpose Machine code Program Analysis System), a tool used for Software Analysis. The paper discusses the procedure of machine code program analysis and introduces both structures and functions of GMPAS. We mainly describe the instruction model used for analysis and scanning module,analysis algorithms and user interface of the system. The system can analyse the machine code program of various kinds of instruction system and can provide the ability of synthetic analysis as well as many forms of output reports of analysed results.

目　　录

一、 前　　言

这一节将简略地介绍机器代码程序分析系统。首先是建立这样一个系统的目的，其次简单地介绍程序分析的概念及其它的用途，最后介绍本文研究的系统特点。

1、问题的提出

计算机软件的解剖、分析和修改是当前计算机发展中的一个重要领域，特别是在我国，进口机器的增加，智能仪器的引进，充分利用它们已成为当前的主要任务。为此，分析、解剖这些机器上的现有软件，并在此基础上进行修改和移植是软件工作者面临的一大任务。但是，国外的软件厂家为了保密，一般只向用户提供机器代码程序（如IBM／PC的COM文件）。由于机器代码程序的可读性差，再加上缺少清晰的、足够的有关文档资料，使得对于这些程序的理解非常难以进行。给分析工作带来了很大的困难。解决的途径之一是将机器代码程序转换为可读的文本（汇编助忆符程序），并从中抽象出程序的结构（由于汇编助忆符程序的阅读也比较困难）。在重新构造程序可读文本和结构的过程中，人的分析工作是艰巨的。但实际上很多分析工作可以设法用机器来做，使分析自动化。

目前已为此目的研制了一些反汇编专用程序[11][12][13][14]，这些程序采用不同方法为分析人员的工作提供了帮助。但它们所具有的功能和所能提供的信息都是有限的，仅仅是针对某一指令系统将相应的源程序反汇编出来，而不具有其它分析能力，人的分析工作仍然很多。

为了使正确的软件分析变得容易，就应提供类似于[6][7]的程序分析工具。它应具有较强的功能，能为分析人员提供各种有用信息，

使分析人员将精力集中于程序算法和功能上，而不必注重一些费时的机械劳动。另外，这样的分析工具也可作为调试工具和文档编制工具前者可以表示程序中所有可能的执行路径；后者可以提供程序文档，使程序设计人员可以化较多时间集中编码，而相应文档的编制由机器来完成。

2、程序分析简介

程序分析是一种静态分析技术[1][2][3][21]，它在不实际运行源程序的情况下，用计算机对源代码进行分析，考查程序运行的性能，确定程序结构并检查其异常情况。它的主要过程是对程序控制流和数据流进行分析。控制流图表达了程序的结构，数据流表述程序变量的数据流动情况。通过考查控制流图中各结点数据流动情况，可以查出程序中某些类型的错误。数据流分析仔细地研究程序的细节，以推出程序的一般结构和性质，这些推导分析的结果可用于程序的文件编制等方面。

由于程序分析是通过分析程序控制流和数据流的所有路径，所以给出的全局信息仅通过单独运行程序是得不到的。静态分析包括程序结构、数据流程、模块结构的分析，以及符号执行和正确性证明。这种分析技术目前主要用于对高级语言书写程序的分析，它的应用范围很广，包括代码优化[1][5]，程序确认[6][7]，程序测试[8][1]及程序复杂性度量[9]，等软件工程的许多方面，成为一种非常有效的技术。

本文将利用静态分析技术对机器代码源程序进行自动分析。由于低级机器语言与高级语言的差别，因而它们在表示方法、处理技术等许多方面有较大的差异，这导致了分析策略和算法的不同。这将在后

面的章节中分别讨论。

3、GMPAS 简介

GMPAS是为机器代码程序自动分析而研制的一个软件工具。它将机器码源程序看成是一个有向图，由于机器码程序中的指令和数据混杂在一起的，这个有向图（只表示程序中的指令部分）是隐含在一群代码中的。

GMPAS的主要功能：

1）能对以二进制或十六进制的机器代码源程序进行扫描，力求搜索出完整的有向图，并以内部代码来表示它（搜索过程中指令的识别是通过调用指令检索器Retrieve[25]完成的）。

2）对内部代码表述的程序有向图进行控制流分析可产生源程序的控制流图。

3）通过分析源程序中的各子程序之间调用与被调用关系可产生源程序调用图。

4）根据程序有向图可产生和源程序指令系统相对应的汇编助忆符程序清单。

5）对源程序变量（为分析方便仅指寄存器）进行静态数据流分析，可得到数据流动的信息。

6）为源程序提供一定查错能力，GMPAS可查出的错误是引用了未定义的寄存器的指令。

7）能对各种指令系统的机器代码程序进行分析，具有通用性。

8）对于分析结果提供各种形式的输出，包括图形、文字和清单。

GMPAS是一个由各个独立分析模块组成的综合分析工具，它具有较强的交互能力和良好的用户接口。

二、概念和模型

在这一节中将介绍并建立系统设计中依据的概念和模型。

1、流　图

定义：流图是一有向图 $G=<V, E, n_0, f>$，它满足下列条件：

1）有唯一的结点 $n_0 \in V$，n_0 没有前驱结点，图中任何一个结点 $v_i \in V$ 都有一条从 n_0 到 v_i 的路径，n_0 称为流图的入口点或首结点。

2）有唯一的结点 $f \in V$，f 没有后继结点，V 中任一结点 $v_i \in V$ 都可达 f。f 称为流图的出口或末结点。

下面将用流图来表达程序结构。

首先定义程序的基本块。它是程序中的一个满足下列条件的最大指令段：（i）指令段在执行时只有单一的入口指令，称为基本块的首指令；（ii）单一的出口指令，称为基本块的末指令；（iii）指令段中的指令是顺序执行的，即总是从首指令开始，经过所有段中的其它指令到末指令结束。

其次考虑到机器代码程序结构差，是不可规约的[1]，因此程序的结束指令可能有多个，故在此规定流图的末结点 $f \subseteq V$ 是一个子集。

定义：程序控制流图是一流图，其中，

1）流图中的每一个结点 $v_i \in V$ 代表程序中一基本块 b_i；

2）流图中每一条边 $e_{ij}=<v_i, v_j> \in E$ 相应于程序中的控制转移；

3）设图中结点 v_i，$v_j \in V$ 相应于程序基本块 b_i，b_j，那么边 $e_{ij}=<v_i, v_j> \in E$ 存在的条件是（i）基本块 b_j 是

紧接着基本块 b_i 之后的；或（ii）基本块 b_i 的末指令是转移指令，其转移去处是基本块 b_j 的首指令。

通过控制流图可以直观地将程序结构表示出来（图2）。但是，要指出控制流图并不是唯一的程序图表示方法，另一种常用的方法叫程序框图（图3），它用于程序设计中。这两种方法的区别可以从图2、3中清楚地看出，控制流图主要是将程序控制结构表示出来，如

```
Start:  SUB A                  }
        LD B, 100H             } b1
        LD HL, Loc+1           }
Sumd:   INC HL                 }
        ADD A, (HL)            } b2
        DJNZ Sumd              }
        LD (Loc), A            }
        HALT                   } b3
        END Start
```

图 1　　例 1

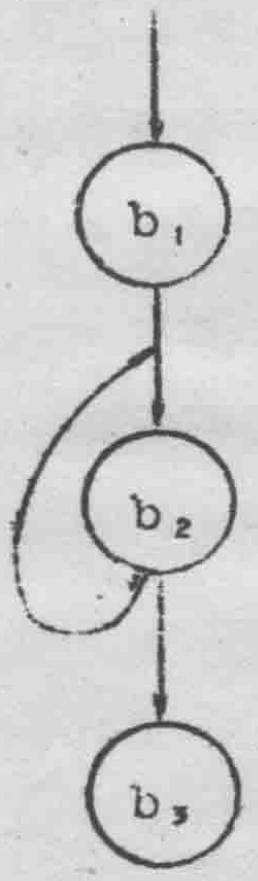

图 2
程序控制流图

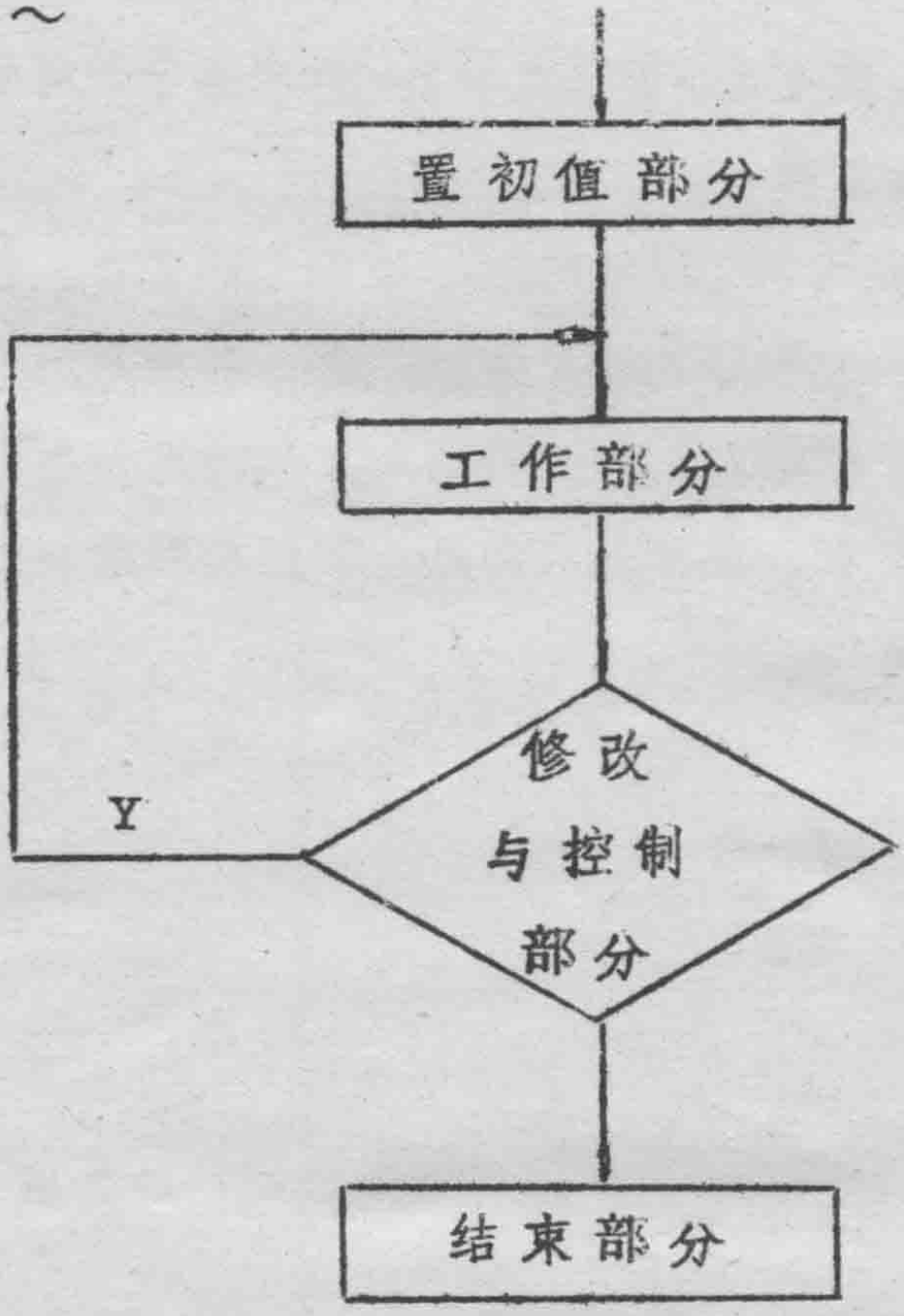

图 3　程序框图

程序中的执行路径和循环等，并按执行顺序给图中结点确定编号，但它不关心执行的计算和完成的功能，而将这些信息隐蔽起来，它就好象是程序的语法结构；相反，框图不仅包含语法，而且主要是包含语义。程序流图加上程序文本等价于程序框图。

由于机器指令功能差，使得相应流图的结构性差，复杂度高，变化多端而不易看清。仅仅控制流图对理解程序整体还是不够的，有必要进一步地抽象到更高层次的程序结构。考查程序中各子程序之间的关系，可以发现它们之间的调用与被调用关系在更抽象的级别上反映了程序的结构及程序的运行情况。对这种情况可用类似于控制流图的方法来表示。

定义：调用图是一流图 G =(n_0， V， E)，其中，

1 ）V代表图中所有的结点的集，V的结点已编号，它们代表程序中的各个子程序。$n_0 \in V$是主程序本身。

2 ）E代表图中所有边的集。若从结点v_i有一有向边$e_{ij} \in E$指向结点v_j，则表示子程序v_i一次或多次调用子程序v_j。

调用图反映了程序与子程序之间的联系（图4）。

2、控制流分析和数据流分析

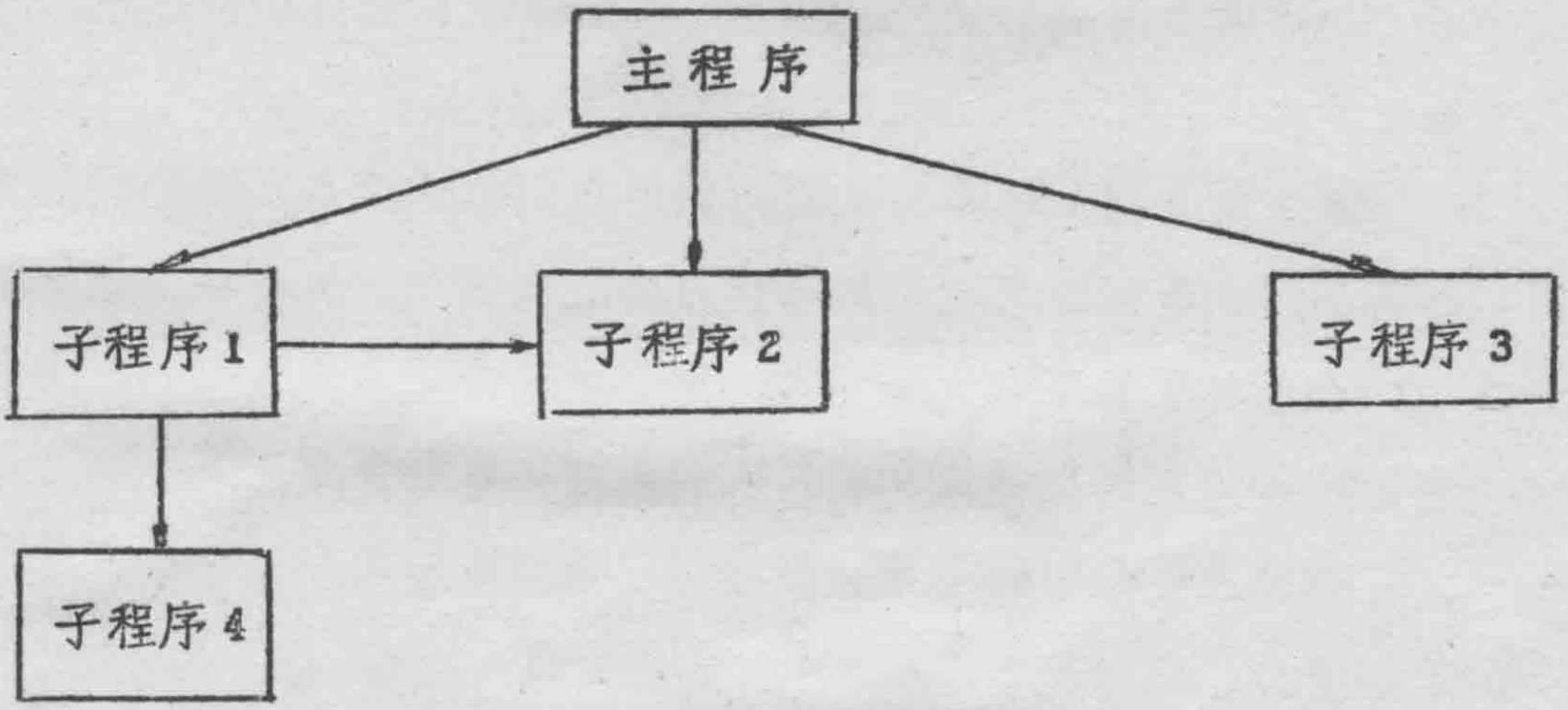

图4　调用图例

控制流分析实质上是整理和确定程序控制流图中的结点关系，也就是给图中的结点确定一个顺序，它是程序分析的基础，其结果是程序控制流图，它通过扫描源程序代码，逐条指令地分析构造程序中各基本块，再逐块分析产生表示基本块间联系的弧。

流图中的路径代表了程序实际运行的路径，它将程序所有可能的执行路径都包含在其中。当然，流图并不是完全正确地描述了程序真实运行时的情况，图中有些路径在执行时是行不通的（路径是否可执行是一个不可判定的问题[1]）。

考虑到机器码程序的特点，同时也为了分析的方便，本文中将整个程序看成是一个流图，在流图分析中将子程序调用看成是一般的转移指令（由于机器指令程序的参数传递一般是隐含的，且不同的程序传递约定各不相同，略去参数传递分析对流图影响不大），返回指令作停机指令看待，只是将调用信息（如地址、调用次数等）和返回地址记录下来，供调用图分析用。这样使控制流分析非常方便，而不用考虑各子程序间数据流动的复杂分析[1]。由于记录了调用信息，所以也不影响调用图分析和数据流分析。

数据流问题是研究控制流图中各结点间数据流动的状况。它可归为二类：

1）对于程序中某给定点，问在控制到达这点之前数据情况如何？即什么样的数据定义可以影响这点的数据使用，又叫向前流问题。

2）对于程序中某给定点，问控制离开这点后数据将发生什么样变化？即这点的数据定义将影响什么样的数据使用，又叫向后流问题。

本文主要关心的是第一类问题，通过解所谓到达——定值流方程来解决。

所谓变量A在某点d的定值到达另一点u（或称变量A的定值点d到达一点u），是指流图中从d有一条到达u的路径，且在该路径径上没有A的其它定值（即定义），假设在程序中某点u引用了变量A的值，则把能到达u的A值的所有定值点全体称为A的引用点u的引用——定值链（记为ud链）。

将全局数据流结构定义成一个代数系统（L，F），其中L是数据值的集合，F是作用在L上的函数的集合（F是在控制流图上的操作）。实际上L可以看成是一半格[1][2][3]，L中最小元素用O表示，它为机器可取最小值，最大元素用Ω表示，它为未定义的值。F假定在并

和联的二种运算下是封闭的，它包含有恒等映射，而且是单调的，可分配的，即

$$\forall f \in F,\ x,\ y \in L,\ x \leqq y \Rightarrow f(x) \leqq f(y)$$

$$\forall f \in F,\ x,\ y \in L,\ f(x \wedge y) = f(x) \wedge f(y)。$$

给定全局数据流代数结构（L，F）和控制流图G，可将流图中的每一条边（m，n）$\in$E与传播函数f（m，n）$\in$F相对应，前者表示数据可能沿此传播的程序流向，后者表示控制从结点m的入口点始到结点n的入口止相应数据的变化。

给定集合S＝｛f（m，n）：（m，n）$\in$E｝和控制流图G，可以定义包含S的、最小的、作用在L上的传播函数集F。

定义了F之后，可以用下列方程组表达向前流问题：

$$x_r = 0$$

$$x_n = \bigcup_{(m,\ n) \in E} f_{(m,\ n)}(x_m)$$

$$(n \in V - \{r\})$$

其中$\forall n \in V$，x_n表示在结点n入口点的数据值集，$x_n \subseteq L$。

在实际分析中，x_n不仅包括所有在n入口点的可用值，而且包括所有对这些值定义的点，即x_n是在n点的引用——定值（ud）链。

3、指令分类模型

前面在介绍控制流分析概念时提到，构造程序控制流图是通过逐条指令扫描来完成的。但是，机器代码程序的特点之一是指令与其使用的数据混杂在起，且都以机器代码表示，二者难以区分。如果仅仅是顺序扫描，必然会造成指令与数据不分，结果不正确，使自动分析无法正确进行。为了正确区分指令与数据，必须顺程序流向扫描，将

指令（在流向上）与数据（不在流向上）区分开。

机器指令有各种类型，它们对程序流向影响不一，即使是针对特定指令系统，流向跟踪也是很困难的。因此必须建立一个通用的处理模式，抓住各种类型指令的共性，用统一的方法处理来保证扫描的通用性和可行性。

不同类型的机器其机器指令系统是不尽相同的，每一种型号机器都有唯一的机器语言，这是与机器硬件有关的。但是，它们各自所包括的类型及完成的功能基本相似，例如 IBM／360 与 PDP 11／23 机器，虽指令系统不同，但它们的指令类型（按功能分）却基本相同，都有进行存、取、算术运算、逻辑运算、控制及转移等几种类型指令。同一种类型指令对于流向的影响是一样的，所以在扫描机器码源程序时，只要对各种类型指令分类进行统一处理，就可建立具有通用性的扫描算法。当然，对于中断指令，不同的机器其差别是很大的，中断指令流向的自动跟踪是非常困难的，本文对中断转移不予考虑（原因见第三节）

怎样分类？各类指令又有何特点？这是通用扫描算法的关键。由于机器指令系统很多，而且还在不断增加，要想建立一个包括各种系统，各种类型的指令模型，不说不可能，至少要花费大量时间和精力。但是，扫描的目的仅是要找出所有源程序的指令，所以扫描方法应是跟踪指令流向，也即程序流程，这要靠在扫描一条指令时能找出下条指令的地址，由这个地址来确定下条指令，即跟踪流向。跟踪流向的关键之处在于扫描时是否能确定指令的正确地址。一般的指令都是顺序执行，其下条指令地址的确定是很方便的，主要是控制转移类型指令，它们往往改变正常的流向。抓住这个特点，我们建立一个适用于扫描的通用指令分类模型，它根据执行时确定下条指令地址的方式不

同将指令分为四种类型：

1）条件控制转移类：

它的特点是执行后有两个转移去处：（i）满足转移条件，其转移去处地址由指令操作数确定；（ii）不满足转移条件，其下条指令地址即为流向。条件控制转移类包括各种条件转移指令。

2）无条件转移类

它的特点是仅有一个转移去处，其地址由指令操作数确定，它包括各种无条件转移指令。

3）停机类

它的特点是执行后没有下条指令，是程序段结束的标志。从程序段开始指令到停机类指令，构成了程序中一条完整的路径。

此类包括停机指令，子程序返回指令及导致程序停止执行的中断指令。

4）顺序类

它的特点是按照指令线性排列顺序，执行完当前指令后，依次执行下条指令。它包括除上述三类之外的所有其它指令。

从上面四类指令特点可以看出，前三类指令都是程序基本块的末指令。

三　机器代码程序分析的讨论

这一节主要讨论机器代码程序与一般高级语言程序的不同之处。机器代码程序分析的特点及GMPAS采取的处理策略。首先介绍所谓“独立”指令和“相关”指令及它们对于程序分析的影响。然后讨论机器码程序分析特点。最后阐述所采用的处理方法。

1。“独立”指令与“相关”指令

此处关心的指令特点主要是指其对于程序流向的影响。当然对于指令其它方面的特性此处的讨论也适用。

“独立”指令是指其下条指令地址的确定完全可以由当前指令本身决定。而和程序的其它指令无关。如高级语言的*goto*语句。机器语言的部分转移指令，在Z80中，如*Call* 05H指令等等。它们的转移去处完全可由其自身决定。

所谓“相关”指令是指其下条指令地址的确定和程序中其它指令有关。这种指令在高级语言程序中不存在。高级语言程序的所有转移去处都可依语言文法在对源程序进行扫描时得到。而对于机器语言，在转移类指令中有许多“相关”指令。如Z80中的JP（HL）。它的转移去处取决于装载寄存器HL的指令。又如IBM／360的BZ 30（2，3）。它的转移去处取决于寄存器2，3的装载指令。

完全由“独立”指令组成的程序。其分析方法比较直观。首先通过找出所有转移指令的转移去处。以确定程序的控制流。然后通过找出所有信息处理指令（对变量进行加工的指令）的处理结果确定数据流结果。将这些数据流的结果作为控制流的函数进行分析，产生有关数据流动的信息。换言之，完全由“独立”指令组成的程序分析控制流与数据流的结果无关。