中国计算机应用文集

（3）

DJS100系列机软件专辑

中國計算機用戶協會

中国计算机应用文集

第三集

DJS100系列机软件专辑

中国计算机用户协会

目　录

∞∞综述与分析∞∞

∞∞开发与研制∞∞

∞∞改进与应用∞∞

软件移植的设想与实现

复旦大学　薛国良　赵隆赓

〔摘要〕本文讨论了对一般软件进行移植的设想和实现方法，并且为持有现成软件的DJS100系列机的用户提供了把它们移植到各类微型机和小型机上去的实现方法。对各类微型机和小型机的用户都有参考价值。

1. 背景

大规模、超大规模集成电路工艺的发展提供了功能齐全、价格低廉的微处理机器件，为微型机的发展开辟了广阔的前景。发展微机系统适合我国的经济条件和技术力量，目前已逐渐成为我国计算机推广应用的主要途径。各种廉价的新型的微机不断推出，小型机也迅速换代，这就向广大老用户和软件工作者提出了一个新课题，即如何多快好省地开发可靠的微型机或小型机软件，如何利用现成的软件，并如何将它们移植到新型的计算机上去，对于新的用户来说，这也是一个感兴趣的问题。

所谓现成软件就是用户已掌握的进口软件和由用户自己开发的使用成熟的软件。

例如NOVA系列机的软件和在DJS100系列机上开发的为大型绘图仪研制的绘图语言，电子传报系统，洪水预报系统，发电厂自动控制系统等等。这些软件的研制成功都花了较大的人力物力，有的耗费了几十个人年。

将现成的软件移植到新型计算机上去，可以用直接的办法，就是既要熟悉现成软件的设计原理和程序框图，又要掌握新型计算机的汇编语言，然后按程序框图重新逐条编制程序。这样做的缺点是明显的。

本文作者之一针对上述软件移植方法的缺点，自1976年起开始着手软件移植方法的研究，根据多年来的实践，提出了进行软件移植的设想：

1. 建立一种公共适用的程序设计语言。
2. 在具体计算机上构造一个通用的软件产生器。

本文主要针对上述设想介绍一些具体的实现方法，并探讨了有关方面的原理。

2. 公共适用的程序设计语言

我们这里讨论的软件主要是指应用软件；从计算机的推广使用而言，我们这里主要是指计算机应用，以及在计算机应用过程中产生的应用软件的移植问题。系统软件有些特殊问题，但这儿提出的原则也可借鉴，一般也是适用的，只要对特殊问题由人工专门处理即可。

随着计算机应用事业的发展和新技术不断推出，许多计算机应用系统所配置的设备不断更新，系统所配置的软件也必须作相应修改。这种修改工作量往往和原生产所花费的工作量是属于同一数量级的。为了解决这类问题，人们早已想到使用高级语言。使用高级语言编制的程序可以在很大程度上避免这个矛盾。但实际上相当一部分应用系统都用汇编语言作工具，原因有二个方面：

1. 汇编语言的编译所产生的目标程序质量一般比较高，而多数应用系统对目标程序质量有一定的要求。

2． 每一台具体的计算机通常都有现成的汇编程序。

3． 对于计算机应用中的某些特殊用途，现有的许多高级语言的语句无法描述和实现。使用这些语言编写程序时经常需要用汇编语言来编写相当一部分程序。例如使用BASIC语言编写应用软件，许多用户事先必须消化BASIC解释系统的编译原理，然后再利用CALL语句，事先用汇编语言编写被调用的子程序以弥补BASIC语言功能之不足。很明显，使用BASIC语言来设计应用软件不是上策。

由于计算机应用的面很广，高级语言又不能普遍适用，针对具体应用的专用的高级语言就不断增多，这类语言据统计就有500种以上，这种局面给软件的移植增加了重重障碍。

本文作者针对应用软件在编制程序过程中离不开直接使用具体机器的汇编语言的特点，自1974年开始在DJS100系列机（简记J100）上着手研制了一种从汇编语言（H100）"自拔"而成的高级程序设计语言JCY作为一种公共的程序设计语言（G100），其目的是使得不同机型（Ji）的用户都能用这种语言来编写程序，以达到便于交流便于移植的目的。

设计公共语言Gi的意义是：

1． 每一台具体计算机Ji都有现成的汇编语言Hi。

2． 所谓公共的程序设计语言Gj，是在具体Ji的Hi基础上扩展一种不依赖于具体计算机i的高级语言G，且定义如下： $Gi = \{Hi, G\}$

3． 在Ji上从Hi"自拔"成Gi相对容易，对已掌握Hi的用户来说只需要几个人月的工作量。JCY语言的设计成功为从Hi自拔成Gi提供了成功的经验。

4． 从Gi程序到Gi程序的移植只需要对Gj中属于Hi的少数指令人工或自动移植成Hj指令即可。

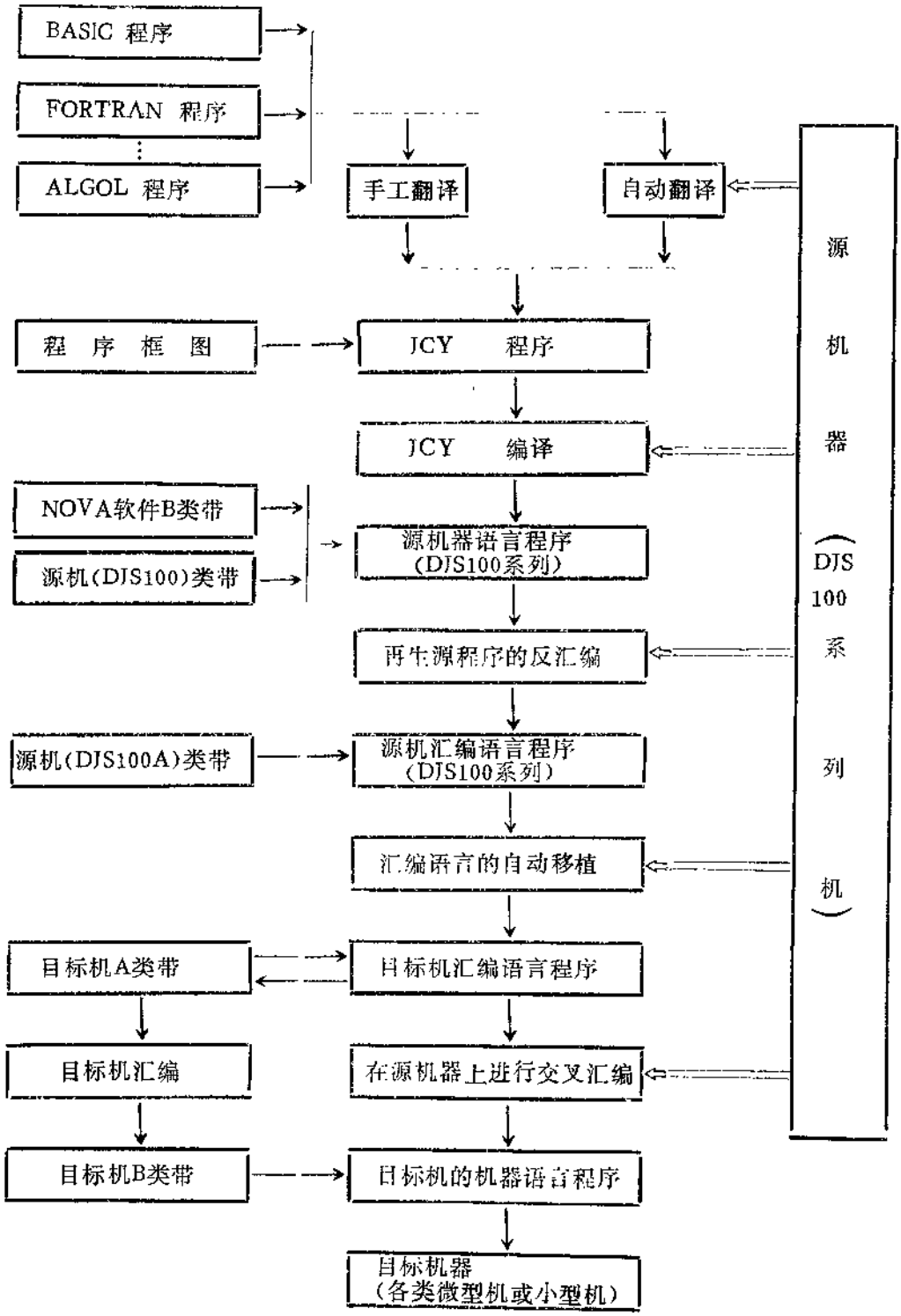
例如1976年用上述办法将富春江水电厂的洪水预报的ALGOL程序移植到DJS131机，先将ALGOL程序改写成G131程序，然后把131机的汇编语言H131自拔成G131，二项工作中的主要部分都是在1976年暑假完成的，同年9月底该厂购买的DJS131机安装验收的同时洪水预报程序系统也在DJS131机上投入运行并验收通过。

软件工程的实践表明软件维护的工作量通常要占整个软件生存期的一半以上。富春江水电厂洪水预报程序系统投入运行之后，发现由于上游修建了多处小水库，以致原设计的数学模型与实际不符。数学模型的修改必然导致程序的修改。由于使用G131语言编写洪水预报程序，修改很方便。如将洪水预报程序移植到DJS180系列机上去，对已配有G180的用户，只要稍加修改就可以直接对用G131语言编写的洪水预报程序在DJS180系列机上进行编译，达到了基本上自动移植的目的。

3． 软件产生器的设想与实现

所谓软件产生器就是将一种在源机器上运行的现存软件通过一个加工系统的加工自动翻译成目标机器上的软件。例如DJS100系列机上成熟的软件通过在该机种上用JCY语言设计的"软件产生器"自动加工成DJS180系列机的软件就是这种软件产生器的特例。

用户所持有的在源机器上使用成熟的软件可能有多种语言形式，例如BASIC、FORTRAN、ALGOL，汇编语言，机器语言（纸带格式）……。正在开发中的软件形式可能尚处于程序框图的形式。在DJS100系列机上实现软件产生器的设想可以用下图来描述。


BASIC 程序
FORTRAN 程序
ALGOL 程序
手工翻译
自动翻译
源机器(DJS100系列)
程序框图
JCY 程序
JCY 编译
NOVA软件B类带
源机(DJS100)类带
源机器语言程序
(DJS100系列)
再生源程序的反汇编
源机(DJS100A)类带
源机汇编语言程序
(DJS100系列)
汇编语言的自动移植
目标机A类带
目标机汇编语言程序
目标机汇编
在源机器上进行交叉汇编
目标机B类带
目标机的机器语言程序
目标机器
(各类微型机或小型机)

本节主要介绍再生源程序的反汇编和汇编语言程序的自动移植。有关JCY语言可参考文献〔1〕，在源机器上进行交叉汇编的方法可以参考文献〔2〕有关高级语言自动翻译问题，将留作以后讨论。

如上图所示，这里的源机器为DJS100系列机。

3.1. 再生源程序的反汇编

对机器指令程序进行反汇编，各类计算机都有现成的反汇编程序。这里所指的再生源程序的反汇编与各类计算机原有的反汇编有较大差别，差别在于前者将产生一种可逆的源程序，即：

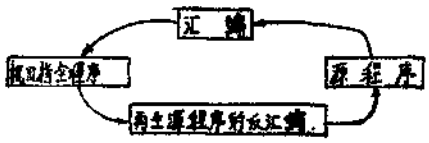
要实现上述可逆过程必须解决下面三方面的问题：

1. 再生源程序的反汇编所产生的源程序应穿孔输出形成一条A类带，或以文件形式存入外存。
2. 在输出的源符号程序中不允许再出现绝对地址。
3. 对指令和常数必须作严格的区别。

第1点很容易解决。以下讨论后两点。

3.1.1.标号的引入

在允许任意定位的源程序中是不允许出现绝对地址的，而普通反汇编所产生的源程序都写成绝对地址的形式，如：




```
STA     0,12001
LDA     1,12002
MOV     1,1,SZR
JMP     12004
JMP     @12003
```

同一段程序用再生源程序的反汇编输出则有如下形式：

```
STA     0,LAB01
LDA     1,LAB02
MOV     1,1,SZR
JMP     LAB03
JMP     @LAB04
```

在某一处有：

```
LAB01:      0
LAB02:      0
LAB04:      LAB05
LAB03:      MOV1,2
```

可以用设置符号表及二遍扫描的方法实现这种功能。

3.1.2.常数和指令的区分

由于计算机的指令，地址和数具有形式上的同一性，地址和数都可看作常数，关键在于常数和指令之间难以区别。但从使用的观点而言，常数和指令还是可以加以区别的，即常数具有被引用的特点，例如：

LDA　　1,12002

其中12002单元中的内容应理解为常数。为简单起见，我们可把12002单元中的内容及STA 0,12004中的12004单元中的内容都看作是常数。在反汇编时常数和指令就可按自己的特性进行还原。对于同时兼作常数和指令的情况，例如程序中有对自身的指令进行修改的情况，最好能同时并列出常数与指令两种形式。对此必须采取逻辑流程分析法，前提是只须知道总入口，这一方法也可以用程序自动实现，不过用此法需多花一次扫描。

3.2.汇编语言程序的自动移植

在进行软件移植时，如要将甲种汇编语言(源语言)程序翻译成乙种汇编语言(目标语言)程序，例如将DJS100系列机的汇编语言H100程序翻译成DJS180系列机的汇编语言H180程序或某微型机(如8080、Z80、8086等)的汇编语言程序，这步工作总的可分为以下三个阶段：

1．建立二者状态空间的对应关系

2．汇编指令的保功能转换

3．优化考虑

这种划分只是逻辑上的，实际着手时并非绝对按此次序，每一阶段都要考虑其它阶段与其相互影响，通盘兼顾。

以下我们分别加以讨论。

3.2.1.状态空间的定义及对应关系

我们把一种计算机的指令系统所相关的存贮单元，通用寄存器、状态位（或寄存器）等都各自看作一个一维状态空间，其值域为其可能处于的所有不同状态。设共有几个这种一维空间V_1，V_2……，Vn，则其状态空间Π定义为笛卡子积：$\Pi = V_1 \times V_2 \times \cdots\cdots \times Vn$

如源机器的状态空间为Π_1，目标机器的状态空间为Π_2，那么所谓从Π_1到Π_2的对应关系就是：对Π_1中的每一个点x_1必须有Π_2中的一个点y_1或一组点y_{11}，y_{12}…，y_{1m}构成的集合Y_1与x_1相对应，记作$x_1 \to y_1$或$x_1 \to Y_1$，并且如果$x_1, x_2 \in \Pi_1$，$Y_1 \subset \Pi_2$，$Y_2 \subset \Pi_2$，$x_1 \to Y_1$，$x_2 \to Y_2$，当$x_1 \neq x_2$时必须有$Y_1 \cap Y_2 = \phi$，而当$x_1 = x_2$时则$Y_1 = Y_2$。其中$Y_1 \neq \phi$，$Y_2 \neq \phi$。

我们用$|\Pi|$表示Π的状态（点）的个数，设

$\Pi_1 = V_1 \times V_2 \times \cdots\cdots \times Vn$，$\Pi_2 = W_1 \times W_2 \times \cdots\cdots \times Wm$那么必然要求$|\Pi_2| \geqq |\Pi_1|$。

一种较方便的构造对应的方法是对Vi，寻找某一Wj，有$|Wj| \geqq |Vi|$，那么就会$Vi \to Wj$。不失一般性，假设下标一致，即i = j；当$m \geqq n$且找到了n个这样的对应：

$V_1 \to W_1$，$V_2 \to W_2$，……，$Vn \to Wn$

那么就完成了状态空间的对应构造。显然，当m>n，$|Wi| \geqq |Vi|$ i = 1,2,…，n时，只须取Π_2的一个子集Π_2^*，然后构造从Π_1到Π_2^*的对应就可以了。

按前面的讨论和记号，可得出结论：

1．可移植的必要条件是$|\Pi_2| \geqq |\Pi_1|$

2．在指定Wi与Vi对应的前提下，可移植的必要条件是$m \geqq n$且$|Wi| \geqq |Vi|$。

实际完成这一阶段的工作并不象想象的那么容易。例如内存单元之间的功能不尽相同，

有些单元有特殊的作用和限制等等，所以要构造出合理的对应关系，往往需要结合下面两个阶段的分析结果来考虑。

3.2.2.汇编指令的保功能转换

每种计算机都有其特定的指令系统，也就是指令的集合。指令可形式地看作是定义在状态空间上的功能函数，任给一个初态，在该函数作用下必能达到一个确定的终态。即，设K_1为Π_1上的指令系统，$x_0 \in \Pi_1$，$f \in K_1$，则必有$x_t \in \Pi_1$使$f(x_0) = x_t$。

再设Π_2上的指令系统为K_2，$\varphi \in K_2$；如对任一$Xoi \in \Pi_1$，有$Yoi \subset \Pi_2$，$Xoi \to Yoi$，$f(Xoi) = Xti \in \Pi_1$，且有Yti，使$Xti \to Yti$，而且$\varphi(Yoi) = Yti$成立，那么就可令φ与f对应，也记作$f \to \varphi$。这时这一对应就称为指令的保功能转换。其中，$\varphi(Yoi)$表示 $U_{yi \leftarrow Yoi}\varphi(yj)$。

例如令同号的通用寄存器相对应，内存单元A与2A相对应，则DJS100的指令 JMP17,2可以直接转换成DJS180指令 JMP36 (R2)。

但实际上这种直接对应是不多见的，而我们也只要求能找出一组K_2中的指令φ_1、φ_2…、φn使得在如上条件下有$\varphi n \varphi n_{-1} \cdot \cdots\cdots \cdot \varphi_2 \cdot \varphi_1(Yoi) = \varphi n(\varphi n_{-1}(\cdots\cdots(\varphi_2(\varphi_1(Yoi))\cdots\cdots)$（n重）$= Yti$

就可以了。例如DJS100机指令ADDL*1,2,SNC可用一组DJS180机指令来对应：

```
MOV    R2,R5
ADD    R1,R5
ROR    R5
BCS    *+m      ; m表示下一条指令所占字节数加2。
```

这样成组的指令可以看作是一条广义指令φ'，即

$\varphi' = \varphi n \cdot \varphi n_{-1} \cdot \cdots\cdots \cdot \varphi_2 \cdot \varphi_1$

由于指令的保功能转换要对Π_1上的任一Xoi都有效，而不同机种的硬件功能不同，因此构造这种转换往往很困难，必须非常小心。首先要对指令系统K_1作深入细致的分析，以求正确把握其中每一条指令的功能；其次要熟悉指令系统K_2，了解其总体功能和特点；二者都必须分别放在各自的状态空间及其对应的背景上考查，只有这样才能准确有效地构造转换。

指令的保功能转换同样存在着是否可能的问题。好在指令系统通常总含有诸如存取、算术逻辑运算、移位、转移、I/O等指令或与它们相当的指令，它们的功能作为整体来考虑，基本上是相当的。这就为移植的可能性提供了保证。但这些指令同样不是简单对应的；有些特殊指令，如对状态位的操作、与中断有关的操作等，往往需特殊处理。当不存在对应的硬件功能时，一种可能有效的简单方法是指定专用的内存单元来进行软件模拟。这一点在前一阶段就应有某种设想。当目标机上的通用寄存器较充裕时，用它们来模拟通常更有效。

3.2.3.优化考虑

实际的翻译可以采用各种不同的途径。我们只讨论用软件实现的方法，不讨论诸如仿真等方法。一般方法有：

1. 程序自动翻译

也就是设计一个专用的软件系统，以源机器的汇编语言程序为输入，自动地输出相应的目标机器汇编语言程序。这比高级程序设计语言程序之间的翻译更复杂。

2. 手工翻译

即由程序员人工进行翻译。

3. 前两种方法的有机结合。

用程序自动进行翻译当然是人们追求的目标。但这样做时通常要考虑一般化的情况，加以统一处理，因此产生的目标程序质量往往不高。同时为简化问题，有时会加上一些合理的限制。但限制再合理，总会有损于翻译的严密性，从严格的定义来看无法保证正确性，而禁止一切限制会使质量进一步下降，甚至无法接受。另一方面手工翻译的质量较高，可以针对具体情况灵活地处理，充分发挥目标机指令系统的优点，但这对程序员的要求高，工作量大，而且出错多而又难免，查错也困难。最好的办法是将两者有机地结合起来，即先用专用软件生成一个尽管质量不高可是正确性有保证的初始结果，然后由人工进行局部优化，也就是说对每一有特定功能的程序片断进行优化，删除重复无用的操作，用更有效的操作替换低效的操作，必要时干脆采用新的方案。

从这一思路可以想到应设法由程序本身来实现这种局部优化工作，即使付出一定的系统开销也是值得的。下面我们探讨一下局部优化的思想和原理。

以前我们讨论的指令的保功能转换，立足点在于将一条源指令转换成一条或一组目标指令，也就是1:n的对应。这样目标程序的长度往往很长。如果考虑引入m:n的对应，也就是同时对一组目标指令加以考查，只要这一组目标指令构成有特定功能的块，就可把这一块同时转化为一组相应的目标指令以完成相应的功能。

例如下列DJS100系列机的指令组：

```
MOVZL   0,1
ADDL    0,0
ADD     0,0
ADD     1,0
```

功能是将AC0内容乘以10。两种不同的变换并列如下：

```
        1:n                          m:n
MOV     R0, R1              MOV     R0, R1
CLR     R4                  ADD     R0, R0
ROL     R1                  ASL     R0
ADC     R4                  ADD     R1, R0
BIC     #177776,R4          ASL     R0
ADD     R0,R0
ADC     R4
BIC     #177776,R4
ASL     R0
ADD     R4,R0
ADD     R0,R0
ADC     R4
BIC     #177776,R4
ADD     R1,R0
ADC     R4
```

```
BIC     #177776,R4
```

其中用R4模拟进位位，目标机为DJS180系列机。可以看出优化的关键在于删除了不必要的对进位位的操作。

但实际上不可能、或很难识别所有的功能块，而且这种识别很费时间和存贮空间，（例如，假定采用存放常见的组合再用匹配算法的作法。不过识别目标程序中的不必要操作是较容易的，这要求对输出缓冲区中的目标指令组进行扫描；缓冲区大小应适当，既要满足可能的超前或回溯扫描又不能占太多的内存。用这一想法，上例中实际的m:n变换输出为：

```
MOV     R0,R1
CLR     R4
ROL     R1
ADD     R0,R0
ASL     R0
ADD     R0,R0
ADD     R1,R0
```

统计资料的积累，对于应设计何种优化算法是有巨大参考价值的。

4. 软件移植的现状与展望

当前国际上由于硬件发展速度极快，不断换代更新，新机种不断推出，硬件的价格性能比持续下降；相比之下软件研制周期仍很长，费用高昂。这促使大多数用户尽量采用软件移植的办法。尤其出现了混机种的计算机网络之后，对可移植性尤其是数据的可移植性的需要更有所增加。从可靠性角度着眼，将成熟的软件移植到新的机器上比重新设计对用户更有吸引力。还有一点，就是成功的高级程序设计语言也需要在新的机器上实现，以便使用由该语言写成的程序，包括各种系统软件也能在新的机器上运行。这一切促使软件移植领域的研究与探讨一直很活跃。

使用一种中间语言去完成移植的设想一直很有吸引力，因为从理论上讲，某种高级程序设计语言可以设计一个编译程序自动转换成这种通用的中间语言，而它与机器的汇编语言之间的距离是很近的，可以相当容易地设计一个翻译程序将这中间语言转换成各种机器的汇编语言或机器语言。如PASCAL语言的P—code就是中间语言一例。国内也出现了XCY语言的X—code。

实质上从观念上看，可以把我们前面论述中假定的DJS100系列机的汇编语言当做是一种中间语言，那么已在DJS100系列机上实现的JCY语言的编译程序就是从程序设计语言到中间语言的编译程序，而将DJS100汇编语言转换成DJS180系列机汇编语言的程序就是翻译程序。不过这仅是从观念上而言的。具体地看，相对地讲DJS180系列机的汇编语言更适合于当这种中间语言，因为它与各种微型机的汇编语言有相当大的相似性，较能满足对中间语言的各项要求。

一般地，人们总是另外设计一种特别的中间语言以满足一定的要求，这种中间语言就对应了某一抽象机。我们提出取某一真实的机器及其汇编语言充当这种抽象机及其对应的中间语言的理由是：

1. 基本上符合人们提出的对中间语言的要求与抽象机应具有的功能。
2. 有利于测试编译程序，因为这是在真实的机器上运行；编译产生的代码可以真实地

运行以检验其正确性，这比抽象机上的中间语言优越。

3. 对国产机而言，一旦某种高级程序设计语言在该机上实现了，那么今后出现任何新的机种，只须考虑设计一个翻译程序和少量人工辅助工作就可将原有的软件移植过去。

4. 微处理机的应用日益广泛，尤其是16位微处理机发展很快，而相应的软件配置远不能适应这一形势。从指令系统角度看，各类微处理机与DJS180系列机有不少类似之处，这有利于可移植性。

随着形势的发展，今后有可能要对中间语言加以扩充，修改甚至重新设计。事实上不可能存在普遍适用的中间语言，这种改变是正常的。但至少可做到在相当一段时间内不必重新设计。

要真正做到软件的高度可移植性，首先应当在设计软件时就考虑这一点，所使用的程序设计语言应是标准化、统一的。然而实际上各种现在流行的程序设计语言都被各个使用者赋于各自不同的理解，各种实现都加上了各自不同的限制与扩充，这些都给可移植性带来了灾难性的影响，成为移植产生的软件出错的主要根源。因此语言的标准化问题应给予充分的重视。与国外设计的各种程序设计语言的“被污染”相比较，我们国内自己设计的几种高级程序设计语言尚“纯洁”，我们希望这些语言或正在研制中的语言在设计、推广运用时能预防产生这类问题。

参考文献

〔1〕薛国良　结构程序设计语言JCY参考手册
《JCY语言用户通讯》第一期

〔2〕INTEL8086/DJS100系列机交叉汇编程序
《JCY语言用户通讯》第一期

〔3〕P·J·Brown等　软件移植　（朱关铭等译）

〔4〕E·G·Duncan　微处理机的程序设计和软件研制　（白英彩译）

〔5〕PDP—11/70 处理机手册　DEC公司编
铁道部直属电子计算所译

〔6〕T·W·PRATT　程序设计语言的设计与实现
（郑人杰等译）

〔7〕杨德元　PASCAL可移植编译程序及其分析说明

〔8〕胡燕武、白光野　中间语言X—code的设计思想
《计算机学报》1982.2期

〔9〕R·E·Berry Exptrience with the Pascal P—compiler,
software—Practice and Experience, 8:5 (1978)

〔10〕P·C·Poole W·M·Waite Portability and adaptability,
in F·L·Bauer,Advanced Course on Software Engineering——Verlay,Berlin,1973

NRDOS 结构分析介绍

100系列软件中心吉林省电子所　NRDOS结构分析组

1980年年底，当时的国家电子计算机总局在长春召开了DJS153机系统软件结构分析任务协调会，会上确定由吉林省电子所和100系列软件中心共同承担NOVA3/4的带内存管理保护部件的实时磁盘操作系统——NRDOS进行结构分析。吉林电子所负责NRDOS的系统生成程序NSYSGEN.SV，磁盘自举程序BOOT.SV和系统程序库SYS.LB的分析任务，100系列软件中心承担NRDOS的程序库A、B、C、I、O的结构分析，经过两个单位的有关同志的努力工作，在两个单位领导和同志们的支持下，现在已经完成了这一工作。共分析程序106KW，画出了各程序的细框图约900张。写出了结构分析报告的初稿，並在哈尔滨市举办了用户培训班，向100系列用户介绍NRDOS的结构，並计划在今年年底在苏州举办一次。NRDOS的结构分析已经过专家的审定並通过了国家鉴定。整个NRDOS的结构分析工作是由16位同志完成的，吉林电子所郭益和分析系统生成程序NSYSGEN.SV，郝瑞瑞分析自举程序BOOT.SV，阎旭初分析系统库SYS.LB.100系列软件中心的董国成、刘本学分析A库；滕惠成、任志远、林墨君分析B、C库；熊允亨、李文英分析O库，韩然分析I库；田希环、张森、王志忠分析键盘命令解释程序；夏业勋、沈跃分析磁盘初始化程序。

NRDOS的结构分析是在RDOS和MRDOS结构分析的基础上进行的，科学院计算所，物理所、1015所、华东师大、黑龙江大学、北京计算机三厂、重庆大学、清华大学、辽源无线电三厂等单位都对RDOS和MRDOS进行了结构分析，他们出版的分析报告和程序框图，是我们分析NRDOS的基础，为我们分析NRDOS创造了有利条件。

NRDOS的结构分析，既有上述有利条件，同时，也存在一些问题，首先是，我们这次结构分析是在没有源程序的情况下进行的，只能靠反汇编打出各模块清单。这样，除了入口、标准外部及外部位移量等符号外，其他各种局部符号、助记符、参数，特别是注解部分的信息，均已全部丢失，这就大大增加了我们分析的难度。尤其是其中一些关键地方，由于没有注解的启示，需要反复琢磨其设计思想，在这方面花费了我们很多时间。其次是，这次分析的工作量有了很大的增加，程序库A、B、C、I、O，都有很多新的增加和改写，仅仅程序库O，就增加了一半的工作量，复盖个数已由原来的40个增加到61个，其他各程序库也有类似的情况，有的是我们第一次进行分析，如系统的生成，库B、CLI和磁盘的初始化等，没有更多的资料可供借鉴，只能靠我们自己摸索。面且，参加这一工作的有不少新同志，所以缺点和错误是肯定存在的。

在此100系列机诞生10周年之际，我们仅以NRDOS结构分析作为向纪念大会的献礼。为了说明起见，我们先回顾一下100系列操作系统的情况。自从第一台DJS130机诞生以来，在100系列的各挡机种上，先后配置的操作系统有：

1. SOS独立操作系统，主要用于设备的输入/输出，是一个单道单任务系统；
2. XRTOS扩展实时操作系统，主要是在SOS上增加了实时和多任务功能；
3. RDOS实时磁盘操作系统；

是一个以磁盘为基础的实时多任务两道作业的操作系统，具有批处理功能，不带内管部件，两道作业与操作系统共享32KW内存空间；

4. MRDOS国产机DJS140上运行，（NOVA840）带内存管理和保护部件MMPV，具有一个用户图和一个通道图；

5. NRDOS国产机153、152、142、155上运行(NOVA3、NOVA4)，具有内存管理部件MMU和内存保护部件MPU，内存管理和保护部件是分开的，NRDOS使用两个用户图和两个通道图，因此，NRDOS也可以称为在NOVA3或NOVA4上运行的MRDOS。

RDOS、MRDOS、NRDOS作为两道的实时多任务的磁盘操作系统，具有基本的共性和特点，它们都是以资源的管理——处理机管理，内存管理、信息管理和设备管理作为自己的设计出发点的。但MRDOS，尤其是NRDOS由于内管部件和其他功能的增加，已经有了很大的扩充。

例如，在NRDOS的内存管理中，增加了虚拟复盖和窗口变换功能，使得用户可以方便地使用扩充的内存地址空间；有协调功能，可以在生成操作系统中指定这一功能，改进系统软资源设置的合理化。NRDOS系统调用命令增加了很多，使用户在编制程序中更加方便。键盘命令功能强大而齐全，运行各种高级语言和实用程序非常方便灵活。

在多任务处理中，增加了操作员与任务的通讯，用户可象使用键盘命令一样，通过控制台来控制多任务的运行，如消去一个任务，改变任务的优先级，周期地生成和执行任务，使指定任务成为挂起，准备和执行状态等等。

从以上介绍可以看出，NRDOS是目前100系列机上所配置的功能最齐全和完善的系统。

大家知道，操作系统是计算机软件的核心和基础，掌握它的设计思想、实现方法、采用的技术，具有极为重要的意义，尤其是对维护系统的安全性和可靠性，改造系统为我所用，增配用户外设等等，只有掌握了操作系统的结构才有可能。

所以，我们分析NRDOS，正是建立在这一种认识上。

NRDOS的立体程序由5个系统程序库组成，这就是我们通常所说的A、B、C、I、O，一个符合需要的系统就是由这些库中的模块经过裁剪组成的，其中A、B、C库是常驻内存的，I库是为系统作初始准备而设置的，一旦运行后，其内存挪作他用。库O是系统复盖部分，主要包括存贮管理和文件系统，除了这5个库外，还有一些必不可少的实用程序，例如，为了生成一个合适的系统，需要有系统生成程序，为了能方便地从磁盘上自举一个操作系统或独立程序，需要有磁盘自举程序，为了用户的单任务或多任务处理和调试需要，必须有多任务处理模块和查错程序，为了建立磁盘上的文件系统，使用有了坏盘区的磁盘，需要有磁盘的预处理程序，即磁盘的初始化程序，还有一个必须用到的用户与操作系统的第一个接口——键盘命令解释程序，所有用户程序，高级语言、实用程序均由它调用，我们认为，掌握了这些，才算掌握了操作系统的主体，所以，我们除了分析NRDOS的主要组成部分外，还对系统的生成，磁盘自举、多任务系统，磁盘初始化和键盘命令解释程序进行了分析。

下面，我们分别介绍一下各个程序模块的简要情况。

一．系统库A：NRDOSA.LB，占用零页152_8个单元，非零页31622_8单元，共由45个程序模块组成，有框图137页，其主要功能模块为：

1. MMPU管理模块NMAPZ

该模块处理各种原因的进管，在带图的系统中，从用户区进入系统区（除硬设备中断

外）以及程序执行过程中遇致陷指令时，都要转入进管处理程序。进管处理程序根据是自动进管、违章致陷及用户自定义指令进管分别做不同的处理。由于NRDOS中算逻指令的不存不跳步指令要致陷以及MMPU的区别，使NRDOS的进管处理程序和MRDOS的进管处理差别较大，功能更为扩充。

2. 系统调度，由三个模块组成：

NSYST：系统入口处理，挂加工单和寻找进程，系统返回处理；

NSCHE：系统监督和各类进程启动程序；

NSYMO：全部Ⅱ类命令的语法检查程序和Ⅰ类命令的检查入口。

3. 复盖装配模块NOVLA

包括各种调用方式的复盖装配程序

4. 双机管理模块NDPMQ

包括各种双机调用子程序，双机部件IPB驱动程序及IPBQ进程加工处理程序，该模块向用户提供了磁盘文件共享功能。IPB的处理已与MRDOS的处理已完全不同。

5. 慢速字符设备驱动，对于每台慢速字符设备，都有相应的模块，存放各设备的DCT，其公用的驱动程序放在模块NIOBU和NRUG4中。TTI驱动程序在NTTYD中，操作员信息在NOPPR中。

6. 系统子程序模块NFILI，NGSUB，NBLKI，这些子程序绝大部分由文件系统调用。

7. 直接盘区读/写模块NRWBL，处理(扩充)直接盘区读写命令的执行（MRDOS该模块放在系统复盖中）。

8. 掉电中断处理模块NPWRF，处理主机掉电及恢复。

9. 各种假名处理模块。

二. 系统库B：NRDOSB.LB，由5个模块组成，不占用零页单元，占用非零页2616$_8$个单元，有框图6张。该库包括磁带和磁带盒的各种处理子程序和驱动模块NMTAD；此外，对于每台磁带机和盒式磁带机，都有相应的模块存放设备控制表DCT和目录控制块。

三. 系统库C：NRDOSC.LB，由60个模块组成，占用非零页单元37506$_8$个。有框图76张。该库有以下重要功能模块：

1. 各种型号磁盘的驱动程序模块，包括固定头磁盘、活动头磁盘、磁盘组和软盘的启动程序和中断处理程序，对于每台磁盘，都有相应的模块存放其DCT和一级分区目录控制块。

2. 各类多路通讯设备处理子程序和驱动程序模块NQTYD，包括QTY，ALM和ULM的中断处理程序，对于每一台通讯设备，都有相应的模块存放DCT和目录控制块。此外，还有一个假名处理模块NKQTY提供这些设备的假名处理。

3. 多处理机通讯转接器MCA驱动模块，其接收机和发送机的驱动程序在NMCAM模块中。

4. 主中断处理模块NINTD，包括主中断处理入口，时钟中断处理和中断解除程序。

5. 各种系统用表，包括系统栈，胞区和系统缓冲区等数据结构。

此外，该库还有系统转贮处理和浮点部件管理等模块。

四. 系统库I：NRDOSI.LB，由7个模块组成，占非零页10623$_8$单元，有框图30张，该

库完成系统的初始准备工作。主要包括给系统指示字置初值，对目录控制块处理，构造系统中断栈，构造系统图，对小CLI的空间移动处理，释放自由页，建立进程控制块链，对加工单的处理，准备数据通道图，构造缓冲区链，填主设备DCT，读磁盘的标识参数，复写坏盘区表，计算复盖文件的开始地址等。

初始准备程序有三个入口：部分初始准备，带复盖的部分初始准备和完全初始准备。这三个入口在给系统指示字和系统用表置初值方面的功能是完全一致的。区别主要是对磁盘文件的处理和是否要输入复盖。

小CLI询问系统的时期和时间，並根据面版开关的设置情况，决定是调CLI执行还是调用户程序RESTART.SV执行。

五．系统库O：NRDOSO.LB，是一个系统复盖文件库，由61个复盖模块组成，每个复盖占一个盘区（256字），其有效使用占非零页31515$_8$个单元，有框图188页。

系统复盖文件用以存放在系统运行中使用频率低的系统程序，所以，它所包含的都是各种系统命令的加工程序，其主要内容有：

1． 有一个系统设备文件用表（SFTAB），它占用一个复盖，其中记录了各种慢速设备的保留文件名，设备码及设备特性，供初始准备目录命令加工中选各种设备文件时使用。

2． 各种文件命令的加工程序，这部分约占用34个复盖。

3． 用户复盖（包括虚拟复盖）、交换、链接及窗口变换的加工程序（即用户存贮管理命令加工）。

4． 各类打开处理。

5． 系统设备的假脱机处理。

6． 系统协调功能处理。

还有读写操作员信息，双机切换，取或置系统当前时间等的加工程序。

六．系统库SYS.LB，这是一个用户程序装配时使用的系统程序库，包括单任务及多任务调度，任务命令处理，操作员信息、查错程序等。有模块50个，框图　　页，占用零页63$_8$个单元，非零页11352$_8$单元，主要功能模块有：

输入出缓冲处理包；

任务操作员通讯处理；

用户复盖管理；

各类任务调用命令处理，

查错处理；

单任务和多任务调度。

七．键盘命令解释程序CLI

CLI由三个文件：CLI.SV，CLI.OL和CLI.ER组成，其中CLI.SV是保存文件，占内存0页53$_8$个单元，非零页12015$_8$个单元，CLI.OL是复盖文件，包括14个复盖，每个复盖长度1.5kw，共计20kw。其中第14个复盖为分析解释各种命令的处理程序。CLI.ER是CLI及NRDOS的各种错误信息文件。出错时，根据错误代码，输出相应的错误信息。它以保存文件的形式出现，但不能执行，其长度为2.5kw。整个CLI有框图187张。CLI的执行分为4个阶段：

1． 初始准备；

2． 语法分析；