

# Z-80目标程序的自动重定位

侯 济 恭

〔计算机科学(电脑)系〕

## 摘 要

本文讨论Z-80目标程序的静态自动重定位问题.文中给出Z-80指令的重定位字典和相应的重定位算法,据此可自动、准确、快速重定位任何Z-80目标程序.

**关键词** 指令系统, 汇编语言, 计算机应用

## 一、前 言

在移植或改造Z-80目标程序时,常需将目标程序从一个区域移动到另一个区域.在设计自动控制程序时,也经常要在配有汇编程序的机器上预先调试好,然后再移回目标机(如单板机)去.诸如此类的问题都必然要遇到变更程序的某些指令,这就是程序的重定位问题.如何快速、准确地重新定位程序,这对系统程序设计人员来讲是非常重要的.本文结合移植dBASE-Ⅱ的经验,介绍一种快速自动重定位方法.

## 二、Z-80 指令集合分析

按所占有的字节数划分,Z-80指令集合可划分为以下4类.

(1) 一字节指令.例:LD A, B; 寄存器B值送累加器A

ADD A, B; 寄存器B与累加器A相加

(2) 二字节指令.例:LD A, 6; 常数6送入累加器A

ADD A, 6; 常数6与累加器A相加

(3) 三字节指令.例:LD A, (200); 200单元内容送累加器A

JP 300; 转移到300单元执行

(4) 四字节指令.例:LD IY, 123; 值123送变址器IY

LD (400), HL; 寄存器HL值送入地址400内

各类指令具有以下特点:一字节和二字节指令是关于寄存器之间的数据传送、算术逻辑

本文1988年1月7日收到.

运算以及 8 位数据传送, 因而它们与地址元关, 亦即指令的驻留区域不影响执行的结果。这二类指令在移动时无须重新定位。

三字节与四字节指令中, 有一大部分与地址有关, 表 1 (a) 之程序, 若从区域 100 移至区域 1000, 则必须修改相应指令的地址码, 变成表 1 (b) 之程序, 才能使该程序正确运行。

表 1 程序移动时指令的重定位

(a) 移动前程序	(b) 移动后程序
100 LD A, ( 112 )	1000 LD A, ( 1012 )
103 LD IY, 112	1003 LD IY, 1012
107 LD ( IY+1 ), A	1007 LD ( IY+1 ), A
10A DEC A	100A DEC A
10B JP NZ, 103	100B JP NZ, 1003
11E JP 0000	100E JP 0000
112 50	1012 50

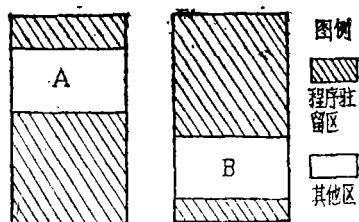
### 三、必须重定位之程序段分析

移动程序时存在以下二种情况 (1) 全局移动: 整个程序从 A 区移至 B 区 (图 1)。

(2) 局部移动: 程序的某一段 S 从 A 区移至 B 区 (图 2)。

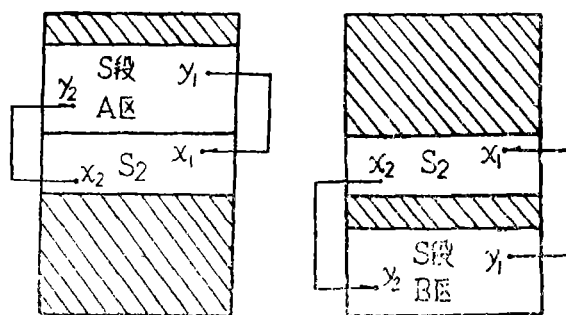
对于全局移动, 无疑必须考察所有组成该程序的指令的重定位问题, 对于局部移动, 问题比较复杂, 必须区别处理。

位于移动区域内的指令重定位时必须考虑到该指令的地址码部分是否与非移动区相关,



(a) 移动前 (b) 移动后

图 1 全局移动示例



(a) 移动前 (b) 移动后

图 2 局部移动示例

右线移动区内指令与非移动区相关;

左线非移动区内指令与移动区相关。

若相关, 则不必重定位, 如图 2 (a) 之线①, 移动区 A 内的指令在点  $y_1$  访问非移动区点  $x_1$ , 显然, S 段程序移至 B 区后, 点  $y_1$  的指令无需更动。位于非移动区内的指令同样必须考虑其指令的地址码部分是否与移动区相关, 若相关, 则必须重定位, 如图 2 (a) 之线②, 位于非

移动区的指令在点 $x_2$ 访问移动区A的点 $y_2$ ，当S段程序移至B区后（图2b），必须更改点 $x_2$ 的指令之地址码。

综上所述，可得如下结论：如果一指令的操作数是地址码且地址码位于移动区内，则该指令必须重新定位。

四、算 法 分 析

为了方便讨论，我们把Z-80指令分解为三部分：操作码 1，操作码 2 和地址码（操作数）。例如：LD（nn），BC 的指令代码是ED43xxxx，则ED为操作码 1，43为操作码 2，xxxx为地址码。JPnn的指令代码是C3xxxx，C3为操作码 1，xxxx为地址码。

不考虑单字节指令，则Z-80指令按操作码个数可划为 2 类。其一是单操作码指令，如转移类指令及八位数据传送指令，其二是双操作码指令，例如关于变址器IX和IY的指令以及输入输出指令等。这一类指令又可分为操作码 1 是ED（如输入指令）和操作码 1 是FD和DD的指令（如与IY和IX有关的指令），有意思的是，FD和DD的第二操作码完全相同，即它们是成对出现。如LD IY，nn和LD Ix，nn的代码分别为FD21xxxx和DD21xxxx。

据此，可建立三个结构完全一样的指令对照表和一个索引表。以此构成重定位字典。

1. 对照表。对照表由三个字段构成（表 2），每个字段占一个字节，各字段含义如下：（1）操作码字段：本字段存放指令操作码。对于单操作码指令，存放操作码 1，对于双操作码指令，存放操作码 2。TAIL单元存放本次待查指令操作码。（2）字节数字段：本字段登记相应指令的字节数，以作为伪程序计数器增量的依据。末栏字节数恒为 1，亦即凡表中不能查到的指令必为一字节指令。（3）修改标志字段：由于 3 字节和 4 字节指令中存在与地址无关的指令，故设立本字段以示区别。其中“Y”的机内代码为 1，“N”则为 0，末栏恒为N。

表 2 对照表		
操作码	字节数	修改标志
C3	3	Y
⋮	⋮	⋮
TAIL	1	N

对照表的详细结构，见附录 1—3。

2. 索引表。索引表由二字段组成（图 3），第一字段为关键字，占一字节，存放指令的操作码1：ED、FD、DD。最末一栏other存放本次待查的指令之操作码 1，表中第二字段为指针，分别指向相应的对照表，本字段占二字节。

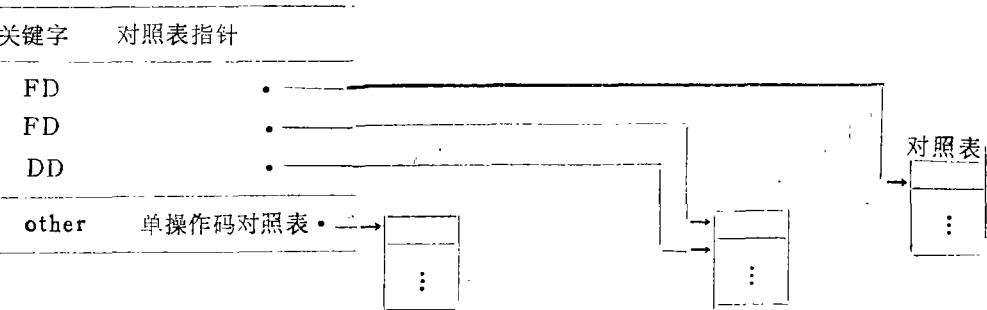


图 3 索引表与重定位字典结构

根据以上的重定位字典结构,很容易实现程序的重定位工作。其基本原理就是模拟CPU运行该程序。为此,设立二个动态指针:伪程序计数器SPC和对照表指针P。前者用于读指令,后者用于查对照表。重定位过程如下:

- (1) 由SPC,取出一指令操作码1;
- (2) 查索引表,得出相应的对照表首址,置入P;
- (3) 查对照表,判断该指令是否必须重定位,是则处理之;
- (4) SPC推进X字节(根据该指令节数而定);
- (5) 重复以上操作,直至到达被处理程序之终点。

将重定位程序分为二部分:一部分为查表程序,另一部分为重定位处理程序。程序结构示于图4。图中一些符号说明如下:

SPC. 操作码1  $\Rightarrow$  X, 将SPC所指向的指令之操作码1送X单元;

P. 字节数+SPC  $\Rightarrow$  SPC, 将P所指向的记录之字节数字段内容与SPC相加,结果送SPC,亦即SPC推进一条指令。

## 五、结 束 语

本程序用Z-80汇编语言编写,在APPLE机上实现。运行效果颇佳,证明其可快速,准确重定位Z-80目标程序。事实上,只须修改本程序的重定位字典内容,就可以达到对其它微型机的目标程序的移植工作,例如8085, 6502汇编目标程序。因此本程序具有普遍的意义与实用价值。

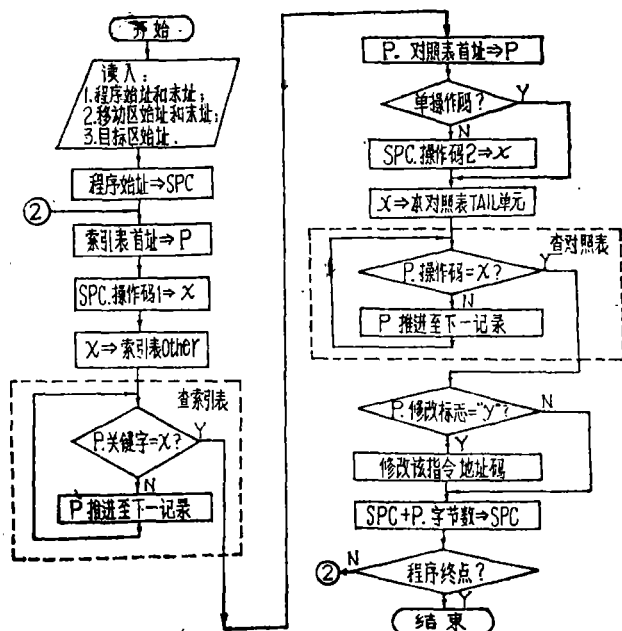


图4 重定位处理程序结构

附录 1 HD类指令对照表

操作码 2	字节数	修改标志	操作码 2	字节数	修改标志
			62	2	N
45	2	N	B3	2	N
4D	2	N	BB	2	N
5F	2	N	AB	2	N
4F	2	N	6F	2	N
47	2	N	67	2	N
A0	2	N	46	2	N
B0	2	N	56	2	N
A8	2	N	5E	2	N
B8	2	N	44	2	N
A1	2	N	4A	2	N
B1	2	N	5A	2	N
A9	2	N	6A	2	N
B9	2	N	7A	2	N
40	2	N	57	2	N
48	2	N	42	2	N
50	2	N	52	2	N
58	2	N	7B	4	y
60	2	N	72	2	N
68	2	N	43	4	y
78	2	N	53	4	y
A2	2	N	63	4	y
B2	2	N	73	4	y
AA	2	N	4B	4	y
BA	2	N	5B	4	y
A3	2	N	6B	4	y

附录 2 DD和FD类指令对照表

操作码	字节数	修改标志	操作码	字节数	修改标志
E3	2	N	70	3	N
23	2	N	71	3	N
2B	2	N	72	3	N
E9	2	N	73	3	N
09	2	N	74	3	N
19	2	N	75	3	N
29	2	N	77	3	N
39	2	N	34	3	N

续附录 2

操作码	字节数	修改标志	操作码	字节数	修改标志
<i>F9</i>	2	<i>N</i>	86	3	<i>N</i>
21	4	<i>y</i>	8 <i>E</i>	3	<i>N</i>
<i>CB</i>	4	<i>N</i>	96	3	<i>N</i>
46	3	<i>N</i>	<i>A6</i>	3	<i>N</i>
4 <i>E</i>	3	<i>N</i>	<i>B6</i>	3	<i>N</i>
56	3	<i>N</i>	<i>AE</i>	3	<i>N</i>
5 <i>E</i>	3	<i>N</i>	<i>BE</i>	3	<i>N</i>
66	3	<i>N</i>	35	3	<i>N</i>
6 <i>E</i>	3	<i>N</i>	38	3	<i>N</i>
7 <i>E</i>	3	<i>N</i>			

附录 3 单操作码指令对照表

操作码	字节数	修改标志	操作码	字节数	修改标志
06	2	<i>N</i>	36	2	<i>N</i>
0 <i>E</i>	2	<i>N</i>	3 <i>A</i>	3	<i>y</i>
16	2	<i>N</i>	32	3	<i>y</i>
26	2	<i>N</i>	01	3	<i>y</i>
1 <i>E</i>	2	<i>N</i>	11	3	<i>y</i>
2 <i>E</i>	2	<i>N</i>	21	3	<i>y</i>
3 <i>E</i>	2	<i>N</i>	31	3	<i>y</i>
<i>D3</i>	2	<i>N</i>	<i>C3</i>	3	<i>y</i>
<i>DB</i>	2	<i>N</i>	<i>C2</i>	3	<i>y</i>
<i>D6</i>	2	<i>N</i>	<i>CA</i>	3	<i>y</i>
<i>C6</i>	2	<i>N</i>	<i>CD</i>	3	<i>y</i>
<i>DE</i>	2	<i>N</i>	<i>DA</i>	3	<i>y</i>
<i>E6</i>	2	<i>N</i>	<i>E2</i>	3	<i>y</i>
<i>EE</i>	2	<i>N</i>	<i>EA</i>	3	<i>y</i>
<i>FE</i>	2	<i>N</i>	<i>F2</i>	3	<i>y</i>
<i>CE</i>	2	<i>N</i>	<i>FA</i>	3	<i>y</i>
<i>CB</i>	2	<i>N</i>	<i>C4</i>	3	<i>y</i>
38	2	<i>N</i>	<i>CC</i>	3	<i>y</i>
30	2	<i>N</i>	<i>D4</i>	3	<i>y</i>
28	2	<i>N</i>	<i>DC</i>	3	<i>y</i>
20	2	<i>N</i>	<i>E4</i>	3	<i>y</i>
10	2	<i>N</i>	<i>EC</i>	3	<i>y</i>
<i>F6</i>	2	<i>N</i>	<i>F4</i>	3	<i>y</i>
18	2	<i>N</i>	<i>FC</i>	3	<i>y</i>

★为LD BC, nn和LD DE, nn两者均可有LD(BC), A和LD(DE), A.

## 参 考 文 献

- [1] 中国科学院, 微型计算机丛书, 计算机研究与发展编辑部, (1982)。
- [2] 许茂元等译, Z-80汇编语言程序设计, 科学技术文献出版社重庆分社, (1981), 11。
- [3] DHITHL, H.M. *An Introduction to Operating Systems*, Addison-Wesley Publish Company, (1983)。
- [4] J.J. 多诺万著, 系统程序设计, 科学出版社, (1981)。

## Automatic Relocation of Z-80 Object Program

Hou Jigong

## Abstract

This paper deals with the static automatic relocation of Z-80 object program. It puts forward a relocation dictionary for Z-80 instructions. It puts forward also a concise and effective algorithm for relocation, with which any Z-80 object program can be relocated automatically, accurately and rapidly.

**key words:** instruction systems, assembly Language, computer applications