iAPX86、88系列服务程序

用户指南

用于8086开发系统

手册号　121616

第二十三册

iAPX86、88系列服务程序
用户指南
用于8086开发系统

手册号　121616

第二十三册

翻　译　李　净

校　对　王兆全

航空工业部第五七四八

序　言

这本手册描述并表示了怎样使用iAPX86,88程序开发实用软件(LINK86·86,Loc86·86，LIB86·86，OH86·86)。当这些程序是在1.0版本执行方式中或按了RUN键后其必须在系列Ⅲ中执行。

iAPX86,88系列系指处理机器芯片的整个系列（8086，8087，8088和8089）。8086是这个系列的第一个成员，因此本手册通常用8086来代表整个系列。本手册供程序员用ASM86，ASM89，PL/M—86，PASCAL—86或任何其他产生8086目标的语言翻译程序来开发8086、8087、8088和8089程序使用的。

本手册分为六章：

- “前言”，本手册应用程序的概述。

“LINK86”，完整地描述了LINK86的调用行及操作。

- “LOC86”，完整地描述了LOC86的调用行及操作。
- “LIB86”，完整地描述了LIB86的调用行及操作。
- “OH86”，完整地描述了OH86的调用行及操作。
- “探讨程序开发”，给出了几种程序开发的完全代注译的情况。

有关资料

下列的手册在你使用iAPX86，88实用软件的工作的各个方面可能是很有帮助的。

- 8086/8087/8088宏汇编语言参考手册用于8086开发系统　　121627
- Intellec Series III Microcomputer Development System Product Overview, order number 121575
- Intellec Series III Microcmputer Development System Console Operaing Instuctions,order number 121609
- Intellec Serie III Microcomputer Development System Programmer Reference, order number 121618
  erder number121618
- MCS-86 Macro Assembly Languagc Reference Manual,order number 9800640
- Macro Assembler Operating Instructions for ISIS-II Users, order number 9008641
- PL/M-86 Programming Manual, order number 9800466

·PL/M-86 Compiler Operator's Manual, order number 9800478
·PASCAL-86 User's Guide, order number 121539
·8089 Macro Assembler User's Guide, order number 9800938
·8086/8087/8088 Macro Assembler Operating Instructions for 8086-Based Development Systems, order number 121628
·The 8086 Family User's Manual, order number 9800722

符号的约定

{} 表示需要包含在大括号内的仅有的一项语法项。

[] 表示包含在中括号内的一项或几项语法项。

…… 表示前面的语法项重复若干次。（省略符号经常被用在中括号内，并带有一个逗号“[,…]”，用其来表示可以重述上述的项，但每次重复必须用一个逗号分开。）

| 竖杠用来分离中括号[]或大括号{}内的各种选择项。

MONOSPACE表示被表示的字符必须象被表示字符那样被重复。

斜体字 表示一个亚元符号，它可以被满足这个亚元符号规定的一项所代替。实际的符号可以是下列中的任何一个：

文件名：
最小尺寸
最大尺寸
段落
补偿值
地址 是数字并且必须遵循数字表达式的INTEL标准(见PL/M86或ASM86)。H下标为十六进制，B下标为二进制，O或Q下标为八进制，D或无任何下标则为十进制。
段名
模块名
类型名
组名
复盖名
公用符号 是由8086目的模块格式定义的。它们可以多达40个字符长，并且在任一命令中以任意次序包括下列的任意字符：
A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ?, @, :, ., _,

黑体字 在交互对话的例子中用黑体字表示用户的回答。

<cr> 表示回车。

目　录

第一章　前言

第二章　LINK86

第三章　LOC86

第一章　前　言

程序开发

程序开发是一个变化复杂的过程。复杂程度取决于所用的开发代码的语言、最终产品的复杂程度和所选择的方法。

图1—1表示开发过程和可用于—以iAPX86,88系列为基体的产品的开发的工具。

这本手册给你为使用iAPX86,88系列实用程序包的四个部分所必须的信息。iAPX-86，88实用程序包允许你在以8086为基体的开发系统上开发你的以8086/8088为基体的产品的程序。

具体地说，这些工具设计成当在8086执行方式时，在一个 Intellec 系列Ⅲ微型计算机开发系统上运行。

本手册所描述的工具有：

·LINK86，它是链接和装配工具。

·LOC86，它是再定位工具。

·LIB86，它是8086目的模块的程序库管理程序。

·OH86，它把8086绝对目标信息转换成十六进制格式。

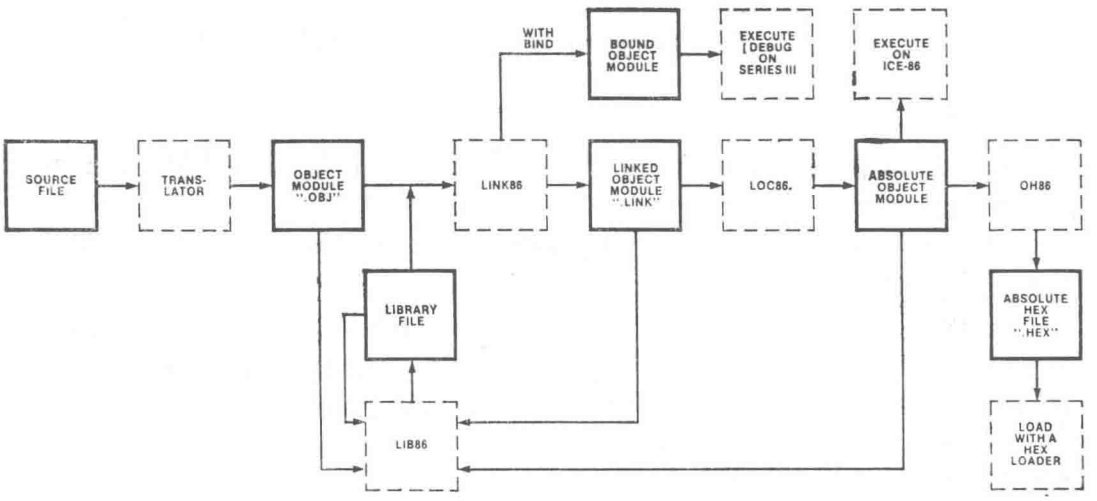

图1—1　iAPX86，88系列开发过程

链接和再定位的方法

LINK86把8086目标模块组合在一起。而LOC86将再定位目标模块转换成绝对目标

模块。ASM86，ASM89，PL/M—86，PASCAL—86以及基至LINK86、LOC86产生8086目标模块。

语言翻译程序通常产生带有未确定的外部参数的8086再定位目标模块。（在一定的环境下，翻译程序能够生产绝对目标模块，但这种情况是很少的，并且不能用于模块设计）。一个定位模块不包含这样的参数，这些参数要求把它们放在8086存贮器中某一特定的位置。8086代码不包括物理地址的参数。翻译模块的输出作为LINK86的输入。

LINK86把8086的一组目标模块组成一个单个的目标模块，並且企图把所有外部符号的说明都和公用符号的说明相匹配。通常LINK86的输出是一个再定位的目标模块：然而，在控制中被详细说明时，LINK86产生一个限定的目标模块。一个限定目标模块可在系列Ⅲ上执行。参见本章后面给出装入时可定位模块的说明。无论LINK86输出是限定的，还是再定位的目标模块，它都可以作为LOC86的输入。

LOC86把再定位（或限定的）目标模块转换成绝对目标模块。这目标模块的每一个段不能被执行或存取，除非它驻留在它的被指定的位置上。

被组合的目标模块在LINK86命令表中被说明。在输入模块中，段组合的顺序和指定给段的绝对地址的顺序是由模块列表的顺序，模块的结构和由命令提供的控制确定的。模块组合的细节在第二章和第三章中被描述。

在链接和再定位期间使用来自目标模块的下列信息：

- 作为段的补偿值给定的段的地址必须被译成存贮器的绝对地址或基本补偿值。
- 段即相同类型邻接的信息块，这些信息通常是代码或数据。
- 组即相同类型的段，这些段只能存放在存贮器的64K字节范围内。
- 级即相同类型的段，这些段享有公共属性，並且保存在一起。
- 公共符号和外部符号，可借助不同目标模块间的地址参数求出。
- 复盖即相同类型的模块，这些模块在执行期间在不同的时刻装入存贮器中。

相对寻址

在程序模块中的指令和数据的相对地址由源翻译程序指定。地址是相对于它们所在的段里的起始地址。相对地址实际上是从段的起点开始的字节数。

LINK86把一个或多个输入模块组合而构成一个输出模块，通过段来进行组合。组合段命令的次序在第四章中说明。

所有的输入段被组合之后，LOC86把绝对的存贮器地址分配给所有的相对地址。所得到的输出模块仅仅当它的段装入到由命令规定的绝对地址上时，才能够被执行。如果LINK86产生了一个限定的目标模块，那么LOC86就不需在系列Ⅲ上执行程序。

外部的参数和公共符号

在另一个不同的目标模块中，指出参数所在的位置的地址，就称为外部参数。一个外部参数不同于一个相对地址。因为产生模块的翻译程序不知道参数符号的位置。当编程序时，你必须声明这些参数是外部的。这就告诉了翻译程序，当然也告诉了再定位和链接命令，这些参数的目标，是在不同的模块中。

包括外部参数的模块被称作是不充分的模块。为了得到充分的模块，必须找到有公共符号的模块，这公共符号和外部参数相匹配。在一个模块中，联系公共符号的是一个地址，这个地址允许位于其它模块中，用适当的外部参数来定位在有公共符号的模块。

当编写这程序时，你必须声明这些符号是作为公共的。这就告诉翻译程序和再定位及链接命令，由其他的模块能够定位这个符号。

如果有外部参数，这些参数为公共符号，并且是不充分的，就会发出警告，并且所得到的模块仍然是不充分的。

程序库的使用

程序库在建立程序的工作中有所帮助。程序库管理程序，LIB86产生並且维护包括目标模块的文件。

LINK86以某种特殊的方法处理程序库文件。如果你指定一个程序库文件作为对LINK86的输入，它就检索程序库，在已经读入的输入模块中使这些模块没有结果的外部参数得到满足。这就意味着，在输入包括外部参数的模块后，程序库将被指定了。如果包括在程序库的模块有一个外部参数，程序库再次被检索，试着满足参数。这个过程继续下去，直到所有的外部参数被满足，或直到程序库中再也找不到一个公共的符号，未匹配这个不满足的外部参数。

当LINK86检索程序库时，通常在满足外部参数的输出中它仅包括程序库模块。如果程序库不能满足外部参数，在输出模块中就不包括来自程序库中的模块。然而，即使对它来讲没有外部参数，LINK86也提供了一种方法，使其无条件地包括一个程序库模块。图1—2表示LINK86对程序库文件的管理。

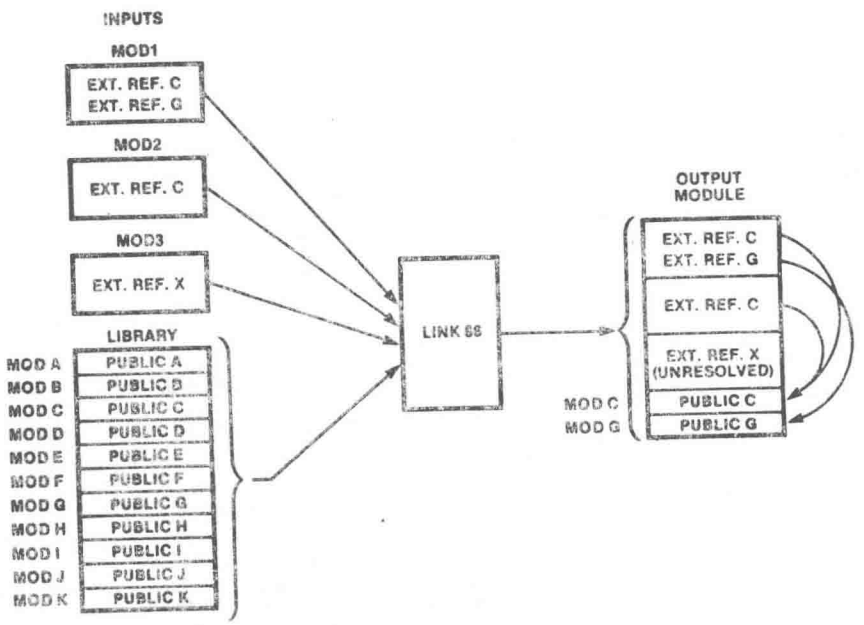

图1—2 用LINK86进行程序库链接

LINK86/LOC86处理

LINK86和LOC86在Intellec系列Ⅲ开发系统上执行。通过系列Ⅲ命令起动执行。和输入模块（LOC86）一起，这些命令可以包括影响它们的输出的控制。对LINK86和LOC86执行不需要控制。对于模块组合，地址分配和输出信息有一些执行缺省。控制给你改变缺省算法和输出的类型的能力。

输入是位于磁盘文件上的目标模块。输入模块包括相对地址、绝对地址、外部参数和公共符号。输入模块必须是按照8086目标模块格式，就象是通过PASCAL—86，PL/M—86编译程序，8086和8089汇编程序和LINK86与LOC86它们自己产生的一样。

LINK86把来自输入模块的段组合起来，且限制目标模块，LINK86安排组中的段和分配偏移值。LOC86按照用命令与/或缺省算法规定的控制安排段和分配绝对地址。处理完成时两种命令把模块和各种错误信息与诊断信息一道输出。图1—3示出LINK86/LOC86过程。

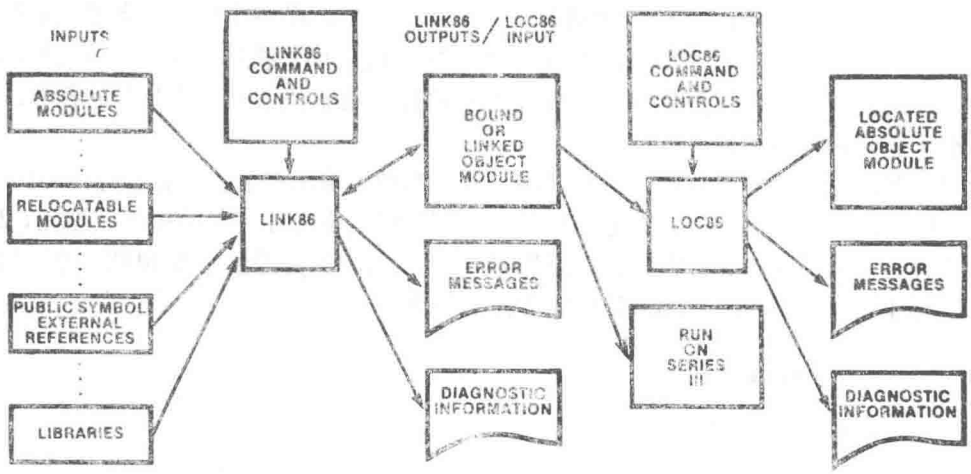

图1—3 LINK86/LOC86过程

8086概述

为了充分地使用再定位和链接命令，你必须搞清楚8086程序是如何构成的及怎样使用存贮器。

存贮器

8086能寻址存贮器高达一兆字节。十进制1兆字节是1048576字节。存贮器地址总是用十六进制表示。存贮器的1兆字节具有的地址：OH到OFFFFFH。

不是所有采用8086的系统都有完整的一兆字节的存贮器。很多系统在可利用的存贮器中将有间隙。存贮器的不同部分可采用不同类型的存贮器芯片来实现。系统监控程序或管理程序通常被存贮在ROM或PROM芯片中。因为在执行过程中它不被修改，它是系统中永久不变的部分。这样就避免每次在系统运转时都需要输入它。所用的数据被放在高速RAM中，因为它经常要被修改。用的比较少的数据通常可以放在低速存速贮器中。

系统存贮器的大小和组成都完全取决于使用的应用系统。

无论你采用什么样的8086系统存贮器，链接和定位用来对你的程序进行链接和定位都是可以实现的。在任一已给的存器贮结构中，它都提供了很灵活的段的位置。

8086寻址方法

8086用20位地址来组成存贮器地址，这20位地址由段地址和相对于段地址的16位偏移值所构成。这意味着用一个单个的段地址，通过仅仅改变偏移值直接寻址64K的存贮器字节。

一个硬件段地址是20位地址。但是段地址被强制放在16（10）H的位置。段地址可被置于任意一个用0结尾的十六进制的地址上：

OH
010H
020H
⋮
OFFFFOH

因为20位段地址的低四位总是0，因此段地址只用十六位就可以表示。

段地址存放在四个16位的段寄存器的某一个中。因为有四个段寄存器，在任何一瞬间，8086能存取存贮器的256K（4×64K）字节。通过改变段寄存器中的数值存取存贮器的整1兆字节。图1—4示出8086寻址概念。

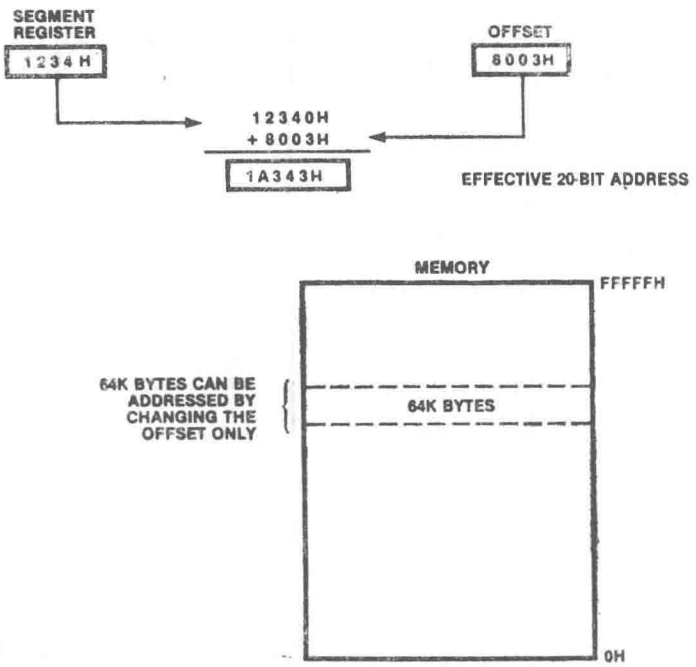

图1—4　8086寻址方式

段

程序是由被称作段的东西组成的，这些段是链接程序和再定位程序的基本单元。对于存贮器硬件，这些基本部分都有特定的目的。被存放在ROM或PROM中的程序部分能够放在与被存放在RAM部分相分离的段中。

8086汇编程序允许程序员命名被开发的程序的段。PL/M86编译程序给段予先生成一个段的名字。

段是存贮器的相邻的部分，它是在编译时被定义的（汇编或编译）。当被定义时，段没必要有一个固定的地址或大小。在定位时，要指定给段一个固定的地址。通过组合段和规定专门大小的控制能够改变它的大小。有些翻译程序可以产生带有绝对地址和具体大小的段的绝对目标指令。

LINK86从所有输入模块中将所有相同的完整名字（段、类型和复盖）和组合类型（存贮器，栈等等）的所有的段组合在一起。段的排列次序是在这些被组合的段的基础上作的。段被组合的方式取决于段的算法（这在下一课题中叙述）和与段相关联的组合特征。

当我们指出组合段时，我们是在谈及段将怎样被装入存贮器中，不是在讲将它们存贮在输出模块中。在LOC86输出模块中的段包括地址，这些地址决定了在存贮器中它们将从那里被装入。象它们在输入模块中一样，按照相同的顺序，它们存放在输出模块中。图1—5示出了输入模块，输出模块和被装入的程序之间的物理关系及输入的程序。

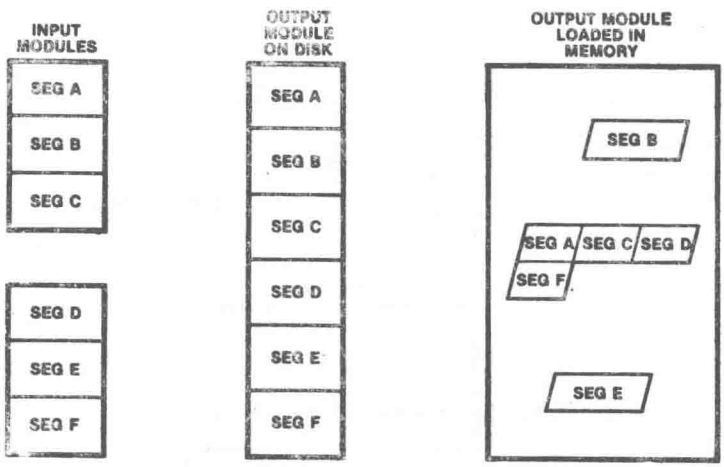

图1-5段的物理意义

段的排列

一个段能有五种排列属性中的一种（並且就一页内的属性来说，有两种）：

·字节，它是指一个段可在任何地址被定位。

·字，它是指一段仅能在一个地址上被定位，这个地址是2的倍数，从0地址开始。

·段落，它是指一段仅能在地址是16的倍数的地址上定位，从0地址开始。

·页，它是指一段仅能在地址是256的倍数的地址上定位，从0地址开始。

·页内，它是指一个段能在上述的属性的无论那一个中被定位，这个属性所规定的定位条件必须遵守，但是它不会跨过一个页的界线。

图1—6示出段的排列界线。

除了字节以外任一排列属性在组合的段之间都能够产生一个间隙。例如，当两页被排列时，段被组合，总有一个间隙，除非在长度上正好是256个字节的精确的倍数。

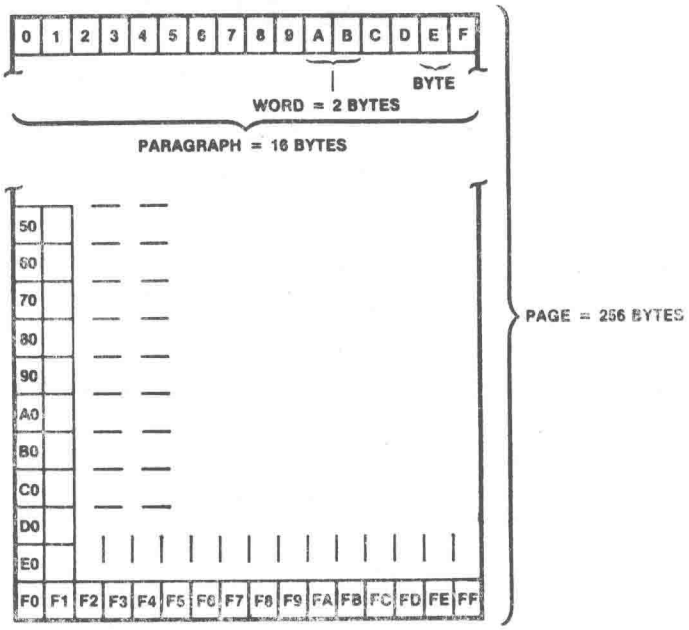

图1-6段排列的边界

段的组合

包括数据和代码的段首尾衔接地被组合。如果排列属性要求它，两段之间可以有一个间隙。对于新的较长的段，段中相对地址应调整。

段的组合有两个特殊情况：堆栈段和存贮器段。PL/M-86用名字STACK和MEMORY来定义这些段。使用ASM86，你必须通过在SEGMENT指令中加STACK或MEMORY参数来定义它们。

当堆栈被组合时，它们被复盖，但它们的长度被加在一起。

当存贮器段被组合时，在公共地址上用它们的地位地址复盖它们。被组合的存贮器段的长度是被组合的最大段的长度。不相对的地址的调整是必要的。如果没有控制使用，通常就在程序段上面的空余部分（在高位存贮器地址）装入存贮器段。

为了确保堆栈段被正确地组合，你应当在每个模块中给它们相同的段名。对于存贮器段也是这样的。如果你把汇编语言程序和PL/M-86程序链接在一起，你应当给它们能与PL/M-86兼容的名字STACK和MEMORY。

段的定位

段是按照在输入模块中遇到它们的顺序被定位的。如果类型（在下一节被描述）被定义，来自类型的段一起被定位。通过使用控制能改变定位算法，在第三章中用命令来

描述定位控制。

段的顺序定位的一种变化就是如何定位MEMORY段。当遇到具有存贮器特征的第一个段时，被放在存贮器段的最后。这意味着所有其它段被定义以后，MEMORY段将被分配在输出模块的最高地址中。

注意

如果在任何LOC86控制中（而不是SEGSIZE）出现MEMORY段的名字或类型的名字，或者它有绝对属性，MEMORY段可以不被定位在模块的顶部。

级

级是段的集合。当段在汇编语言中被定义时，一个级名被指定。用予先定义的段名产生由PL/M产生的段。能够给任何数量段以相同的级名。级名能扩展到模块边界以外，在被组合的不同的模块中能够使用相同的级名。

级的主要目的是将段组合在一起来共享一个公共属性（以任意次序），並且是为了通过指定的唯一的级名在定位时改变这个集合。

在输出模块的存贮器地址空间中，带有相同级名的所有的段被定位在一起（你可以通过在有LOC86 ORDER控制或LOC86 ADDRESSES控制段的指定位置，不考虑级的集合）。

级给你集合的第二个含义，类似输出模块中的段。首先给出相同名字的段。如果你开发几个被组合的模块，你可以要求给出的段在每个模块中包括可执行的代码名CODE。在包含可执行代码的模块内如果有几个不同名字的段，你可以要求给出这些段的CODE的类型名，可使它们在一起被定位，但不被组合。（对于段和类型能够使用相同的名字）。

组

组也是段的集合。在8086目标模块中组定义了寻址范围界限。组规定了段的集合，这些段必须在64K字节范围内被定位。这就意味着能用某个段寄存器的偏移值寻址段的整个组。或者，换句话说，当寻址一组中任何段时，不需要改变段寄存器。这就允许高效率的在模块内寻址。

组寻址总是在16的倍数的地址上开始（即，一个段落边界）。再定位和链接程序（R&L）不改变一个组的段，确保它们落在64K字节范围之内。然而，如果它们没有装配在这个范围里，就发出警报信息。

包括在一个组内的段在输出模块中不必都是邻接的。仅仅要求在组中定义的所有的段必须都落在组的开始地址的64K字节内。

复盖

有时你的8086程序太大不能装配到系统中的可用的存贮器中。复盖允许程序大于可

用的存贮器。

典型的情况是，复盖由代码和数据组成，这些代码和数据在程序执行的一个阶段中执行，但在任何其它时间不使用它。一但执行完，这个代码使用的存贮器能够用在某一其它阶段使用的代码和数据重写。在执行过程中不同时间占有存贮器相同部分代码的部分被称作复盖。

被复盖的程序部分总是驻留在存贮器中；它通常是由主程序模块，经常被使用的程序和装入程序组成。程序的这个部分被称为根。图1—7示出了使用复盖的一个程序的存贮器的配置。

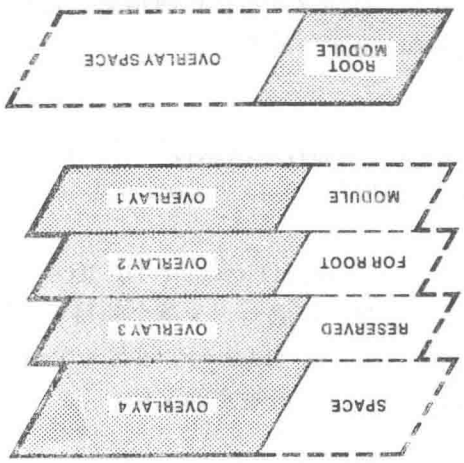

图1-7　在有复盖程序的存贮器配置

位置独立的代码和装配时可定位的代码（PIC/LTL）

PIC/LTL是指限制目标模块。LTL程序能够被装入到存贮器的任何地方。（假设排列属性被承诺）。允许指出段的基地址（段寄存器）。当它决定每一段在哪里定位时，输入程序必须求出这些段基地址的参数。在执行LTL程序以前，装入程序也必须初始化段寄存器。

一个PIC程序就是一个LTL程序，但是它不包括段基地址的参数。为了执行这些程序，输入程序需要把程序放到存贮器中（识别排列属性），同时初始化段寄存贮器并且运行。不需要安排段的基地址。

第二章　LINK86

LINK86联合8086模块並且求解出各个独立翻译的模块之间的参数。LINK86取得文件的表格和控制将其作为输入，並且产生两个输出文件：打印文件和目标文件。

图2—1说明链接过程。输入文件可以是任何链接文件（从一个翻译程序、LINK86、LOC86或8086库文件输出）。打印文件包括诊断信息。目标文件是一个限定在装入时可定位的模块，或是一个简单的再定位模块。

LINK86调用行

当你的Series Ⅲ是在8086执行方式时，必须执行LINK86。对于有关如何启动8086执行方式和在8086执行方式时怎样执行程序的详细情况，参见Series Ⅲ控制台操作说明书。

调用行的一般语法是：

〔：Fn：〕LINK86〔.86〕input list〔TO output file〕〔controls〕

输入表是一个或更多项被链接在一起成为一个单个的目标模块。表的元素既可以是一个输入表控制（PUBLICSONLY）也可以是一个包含一个目标模块的文件。用下列方法中的任何一个都可以生成在输入表中所规定的文件：

- 翻译程序目标文件
- 来自LIB86的程序库文件（或是程序库文件的一个单个模块）
- 来自LINK86的一个局部被链接的目标模块
- 来自LOC86的一个被定位的目标模块输入表中模块的顺序影响输出表中段的顺序。

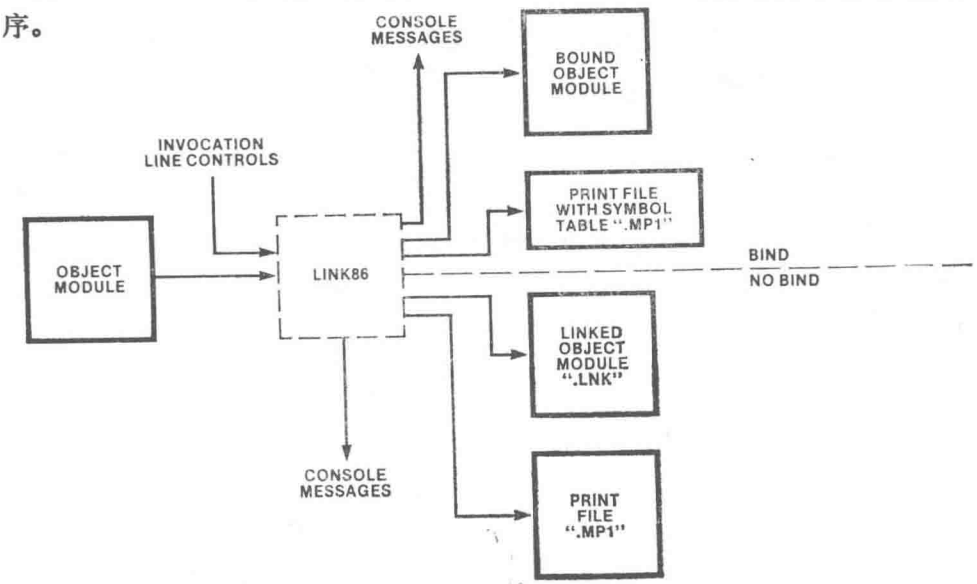

图2-1　LINK86输入和输出文件

TO output file指定此文件file去接受被链接的目标模块。如果输出文件（output file）没有被指定，那么输出就指定给某个文件，这个文件与在输入表中的第一个元素有相同的路经名，但是它的扩展部是“LIK”。如果表中的第一个元素是PUBLICSONLY控制，那么在它的变元中的第一个文件名被用作为缺省名。

如果指定BIND控制，那么用于输出文件的缺省名没有扩展部，並且目标模块没有定位就能够在Series Ⅲ中执行。参见第六章的例子。

控制能够是在表2—1中所描述的控制的任一子集：

表2—1 LINK86控制摘要

控　　制	缩　写	缺　省
BIND	BI	NOBIND
NOBIND	NOBI	
COMMENTS	CM	COMMENTS
NOCOMMENTS	NOCM	
LINES	LI	LINES
NOLINES	NOLI	
MAP	MA	MAP
NOMAP	NOMA	
MEMPOOL(最小尺寸〔，最大尺寸〕)	MP	不可用
NAME(模块)	NA	不可用
OBJECTCONTROLS C	OC	不可用
{LINES | NOLINES | COMMENTS | NOCOMMENTS | SYMBOLS | NOSYMBOLS | PUBLICS | EXCEPT(符号)〔，……)〕 | NOPUBLICS〔EXCEPT(符号〔，……) | TYPEN | OTEPE PURGE | NOPURGE}〔，……〕		
ORDER({组({段〔}〕/级〔/复盖〕〕} 〔，……〕)} 〔，……〕)	OD	不可用
OVERLAY〔(复盖)〕	OV	NOOVERLAY
NOOVERLAY	NOOV	