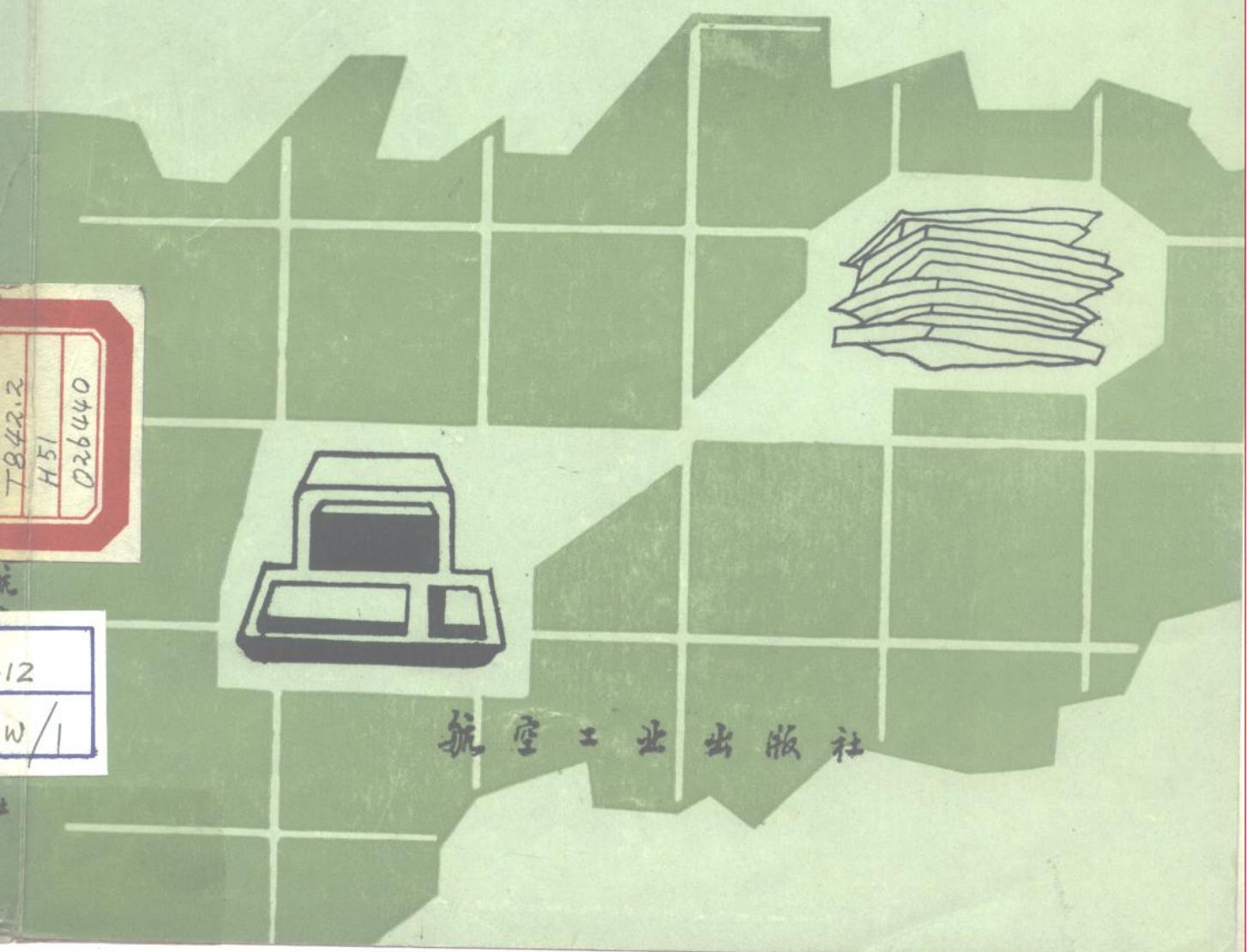
PL/M-86程序设计语言

PL/M-86
程序设计语言

胡汉文 吴立华 傅铅生 编著



T842.2
H51
026440

12
W/1

航空工业出版社

TP312
HHWY/1

南京大学无线电系 图书

PL/M-86
程 序 设 计 语 言

胡汉文 吴立华 傅铅生 编著

南京航空学院 藏书
026440 158
0026912

航 空 工 业 出 版 社

内　容　简　介

本书全面地介绍了美国 Intel 公司专门为微型计算机设计的 PL/M-86 高级语言，同时介绍了结构化程序设计的基本方法。

本书针对读者自学特点，对各部分内容作了较合理安排，由浅入深，循序渐进，突出重点，并附有不少程序设计实例，为使读者使用方便，还介绍了南京航空学院研制的 I/O 软件。

本书适用于计算机类、控制类、导航类和测试类等专业的师生，以及从事软件工程的技术人员。

JS0621

PL/M-86

程序设计语言

胡汉文 吴立华 傅铅生 编著

航空工业出版社出版
新华书店北京发行所发行　　各地新华书店经售
南京航空学院印刷厂印刷

开本：787×1092 毫米 1/16　　印张：12.75
1987 年 11 月第一版　　1987 年 11 月第一次印刷
印数：1-2450　　字数：306 千字
统一书号：15448·28　　定价：2.90 元
ISBN 7-80046-025-8/TP·004

前　言

PL/M-86 语言是 Intel 公司为其微型计算机开发系统软件和应用软件专门设计的一种语言，也是世界上第一种专门为微型计算机设计的高级语言。它的文本是 Intel 公司1976年公布的，先有 PL/M-80 语言，它用于 8080/8085 系列微型机，后在 PL/M-80 的基础上作了不少修改和扩充，将它用于 8086/8088 16 位微型计算机上，并称它为 PL/M-86 语言。这是一种块式结构语言，比较适用于结构化程序设计。

PL/M-86 语言不仅具有一般高级语言的特点，而且还能象汇编语言那样直接利用 CPU 的硬件特性进行程序设计。例如，它的输入/输出指令能直接访问口地址，因此可以通过口地址对 8086 的外围芯片如 8251，8253-5，8259A 等进行通讯，这对采用 8086 来组建各种控制系统和测试系统等都是非常有利的。另外，PL/M-86 语言本身还提供了不少内部过程，用户在编写程序时可直接调用。还应该指出的，IRMX86 操作系统本身也是用 PL/M-86 语言编写的，它向用户提供了许多系统调用命令，并且都是以 PL/M-86 语言形式提供的，所以用户在编写程序时，如果需要，可直接进行系统调用。

用 PL/M-86 高级语言开发的系统软件和应用软件都具有较高的程序质量，表现在1)可维护性好，即便于发现程序中的错误，便于修改和增添。2)可靠性高，用 PL/M-86 语言编写程序容易达到正确的目的。同时用它编写的程序即使在硬件出某些故障的情况下也能进行一定的运行。3)占内存容量小，运行速度快。若程序的长度超过 2K，则占内存大小和运行速度都和汇编语言相接近，程序越大越接近。4) 可理解性好，用 PL/M86 语言编写的程序是块式结构，层次清晰，便于理解和阅读。另外从开发软件的速度和成本来看，用 PL/M-86 高级语言编写要比用汇编语言编写的速度要快约五倍，成本也就相应的降低，可见 PL/M-86 语言是开发 Intel 16 位微型机系统软件和应用软件的十分有效的工具。

随着微机的广泛应用，大量的应用软件需要开发。应用 Intel 16 位微机的广大工程技术人员迫切需要掌握 PL/M-86 语言，但苦于没有一本比较系统而完整的且便于自学的教科书或参考书。本书是编者多年应用该语言取得实际成效的基础上所作的总结和归纳，它不仅系统地介绍了 PL/M-86 语言的基本概念、语句、过程、内部过程和程序结构等，而且还例举了不少考机过的程序实例。从内容处理上由浅入深，便于自学。本书可作为高年级大学生和研究生教材，也可作为广大工程技术人员开发系统软件和应用软件工作的参考书。

由于编者水平有限，错误在所难免，请读者批评指正。

编　者

1987. 5. 5.

目　录

试读结束：需要全本请在线购买：　www.ertongbook.com

第一章　PL/M-86语言的基本概念

1.1　PL/M-86 语言简介

PL/M-86 语言是由PL/M-86 语句组成的，PL/M-86 语句分两大类。

(1) 说明(DECLARE)和过程(PROCEDURE)语句，DECLARE 语句主要用来定义计算对象，如说明变量的类型以及为它们分配存储单元。变量可以是标量（纯量），或是数组，或是结构。PROCEDURE 主要用来组成一个过程，以供调用。过程总是以一个 PROCEDURE 开头，并以一个与其相匹配的 END 语句结尾，详细情况将在以后介绍。

(2) 执行语句有

赋值语句
GOTO 语句
IF 语句
简单 DO 语句
重复 DO 语句
DO WHILE 语句
DO CASE 语句
END 语句
CALL 语句
RETURN 语句
HALT 语句
ENABLE 语句
DISABLE 语句
空语句
CAUSE $ INTERRUPT 等

大多数执行语句将产生机器代码，PL/M-86 编译器接受这些语句作为输入，并产生一个机器代码模块作为输出，一条 PL/M-86 语句可由编译器 PLM86 翻译成为一条 8086 指令或一串指令，或者仅分配存储单元。

PL/M-86 语言除基本语句外，还有不少内部过程可以调用，另外，可使用系统调用命令进行系统调用。

一个“编译块”是 PL/M-86 语言的一个模块。一个编译块不一定是一个完整的程序。在编译后，如果该编译块是一个完整的程序，则可用 LINK86 将其和有关库文件连接起来，并经定位后便可执行。如果一个程序是由几个模块组成的，则可用 LINK 86 将这几个模块和有关的库文件连接起来，并经定位后便可执行。

1.2　PL/M-86 语言的基本符号、标识符和保留字

1.2.1　基本符号

英文字母

大写：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

小写：a d c d e f g h i j k l m n o p q r s t u v w x y z

数字：ϕ 1 2 3 4 5 6 7 8 9

算术运算符：+ − * / MOD

关系运算符：< > = <= >=

逻辑运算符：NOT AND OR XOR

分界符：· ( ) , : ; '

特殊字符：@ $ — := /* */

大、小写英文字母彼此不加区分，除非在字符串中，在上述基本符号中极大多数是大家所熟悉的。现仅将几个较特别的符号进行如下说明：

符 号	名 称	用 途
@	at符	引用地址
:=	赋值号	内嵌赋值时的赋值号
$	美元符号	①数和标识符中加入 $，是为了改善可读性，但无实际意义。 ②控制行的首符号
—	下划线	可作标识符中的符号
/*	注释前分界符	进行程序注释用
*/	注释后分界符	

1.2.2 标识符和保留字

标识符是用来命名变量、过程、宏说明和标号等。一个标识符最多可用31个字符长，第一个字符必须是英文字母，其余的可以是字母、数字或下划线，在标识符中，美元符$总是被编译程序略去。在标识符中应用美元符$是为了改善可读性，但不允许用美元符$作为标识符的第一个字符。下面是正确的标识符：

PHI, PSI, GAMMA, THETA, INPUT$COUNT

OUTPUT$COUNT, STINS37, ABC, DASTEST, L1 2 3 4等。

有许多标识符在形式上是合法的，如

ADDRESS, CASE, DATA, DECLARE 等，但它们在意义上预先已有意义，实际上是PL/M-86语言的一部分，故称它们为保留字，保留字不能用作标识符（保留字见附录C）

1.3 常数

常数分数值常数和字符串常数两大类，常数的值在程序运行时是不会改变的。

1.3.1 数值常数可分纯数常数和浮点常数

纯数常数可用二进制、八进制、十进制或十六进制表示，而浮点常数只能用十进制表

示，纯数常数又可分为BYTE（字节）型，WORD（字型）、DWORD（双字型）、INTEGER（整型）、SELECTOR(POINTER的基址)和POINTER（指针型），而浮点常数则是带小数的十进制数，它是REAL（实型），纯数常数的取值范围：

BYTE型　　0～255　　占一字节存储单元，

WORD型　　0～65535　　占相邻两个字节存储单元，

DWORD型　　0～4294967295　　占相邻四个字节存储单元，

INTEGER型　　－32768～＋32767　　占相邻两个字节存储单元，

SELECTOR型　占相邻两个字节存储单元，

POINTER型　　占相邻四个字节存储单元。

如5，36Q，0111B，25FE，15D，65Q，65535，42578855等都是纯数常数的正确表示，其中B——二进制，Q和O——八进制，H——十六进制，D或缺省——十进制。

浮点常数是由十进制小数点所标识，如18.3，275.56，89.，0.178，12.0等都是正确的浮点数表示。浮点数还可以带指数部分，如10.0E—01，84.56E—03，465.55E＋02。0.00055E＋10等都是正确的表示，如1E—07，53E＋02则是错误的，因尾数部分不带小数点。

浮点数为REAL型即实型数，占相邻4个字节，其取数范围为1.17×10^{-38}～3.37×10^{38}（绝对值）。

为了改善可读性，二进制数字符之间往往插入美元符$，如011$1001$0101B此数与0111001010 1B完全等值。

1.3.2　字符串

字符串是用单引号括起来的可打印的ASCII字符的集合，编译程序将字符串表示为ASCII代码存于存储器中，一个字符翻译为一字节，如'D'，'ABC123'，'WHO ARE YOU'等均为字符串，字符串常数的最大长度受PLM 86编译程序的限制，一般不超过255。

1.3.3　逻辑值

BYTE型0FFH代表"真"，BYTE型00H代表"假"。

1.4　变量和运算

变量是其值在运算过程中可以改变的量，它不写成具体数的形式，而是用一个名字即标识符来表示，这个名字便称为变量名。既然变量包括标量（纯量）、数组和结构，故变量有标量变量（简称标量），它总是表示一个具有单一数值的变量。下标变量，它用来表示数组的某个元素的，下标变量是带有下标的。结构元素变量，它用来表示结构中某一元素。

变量的类型和常数一样，可分为BYTE（字节型），WORD（字型），INTEGER(整型），SELECTOR（POINTER的基址）、POINTER（指针型）和REAL（实型）。

PL/M-86语言可进行算术运算、关系运算和逻辑运算，其中算术运算分三种不同类型，即无符号、有符号和浮点运算。对于任何计算所用的运算类型取决于所涉及的变量和数据类型。

1.4.1　BYTE型、WORD型和DWORD型

BYTE型变量的取值是一个8位二进制数，其取值的范围为0～255，它占据8086存储器的一个字节。

WORD型变量的取值是一个16位二进制数，其取值范围为0～65535，它占据存储器

的相邻两个字节。

DWORD 型变量的取值是一个32位二进制数，其取值范围为0～4294967295，它占存储器的相邻四个字节。

对于BYTE型、WORD型和DWORD型变量的运算用无符号整数运算，所有 PL/M-86 运算符都可以进行运算。算术运算和逻辑运算产生 BYTE 型、WORD 型或 DWORD 型结果，这依赖于运算和运算对象，关系运算的结果总是产生BYTE型“真”或“假”。

用无符号运算，若从小值减去大值，结果是两个值差值的绝对值补码，假如BYTE型值 0 (0000$0000B)减去BYTE型值 1 (0000$0001B)，结果是BYTE型值 255(1111$1111B)。

除法运算的结果总是截断（向下舍入）到一个纯数，例如WORD 型值 7(0000$0000$0000$0111B)被 BYTE 型 2 (0000$0010B)除，结果是WORD 型 3 (0000$0000$0000$0000$0011B)。

1.4.2 INTEGER（整型）变量

INTEGER (整型) 变量取值是有符号的整数，其取值范围为－32768～＋32767，在机器内部一个INTEGER值用补码表示，它占存储器两个相邻字节。

对 INTEGER 变量的运算，用有符号的整数算术运算产生整数结果，关系运算将产生 BYTE 型结果，即“真”或“假”。

对INTEGER型只能进行算术运算和关系运算，不允许进行逻辑运算。

1.4.3 REAL（实型）变量

一个 REAL 型变量的取值是有符号的浮点数，其取值范围通常为1.17×10^{-38}～3.37×10^{38}（绝对性）。它占存储器四个相邻字节。

REAL 型算术运算将产生REAL型结果，关系运算将产生 BYTE 型结果“真”或“假”。

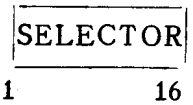
REAL型不允许用MOD运算和逻辑运算。

1.4.4 SELECTOR（POINTER的基址）和POINTER（指针型）变量

POINTER变量的取值是一个8086存储单元的地址值，通常是占存储器四个相邻字节，但有时也可能是只占两个字节。POINTER变量的重要应用是作为有基变量的基（将在以后说明）。

POINTER 型只能进行关系运算，将产生BYTE型“真”或“假”的结果，它不允许进行算术和逻辑运算。

SELECTOR型变量是表示POINTER型的基地址，它占POINTER型的位如下：



SELECTOR型变量可用INITIAL或 DATA赋初值，也可用赋值语句赋初值。另外，还可用内部函数SELECTOR$OF从POINTER型数值中求出 SELECTOR 型的值（将在以后介绍）

1.5 数组和下标变量

数组是取一组常数量，它借助于下标来区别各个数组元素，数组的下标是从零开始的，例如GAMMA(99)，这表示数组中的第 100个元素，而 GAMMA(99) 则是下标变量，因为它

试读结束：需要全本请在线购买： www.ertongbook.com

带有下标。下标可以是另一个变量，或者是能产生字节、字或整型值的 PL/M-86 表达式。例如：VECTOR(LAST(5)+5) 是完全正确的，下标中 LAST(5) 又是另一个数组的下标变量。

数组是用一个“维数区分符”来说明的。“维数区分符”是一个括在括弧中的纯数常数，它表示数组元素的个数。

1.6 简单说明语句

在 PL/M-86 程序中所用的变量在它们被引用之前必须要用说明语句加以说明，但只能说明一次，重复说明是非法的。说明语句可出现在程序块的开头或在外层程序块中。

最简单的说明仅仅说明变量类型，一个变量的说明语句是由保留字 DECLARE 开始，每个变量的标识符或用括弧括起来的一组标识符后面跟七个保留字(BYTE, WORD, DWORD, INTEGER, REAL, POINTER, SELECTOR)之一，最后用分号(;)结束，这便说明变量的类型了。例如：

```
DECLARE    VAL    BYTE;
  说明变量VAL是BYTE型。
DECLARE    (OLD, NEW) BYTE;
  说明变量OLD和NEW都是字节型的。
DECLARE    VALUE    WORD;
  说明变量VALUE是字型的。
DECLARE    DWW    DWORD;
  说明变量DWW是双字型的。
DECLARE    INT    INTEGER;
  说明变量INT是整型的。
DECLARE    (THETA,  GAMMA, PSI)  REAL;
  说明变量THETA, GAMMA, PSI 都是实型的。
DECLARE    PTR    POINTER;
  说明变量PTR是指针型的。
DECLARE  A(100)    BYTE;
  说明数组A 为 100 个元素，它们是字节型的。
DECLARE  (WIDTH, LENGTH, HEIGHT)  (100) REAL;
  它等价于下列说明
DECLARE  WIDTH(100)  REAL;
DECLARE  LENGTH(100)  REAL;
DECLARE  HEIGHT(100)  REAL;
```

说明数组 WIDTH, LENGTH, HEIGHT都是100个元素，都是实型的。

1.7 有关结构问题

1.7.1 结构

结构是用一个结构名(标识符)来引用一组结构成员，这些结构成员可以有不同类型。

结构的每个成员有一个成员标识符。结构成员的说明必须分别进行，即使类型相同，也应如此。例如：

```
DECLARE AIRPLANE STRUCTURE (SPEED REAL,
    ALTITUDE REAL);
```

在上述说明语句中，STRUCTURE 是说明结构的关键字，AIRPLANE 是结构名，SPEED和 ALTITUDE 是成员名。这样可用 AIRPLANE. SPEED 来表示结构的第一个成员，AIRPLANE · ALTITUDE 来表示结构的第二个成员。

如果将说明语句写成

```
DECLARE AIRPLANE STRUCTURE (SPEED,ALTITUDE) REAL;
```

这是错误的。

另外应该注意的结构成员不能是有基的，也不能有任何属性。

1.7.2 结构数组

结构数组，其结构名实际上是数组名，而成员名是标量变量名，结构数组说明如下：

```
DECLARE AIRPLANE(10) STRUCTURE (SPEED REAL,
                ALTITUDE REAL);
```

此说明语句说明了与结构名 AIRPLANE 相连系的 10 个结构，下标从 0 到 9。每个结构有两个成员，因此分配了 20 个实型变量的存储单元。每个结构数组成员都可以引用。例如要引用第 5 个结构成员ALTITUDE，便可写

```
AIRPLANE(4) · ALTITUDE
```

1.7.3 结构内数组

其结构本身仍然是一个结构，而其成员中郤包含了数组，其说明语句如下：

```
DECLARE A STRUCTURE (B(10) REAL, C(10)
                REAL, BI BYTE);
```

这个结构A包含了两个10个实型元素的数组，一个字节型变量，若想引用结构数组A·C的第 5 个元素，则可写A · C(4)。

1.7.4 结构数组内数组

结构数组内数组，其结构本身是一个数组，结构成员中又包含数组，如下说明语句：

```
DECLARE A(10) STRUCTURE (B(10) REAL);
```

标识符A可引用10个结构的数组，每个结构包含有一个10个实型元素的数组，这相当于BASIC、FORTRAN 中的二维数组。这个结构数组内数组可容纳 10×10 的实型阵矩。若想引用第 5 个结构中数组B的第 5 个元素，则可写

```
A(4) · B(4)
```

注意，每个 B数组的元素是相邻存储的，B 数组本身又是A数组的元素，故也是相邻存储的。又如：

```
DECLARE A(10) STRUCTURE (B(10) REAL, C(10) REAL,
        BI BYTE);
```

此说明语句说明了结构数组A中包含有两个有10个实型元素的数组成员和一个字节型成员。

1.8 对数组和结构的引用

在上一节中，我们已例举了不少数组和结构的引用，引用可以是“完全限定的”，“部分限定的”和“非限定的”。

1.8.1 完全限定的引用

例如有下列说明语句

```
DECLARE  AD(100)  BYTE;
DECLARE  ABC  STRUCTURE(D1(100)  BYTE, D2(10)  WORD);
DECLARE  A(10)  STRUCTURE  (B(10)  BYTE, D(10)  BYTE);
```

则AD(79)，ABC · D_2(4)，A(5) · B(5)等都是“完全限定”的引用，它们都代表一个标量。

1.8.2 部分限定和非限定的引用

首先介绍一下@算符，@算符有两个作用：

(1) 地址引用，其形式为@标量变量名或@数组名或@结构名，地址引用的是该标量或数组、结构的第一个元素在运行时的实际地址。可见，地址引用是用@算符构成的。它取的是一个 POINTER 型值（即存储单元的实际地址）。地址引用的重要作用是为 POINTER 型变量赋值。如PTR=@NUM;

(2) 将常数表中常数分配内存单元，例如

@('NEXT VALUE')，即将字符串'NEXT VALUE'分配内存单元。

部分限定和非限定的引用仅在位置引用及内部过程LENGTH、LAST 和 SIZE 中允许使用。

例如有下列说明语句

```
DECLARE  ITEMS(100)  BYTE;
DECLARE  NODE(25)  STRUCTURE(SUBLIST(100)  BYTE,
            ERANK  BYTE);
DECTARE  RECORD  STRUCTURE (KEY BYTE, INFO  WORD);
```

根据以上说明，@NODE(15)和@NODE(12)·SUBLIST等都是部分限定引用，@NODE(15)是结构NODE(15)的第一个字节的地址，而NODE(15)本身是结构数组 NODE 的一个元素。同样，@NODE(12) · SUBLIST 是结构数组中数组 NODE(12) · SUBLIST 的第一字节的地址，而 NODE(12) · SUBLIST 本身是结构 NODE(12)的一个成员，而NODE(12)又是NODE的一个元素。应注意@NODE· SUBLIST是不允许的。

非限定的引用是引用一个结构和数组的标识符，即结构名和数组名，而不带成员标识符和下标，如@ ITEMS 和 @ RECORD 都是非限定的引用，它是引用整个结构和数组，@ITEMS是整个数组第一字节的地址，而@RECORD是整个结构的第一个字节的地址。

1.9 关于有基变量问题

1.9.1 有基变量

所谓有基变量是由另一个变量所指向的量。这另一个变量称为该变量的“基”，有基变量不是由编译程序来分配存储单元的，在程序运行的不同时刻，它可能实际上处于存储器中

不同位置，因为它的基可以由程序改变。一个有基变量的说明方法，首先说明它的基必须是指针型或字型，然后说明有基变量本身。

例 DECLARE ITEM$PTR POINTER;
DECLARE ITEM BASED ITEM$PIR BYTE;

其中BASED是有基变量说明的关键字，有了这样的说明，对ITEM的引用实际上是引用ITEM$PTR当前值所指向的字节值。若

ITEM$PTR=34AH;
ITEM=77H;

这将加载字节值77H到地址为34AH的存储单元中。

对有基变量的基有下列限制：

(1) 基必须是指针型或字型。字型是为了和PL/M-80程序兼容才允许这种基，字型的基并不总是给出正确的结果，故基最好用指针型。

(2) 基不能有下标，即不能是一个数组元素。

(3) 基本身不能是一个有基变量。

例 DECLARE (AGE$PTR, INCOME$PTR, RATING$PTR,
CATEGORY$PTR) POINTER;
DECLARE AGE BASED AGE$PTR BYTE;
DECLARE (INCOME BASED INCOME$PTR, RATING BASED
RATING$PTR) WORD;
DECLARE (CATEGORY BASED CATEGORY$PTR) (1) WORD;

例中第一个说明语句说明AGE$PTR, INCOME$PTR, RATING$PTR和CATEGORY$PTR为指针型变量。

第二句说明AGE是字节型有基变量，它的基为AGE$PTR。

第三句说明INCOME, RATING为字型有基变量，它们的基分别为INCOME$PTR和RATING$PTR。

第四句说明CATEGORY为一个字型的有基数组，它的基为CATEGORY$PTR。此时，CATEGORY$PTR的值是CATEGORY第一个元素的地址，其它元素跟在后面相邻存放。其中维数区分符1并不表示该有基数组只有一个元素，有基数组的维数区分符的大小可以是任意的，只要是非0的正整数，因为有基数组实际上不用给它分配存储单元，所以维数区分符具体为何值是无关紧要的。

在符号CATEGORY BASED CATEGORY$PTR外面的括弧是可有可无的，加上只是改善语句的可读性。

1.9.2 地址引用和有基变量应用举例

地址引用的重要用途是提供基的值。因此@运算符连同有基变量的概念，使PL/M-86具有处理指针的能力。

例如有下列三个不同的变量

NORTH$ERROR, EAST$ERROR及HEIGHT$ERROR，我们想用一个标识符ERROR，能在不同时刻分别引用这些变量。为此可写下列几个语句：

```
DECLARE (NORTH$ERROR, EAST$ERROR,
HEIGHT$ERROR) REAL;
DECLARE ERROR$PTR POINTER;
DECLARE ERROR BASED ERROR$PTR REAL;
.....
ERROR$PTR=@NORTH$ERROR;
```

这时，ERROR$PTR的值是NORTH$ERROR的地址，对ERROR的引用实际上是引用NORTH$ERROR。

若再写

```
ERROR$PTR=@EAST$ERROR;
```

此时，引用ERROR实际上是引用EAST$ERROR。

这种技术对于处理复杂的数据结构及传送地址作为过程的参数是非常有用的。

1.10 存储的相邻性

下列情况是相邻存储的

(1) 一个数组的元素是相邻存储的，第0个元素在最低的地址，最后一个元素在高地址（对有基数组不分配存储，但它的元素可以认为是相邻存储的）。

(2) 结构的成员是按他们说明的顺序相邻存储的（对一个有基结构不分配存储单元，但它的成员可以认为是相邻存储的）。

(3) 在同类型变量说明时，在括弧中所列变量是按说明的次序相邻存储的（若有基变量出现在一个括弧的变量表中，则在分配存储时略去它）。

1.11 PL/M-86表达式

表达式有算术表达式，关系表达式和布尔表达式三种。

1.11.1 PL/M-86 算术表达式

算术表达式是用算术运算符+、－、*、/和MOD将运算对象（变量、常数等）连接起来有意义的式子。

例 $x+y$

$x*y-z$

$(A+B+C)*D-x/y$

$3.1415926*R*R$

A MOD B 等都是正确的算术表达式。

算术表达式运算规则见表达式规则一览表。

MOD（模）运算与除法运算基本相同，但它所得的结果不是商，而是余数，MOD运算只能用于BYTE、WORD和INTEGER运算数，而不能用于实型运算数。

MOD运算举例：设INTEGER变量 A和B，它们分别为35和16，则A MOD B其结果为3，－A MOD B为－3。

表达式规则表只适用于如表达式A＋3＊B的情况。若表达式为3－5＋A，则凭表达式规则表就不能完全说明问题，因为在此表达式中有两个纯数常数要进行减运算，即3－5，我们称3－5为常数表达式。其运算的结果数取决于变量A的类型，如果A的类型为BYTE型、

WORD型或DWORD型则3－5在“无符号上下文”中，要用无符号运算去计算3－5，得BYTE型结果254，然后用无符号运算再去加A，如果A为INTEGER型，则3－5在“有符号的上下文”中，要用有符号运算去计算3－5，其结果为INTEGER型数－2，然后用有符号运算再去加A，若A为REAL型或POINTER型，则表达式是非法的。

又例如表达式3－5＋500＋A，如A为BYTE型、WORD型或DWORD型，则常数表达式3－5＋500在“无符号的上下文”中，3－5当作BYTE型对待，500当作WORD型对待，3－5运算结果得245，然后将245扩展成WORD型或DWORD型，再与500相加得WORD型或DWORD型结果754。此时表达式已成为754＋A。如A为INTEGER型，则3－5＋500在“有符号的上下文”中，因此将这三个常数都当作INTEGER型对待。这时用有符号运算3－5得INTEGER型－2，再与500相加，得INTEGER型498，此时表达式已成为498＋A。

1.11.2 关系表达式

关系表达式是由两个同类型的算术表达式或两个字符串用一个关系运算符连接起来的式子。关系运算符有< <= = >= > <>，例如 5<3，x<=0，x>y，6<=4，CHR='A'，3＋5＊A>4－B等均为关系表达式。

关系运算符总是二目运算符，它作用在两个运算对象上产生一个字节型的结果。如果两个运算对象是同类型的，用无符号运算来比较两个字节型的值或两个字型的值，用有符号算术运算来比较两个整型值或两个实型值。指针型值按照8086地址顺序来进行比较。除允许字节型和字型可进行混合运算外，其它类型混合运算是非法的。

1.11.3 布尔表达式

布尔表达式是用逻辑运算符NOT AND OR XOR将BYTE型、WORD型、DWORD型值或纯数常数连接起来或者将两个关系式连接起来的式子。

表达式规则一览表

变量类型	算术运算种类	运算数类型	算术运算	结果	备注
DWORD型、WORD型和BYTE型	无符号运算	BYTE与WORD	＋或－或＊或/或MOD	BYTE WORD	范围：0至255 范围：0至65535
		BYTE与WORD变为WORD与WORD	任意	WORD	字节型运算数用高8位是0的方法扩展成字型值
		BYTE与DWORD变为DWORD与DWORD	任意	DWORD	字节型运算数用高24位是0的方法扩展成双字型值
		WORD与DWORD变为DWORD与DWORD	任意	DWORD	字型运算数用高16位是0的方法扩展成双字型值
		BYTE与小于256的纯数常数	＋或－或＊或/或MOD	BYTE WORD	常数作为BYTE型 常数作为WORD型
		BYTE或WORD与大于255，小于65535的纯数常数	任意	WORD	常数作为WORD型
		BYTE或WORD与大于65535的纯数常数	任意	DWORD	常数作为DWORD型
		DWORD与小于4294967295的纯数常数	任意	DWORD	常数作为DWORD型

试读结束：需要全本请在线购买： www.ertongbook.com