微型计算机实用大

第11篇 计算机语言

11.1 汇编语言

11.1.1 计算机语言概述

计算机语言可分为三类：机器语言，汇编语言和通用语言。前两种语言是面向计算机的，一般称为低级语言；后一种语言是面向程序设计人员的，一般称为高级语言。

机器语言 计算机的电子部件，都是由电子线路按着某种逻辑需要组合在一起的，这些电子线路无外乎是各种门电路，触发器，寄存器，译码器，驱动器等等，这些部件只有两种状态：高电平或低电平，也就是说它们只能识别高低电平，若我们将高电平规定为逻辑"1"，低电平规定为逻辑"0"，那么它们只能认识"1"和"0"。如果我们要计算机做什么工作，就必须用"1"和"0"的序列(若干个"1"和"0"的组合)告诉计算机，计算机则按"1"和"0"信号，实现相应的逻辑功能，这些能使计算机实现某种逻辑功能的"1"和"0"的组合，称为机器指令代码，每台计算机都有若干个不同的"1"和"0"的序列来使计算机完成不同的工作，即若干条机器指令代码，这些机器指令代码的集合称为机器语言，亦称指令系统。不同的计算机，它的机器指令代码的种类的多少不同，每条指令代码的组合形式也不相同。

用机器指令代码编制的解题步骤，即用机器代码组成的程序称为机器语言程序。机器语言程序中，计算机完成的操作，内存单地址，寄存器等都是用二进制代码表示。

用机器语言代码编制程序是非常困难的，如果若干个代码中有一个"0"写成了"1"，或"1"写成了"0"，那么这个程序就不能实现预定功能。所以机器语言有下述缺点：

①机器指令难记；

②编制程序困难，极易出错，出错后难于查出和纠正；

③机器语言编制的程序通用性差。

但也有以下优点：

①计算机能直接认识、执行；

②算法刻画细致；

③程序紧凑，占用内存少，执行速度高；

④能充分利用、发挥计算机硬件的功能。

汇编语言 计算机诞生初期，人们都是用机器语言编制程序，随着时间的推移，需要计算机解决的问题越来越多，越来越复杂，机器语言不能满足人们的要求，于是人们就采用容易记忆的符号，如英文单词或英文词组的缩写(称为助记符号)来代替机器指令，即用助记符号表示机器指令的操作，内存单元地址，寄存器的名字等，因而就产生了符号语言。如用ADD代表加法操作，SUB代表减法操作，MUL代表乘法操作

等等。但用符号语言编制的程序计算机不能直接认识和执行，必须将其译为机器语言程序，然后计算机才能认识执行。

将符号语言翻译为机器语言程序，可由人工完成，也可由机器完成。人工翻译极易出错，现在一般用机器完成。让计算机完成翻译工作，就要对符号语言所使用的字符，组成符号的规则，语句的书写格式做一些必要的规定，这就是所谓的词法和语法，而且还需加一些必要的说明，如数据进制，表示方法，存储形式，存储位置，指令代码存放位置，保留多少工作单元，翻译到哪里结束，模块间如何联系等等，这些用于说明的命令，称为汇编命令或伪指令。

机器指令的助记符号，伪指令助记符号及使用规则(词法，语法)就构成了汇编语言。用汇编语言编写的程序必须经过翻译，变换为机器语言程序，完成这种翻译工作的软件称为汇编程序，这种翻译过程称为汇编过程，或简称汇编。从这个意义上来讲，用汇编语言的语句编制的程序称为源程序，而翻译后产生的机器语言程序称为目标程序或目标代码。

在不同机器上，执行汇编程序的过程也略有差异。在IBM PC机上执行汇编程序的过程如下：

①用编辑软件将汇编语言源程序送入计算机，并以.ASM为文件的扩展名，存于磁盘；

②启动汇编程序，将汇编语言源程序读入内存，译为机器语言程序，并以.OBJ为文件的扩展名存于磁盘；

③启动连接程序将目标程序读入内存，形成可执行程序，并以.EXE为文件的扩展名存于磁盘；

④用纠错(动态调试)程序进行调试，若出错，重复①~④，直至达到设计要求；

⑤对调试好的程序(.EXE文件)，在任何情况下，在系统提示符下，直接键入文件基本名，就可执行，从而实现预定功能。

用汇编语言编写程序，即使是完成一个简单的工作也需大量的语句，故给使用者带来不便，为节省编程和输入程序的工作量，汇编程序又提供了一种让人们利用一个符号(字符串或称为名字)，来代替多次重复出现的一组指令语句(称为宏定义)，在对其汇编时，再用相应的那组指令语句的功能(称为宏扩展)，这个用来代替一组指令的符号称为宏指令。具有处理宏定义，宏扩展功能的汇编程序称为宏汇编程序。这种具有宏功能的汇编语言，称为宏汇编语言。

宏汇编语言程序由机器指令语句、伪指令语句和宏指令语句组成。每条机器指令语句与唯一的一条机器指令代码对应，而宏指令语句对应于一组机器指令语句。伪指令语句没有机器指令与之对应。

汇编语言与机器语言比较，有下述优缺点。

优点：

①汇编语言语句容易记忆；

②汇编语言程序容易编制，调试，修改；

③算法刻画细致；

④一旦转换为机器语言程序后，占用内存少，运行速度高；

⑤能充分利用、发挥计算机硬件功能。

缺点：

①计算机硬件不能直接认识，执行；

②通用性差。

通用语言 通用语言是对计算机操作步骤进行描述的一整套标记符号、表达格式、结构及其使用的语法规则。

尽管汇编语言的出现简化了程序设计，它与机器语言相比较，具有明显的优点，但用汇编语言编制较复杂的运算和处理程序，是一件非常麻烦的事，而且要求编程人员对计算机硬件有比较清楚而具体的了解。通用语言对编程人员的计算机硬件知识几乎没有要求，编程时只要考虑解决问题本身的逻辑，用相应的通用语言语句进行描述就可以了。所以通用语言对于使用者来说是一种高级语言。

目前世界上有数百种通用语言，常用的有：FORTRAN，Pascal，C，BASIC，COBOL，LISP，PROLOG，Ada 等等。各种语言都有它自己的特点及适用范围。

用通用语言编制的程序，也需经过翻译，变换为目标程序，然后再经过连接形成可执行程序。将通用语言程序转换为目标程序的软件有编译程序和解释程序。每种通用语言有它自己的编译程序或解释程序。

通用语言具有如下优缺点：

优点：

①语句容易记忆；

②编程、调试、修改容易；

③容易表达数据的结构和解决问题的算法；

④通用性强。

缺点：

①计算机硬件不能直接认识、执行；

②通用语言程序转换为执行程序后，占用内存多，运行速度低；

③不能充分利用、发挥计算机硬件的功能。

11.1.2 汇编命令(伪指令)

汇编命令(伪指令) 是用来告诉汇编程序如何对汇编语言源程序进行汇编的命令。汇编命令只为汇编程序所识别，在目标程序中没有与之对应的机器指令代码，故亦称为伪指令。

不同的机器(尽管是CPU 型号相同)所具有的汇编程序的功能也不尽相同，它的功能的强弱主要取决于它的汇编命令的种类和多少，汇编命令的种类越多，编程人员编写汇编语言程序时越方便。

符号定义命令 符号定义命令用来对标号(名字或变量)进行定义，即给一个名字赋值。

符号定义命令有两类：一类是对标号定义后，不得再对同一个标号定义的命令，另一类是，对标号定义后，还可以重新定义。

一般汇编程序的符号定义命令使用：EQU、=、DL(DEFL)、SET 等。

数据说明命令 用汇编语言编写程序时，用到的存储单元的数据、数据的表示方法、存储位置、存储格式等等，都由编程人员安排。

数据说明命令是编程人员安排数据的命令，即将程序所用到的数据存储在内存中，如有必要，则还可以定义符号地址。

一般汇编程序的数据说明命令使用：DB(DEFB)、DW(DEFW)、DD、DC、DQ、DT 等。

保留内存单元命令 汇编语言程序在进行运算和处理过程中，需要一些存储单元保存中间结果和最终结果，在编程时，就需预先留出一些存储单元，称为保留内存单元，这时，也需要用汇编命令说明保留的内存单元数目。

一般汇编程序保留内存单完命令使用：DS、DM 等。

代码定位命令 汇编语言程序被转换为可执行程序后，在运行时放入内存，那么，放入内存中的位置，各段代码之间的位置也需要在编写程序时进行说明。

如果内存可以分成若干段，或者程序代码可以分为指令代码、数据代码等等，都需在源程序中进行说明，这些说明都要用到代码定位命令。

一般汇编程序的代码定位命令使用：ORG，REL，DATA，ABS，COM，SEGMENT 等等。

包含说明命令 当要解决的问题所需程序较大时，为了编辑程序方便，把源程序分成若干段进行编辑，以不同的文件名存在磁盘上，而在汇编时，把它们做为一个源文件进行汇编，这时，在源程序中需要说明将

那一段源程序放在什么位置，那么，就在相应的位置上写上一个包含说明命令，及要插入此位置的源程序段的文件名，汇编程序汇编此源程序时，就把包含命令指出的源程序，从磁盘上读入内存并翻译成目标程序。

一般汇编程序使用：INCLUDE。

公共标号说明命令 当要解决问题的程序很大，需多个人协同工作时，则把程序分解成若干个逻辑功能模块，由多个人编制，每人完成其中的一块或几块，每人编制程序名，各自独立的汇编，生成目标程序，然后通过连接程序将这些目标程序连接在一起，形成一个完整的执行程序。各个模块之间可能要有联系，模块之间相互引用数据或相互调用子程序，这时每个模块中允许其它模块引用的标号，存储单元的符号地址、子程序名等，必须进行说明，没有说明的标号或名字，其它模块不得引用，用于说明可被其它模块引用的标号的说明命令称公共标号说明命令。

一般汇编程序的公共标号说明命令使用：PUBLIC，ENTRY。

外部标号说明命令 一个模块要引用其它模块的标号或名字，称为外部标号，在源程序的开始也必须加以说明，未说明的外部标号不能引用。这个命令与其它模块中的公共标号说明命令是相对应的。

一般汇编程序的外部标号说明命令使用：EXTRN。

打印格式说明命令 汇编程序将源程序汇编后，除产生目标程序(.OBJ 文件)外，还生成一个程序清单文件(.LST 文件)，清单文件中含有目标程序与源程序(语句对应)，以及程序中所使用的各个标号及其值的列表。如有必要，可以用打印机打印出来。打印的格式，如哪些行打印，哪些行不打印，打印的标题是什么，在什么位置分页等等，都可以在汇编语言源程序中用打印格式说明命令进行说明。

一般汇编程序的打印格式说明命令使用：LIST，TITLE，PAGE 等等。

条件汇编说明命令 条件汇编命令允许汇编程序根据条件决定某段程序是否参加汇编，编制程序时，可考虑所有的情况，在汇编程序时，根据某些具体情况，改变某些条件的值，则可以汇编产生不同的目标程序，以适应不同的应用场合。

一般汇编程序的条件汇编说明命令使用：IF，ENDIF 等。

宏定义说明命令 宏定义说明命令用于定义一条宏指令。一条宏指令是一组完成某种功能的指令，即一段程序，这段程序在编制源程序时要多次重复地使用，为了减少书写次数，把它定义为一个名字，称为宏定义，这个名字称为宏名字，这段程序称为宏体，一旦作了宏定义后，宏名字就可在源程序中像指令一样使用，也称宏名字为宏指令。编制程序用到某种功能程序段时，写上相应的宏指令，称为宏调用；汇编程序对源程序汇编时，遇到宏名字，就用相应的程序段(宏体)来代替它，称为宏扩展。具有这种宏处理功能的汇编程序称为宏汇编程序。宏定义命令在定义宏指令时，还允许带形式参数，宏调用时，写上实际参数，在宏扩展过程中，则用实际参数代替形式参数。

一般汇编程序的宏定义说明命令使用：MACRO，MEND 等。

源程序结束说明命令 源程序结束命令用以说明源程序模块的结束和说明程序的启动地址。汇编程序遇到此命令时，即结束汇编，所以这条命令一般位于源程序模块的最后，只有主程序模块的源程序结束命令才指出程序的启动地址，其它源程序模块(非主模块)不能用此命令说明程序的启动地址。

一般汇编程序的源程序结束说明命令使用：END

11.1.3 汇编方法与程序的装入

汇编程序是将汇编语言源程序转换为目标程序的软件。

宏汇编程序是具有宏处理功能的汇编程序。

汇编程序生成的目标程序有的可以直接运行，有的需要经过连接和装入后才能直接运行，这决定于汇编程序的类型，主要有下述几种情况：

1. 汇编后立即装入内存运行

这种汇编程序将源程序按绝对地址格式产生目标程序，并将其立即装入内存执行，这种汇编程序的优点是实现起来简单，容易理解。缺点是：

(1)缩小了用户程序使用的存储区(因为汇编程序仍驻留在内存中)；

(2)一次只能汇编一个源程序模块，若一个程序较大，不能由多个人编写分别调试；

(3)程序每执行一次，就要汇编一次，浪费时间。

2. 汇编后生成目标程序，由装入程序装入并运行

这种汇编程序先将源程序和各个模块分别汇编成绝对地址的目标程序，存入磁盘，然后由装入程序将它们装入内存的相应存储区，并执行。优点是：

(1)供用户使用的存储区大，尽管装入程序本身仍在内存中，但它占用内存较汇编程序少得多，所以留给用户的存储容量较大；

(2)可将一个大程序分成多个模块，分别汇编调试；

(3)可连接不同的目标程序模块，甚至不同语言程序的目标程序模块。

缺点是：内存的管理是在用户编制程序时安排，若管理失误，会使各模块占用内存重迭，或使程序模块在内存之间的间隙大，浪费内存空间。

3. 直接连接和装入

汇编程序将源程序汇编为相对地址的浮动程序，各个目标模块的地址均从零开始，然后由连接程序将这些模块连接成一个完整的绝对地址的可运行的目标程序，需要时装入内存并运行。优点是：

(1)大型程序可分成若干模块，由多个人分工编制、汇编、调试；

(2)可连接几个目标模块，甚至是不同语言程序的目标程序模块；

(3)连接在一起的模块紧凑，模块间没有间隙，内存利用率高。

缺点是：

(1)连接和装入程序较复杂；

(2)程序的各个模块全部连接在一起，运行时一起调入内存，占用内存较大。

4. 动态连接和装入

汇编程序将源程序汇编为相对地址的浮动程序，但在程序运行时，一个模块只有在被调用的时候才被装入内存并进行连接。这样做的优点是：

(1)程序可以分成若干个模块，分别编制调试；

(2)由于各模块不是同时调入内存，所以扩大了程序使用的内存空间；

(3)可以连接几个模块，甚至是不同语言程序的目标程序模块；

(4)在多用户系统中，一个模块可由多个用户共享。

缺点是：

(1)连接和装入程序较复杂；

(2)在程序运行中，某个模块被多次使用，就需多次装入和多次连接，浪费时间，效率低。

目前在8位微机系统中多数为第二种汇编程序，有些8位微机是第三种汇编程序。

IBM PC宏汇编程序属于第三种汇编程序。汇编程序产生的目标程序(.OBJ文件)经过连接程序进行连接，形成可执行程序(.EXE文件)。

11.1.4 Z-80汇编语言

由于使用Z-80 CPU的微型机的型号很多，而各种机器使用的汇编程序不尽相同，各种汇编程序允许用户的汇编命令、语句格式、数据表示方法等也各有差异，但大同小异。本节仅以CROMENCO微型机的宏汇编程序为例，介绍Z-80汇编语言的语法规则、数据表示方法及汇编命令(伪指令)。但不介绍Z-80的指令助记符及书写形式，这部分内容详见有关条目。

字符集

①任何可打印的ASCII码字符(包括空格符)；

②三个控制字符：CTRL-I，CTRL-N，CR

CTRL-I：制表符

CTRL-N：把一行字符在打印机上展开的控制符

CR：回车符

名字

①保留字：汇编程序已赋于意义的字符串，如指令的助记符、伪指令助记符、寄存器名字和由字符串组成的各种运算符，标志位助记符等；

②用户自定义名字：用户为了引用数据、数据地址或指令地址方便，按一定的规则起的名字，有时亦称变量或标号。宏定义伪指令定义的名字，称为宏名字或宏指令；

③命名规则：

组成名字的字符必须是：A～Z，a～z，0～9，·和$；

名字长度≤80，但前6个字符相同者，汇编程序认为是同一名字；

标号的第一个字符不能是0～9的数字；

保留字不能用作自定义名字。

语句格式 汇编程序由伪指令语句、机器指令语句和宏指令语句组成。每种语句可分为4个域，其格式如下：

[标号] <操作符> <操作数> [;注释]

1.标号域

标号是用户自定义名字。用以表示操作数、操作数或工作区的地址、机器指令地址或宏名字。

标号域是可选项，可有可无，视需要而定。标号从一行的第一列开始书写，若不从第一列开始，则要以“:”结束。

标号的定义：标号在标号域中出现称为定义标号。

标号的引用：标号在操作符或操作数域中出现称为引用标号。

重复定义标号：同一标号(名字)在标号域中出现两次或两次以上，此标号则为重复定义标号。

未定义标号：未在标号域中出现，而在操作符域或操作数域中引用的标号，称为未定义标号。

重复定义标号和引用未定义标号都是一种语法错误，汇编程序在汇编过程中进行检查，若发现则显示出错误信息。

2.操作符域

表示语句所要完成的操作，它可以是指令的助记符、伪指令助记符或宏指令。

操作符域可从一行的除第一列以外的任何位置开始。操作符域前可以有标号，若标号未以“:”结束，则标号与操作符之间至少有一个作为分隔符的空格或制表符。

3. 操作数域

若为指令语句，则为机器指令要操作的数据或数据的地址；若为伪指令语句，则为伪指令要初始化内存单元的数据、保留存储单元的个数或所需的参数；若为宏指令语句，则为实际参数。

对一个语句来说，因操作符不同，所需的操作数个数或参数的个数也不同。

在指令语句中，最多为两个操作数，二者之间用逗号分隔，右侧的操作数为源操作数，左侧的为目标操作数。有的指令语句只有一个操作数，有的指令无操作数。

操作数通常由寄存器名字、常数、标号(名字)或表达式组成。

4. 注释域

用以说明伪指令语句、指令语句或程序段在程序中的作用，便于程序的阅读、交流及维护。注释域的格式较自由，以“;”开始的可见ASCII码串，注释以回车符结束。注释在一行写不完时，可写多行，每行必须以“;”号开始。

注释域可以跟在任何一个合法的操作符(无操作数时)或操作数之后，注释与操作符(或操作数)之间可以有间隔符，也可以没有，但必须以“;”号开始。

注释域是可选项，视需要而定。

数据在机内的表示方法 数据可分为带符号数(绝对数)和不带符号的数。带符号数用补码表示。用数据定义伪指令定义数据时，只能用一个字节或一个字的长度表示。

①无符号数：字节数据范围0～255，字数据范围0～65535。

②带符号数：字节数据范围－128～127，字数据范围－32768～32767。

数据的书写形式 在汇编语言源程序中，可用下述几种方法表示数据：

1. 二进制常数

由二进制数字0和1组成，数据以B结尾。如：1011B，101010B等。

2. 八进制常数

由八进制数字0～7组成，数据以Q结尾，如：132Q，276Q等。

3. 十进制常数

由十进制数字0～9组成，用D或不用D结尾均可。如123，259D等。

4. 十六进制常数

由数字0～9和字母A～F组成，数据以H结尾。若数据以A～F的字母开头，必须冠以0，以便与保留字或自定义名字相区别。如：1235H，12A8H，0A27H等。

5. ASCII码常数

用单引号括起来的可见字符串(含空格)，如：‘AB’，‘HOW ARE YOU?’等。

6. 特殊字符$

对于任何允许把表达式做为操作数的操作码，操作数都可用“$”字符表示，$表示当前地址计数器的值或当前程序计数器PC的值。

7. 标号(名字)

伪指令语句、指令语句的或用伪指令定义的名字也可以做为数据使用。如：VAL EQU 15。在程序中引用VAL时，则其值为15。

8. 表达式

将数据用运算符号连接起来的有意义的式子称为表达式。表达式的值是在汇编程序对源程序汇编的过程中求值的，只要表达求出的值是合法值，就可以代表内存单元地址或常数。

运算符号 汇编程序允许使用算术运算符、逻辑运算符和关系运算符。

1. 算术运算符

算术运算符有+，－，＊，/，MOD，SHL 和 SHR。

+，－，＊，/：是算术运算中的加、减、乘和除符号；

MOD：计算一个除法的余数，例如：26 MOD 4 = 2；

SHL n：逻辑左移n 位，例如：35 SHL 2 = 140，即35 逻辑左移2 位，每左移一位相当于乘2，左移2 位，相当于乘4；

SHR n：逻辑右移n 位，例如：32 SHR 2 = 8，即32 逻辑右移2 位，每右移一位相当于除以2，右移2 位，相当于除以4。

2. 逻辑运算符

逻辑运算符有：AND(与)，OR(或)，XOR(异或)和NOT(取反)。

这些逻辑操作都是按位操作。

AND： 逻辑与，两个数的对应位同时为1，对应位的结果为1，否则为0。例如：35 AND 7 = 5

OR： 逻辑或，两个数的对应位同时为0，对应位的结果为0，否则为1。例如：35 OR 7 = 37

XOR： 逻辑异或，两个数的对应位相同，对应位的结果为0，否则为1。例如：35 XOR 7 = 32

NOT： 取反，原数据位为0，其结果对应位为1，原数据位为1，结果对应位为0。例如：NOT 35 = 218

3. 关系运算符

关系运算符有：GT(或>)，GE≥，LT(或<)，LE，EQ(或=)，NE。

关系运算符的运算结果是一个逻辑值“真”或“假”，若关系成立，运算结果为真，否则运算结果为假。

GT 或>：大于，若运算符左侧的数据大于右侧的数据，运算结果为“真”；

GE：大于或等于，若运算符左侧的数据大于或等于右侧的数据，运算结果为“真”；

LT 或<：小于，若运算符左侧的数据小于右侧的数据，运算结果为“真”；

LE：小于或等于，若运算符左侧的数据小于或等于右侧的数据，运算结果为“真”；

EQ：等于，若运算符左侧的数据等于右侧的数据，运算结果为“真”；

NE：不等于，若运算符左侧的数据不等于右侧的数据，运算结果为“真”。

4. 运算符的优先级

上述运算符的优先级别，由高至低排列如下：

＊，/，MOD，SHL，SHR

+，－

NOT （最高）

AND

OR，XOR

），〈，=，GT，GE，LT，LE，EQ，NE （最低）

同一行的运算符优先级别相同。在表达式的运算中，若有多个运算符，先处理优先级高的运算符，后处理优先级低的运算符，若级别相同，则按出现次序由左至右顺序处理。

在表达式中，还允许使用括号，若有括号，先算括号内的式子，再算括号外的式子。

伪指令 在下述伪指令格式说明中，带有方括号项，均为可选择项，可有可无，视需要而定，未加方括号项为必选项。

1. 符号定义伪指令

(1)等价伪指令 EQU (Equate)

格式：<LAB:> EQU <EXP>

功能：将 EXP 求值后赋给左侧标号 LAB。

说明：LAB 为用户自定义名字，EXP 为常数、表达式或另外一个标号。

(2)定义标号伪指令DL(或DEFL)(Define Lable)

格式：<LAB:> DL <EXP>

功能：与EQU功能相似，也是把EXP求值后，赋给左侧标号。

说明：LAB,EXP的意义同EQU伪指令。DL与EQU的区别是用EQU定义的标号，只能定义一次，用DL定义的标可以定义多次，且总是认为最近次定义的值有效。

2. 数据定义伪指令

(1)定义字节伪指令DB(或DEFB)(Define Byte)

格式：[LAB:] DB <EXP1 [,EXP2, ...]>

功能：用EXP1, EXP2, ...的求值初始化一个或多个内存字节单元。

说明：LAB为自定义标号，是可选项，EXP1, EXP2,... 是常数、字符串、名字或表达式。EXP的求值必须是0～255或－128～127范围内的数值，若为字符串，则须用单引号括起来。

(2)定义字伪指令DW(或DEFW)(Define Word)

格式：[LAB:] DW <EXP1 [,EXP2, ...]>

功能：用EXP1, EXP2, ... 的求值初始化一个或多个内存字单元。

说明：LAB为自定义标号，是可选项，EXP1, EXP2, ...是常数、名字或表达式。EXP的求值必须是0～65535或－32768～32767范围的数值，若为字符串，必须用单引号括起来。

(3)定义信息伪指令DM(或DEFM)(Define Message)

格式：[LAB:] DM <EXP1 [,EXP2, ...]>

功能：用EXP的求值初始化内存单元。

说明：与DB的功能相似，二者区别只是当EXP为字符串时，字符串中的最后一个字符的最高位为1。

(4)定义存储单元伪指令DS(或DEFS)(Define Sterage)

格式：[LAB:] DS <EXP>

功能：保留EXP个存储单元。

说明：EXP为一个常数或表达式。

3. 代码说明伪指令

(1)代码定位伪指令ORG (Origine)

格式：[LAB:] ORG <EXP>

功能：用EXP的求值初始化汇编地址计数器，此后的机器代码从EXP指出的内存单元开始存放。

(2)绝对源代码段说明伪指令ABS (Absalute)

格式：ABS

功能：将ABS之后的语句汇编为绝对地址，直至给出另外的代码段说明伪指令为止。

(3)浮动源代码段说明伪指令REL (Relocatable)

格式：REL

功能：将REL之后的语句汇编为浮动地址，直至给出另外的代码段说明伪指令为止。

(4)公用数据区说明伪指令COM (Common)

格式：COM <LAB>

功能：由标号LAB开始的存储区为多个模块公用的数据区。

说明：公共数据区用于多个模块之间传送变量，或者用于机器语言程序与FORTRAN语言程序之间传送变量。当多个模块使用一个公共区时，公共区的长度应相同或者第一个模块中必须包含最长的公用区说明。此伪指令后由数据定义伪指令语句组成。

(5)数据段说明伪指令DATA (Data)

格式：DATA

功能：此伪指令之后为数据区，链接程序将其代码理解为浮动的。

(6)源程序结束伪指令END (End)

格式：[LAB:] END [START]

功能：源程序到此结束。

说明：LAB为自定义标号，是可选项。START为程序的启动地址。当多个程序模块连接在一起时，只有主模块必须有启动地址，其它程序模块只能以END结束，但不能有启动地址。

4.模块连接伪指令

(1)模块入口说明伪指令ENTRY (Entry)

格式：ENTRY <LAB1 [,LAB2, ...]>

功能：用以说明本模块中允许其它模块使用的程序入口地址或数据区地址。

说明：LAB1，LAB2，...是本模块中可被的其它模块引用的标号，可以是子程序入口，程序段入口或数据的地址，此伪指令语句一般位于模块的顶部，所列标号必须是其它模块中已用EXT伪指令说明的标号。

(2)外部标号说明伪指令 EXT(或EXTRN) (External)

格式：EXT <LAB1 [,LAB2, ...]>

功能：用以说明本模块将引用的其它模块中的标号。

说明：LAB1，LAB2，...是本模块中将要引用的其它模块中的标号，可以是子程序入口、程序段入口或数据区的地址，此伪指令语句一般位于模块的顶部。所列标号必须是其它模块中已用ENTRY伪指令说明的标号。

(3)包含伪指令 *INCLUDE (Include)

格式：*INCLUDE <d:FNAME>

功能：将语句中指出的源文件插入到本模块中此伪指令的当前位置处。

说明：d:为磁盘驱动器的名字，FNAME为源文件在磁盘上的文件名。要特别注意一点，FNAME文件中不得有END语句，因为当FNAME插入到本模块时，汇编程序遇到了END后，就认为源程序结束了，将终止对当前文件的汇编。

(4)模块命名伪指令 NAME(Name)

格式：NAME <MNAME>

功能：为模块命名。

说明：NAME语句并非必须有，只是经它命名的模块将作为.REL文件的一部分，库管理程序可能借助于模块名来定位.REL文件。MNAME为用户起的模块名。

5.宏操作伪指令

宏定义(MACRO)与宏结束(MEND)伪指令

格式：<MCNAME:> MACRO [#PR1, #PR2, ...]

.
.
. 宏体

MEND

功能：用于定义一条宏指令。

说明：MCNAME是用户自定义的宏名字，即宏指令。PR1，PR2，...为形式参数，这些形式参数必须以#开头。宏体是实现某种功能的一段程序。一旦用宏定义定义了宏名字之后，在编写源程序时，在源程序的操作符域使用这个名字，就如同使用宏体所代表的程序段。

程序中使用宏指令的好处是：多次重复使用的程序不必重复书写；可以改善源程序的可读性，这样编

制的程序比使用子程序运行速度高。

6. 条件汇编伪指令

条件汇编开始(IF)和条件汇编结束(ENDIF)说明伪指令

格式：IF <CONDITON>

.
.
. 条件体

ENDIF

功能：CONDITION 是一个条件表达式，若条件成立，则将条件体汇编为机器代码，否则不进行汇编。

说明：条件体是一段程序。条件表达式的值不得超过字数据的长度，只要表达式的结果非0，则条件成立。

7. 打印格式说明伪指令

(1)打印标题说明伪指令 TITLE (Title)

格式：TITLE <BODY OF TITLE>

功能：打印清单的每页的顶部打印一个标题。

说明：BODY OF TITLE 是要打印的标题，可以是行长所允许的字符串。

(2)产生清单说明伪指令 LIST (List)

格式：LIST [OPT1，OPT2，OPT3]

功能：用以选择打印清单的内容，以删掉不需要的或重复的部分。

说明：OPT1，OPT2，...是打印清单说明的可选项，它可以是 GEN，NOGEN，COND，NOCOND，ONTOFF。

GEN：在每一个宏调用之后，打印宏指令的相应宏体，直至遇到NOGEN。

NOGEN：与GEN 相反，在宏调用后不打印宏指令的相应宏体，直至遇到GEN。

COND：打印条件汇编的条件体，直至遇到NOCOND。

NOCOND：不打印条件汇编的条件体，直至遇到COND。

ON：打印程序清单。

OFF：不打印程序清单。

这6个可选项，可分为3对，GEN-NOGEN，COND-NOCOND 和ON-OFF，每对之间是矛盾的，选用时，只能选中一对中的一个。源程序汇编开始时，隐含为GEN，COND 和ON。

(3)换页说明伪指令 FROM (From)

格式：FROM

功能：打印清单时，迫使打印机换页。

11.1.5 IBM PC 宏汇编语言

IBM PC 及兼容机所使用的宏汇编程序的版本较多，但功能类似，仅有极少数伪指令的功能稍有差异，高版本的汇编程序功能略强。本节不介绍8086/8088 指令助记符及书写格式，见有关条目。

字符集

①任何可打印的ASCII 码字符(包括空格符)

②控制字符：制表符，回车符，换行符，换页符。

名字

(1) 保留字：汇编程序已赋于意义的字符串，如指令的助记符、伪指令助记符、寄存器的名字及由字符串组成的各种运算符、标志位助记符等。

(2) 用户自定义名字：用户为了引用数据、数据地址或指令地址方便，按命名规则起的名字，亦称标号或变量。宏定义伪指令定义的名字，称为宏名字或宏指令。

(3) 命名规则：

①组成名字的字符必须是：A～Z，a～z，0～9，？@，_(下划线)等；

②名字的长度≤8；

③标号的第一个字符不允许是0～9的数字；

④保留字不能用作自定义名字。

语句格式 汇编语言程序中有三种类型的语句：机器指令语句、伪指令语句和宏指令语句，这三种语句都有4个组成部分：标号、操作符、操作数和注释。格式如下：

[标号] <操作符> <操作数> [;注释]

1. 标号域

标号是用户自定义名字，用以表示操作数、操作数的地址、机器指令的地址、过程名字或宏指令(宏名字)。当标号表示操作数地址时，亦称为变量。

标号在语句中是可选项，视需要而定，一般情况下，数据区的首地址、子程序的入口、宏指令定义、转移指令的目标指令语句、主模块的启动指令语句都应有标号。

标号的定义：标号在标号域中出现称为定义标号。

标号的引用：在操作符、操作数域出现标号，称为引用标号。

标号的重复定义：指同一个标号在标号域中出现多次，即为重复定义。

未定义标号：在操作数或操作符域引用了未在标号域中出现的标号。

重复定义标号、未定义标号都属于语法错误。

标号的属性：标号有与它们用途相关的三种属性：段属性、偏移量属性和类型属性。

段属性：标号总是处于某个段中，其所处段的基地址即为其段属性。标号的段属性值可由SEG算符求得。

偏移量属性：标号与其所处段基址的距离为其偏移量属性。标号的偏移量属性可以OFFSET算符求得。

类型属性：数据定义伪指令语句的标号(亦称变量)，具有BYTE(字节)、WORD(字)、DWORD(双字)、QWORD(四字)和TBYTE(十字节)类型。指令语句或过程所用标号具有FAR(远)和NEAR(近)类型。

类型属性值如表11.1-1所示。

表11.1-1 标号类型属性值

属性	BYTE	WORD	DWORD	QWORD	TBYTE	NEAR	FAR
值	1	2	4	8	10	−1	−2

标号的类型属性值可由算符TYPE求得。

2. 操作符域

机器指令语句的操作符域，就是机器指令的助记符号，它表示指令要完成的操作。

伪指令语句的操作符域是伪指令助记符号，告诉汇编程序对其后的参数如何使用或处理，或者对其后的语句如何汇编。

宏指令语句的操作符域，是宏指令。

标号域与操作符域至少用一个空格或制表符分隔，若机器指令语句，标号一定以冒号":"结束，这时

“:”与操作符域之间的分隔符可有可无。

3. 操作数域

机器指令语句的操作数为指令的操作对象，一般为寄存器名字，以某种寻址方式表示的内存单元的有效地址或指令语句的有效地址等。

机器指令语句的操作数的个数因指令不同而异，有的指令有两个操作数，书写时，两个操作数之间用一个“,”分隔，左边的为目标操作数，右边的为源操作数；有的指令只有一个操作数，或是源操作数，或为目标操作数；有的指令没有操作数，或有操作数但不写出来，即隐含在指令中。

伪指令语句的操作数域为伪指令所需的可选参数或数据。

宏指令语句的操作数域为宏指令所需的实际参数。

操作符与操作数域之间至少用一个空格或制表符分隔。

4. 注释域

用以说明语句在程序中所起的作用。

注释域在语句中是可选项，需要时，可对语句或程序段加以说明，以便增加程序的可读性，便于软件的交流和维护。

注释以“;”开始，以回车符结束，一行写不下时，可写多行，但每行必须以“;”开始。

操作数域与注释域间可以用空格或制表符分隔，也可以不用。

数据在机内的表示方法 数据可分为带符号数和不带符号数(绝对数)，带符号数用补码表示。用数据定义伪指令定义数据时，可以用字节、字、双字、四字和十字节表示一个数据。

各种数据的表示范围如下：

字节数据：无符号数 0～255，带符号数－128～127。

字数据：无符号数 0～65535，带符号数－32768～32767。

双字数据：无符号数 $0\sim2^{32}-1$，带符号数 $-2^{31}\sim2^{31}-1$。

四字数据：无符号数 $0\sim2^{64}-1$，带符号数 $-2^{63}\sim2^{63}-1$。

十字数据：无符号数 $0\sim2^{80}-1$，带符号数 $-2^{79}\sim2^{79}-1$。

数据还可以用浮点格式表示，汇编程序用浮点格式表示数据时，使用 4 个字节：1 个字节表示阶码，3 个字节表示尾数。

阶码用过余 128 码表示。

尾数用小数原码表示，小数点的位置确定在最高位之前，最高位一定为1，且隐含，用最高位表示数据符号，“0”表示正，“1”表示负。

例如：36.625 表示为 86128000H；

－36.625 表示为 86928000H。

浮点数据的表示范围：

正数：$2^{127}\times(1-2^{-24})\geqslant N\geqslant 2^{-127}\times(2^{-1})$

负数：$2^{127}\times(-(1-2^{-24}))\leqslant N\leqslant 2^{-128}\times(-2^{-1})$

零：阶码和尾数同时为 0

数据的书写形式 在汇编语言源程序中，可用下述几种方法表示数据：

(1)二进制常数。以字母B结尾的若干“0”和“1”组成的数字串。如：100110B

(2)八进制常数。以Q结尾的，由数字0～7组成的数字串。如：127Q，325Q

(3)十进制常数。整数：用D或不用D结尾的，由数字0～9组成的数字串。如：21D，519

实数：数据中有小数点时，汇编程序将其译为浮点格式。

(4)十六进制常数。以字母H结尾的，由数字0～9和字母A～F组成的数字串。如：23H，1A6BH

为了避免与标号混淆，以字母A～F开头的十六进制数，必须冠以0。如：0A5H，0E12DH

(5)科学表示法的常数。由0～9的数字，字母E，小数点及+或-号组成的字符串。如-0.3625E+2，0.75E-2

(6)ASCII码常数。用单引号或双引号括起来的可见字符串(含空格符)。如：'ABC'，"GOOD-BYE"

(7)标号(名字)。用符号定义伪指令定义的标号(名字)，或指令语句、伪指令语句的标号都可当数据使用。如：VAL=10，在程序中引用VAL时，其值为10。

(8)特殊符号$。美元符$代表当前汇编计数器的值。在书写源程序时亦可使用。

(9)表达式。用运算符号将上述常数、符号等连接起来的有意义的式子称为表达式。

运算符号 IBM PC汇编程序允许使用的运算符种类很多，有算术运算符、逻辑运算符、关系运算符、分析算符、组合算符和记录专用算符。

1.算术运算符

算术运算符有：+，-，*，/，MOD，SHL和SHR。

(1)+，-，*，/的作用及意义与算术中所用的+，-，×，÷符号相同；

(2)MOD是取除法后的余数，如：35 MOD 4 = 3；

(3)SHL n是逻辑左移n位，如：35 SHL 2 = 140，左移一位相当于乘2；

(4)SHR n是逻辑右移n位，如：128 SHR 4= 8，右移一位相当于除以2。

2.逻辑运算符

逻辑运算符有：AND，OR，XOR和NOT，逻辑运算是按位操作的。

(1)AND：逻辑与，如：76H AND 23H = 22H

(2)OR：逻辑或，如：76H OR 23H = 77H

(3)XOR：逻辑异或，如：76H XOR 23H = 55H

(4)NOT：逻辑非，如：NOT 76H = 89H

注意，在机器指令的操作中也有逻辑与、逻辑或、逻辑异或和逻辑非操作，它们的助记符号与上述逻辑运算符号完全相同，但是运算符的运算是在汇编程序对源程序的汇编过程中求值，而与、或、异或、取反指令是在计算机执行机器语言程序时进行相应操作获得运算结果。运算符的两个操作数只能是常数或标号，指令的两个操作数可以是寄存器或内存单元中的数据，源操作数也可以是常数或标号。

3.关系运算符

关系运算符有EQ，NE，GT，GE，LT和LE。关系运算的两个操作数必须是数值或表达式，关系运算符的结果是逻辑值：真值或假值，真值用FFFF表示，假值用0表示。

(1)EQ：等于，运算符左侧表达式的值等于右侧表达式的值时，运算结果为真；

(2)NE：不等于，运算符左侧表达式的值不等于右侧表达式的值时，运算结果为真；

(3)GT：大于，运算符左侧表达式的值大于右侧表达式的值时，运算结果为真；

(4)GE：大于或等于，运算符左侧表达式的值大于或等于右侧表达式的值时，运算结果为真；

(5)LT：小于，运算符左侧表达式的值小于右侧表达式的值时，运算结果为真；

(6)LE：小于或等于，运算符左侧表达式的值小于或等于右侧表达式的值时，运算结果为真。

4.分析算符

分析算符有：SEG，OFFSET，TYPE，SIZE和LENGTH。分析算符用于取得标号的各种属性值或标号所分配的内存空间的大小。

(1)SEG算符

格式：SEG LAB

说明：LAB 为用户自定义的标号，运算结果为标号所处段的基址值(16 位)。

(2)OFFSET 算符

格式：OFFSET LAB

说明：LAB 为用户自定义标号，运算结果为标号距其所处段基址的距离(16 位)。

(3)TYPE 算符

格式：TYPE LAB

说明：LAB 为用户自定义标号，运算结果为标号类型属性值。

(4)SIZE 算符

格式：SIZE LAB

说明：LAB 为用户用数据定义伪指令定义的标号，而且数据项仅为DUP 形式，运算结果为伪指令分配给标号的内存字节数。

例如：ADR1 DB 10 DUP(0)
ADR2 DW 15 DUP(0)
ADR3 DD 5 DUP(0)

则：SIZE ADR1 = 10
SIZE ADR2 = 30
SIZE ADR3 = 20

(5)LENGTH 算符

格式：LENGTH LAB

说明：LAB 为用户用数据定义伪指令定义的标号，而且数据仅为DUP 形式，运算的结果为与标号类型相联系的数据个数。

仍以上例中三个数据定义伪指令语句为例。

则：LENGTH ADR1 = 10
LENGTH ADR2 = 15
LENGTH ADR3 = 5

SIZE,LENGTH,TYPE 三者之间的关系为：

SIZE LAB = LENGTH LAB * TYPE LAB

5. 组合算符

组合算符有 PTR,THIS 和 SHORT。用于建立或临时修改标号类型或存储器操作数存储单元的类型。

(1)PTR 算符

格式：NEWTP PTR EXP

说明：NEWTP 为要建立(或要改变的)新的类型，可以是类型助记符 BYTE,WORD,DWORD,QWORD,TBYTE,NEAR 或FAR。EXP 为已定义的标号或存储器操作数的各种寻址方式的表达式。

例如：MOV AX, WORD PTR ADR1
MOV BYTE PTR [BX], 35H

(2)THIS 算符

格式：THIS NEWTP

说明：NEWTP 为要建立(或要改变的)新的类型助记符号，用来建立一个标号的类型属性，该标号的段和偏移量属性就是其下一个存储单元的段和偏移量，而其类型为NEWTP。

(3)SHORT 算符

格式：SHORT LAB

说明：LAB 为指令语句的标号，SHORT 算符用于指明转移指令的目标指令在－128～127 的范围内，以使转移指令译为两个字节的指令。

6. 字节分离算符

字节分离算符有HIGH 和LOW。该算符用于截取字数据的高字节或低字节。

例如： VAL EQU 1234H
HIGH VAL = 12H
LOW VAL = 34H

7. 记录专用算符

记录专用算符有MASK 和 WIDTH。

(1)MASK 算符

格式：MASK RFL

说明：RFL 为记录的字段名。该算符的结果为RFL 字段的掩码，即对应于RFL 字段的各位为1，其余位为0 的代码。

(2)WIDTH 算符

格式：WIDTH RFL

说明：RFL 为记录的字段名。该算符的运算结果为RFL 字段的宽度，即该字段所占用的二进制位的位数。

8. 运算符的优先级

下面由高至低列出运算符的优先次序，同一行的运算符优先级相同。

(1)SIZE,LENGTH,MASK,WIDTH
(2)SEG,OFFSET,TYPE,PTR,THIS
(3)HIGH,LOW
(4)*,/,MODE,SHL,SHR
(5)+,－
(6)EQ,NE,GT,GE,LT,LE
(7)NOT
(8)AND
(9)OR,XOR
(10)SHORT

在同一个表达式中出现多个运算符时，按运算符的优先级别运算，先算优先级别高的，再算优先级别低的，同级运算符，则按出现次序由左至右顺序计算，括号可以改变优先次序，先运算括号内的各项，再运算括号外的各项。

伪指令 在下述伪指令语句的格式说明中，带有[ ]的项是可选项，即可有可无，视需要而定。

(1)符号定义伪指令

①等价伪指令EQU (Equate)

格式：<LAB> EQU <EXP>

功能：将EXP 求值后赋给标号LAB。

说明：LAB 为用户自定义标号，EXP 为数值表达式，可以是常数或其它名字等。

②等号伪指令=

格式：<LAB> = <EXP>

功能：将EXP 求值后赋给标号LAB。

说明：LAB 为用户自定义标号，EXP 为数值表达式，可以是常数或其它名字。EQU 和=的功能类似，