目　录

第五章　宏汇编程序

MACRO 是一个两遍扫描的宏汇编程序，它要求12K或更大配置的RT-11系统(或后台区)。MACROS (以后均译为宏)是源语言命令语言中的一些指令，它们等价于指定的一系列机器指令或机器命令，具有最小内存配置的用户必须使用汇编程序（ASEMBL）和展开程序（EXPAND），而在汇编任何程序之前，应该阅读本章及第十和十一章（本章指出了汇编程序（ASEMBL）所不具有的宏特性，另外许多特性汇编程序没有，但展开程序却具有）

宏的某些主要特性有：

1、汇编功能的程序控制。

2、说明输入或输出文件的设备和文件名。

3、在命令输出设备上打印错误清单。

4、按字母顺序的格式化的符号表列表。

5、产生可重定位的目的模块。

6、说明全程符号，实现目的模块的联接。

7、各种条件汇编命令。

8、各种程序进行分段的命令。

9、用户定义的宏命令。

10、系统宏的广泛集合。

11、包括相互引用列表（CREF）在内的广泛的列表控制。

宏汇编程序的操作指南在 5.7 节中描述。

译注：本章是借用西北工业大学 608 教研室的译稿编译而成。

5·1 源程序格式

源程序是由一系列源程序行所组成；每一源程序行包含一个汇编语句並跟一个语句终结符。终结符可以是一个换行字符（它使行计数增1）或一个换页符号（它使行计数重新开始並使页计数增1）。

注

编辑程序对在源程序中所遇到的每一个回车后面自动地添上一个换行，为了列表格式，如果在换行或换页前面没有回车，宏就自动地嵌入一个回车。

一个汇编语句行最多能够包含132（十进制）个字符（语句终结符除外），超过这个界限，多出的字符被忽略並且给出错误标志。

5·1·1 语句格式

一个语句最多能够包含四个域。它们依出现的次序和指定的终结符号来识别，宏汇编语言语句的一般格式是：

标号：操作符　操作数（一个或多个）　；注释

标号和注释域是任选的，操作符和操作数域是相互依赖的；其中之一可以被省略，这取决于另一个的内容。

汇编程序逐个解释或处理这些语句，不是产生一个或多个二进制指令或数据字，就是执行一次汇编处理。一个语句可以只包含这些域中的一个，也可以包含所有的四个。语句中允许空白行。

某些语句只有一个操作数，例如：

```
CLR   R0
```

而另外一些有两个：

```
MOV   #344, R2
```

汇编语言的语句必须完全在一行上。不允许有继续行（如果试图用换行来得到继续行，那么汇编程序就把这个换行解释为语句终结符。）

宏源语句可以通过编辑程序插入横表符实现格式化，该横表符能调整语句域位置，例如：

标号域	操作符域	操作数域	注 释 域
CHECK:	BIT	#1,R0	; IS NUMBER ODD?
	BEQ	EVEN	; NO,IT'S EVEN
	MOV	#-1,ODDFLG	; ELSE SET FLAG
EVEN:	RTS	PC	; RETURN

5·1·1·1 标号域—— 标号是一个用户定义的符号，它的前六个字符是唯一的，並被赋予指令计数器的当前值，並且送入用户定义符号表。这个标号的值可以是绝对的（在内存中固定的不依赖于程序的位置）或是浮动的（在内存中不固定），这取决于指令计数器的值当时是绝对的还是浮动的。

标号是程序中访问指定存贮单元的符号表示。如果有标号，总是出现在语句的第一个域，並且必须以冒号来终结。例如，如果当前的存贮单元是绝对的100（八进制）语句。

```
ABCD:   MOV   A, B
```

把值100（八进制）赋给标号ABCD。随后引用ABCD就是访问存贮单元100（八进制）。在这个例子中，如果指令计数器在这程序段中被定义为浮动的，那么ABCD的最终值将是100（八进制）加上一个由LINK在重定位这段代码时所指定的值，这个值被称为浮动常数（ABCD的最终值在联接前是不知道的。具体内容将在这章最后和第六章中讨论。）

在一个标号域中可以出现多于一个标号。在这种情形，域中的每一个标号被赋与同一个值。例如，如果当时的指令计数器是100（八进制），语句：

```
ABC:ERREX:MASK:  MOV   A,B
```

中三个标号— ABC，ERREX 和 MASK —中的每一个都等于值 100（八进制），用做标号的符号在用户程序中不可以被重复定义，如果重复定义了标号，会在汇编清单中产生一个错误标志。

5·1·1·2 操作符域—— 操作符域跟随在语句的标号域后面，它可以包含一个宏调用，一个指令助记符或一个汇编命令。操作符的前头可以有一个或多个标号，也可以没有，在操作符后面可以跟一个或多个操作数和注释或直接跟注释。开头的和后继的空格和横表都被忽略。

当操作符是一个宏调用时，汇编程序嵌进相应的代码将其展开。当操作符是一个指令助记符时，它指明了要生成的指令以及对跟在后面的操作数（一个或多个）要执行的动作。当操作符是汇编程序命令时，它指明某一功能，或在汇编时要被执行的动作。

操作符可以由空格，横表或任何非字母数字字符（符号成份）来终结。

考虑下面的例子：

```
MOV  A,B     （以空格终结操作符 MOV）
MOV  @ A,B   （以@终结操作符MOV）
```

当语句行不包含操作数或注释时，操作符被回车后跟以换行或换页字符所终结。

一个空白操作符域被解释为一个·WORD 汇编命令。（参见§5·5·3·2 节）

5·1·1·3 操作数域—— 操作数是由操作符操作的语句部分。操作数可以是表达式，数，或符号变量或宏变量（在操作符所能操作的

的范围）当多个操作数出现在语句当中时，每个操作数要用下面这些符号中的一个所分隔开，这些符号是：逗号，横表，空格或是把一个或多个操作数括起来的成对尖括号（见5·2·1·1节）。用多个定界符来分隔操作数是不合法的（除非是空格和横表，空格和／或横表的任何组合表示一个定界符）。一个操作数的前面可以是操作符，标号或另一个操作数，后面可跟以注释。

操作数域后面跟以注释时，被分号所终结。当以操作数来结束语句时，操作数域以语句终结符所终结，例如：

```
LABEL:MOV   A,B ;   COMMENT
```

MOV 和 A 之间的空格终结了操作符域并且开始操作数域，逗号分隔开操作数 A 和 B，分号终结操作数域并开始注释域。

5·1·1·4 注释域—注释域是任选的，它可以包含除 NULL，RUBOUT，回车，换行，竖表或换页外的任何ASCII 字符，在注释域中出现的所有其它字符，甚至具有确定用法的特殊字符都被汇编程序所忽略。

注释域的前面可以是另外三个域型中的任何一个或全部，也可以全没有。注释必须以分号开始而以语句终结符结束。

注释不影响汇编处理或程序执行，但是为了今后的分析，调试，或说明的目的，它们在源程序列表中是有用的。

5·1·2 格式控制

源程序的横向格式或行格式是由空格和横表字符所控制。这些字符不影响汇编处理，除非它们被嵌入在符号，数或ASCII 正文中，或它们是被用作操作符域的终结符。因此，这些字符能够被用来提供一个整齐的源程序，语句能够被写为：

```
LABEL:MOV( SP)+,TAG; POP VALUE OFF STACK
```

或者，使用格式字符，它能被写为：

```
LABEL:  MOV   (SP)+,TAG , POP VALUE OFF STACK
```

这样在源程序清单的上下文中，就比较容易阅读。

竖格式，即一页的大小，是由换页字符控制，在第n 行之后被嵌入一个换页（CTRL FORM），从而产生一个n行的页（参看5·5·1·6节关于宏汇编的页格式说明以及5·5·1·2 节关于汇编清单输出说明）

5·2 符号和表达式

这节叙述合法的宏汇编表达式的各种成分：汇编字符集，符号结构，数，运算符，项和表达式。

5·2·1 字符集

下面的字符在宏汇编的源程序中是合法的：

1、字母A到Z。大写和小写字母两者都是可以接受的，虽然，在输入时，小写字母被转换为大写字母。小写字母的输出仅仅能够把它们的ASCII 值送到输出设备上。如果 ·ENABL LC 是起作用的话，这种转换对于 ASCII ,ASCIZ '（单引号）或 "（双引号）语句是不真实的。

2、 0到9的数字。

3、字符·（句号或小数点）和$（美元符）被保留用作系统程序符号（用作局部符号除外，见5·2·5 节）。

4、下面这些特殊字符

字符	名称	功能
carriage	回车	格式字符
line feed	换行	格式字符
form feed	换页	格式字符
vertical tab	竖表	格式字符
:	冒号	标号终结符

字符	名称	功能
=	等号	直接赋值指示符
%	百分号	寄存器项指示符
TAB	横表	项或域终结符
SPACE	空格	项或域终结符
#	数符	立即表达式指示符
@	间接符	间接地址指示符
(	左圆括弧	寄存器开始指示符
)	右圆括弧	寄存器终结指示符
,	逗号	操作数域的分隔符
;	分号	注释域的指示符
<	左尖括弧	变量或表达式的开始指示符
>	右尖括弧	变量或表达式终结的指示符
+	加号	算术加运算符或自增指示符
-	减号	算术减运算符或自减指示符
*	星号	算术乘运算符
/	斜线	算术除运算符
&	ampersand	逻辑“与”运算符
!	感叹号	逻辑“或”运算符
"	双引号	两个ASCII 字符指示符
'	单引号	单个ASCII 字符指示符
↑	向上箭头	通用的一元操作符，变量指示符
\	反斜线	宏数字变量指示符(ASEMBL 中不能用)

5·2·1·1 分隔字符和定界字符—— 这章的其余部分涉及到合法的分隔字符和宏变量定界符，这些项被定义在下面的表5－1中。

表5－1　　合法的分隔字符

字符	定　义	用　　法
空格	一个或多个空格和／或横表	空格作为合法分隔符仅仅是对于变量操作数，而表达式中空格是被忽略的
,	逗　号	对于表达式和变量操作数两者来说逗号都是合法分隔符
<……>	成对尖括弧	成对尖括弧被用来括住一个变量，特别是当那个变量包含分隔字符时。成对尖括弧可以在程序的任何地方括住一个表达式，以便作为一项来处理。（当变量包含一元操作符时，尖括弧结构将被使用
↑ \……\	向上箭头结构，在这里，由任何成对印刷字符所括的参数跟在向上箭头的后面	这种结构在功能上等价于成对尖括弧，一般仅仅在变量包含尖括弧时才被使用

宏变量可以几种形式出现，以便允许有特殊情形。在分隔变量时要遵守的一些规则是：

1、如果变量串不包含分隔字符（它们在表5－1中没有被定义）