第3章 汇 编 器

汇编器的功能是将汇编语言源代码转换为 COFF 格式的机器语言目标代码。源文件通常包含以下几种汇编语言要素：

- 汇编器伪指令
- 宏伪指令
- 汇编语言指令

3.1 汇编器概述

C54x 的汇编器 cl500 可完成下述功能：

- 将文本格式的源文件转换为可重新定位的 C54x 目标文件。
- 根据需要产生源文件列表，并且可以定制源文件列表。
- 允许用户将代码分割成若干个段，并为每个代码段保留一个段程序计数器 SPC。
- 定义和引用全局符号，并根据需要在源文件列表中添加交叉引用列表。
- 对程序块进行条件汇编。
- 支持宏操作，允许内嵌定义宏，或者在库中定义宏。

汇编器既可以接受汇编语言源文件作为输入，也可以接受由 C/C++ 编译器生成的汇编语言源文件。

3.2 调用汇编器

汇编器调用格式如下：

```
cl500 [input file [object file [listing file] ] ] [ - options ]
```

其中各个字段的含义如下。

cl500 调用汇编器的命令，它以扩展名为.asm 的文件作为汇编源程序。

input file 汇编语言源文件名。如果用户没有提供扩展名，则汇编器默认扩展名是.asm，除非用户使用了-f 选项。如果用户没有提供输入文件名，汇编器将提示用户输入。源文件或者为代数指令格式，或者为助记符指令格式，但不能二者都有。汇编器默认的指令格式为助记符格式，用户可以使用-mg 选项来声明源程序为代数格式。

object file　汇编器创建的 C54x 目标文件名，默认扩展名为.obj。如果没有提供目标文件名，汇编器将使用源文件名，加扩展名.obj，作为输出目标文件名。

listing file　汇编器创建的可选的列表文件名。如果没有给出列表文件名，汇编器将不产生列表文件，除非用户使用了-l(小写的 L)或-x 选项，这时输出列表文件使用源文件名，以.lst 为扩展名，并将该文件存储在源文件所在的目录下。如果用户只给出列表文件名，但是没有给出扩展名，则汇编器默认扩展名是.lst。

options　用户使用的汇编器选项，不区分大小写，可以出现在命令行中 cl500 之后的任意位置。每个选项前面都要加一个短线连字符“-”。没有参数的多个单字母选项可以连写，例如，-l -c 可以写做-lc。但是有参数的选项(例如-i)就必须单独指定。

以下是各个汇编器选项的介绍。

-@

-@选项后紧随一个文件名，将该文件的内容添加到命令行中，用于避免命令行长度受主机操作系统的限制。在一个命令文件中，文件名和选项参数中包括的空格或连字符必须用引号括起来，例如“this-file.asm”。

-a

使用-a 选项时，将创建一个绝对列表文件，并且汇编器不会产生目标文件。-a 选项与绝对列表程序一起使用。

-c

使汇编器不区分汇编语言文件中的大小写。例如，-c 选项将使大写的符号 ABC 与小写的符号 abc 等效。如果不用该选项，默认情况下区分大小写。区分大小写主要是对符号名的限制，对助记符和寄存器名无影响。

-d

-d name[=value] 用来设置 name 符号，等效于在汇编语言文件的开头插入 name .set value 语句。如果没有给出 value 值，符号将被设置为 1。

-f

默认情况下，汇编器给源文件加扩展名.asm。如果使用-f 选项，汇编器就不会加扩展名。

-g

在源程序调试器中使能汇编器源程序调试，源文件中每行的行信息输出到 COFF 文件。需要说明的是，用户不能在已经包含.line 伪指令的汇编代码中使用-g 选项(例如由 C/C++编译器用-g 选项产生的代码)。

-h、-help、-?

这 3 个选项是等效的，显示出所有可用的汇编器选项的一个列表。

-hc

-hc filename 使汇编器复制文件到汇编模块中，该文件被插入到源文件语句的前面。复制的文件也出现在汇编列表文件中。

-hi

-hi filename 使汇编器在汇编模块中导入(包含)指定的文件。该文件放在源文件语句之前。导入的文件不出现在汇编列表文件中。

-i

为由.copy、.include 或.mlib 伪指令命名的文件指定一个路径名。该选项的使用格式为-i pathname。

-l

小写的 L,用来控制产生列表文件。

-ma

(ARMS 模式)告知汇编器在处理源文件时 ARMS 状态位将被使能。默认情况下,汇编器假定该位为禁止状态。

-mb

改变并行指令间总线冲突的诊断信息级别。默认情况下,总线冲突报告为错误。使用-mb 选项后,总线冲突被报告为警告信息。

-mc

(CPL 模式)告知汇编器处理源文件时 CPL 状态位将被使能,这将导致汇编器强制使用 SP 相对寻址语法。默认情况下汇编器假定该状态位为禁止状态。

-mf

使汇编器调用使用扩展地址。

-mg

声明源文件中包含代数指令。

-mh

使汇编器产生执行速度更快的代码。默认情况下汇编器将产生尺寸较小的代码。只有 asm500 使用该选项。

-mk

指定 C54x 为大存储器模式。该选项将符号_large_model 设置为 1。使用该选项时,汇编器将目标文件标记为大存储器模式文件,将该信息提供给连接器来检查小模式和大模式目标程序块的非法组合。

-ml

(C54x 兼容模式)告知汇编器,在源文件处理过程中 C54CM 状态位将被使能。默认情况下,该位是禁止状态。

-mn

使汇编器在延迟跳转或调用指令中,取消延迟时隙中的 NOP 操作。

-mt

告诉汇编器在处理源文件时,SST 状态位将被禁止。默认情况下,汇编器假设该位处于使能状态。只有 asm500 使用该选项。

-mv

使汇编器使用变长指令的最长形式(P24)。默认情况下,汇编器总是试图将变长指令转化为最短长度。

-mw

禁止汇编器输出警告信息,只有 asm55 支持该选项。

-purecirc

告知汇编器，C54x 文件使用循环寻址(不使用线性/循环模式位)。只有 asm500 支持该选项。

-pw

对代码的流水线冲突产生警告信息。汇编器不能检测到所有的流水线冲突，只能在线性代码中检测到。当检测到冲突后，汇编器将发出警告信息，并且报告解决该冲突所需要添加的延迟时隙数。

-q

取消任何交互提示和处理过程信息，使汇编器在静态模式下工作。

-r

-r[num]选项将根据 num 值限制 remark 类型的警告。remark 是一种比 warning 严重程度略低的警告信息。如果该选项中未指定 num 值，所有的 remark 将被禁止。

-s

将所有定义的符号放在目标文件符号表中。汇编器通常只将全局符号放在目标文件符号表中，当使用-s 选项时，标号和汇编时常量也放在该表中。

-u

-u name 取消对预定义常量 name 的定义，优先级高于-d 选项，使-d 选项对常量的定义失效。

-v

-v value 告知汇编器为哪种型号的处理器产生目标代码。value 有以下几种选择：541、542、543、545、545lp、546lp、548、549。

-x

产生一个交叉引用列表，并且放在列表文件的最后。同时也将交叉引用信息加入到目标文件中，为交叉引用列表器所用。

用户也可以直接调用汇编器 asm500，其语法结构是：

```
asm500 [input file [object file [listing file] ] ] [ - options ]
```

其中各个字段的含义如下。

asm500　调用汇编器的命令。

input file　汇编语言源文件名称。如果用户没有提供扩展名，则汇编器默认扩展名是.asm，除非用户使用了-f 选项。如果没有输入文件名，汇编器将提示用户输入。

object file　汇编器生成的 C54x 目标文件名，默认扩展名为.obj。如果没有指定目标文件名，汇编器将使用与源文件相同的名称，只是将扩展名改为.obj。

listing file　汇编器输出的可选的列表文件名。如果没有指定列表文件名，则汇编器将不产生列表文件，除非使用了-l(小写的 L)或-x 选项，这时输出列表文件以源文件名为名，以.lst 为扩展名，并将该文件存储在源文件所在目录下。如果没有指定扩展名，则汇编器默认扩展名是.lst。

options　与调用 cl500 命令的选项基本相同，只是没有-v、-mf、-mg 和-mb 等 4 个选项。

3.3 C54x 汇编器的特点

本节介绍 C54x 汇编器的主要特点，包括：

- 字节/字寻址
- 并行指令规则
- 变长指令长度的确定
- 存储器模式
- 使用 MMR 地址时的汇编器警告

3.3.1 字节/字寻址

C54x 存储器对于代码的访问按 8 位字节寻址，对于数据的访问则按 16 位字寻址。汇编器和连接器能够掌握地址、相对偏移量以及给定段中数据单元的字宽（对于数据段为字，代码段为字节）等信息。

注意：.struct 和.union 结构中的偏移量

无论当前是什么段，由.struct 或.union 定义的结构的偏移量总是以字为单位。汇编器总是假设.struct 和.union 用于对数据的操作，而不是对代码的操作。

3.3.1.1 代码段的定义

若存在以下两种情况之一，汇编器将该段认做代码段。

- 该段以.text 伪指令引入。
- 该段至少含有一条指令。

如果一个段中没有出现.text、.data 或.sect 伪指令，汇编器将它默认为.text（代码）段。由于段的类型决定了汇编器对偏移量和尺寸的计算方式，因此在将位域汇编到某段之前，必须明确当前工作在代码段还是数据段。

3.3.1.2 汇编程序与基本地址单位

汇编器和连接器都假定在用户代码中，数据段内容使用字地址和偏移量，程序段内容使用字节地址和偏移量。

- 如果地址通过程序地址总线发出（例如，地址作为调用或转移指令的目标地址），则处理器认为它是一个全 24 位地址。这里使用的常数应该以字节为单位表示。这时，在代码段定义的标号能被汇编器和连接器正确处理，但是在数据段定义的标号不能正常使用。
- 如果地址通过数据地址总线发出（例如，某个地址表示一个读或写操作的存储器单元），则处理器把它当作 23 位字地址。这里使用的常数应该以字为单位表示。这时，在数据段定义的标号能被汇编器和连接器正确处理，但是在代码段定义的标号

不能正常使用。

- 汇编器列表文件中的 PC 值列从该段的起始处开始按照一定的单位计数。代码段的 PC 以字节为单位计数，而数据段的 PC 则以字为单位计数。

  例如：

```
1  000000           .text        ; PC 以字节为单位计数
2  000000 2298      MOV AR1,AR0
3  000002 4010      ADD #1,AC0
4
5  000000 .data                  ; PC 以字为单位计数
6  000000 0004      .word 4,5,6,7
   000001 0005                   ; PC 为 1 字
   000002 0006                   ; PC 为 2 字……
   000003 0007
7  000004 0001  foo .word 1
```

- 数据定位操作伪指令(如. byte、. ubyte、. char、. uchar 和. string 等)在代码段中为每个字符分配一个字节，在数据段中为每个字符分配一个字空间。但是，TI 强烈推荐用户仅在数据段中使用这些伪指令。
- 如果伪指令带有一个表示地址的参数，该参数在代码段看作以字节为单位，在数据段看作以字为单位。

  例如：

```
.align 2
```

该语句在代码段中将 PC 对齐到 2 字节(16 位)边界，在数据段中将 PC 对齐到 2 字(32 位)边界。

下面是 C54x 的数据和代码样例。

例 3-1　C54x 的数据样例

```
.def Struct1, Struct2
.bss Struct1, 8                   ; 为 Struct1 分配 8 个字空间
.bss Struct2, 6                   ; 为 Struct2 分配 6 个字空间
.text
MOV *(#(Struct1 + 2)),T0          ; 加载 Struct1 中的第三个字
MOV *(#1000h),T1                  ; 0x1000 是绝对的字地址
```

例 3-2　C54x 的代码样例

```
.text
.ref Func
CALL #(Func + 3)                  ; 跳转到"Func + 3 字节"的地址
CALL #0x1000                      ; 0x1000 是一个绝对字节地址
```

3.3.1.3　将代码当作数据使用，以及将数据当作代码使用

汇编器不支持将代码地址作为数据地址使用(例如，在程序空间读写数据)。同样，汇编器也不支持将数据地址作为代码地址使用(例如，程序跳转到一个数据标号)。这是因为可寻址单元的尺寸不同，代码标号地址是 24 位字节地址，而数据标号是 23 位字地址。因此，

要求用户遵守以下规则：

- 代码和数据不能混在同一个段中。所有数据(包括常数)必须与代码放在不同的段中。
- 如果应用程序对程序段中的一个位域进行读写操作，将不起作用。

3.3.2 并行指令规则

汇编器按照 C54x 指令集参考手册中的规则对并行指令对进行语义检查。为了符合并行原则，汇编器可能会交换指令对中两个指令的顺序。例如，下面的两组指令都是合法的，并汇编成相同的目标代码：

```
AC0 = AC1 || T0 = T1 ^ #0x3333
T0 = T1 ^ #0x3333 || AC0 = AC1
```

3.3.3 变长指令长度的确定

默认情况下，汇编器总是希望变长指令的长度尽可能短。例如，汇编器将尽量选用以下 3 个无条件转移指令中最短的：

```
goto L7
goto L16
goto P24
```

但是，如果变长指令的地址在汇编时不能确定，例如该地址是在其他文件中用符号定义的，则汇编器将选择指令的最长可能形式。在上例中，将选择 goto P24 格式。

下面的一组指令用来确定长度：

```
goto L7, L16, P24
if (cond) goto l4, L8, L16, P24
call L16, P24
if (cond) call L16, P24
```

在某些场合，用户可能希望汇编器选择某些指令的最长可能(例如 P24)形式，因为一些指令的 P24 形式比同一指令的较短形式执行速度快。例如 goto P24 指令使用 4 字节和 3 个周期，而 goto L7 指令使用 2 字节，但是执行时间为 4 个周期。

用-mv 汇编器选项或. vli_off 伪指令，可以使下列指令保持最长形式：

```
goto P24
call P24
```

-mv 汇编器选项使汇编器在整个文件中禁止以上两个指令进行长度选择操作。. vli_off 和. vli_on 伪指令可以在汇编语言文件中定义允许变长指令长度选择的区域。如果命令行选项和伪指令有冲突，伪指令优先。所有其他的变长指令将继续被汇编器确定为它们的最短可能形式，无论是否使用了-mv 选项或. vli_off 伪指令。

. vli_off 和. vli_on 伪指令的作用域是静态的，而且不受汇编语言程序控制流的影响。

3.3.4　存储器模式

汇编器提供了 3 位存储器模式位(对应着 9 个存储器模式),分别是 C54x 兼容模式位 C54CM、CPL 位以及 ARMS 位。汇编器根据模式的指定决定是否接受输入文件。由于模式的不同,同一个输入文件可能会产生不同的汇编输出。

存储器模式取决于 C54CM、CPL 和 ARMS 3 个状态位的取值组合。汇编器不能自动获得这 3 个状态位的值,用户必须使用汇编器伪指令或者命令行选项来告诉汇编器这些模式位的值。如果某个指令修改了 C54CM、CPL 或 ARMS 状态位的值,后面必须紧跟一条相应的汇编器伪指令告知汇编器。当汇编器得知这些位值的改变之后,能够给出该模式下关于语法和语义的错误和警告信息。

3.3.4.1　C54x 兼容模式

如果一个源文件是由 C54x 代码生成的,需要 C54x 兼容模式。除非生成的源代码变为 C54x-native 模式,否则要用命令行选项-ml 汇编该文件。也可以使用. c54cm_on 和. c54cm_off 伪指令来指定 C54x 兼容模式区域。. c54cm_on 和. c54cm_off 伪指令不带参数。如果命令行选项和伪指令有冲突,伪指令优先。

. c54cm_on 与. c54cm_off 伪指令的作用域是静态的,不受汇编语言程序控制流的影响。位于. c54cm_on 与. c54cm_off 伪指令之间的所有代码被汇编成 C54x 兼容模式。

在 C54x 兼容模式下,AR0 用于替换存储器操作数中的 T0(C54x 的变址寄存器)。例如,在 C54x 兼容模式下,*(AR5+T0)是错误的,应该改为 *(AR5+AR0)。

3.3.4.2　CPL 模式

CPL 模式影响直接寻址。汇编器无法自动获得 CPL 状态位的值,用户必须使用. cpl_on 和. cpl_off 伪指令来确定 CPL 的取值。如果某个指令修改了 CPL 状态位的取值,后面必须紧跟一个. cpl_on 或. cpl_off 伪指令来告知汇编器。. cpl_on 伪指令使 CPL 状态位置 1,它等效于使用-mc 命令行选项; . cpl_off 伪指令使 CPL 状态位置 0。. cpl_on 和. cpl_off 伪指令不带参数。如果命令行选项和伪指令有冲突,伪指令优先。. cpl_on 与. cpl_off 伪指令的作用域是静态的,不受汇编语言程序控制流的影响。. cpl_on 与. cpl_off 伪指令之间的所有代码被汇编成 CPL 模式。

在 CPL 模式下,直接存储器寻址以堆栈指针(SP)为参考。直接存储器寻址的语法格式是 *SP(dma),dma 可以是常数或连接时确定的符号表达式。

默认状态下(. cpl_off),直接存储器寻址以数据页寄存器(DP)为参考,语法格式是@dma。这里的 dma 可以是常数或连接时确定的符号表达式。汇编器计算 dma 和 DP 寄存器值的差,并将差值写入输出文件。

DP 寄存器可以在文件中引用,但是不能在该文件中定义(它是外部设置的)。因此用户必须在使用 DP 之前通过. dp 伪指令把 DP 的值告诉汇编器。如果某个指令修改了 DP 寄存器,后面必须紧跟一个相应的汇编器伪指令告知汇编器。该伪指令的语法如下:

```
.dp dp_value        ; dp_value 为常数或者是符号表达式
```

如果文件中没有使用.dp 伪指令,汇编器将假设 DP 寄存器的值为 0。.dp 伪指令的作用域是静态的,不受汇编语言程序控制流的影响。伪指令设定的值将被持续使用,除非遇到下一个.dp 伪指令,或者源文件结束。

无论 CPL 模式是否被指定,汇编器对 MMR 页和 I/O 页的直接存储器寻址操作是相同的。对 MMR 页的访问在语法上由 mmap()限定词标识;对 I/O 页的访问由 readport()和 writeport()限定词来标识。对它们的直接存储器寻址操作都以 0 地址为参考。

3.3.4.3 ARMS 模式

ARMS 模式影响间接寻址,并且在控制代码中是很有用的。汇编器无法跟踪 ARMS 状态位的值,因此用户必须用.arms_on 和.arms_off 伪指令将 ARMS 值告知汇编器。如果某个指令修改了 ARMS 状态位,后面必须紧跟一个相应的汇编器伪指令告知汇编器。.arms_on 伪指令使 ARMS 状态位置 1,它等效于使用了-ma 命令行选项;.arms_off 伪指令使 ARMS 状态位置 0。.arms_on 和.arms_off 伪指令都没有参数。当命令行选项和伪指令发生冲突时,伪指令优先。

.arms_on 和.arms_off 伪指令的作用域是静态的,不受汇编语言程序控制流的影响。所有在.arms_on 和.arms_off 伪指令之间的汇编代码都被汇编成 ARMS 模式。在 ARMS 模式下(.arms_on),将使用对间接存储器访问的短偏移量修改方式。这种修改比代码尺寸的优化更加有效。

3.3.5 使用 MMR 地址时的汇编器警告

当存储器映射寄存器(MMR)被用作单存储器操作数(Smem)时,助记符汇编器 asm500 将发出一个“Using MMR address”警告。该警告表示汇编器认为 MMR 被用作以 DP 为参考的直接寻址操作数。如果该指令为写操作,DP 必须为 0。例如,对于下面的源语句:

```
ADD SP, T0
```

汇编该语句时将输出一条“Using MMR address”警告信息,如下所示:

```
"file.asm", WARNING! at line 1:[W9999] Using MMR address
```

汇编器警告,该指令等效于:

```
ADD 地址(DP + MMR address of SP)的值, T0
```

只有当 DP 为 0 时,SP 的值才被访问。该指令最好应该改写为:

```
ADD mmap(SP), T0
```

当已知 DP 为 0 时,要执行这样的调用,用户可以使用‘@’符号来避免警告信息:

```
ADD @SP, T0
```

3.4　为汇编器的输入命名备用的文件和路径

.copy、.include 以及.mlib 伪指令用来指示汇编器从外部文件读入代码。其中,.copy 和.include 伪指令指示汇编器从其他文件中读取源程序语句,而.mlib 伪指令指定一个包含宏函数的库。上述伪指令的语法格式如下:

```
.copy "filename"
.include "filename"
.mlib "filename"
```

其中 filename 是要求汇编器读入源语句的一个 copy/include 文件名,或者是包含宏定义的宏库。文件名可以是完整的路径名、部分路径名或者没有路径信息的文件名,汇编器按照以下顺序搜索文件:

1) 当前正在汇编的源文件所在的目录。

2) 用-i 选项指定的目录。

3) 用环境变量 C54X_A_DIR 和 A_DIR 设置的目录。

4) 用环境变量 C54X_C_DIR 和 C_DIR 设置的目录。

用户可以用-i 汇编器选项或者 C54X_A_DIR 和 A_DIR 环境变量来增强汇编器的路径搜索能力。

3.4.1　使用-i 汇编器选项

-i 汇编器选项指定一个包含 copy 和 include 文件或宏库的备用目录,其语法格式如下:

```
asm500 -i pathname source filename
```

每个-i 选项指定一个路径名,对用户指定的路径数量没有限制。在汇编源文件中可以使用.copy、.include 或.mlib 伪指令而不指定具体的路径信息。如果汇编器在当前源文件所在目录下没有找到文件,就会到-i 选项指定的各个路径下搜索。

例如,假设被调用的文件 source.asm 在当前目录,source.asm 中包含如下伪指令语句:

```
.copy "copy.asm"
```

同时假设文件存储在如下目录中:

```
对于 Windows 操作系统        c:\tools\files\copy.asm
对于 UNIX 操作系统           /tools/files/copy.asm
```

用户应该针对不同的操作系统分别输入以下命令:

操作系统	输入
Windows	asm500 -ic:\tools\files source.asm
UNIX	asm500 -i/tools/files source.asm

汇编器首先在当前目录搜索 copy.asm，因为 source.asm 在该目录。然后汇编器在 -i 选项所指定的目录中搜索。

3.4.2 使用环境变量(C54X_A_DIR 和 A_DIR)

环境变量是用户定义的一个系统符号，并赋给它一个字符串。汇编器使用环境变量 C54X_A_DIR 和 A_DIR 来命名包含 copy/include 以及宏库的备用路径。

汇编器首先查找 C54X_A_DIR 环境变量，然后读取它并执行相应操作。如果没有发现这个变量，就读取 A_DIR 环境变量并执行相应操作。如果两个变量都置位，处理器专用变量被设置。当用户同时对不同的处理器使用 TI 的开发工具时，处理器专用变量很有用。

如果汇编器找不到 C54X_A_DIR 和 A_DIR 变量，就查找 C54X_C_DIR 和 C_DIR 变量。

给环境变量赋值的命令格式如下：

操作系统	输入
Windows	set A_DIR= pathname;another pathname ...
UNIX	setenv A_DIR "pathname;another pathname ..."

其中的路径名参数 pathname 是包含 copy/include 文件或 macro 库的目录。用户可以用分号或者空格将各个路径名分开。如果汇编器在当前源文件所在目录下以及-i 选项指定的路径下没有搜索到指定的文件，就会到环境变量所指定的目录下继续搜索。

例如，假设文件 source.asm 包含如下语句：

```
.copy "copy1.asm"
.copy "copy2.asm"
```

同时假设文件存在于如下目录中：

Windows	c:\tools\files\copy1.asm
	c:\dsys\copy2.asm
UNIX	/tools/files/copy1.asm
	/dsys/copy2.asm

用户可以用下面的命令设置查找路径：

操作系统	输入
Windows	set A_DIR=c:\dsys
	asm500 -ic:\tools\files source.asm
UNIX	setenv A_DIR "/dsys"
	asm500 -i/tools/files source.asm

汇编器首先在当前目录下查找 copy1.asm 和 copy2.asm 文件，因为 source.asm 在当前目录。然后汇编器在-i 选项定义的目录下查找，并找到了 copy1.asm。最后汇编器在 A_DIR 指定的目录下可以找到 copy2.asm。

需要说明的是，环境变量的设置保持不变，除非重新启动或者输入如下命令使变量复位：