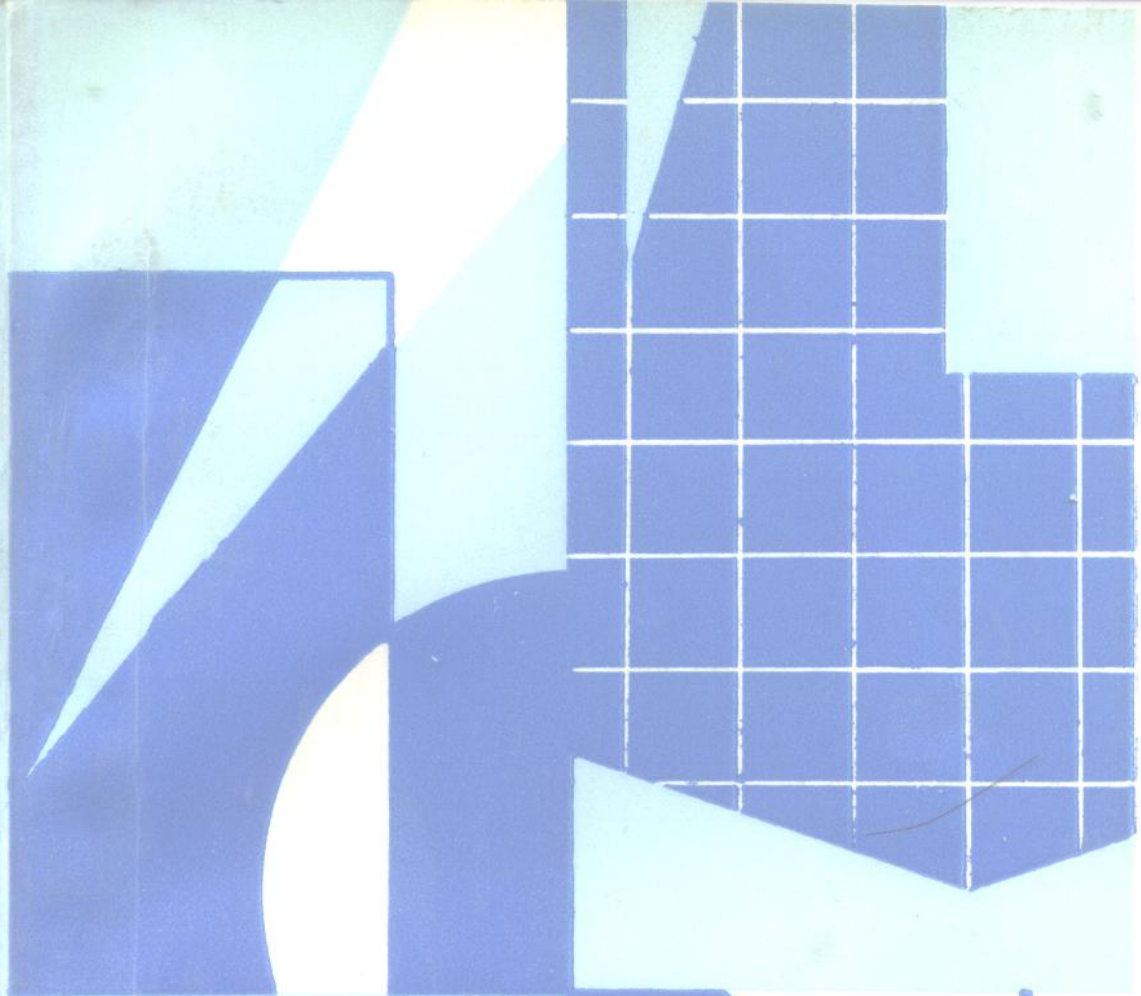


汇编语言程序设计

奚抗生　吴英泽　苗克坚　编

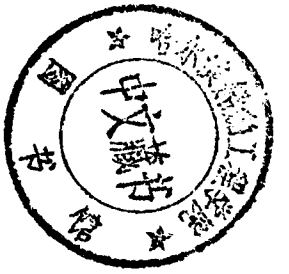
奚抗生　主编

航空工业出版社

汇编语言程序设计

汇编语言程序设计

奚抗生　主编



航空工业出版社
1994

（京）新登字 161 号

内 容 提 要

本书以 IBM PC 为背景机，系统介绍了 8086/8088 宏汇编语言程序设计的基本理论和方法。

全书共分十章，分别介绍汇编语言的基本概念，8086/8088 的硬件结构和指令系统，汇编语言的语法和程序设计的基本方法，I/O、中断和常用于程序的设计方法，汇编语言程序和高级语言程序的接口技术，以及 80286/80386 汇编语言的基本特点和功能。

本书通俗易懂、深入浅出、理论联系实际，每章包含丰富的实用程序，可作为大专院校计算机专业和自动化专业的教材，也可作为工程技术人员自学的参考书。

图书在版编目（CIP）数据

汇编语言程序设计/奚抗生，吴英泽，苗克坚编；奚抗生主编. -北京：航空工业出版社，1994. 7

ISBN 7-80046-798-8

Ⅰ. 汇… Ⅱ. 奚… Ⅲ. 微型计算机-汇编语言-程序设计 Ⅳ. TP313

中国版本图书馆 CIP 数据核字(94)第 03732 号

航空工业出版社出版发行

（北京市安定门外小关东里 14 号 100029）

南京航空航天大学印刷厂印刷 全国各地新华书店经售

1994 年 7 月第 1 版 1994 年 7 月第 1 次印刷

开本：787×1092 1/16 印张：15.75 字数：392 千字

印数：1—1900 定价：14.50 元

前　言

汇编语言和高级语言一样，都是计算机的重要编程工具。高级语言易学、易用、通用性强，受到了广大用户的欢迎；而汇编语言则是一种面向机器的语言，它与计算机的硬件紧密联系，程序中的指令与计算机的机器指令一一对应，目标代码短，执行速度快，能充分发挥计算机的硬件功能和特点，解决了有些高级语言不能解决的问题，为各种计算机应用，尤其为实时处理和过程控制，提供了良好的语言工具。

8086/8088 宏汇编语言不仅有一组功能很强的指令系统，而且像高级语言那样，对程序中的数据使用类型说明，并提供了结构化的程序设计方法，整个程序清晰易读，编程也比较方便。因此，本书以这种汇编语言为基础，全面论述了汇编语言程序设计的基本方法。有了这个基础，读者可进一步了解和掌握其它类型的汇编语言。

本书的基本构思是：首先，在第一章介绍汇编语言的基本概念，主要包括什么是汇编语言、什么是汇编程序以及如何开发汇编语言程序等等，从而使读者对汇编语言程序设计的全过程有一个初步的了解。然后，在第二章和第三章分别介绍 8086/8088 的硬件结构和指令系统，为学习汇编语言程序设计的原理和方法打下必要的基础。从第四章开始，先介绍汇编语言的语法和各种伪指令，然后在第五章进一步介绍汇编语言程序设计的四种典型方法，即顺序程序设计、分支程序设计、循环程序设计以及子程序设计。由于这四种程序设计方法均未涉及计算机与外部设备交换信息的特点，因此在第六章进一步介绍了输入输出程序设计的基本方法，并在第七章对中断程序设计方法，尤其对 8086/8088 的中断系统和功能调用，作了更深入的讨论，使读者掌握系统硬件资源和软件资源的使用方法。为了提高编程能力和程序设计技巧，第八章还介绍了各种典型的应用程序设计方法，这些应用程序都曾通过上机验证，读者可以参考使用。考虑到汇编语言和高级语言的优点和不足之处，第九章特别介绍了汇编语言和常用高级语言的接口技术，使读者在编程过程中能够充分发挥这两种语言的特点，根据任务的实际需要综合使用。最后，考虑到微型计算机的飞速发展，第十章概括介绍了 80286/80386 汇编语言的基本特点和扩充功能，为今后掌握这些内容打下一定的基础。

本书前言第一、六、七、八、九章由南京航空航天大学奚抗生执笔，第二、三章由西北工业大学苗克坚执笔，第四、五、十章由沈阳航空工业学院吴英译执笔。全书由奚抗生主编，并最后修改定稿。

本书承蒙东南大学计算机系叶建勋教授仔细审阅并提出了许多宝贵意见，南京航空航天大学林钧海教授对本书的出版也给予了热情的支持和帮助，在此表示衷心的感谢。

限于水平，书中错误和不妥之处在所难免，敬请广大读者批评指正。

编　者

1993 年 10 月

目　录

第一章 概 论

汇编语言是机器语言的符号表示，用汇编语言编写的程序叫汇编语言源程序，把汇编语言源程序翻译成机器语言目标程序的过程称为汇编。汇编工作可以采用人工查表方法，但通常是由汇编程序来完成的。

本章目的主要是介绍这些基本概念，使读者对汇编语言程序设计的全过程有一个初步的了解。

1.1 什么是汇编语言

从前修课程已经知道，计算机的基本操作是由二进制代码实现的。能完成计算机基本操作的二进制代码串称为机器指令，全部机器指令的集合构成了计算机的指令系统。由于这个指令系统能够直接被机器理解和执行，因此称它为机器语言。

机器语言是一种面向计算机的程序设计语言，不同的计算机具有不同的机器指令代码。下面是用 8086/8088 机器语言编制的计算二数之和的程序。假设被加数存放在当前数据区的 0200H 号单元，然后把它取到累加器 AL 中与立即数 12H 相加，运算结果存入 0201H 单元。为了便于阅读，分别用二进制和十六进制代码列出了它的程序清单。

二进制	十六进制	
10100000	A0	；(0200H) →AL
00000000	00	
00000010	02	
00000100	04	；(AL) +12H→AL
00010010	12	
10100010	A2	；(AL) → (0201H)
00000001	01	
00000010	02	
11110100	F4	；停机

这个程序虽然很短，但阅读和编写却不太容易，既费时间，也容易出错。为此，人们首先提出了用帮助记忆的符号指令代表机器指令。例如，用 MOV AL，DATA 代表 A00002H，表示从符号地址 DATA 取数至 AL；用 MOV SUM，AL 代表 A201H，表示把 AL 中的内容存入符号地址 SUM 中；用 ADD AL，12H 代表 0412H，表示 AL 的内容与立即数 12H 相加；用 HLT 代表 F4H，表示机器暂停执行指令。这里，MOV 是英文 MOVE 的缩写，ADD 是英文 Addition 的缩写，HLT 是英文 HALT 的缩写，AL 是累加器(Accumulator)，DATA 和 SUM 是帮助记忆的符号地址。这样，每条指令的意思就比较清楚，整个程序也容易看懂了。

但是应注意到，用这种符号指令编写的程序，计算机是不认识的，必须把它翻译成二

进制机器代码才能在计算机上运行。为此，用符号指令编写程序时，还必须提供帮助计算机翻译符号指令的定向信息，例如程序代码存放在何处、到何处存取数据、数据存放采用什么形式等等。这些帮助计算机翻译的定向信息不会像符号指令那样产生一一对映的机器指令代码，因此通常称它为伪指令。

需要指出的是，无论是符号指令或者是伪指令，为了使计算机能够理解它们，在书写的时候必须遵循事先约定的语法规则，例如引用哪些字符（即字符集）、缩写符号的组成规则、数据表示方法、语句格式和程序结构等等。

这样，借助于符号指令、伪指令以及它们的使用规则，就构成了所谓的汇编语言。下面，我们把前面用机器语言编制的二数求和程序用汇编语言重新编写如下：

```
DSEG        SEGMENT     'DATA'                  ; 数据段开始
DATA1       DB          15H
SUM         DB          0H
DSEG        ENDS                                ; 数据段结束
SSEG        SEGMENT     STACK 'STACK'           ; 堆栈段开始
            DB          20 DUP (0)
SSEG        ENDS                                ; 堆栈段结束
CSEG        SEGMENT     'CODE'                  ; 代码段开始
            ASSUME      CS : CSEG, DS : DSEG
            ASSUME      SS, SSEG
START :     MOV         AX, DSEG                ; 设置数据段基址
            MOV         DS, AX
            MOV         AL, DATA1               ; 取数到 AL
            ADD         AL, 12H                 ; 求和
            MOV         SUM, AL                 ; 保存结果至 SUM
            HLT
CSEG        ENDS                                ; 代码段结束
            END
```

这段程序虽然有些细节暂时还不能解释清楚（这些内容将在以后各章详细介绍），但是整个程序的意思是很容易看懂的。它以 SEGMNT 和 ENDS 把整个程序分成若干段，其中以 DSEG 为段名的数据段描述了程序所引用的数据内容、存放形式以及存放地址，而以 CSEG 为段名的代码段则清楚地说明了程序执行的基本操作、访问地址以及与其他段之间的联系。

因此，用汇编语言编写程序，其优点是明显的，不仅书写方便，而且容易阅读和记忆。但是正如前面所述，汇编语言只是机器语言的一种符号表示，必须经过翻译，才能在机器上运行，这就是下面需要进一步介绍的内容。

1.2 汇编程序

把汇编语言程序翻译成机器语言程序的过程称为汇编。这个汇编工作可以由人工完成，

也可以由机器完成。但人工翻译既繁琐，也容易出错，所以一般由计算机完成。

计算机完成汇编任务是由汇编程序（Assembler）这样一个软件工具实现的。如图 1-1 所示，汇编程序加工的对象是汇编语言程序，称为源程序（Source Program），而汇编后产生的结果是机器语言程序，称为目标程序（Object Program）。

常见的汇编程序有两种类型：

一、行汇编

这种方法按行对汇编语言源程序逐条汇编，常见的 IBM PC 动态调试程序 DEBUG 中的 A 命令就属于这种汇编类型。下面是这种汇编方法的基本操作过程。

```
C>DEBUG <CR>
-A<CR>
20D9:0100  MOV AL,[0200]    <CR>
20D9:0103  ADD AL,12        <CR>
20D9:0105  MOV [0201],AL    <CR>
20D9:0108  HLT              <CR>
20D9:0109  ^C
```

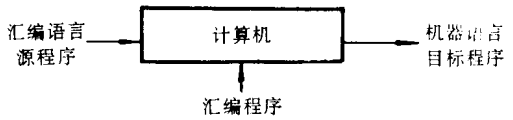


图 1-1 汇编过程示意图

上面的操作过程中，带下划线的字符由用户键入，每当输入完一条汇编指令并收到回车信号<CR>后，汇编命令自动检查语法错误。若有错误，显示“?”，要求重新输入。否则，在相应的地址空间生成机器语言代码，并进入下一行等待汇编指令输入。

汇编命令以 ^C 退出，返回 DEBUG 命令状态，重新显示提示符“—”并在用户指定的地址空间生成机器语言目标代码。若用户在 A 命令后不指明地址，则程序的起始地址就是 DEBUG 指针所指向的当前地址××××：0100，下面是在 DEBUG 环境下，用 D 命令显示的机器语言目标代码：

```
-D100 110 <CR>
20D9:0100 A0 00 02 04 12 A2 01 02-F4 00 00 00 00 00 00 00  ............
```

该目标代码程序可以在 DEBUG 环境下直接运行，也可以作为扩展名为 .COM 的文件存入磁盘，当需要时重新读入内存运行。

用这种汇编方法简单直观，目标程序占用最小的内存空间。但应注意到：(1) 这种方法是按绝对地址汇编的，地址空间不能浮动，程序长度限制在 64KB 范围内；(2) 目标程序没有与其他程序模块连接的信息。若需要连接，必须对被连接的程序各自独立汇编，然后由用户或安装程序实现各目标程序在内存中的地址空间分配，这样做可能会产生各程序间太大的空隙，造成内存空间的浪费，也可能会造成内存空间的冲突，这显然是不希望的。

二、宏汇编

这种方法首先对扩展名为 .ASM 的汇编语言源程序进行汇编，产生扩展名为 .OBJ 的可重定位目标代码文件，然后用连接程序 LINK 连接一个或多个 .OBJ 模块（包括用户库文件），生成一个扩展名为 .EXE 的可执行目标文件。图 1-2 是这种汇编方法的流程图。

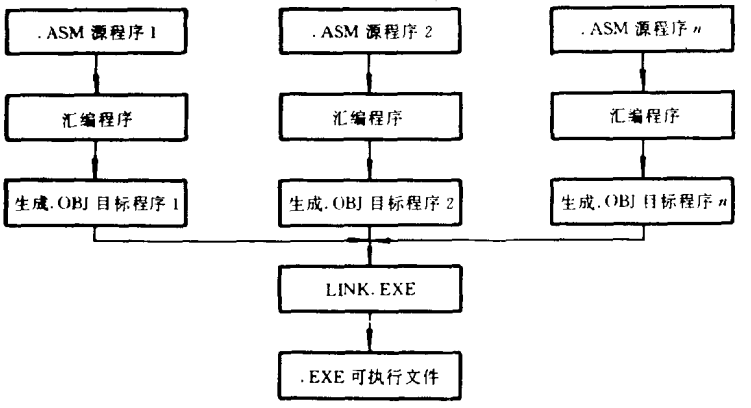


图 1-2　宏汇编流程图

早期能完成上述汇编功能的汇编程序叫ASM(通常叫小汇编程序),其汇编功能较弱。后来经过扩展,增加了宏处理和条件汇编等多种功能,并可支持 8087 协处理器的操作,程序的开发和调试手段也比较完善,这种汇编程序叫宏汇编 MASM。

目前市场上出现的宏汇编程序有多种版本,常见的有:

IBM Personal Computer Macro Assembler Version 2.00
Microsoft Macro Assembler Version 5.00
Turbo Editasm (TASMB) Version 1.03B

这种汇编方法的优点是:(1) 可连接几个程序模块,甚至是不同语言写成的模块;(2) 连接后的目标程序非常紧凑,没有间隙,节省内存空间;(3) 可利用不同的存储段实现不同的功能,适合于大型程序的汇编,程序长度不限于 64KB。

下面将进一步介绍这种宏汇编方法及开发软件的一般过程,使读者对汇编语言程序设计方法建立完整的概念。

1.3　汇编语言程序开发过程

用汇编语言开发软件的流程如图 1-3 所示,一般包括四个步骤。

一、建立源程序

建立汇编语言源程序可以使用任何以 ASCII 码格式产生文本文件的编辑程序,所编辑的文件应该没有加底线、对齐、上下标、粗体字符等控制码,因为这类信息对汇编程序是多余的。

EDLIN 是一种常用的行编辑程序,但目前比较流行的是 Wordstar, WPS, PE2 等全屏幕编辑程序,它们都可以方便地把光标设置到屏幕的任何位置进行文本编辑,包括修改、插入、删除、查找、排版等等。有关这些全屏幕编辑程序的基本操作命令将在实验中掌握、熟悉。

必须注意的是,所建立的汇编语言源程序存盘时,必须以 .ASM 作为扩展名,这样才符合宏汇编的要求。

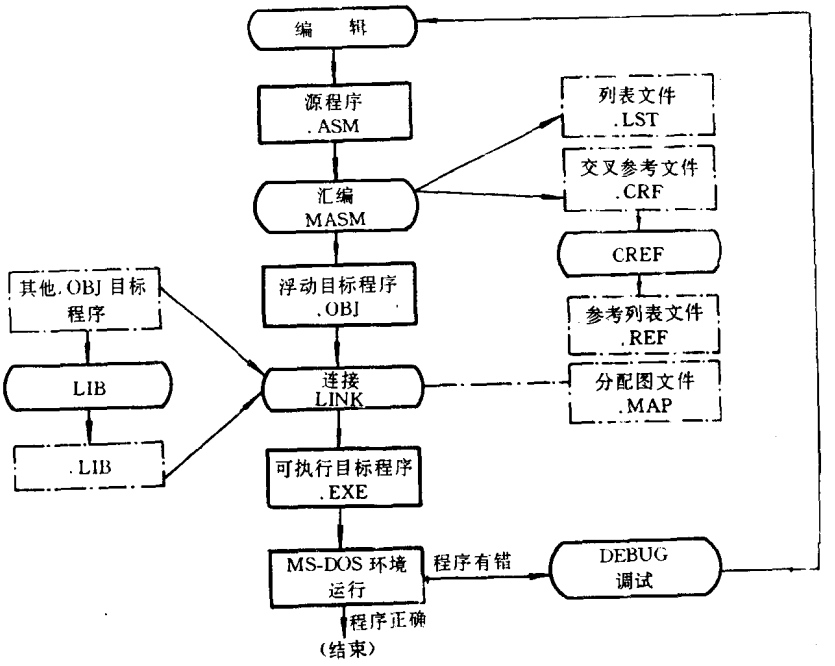


图 1-3　汇编语言程序开发过程

二、产生目标码

建立了汇编语言源程序后，就可以用宏汇编程序对它进行汇编。具体过程如下：

在 DOS 状态下，键入 MASM，调入宏汇编程序。调入以后，先显示版本号，然后依次提出四个问题，如下所示：

Source Filename [.ASM]: Sample
Object Filename [Sample .OBJ]:
Source Listing [Nul .LST]: Sample
Cross Reference [Nul .CRF]: Sample

第一行询问汇编源程序名。用户输入文件名后（例如 Sample），则显示第二个提示，询问目标程序的文件名，括号内为机器默认的文件名，通常直接回车，表示采用默认的文件名。接着出现第三个提示，问是否要建立列表文件，该文件列出了汇编出的目标代码以及与其有关的地址、源语句和符号表，可供打印输出。若需要这个文件，则输入该文件名并回车，否则直接回车。接着出现第四个提示，询问是否要建立交叉参考文件，该文件是一个中间文件，可供 CREF. EXE 程序建立一个扩展名为 .REF 的参考列表文件，以便用户了解源程序中符号和变量的引用情况。若需要则输入文件名，若不需要则直接回车。下面是生成 .REF 文件的操作步骤：

C>CREF<CR>
Gref Filename [.CRF]: Sample<CR>
List Filename [Sample .REF]: <CR>

当用户回答了第四个询问后，汇编程序就对源程序进行汇编。若汇编过程中发现源程序中有语法错误，则列出有错误的语句和错误的代码，并且指出是什么性质的错误和错误的总数。此时可分析错误，然后再调用编辑程序加以修正。修正后重新汇编，直至汇编正

确为止。

三、连接

所谓连接是用连接程序 LINK .EXE 把若干个经汇编产生的目标文件及指定的库文件连接起来，产生可执行的 .EXE 文件。具体连接过程如下。

在 DOS 状态下，键入 LINK，调入连接程序。调入以后，先显示版本号，然后依次提出四个问题，如下所示：

```
Object Moedules  [.OBJ]: Sample
Run File         [Sample.EXE]:
List File        [Nul.MAP]: Sample
Libraries        [.LIB]:
```

第一行询问要连接的目标文件名，用户输入文件名作为回答。如果有多个要连接的目标文件，应一次输入，各目标文件名之间用“+”号相连。第二个提示询问要产生的可执行文件名，一般直接回车就采用了括号内默认的文件名。第三个提示询问是否建立内存分配图文件，该文件包含了源程序中每一个段的列表，并显示出在运行文件中的偏移地址，供用户调试程序时使用。若需要这个文件，则输入文件名再回车，否则直接回车。最后提示是否用到库文件，要求用户输入源程序访问的库文件名。所谓库文件，就是若干可重定位的目标代码模块的集合，它可以由库管理程序 LIB.EXE 建立和修改。若没有库文件，则直接输入回车即可。

程序在连接过程中若出现错误，则显示有关错误信息，例如：

```
Warning: No Stack Segment
There was 1 error detected
```

有错误就要重新调用编辑程序进行修改，然后重新汇编，再经过连接，直至没有错误为止。

四、运行和调试

经过上述汇编、连接后产生的 .EXE 可执行文件可以在 DOS 状态下直接输入文件名运行该程序。如果出现错误，可用动态调试程序 DEBUG 进行调试。找出错误后，再重复上述过程，直到程序能够正确运行为止。

下面列出了前面计算二数之和的源程序经汇编、连接后所生成的列表文件、参考列表文件，以及内存分配图文件清单，供参考。

(1) 列表文件 (SAMPLE .LST)

```
Microsoft (R) Macro Assembler Version 5.10                  10/7/93 11:48:49
                                                            Page 1-1

 1  0000                    DSEG    SEGMENT 'DATA'
 2  0000 15                 DATA1   DB    15H
 3  0001 00                 SUM     DB    0H
 4  0002                    DSEG    ENDS
 5
 6  0000                    SSEG    SEGMENT STACK 'STACK'
 7  0000 0014 [                     DB  20    DUP (0)
 6
```

```
 8       00
 9                ]
10
11  0014                SSEG    ENDS
12
13  0000                CSEG    SEGMENT 'CODE'
14                              ASSUME  CS : CSEG, DS : DSEG
15                              ASSUME  SS : SSEG
16  0000 B8 —— R        START:  MOV     AX, DSEG
17  0003 8E D8                  MOV     DS, AX
18  0005 A0 0000 R              MOV     AL, DATA1
19  0008 04 12                  ADD     AL, 12H
20  000A A2 0001 R              MOV     SUM, AL
21  000D F4                     HLT
22  000E                CSEG    ENDS
23                              END
```

Microsoft (R) Macro Assembler Version 5. 10 10/7/93 11 : 48 : 49
Symbols-1

Segments and Groups:

Name	Length	Align	Combine	class
CSEG.....	000E	PARA	NONE	'CODE'
DSEG.....	0002	PARA	NONE	'DATA'
SSEG.....	0014	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr
DATA1.....	L BYTE	0000	DSEG
START.....	L NEAR	0000	CSEG
SUM.....	L BYTE	0001	DSEG
@CPU.....	TEXT	0101h	
@FILENAME.....	TEXT	SAMPLE	
@VERSION.....	TEXT	510	

20 Source Lines
20 Total Lines
14 Symbols
47584+415147 Bytes symbol space free
0 Warning Errors
0 severe Errors

(2) 参考列表文件 (SAMPLE .REF)

Microsoft Cross-Reference Version 5. 10 Thu Oct 07 11 : 49 : 10 1993

Symbol Cross-Reference (#definition, +modification) Cref-1

@CPU..... 1#
@VERSION..... 1#