Turbo Assembler（汇编）

用户手册

基础实用篇

丛海莱　编译

北京联想计算机集团公司

一九九〇年八月

Turbo Assembler(汇编)用户手册

——基础实用篇

丛海莱　编译

北京联想计算机集团公司

一九九0年八月

前　言

Turbo Assembler 是美国 Borland 公司推出的能充分发挥 80×86 系列芯片的汇编语言软件包，在系统的低级控制、实时处理、硬件接口等方面，汇编语言都是把高级语言和各种硬件结构结合起来的工具，它是软件开发人员应该掌握的最有效的计算机语言。

汇编语言的最主要的优点在于：一、能以极快的速度执行；二、能够与操作系统接口，直接控制显示器、打印机和重要的磁盘存储操作。Turbo Assembler 比起其它汇编语言来，更是胜上一筹。

美 Borland 公司副总裁 Dickerson 先生说：Turbo Assembler 为汇编语言的学习者或程序员提供了一个更好的环境，是学习和设计汇编程序的最好软件包。

若过去使用 Microsoft Assembler 的程序员，现在想改学 Turbo Assembler，不需要任何修改即可完全兼容。

Turbo Assembler 用户指南全套资料共为三册：《Turbo Assembler(汇编)用户手册——基础实用篇》、《Turbo Assembler(汇编)用户手册——高级技术篇》、《Turbo Assembler(汇编)参考手册》。全书全面系统地介绍 Turbo Assembler 软件包的安装、使用，汇编语言的基础知识，初级、中级和高级程序设计技术，Turbo Assembler 与其它语言的接口等内容，并按功能依字母顺序列出 Turbo Assembler 中的各种预定义符、运算符和伪指令，是一套学习并提高汇编语言程序设计技巧的必备参考书。

本书由编译者在长期从事汇编语言程序设计的基础上，根据美国 Borland 公司《Turbo Assembler User's Guide》和《Turbo Assembler Reference Guide》，经过认真编译、整理而成，由于时间伧促，加上编译者水平有限，书中错误之处在所难免，敬请广大读者指正。

本书在编译过程中，得到了许多同志的热情帮助和支持，编译者在此表示衷心的感谢！

编译者

一九九0年八月

目　录

第一章　开始

在我们帮助你提高汇编语言的程序设计能力以前，你必须做一件额外的事情，那就是把你的 Turbo Assembler 盘片各做一份备份，以供操作练习之用，然后将源盘放到安全的地方。

如果你想用 Turbo Assembler 来取代 MASM，请阅读附录 B，看看它们之间有何不同。

§1.1　磁盘上的文件

* RASM.EXE: Turbo 汇编程序
* TLINK.EXE: Turbo 连接程序
* MAKE.EXE: 命令行 MAKE 实用程序
* README.COM: 用来显示 README 文件的程序
* README: 有关本软件及文件说明最新资料
* TCREF.EXE: 原始文件调试信息实用程序
* OBJXREF.COM: 目标程序交叉参考实用程序
* GREP.COM: 文件搜索实用程序
* TOUCH.COM: 用来处理文件更新的实用程序

§1.2　安装 Turbo Assembler

使用 Turbo Assembler 是十分方便的。如果你的系统有硬盘，请为 TASM.EXE 建一个目录（因为你会常常用到它），然后将 TASM.EXE 拷贝到该目录下。如果没有硬盘，将可将 TASM.EXE 放在磁盘上。

做完上述工作后，你可把所想使用的实用程序也拷贝到该目录下，这样就完成了安装动作，下一章我们会教你利用 TASM 来做程序设计的一些基本步骤。

第二章　利用 Turbo Assembler 进行程序设计

如果你从未编过汇编语言程序，现在正是学习的大好时机，你也许听说过汇编语言是一门只适合那些行家和高手的高级艺术，但可千万别相信，其实汇编语言只不过是一种人性化的机器语言，同时可以具有你所希望的高级逻辑。此外，它还有强大的功能（事实上，汇编语言是唯一能充分发挥 8086 家族功能的程序语言。注：8086 家族处理器为 IBM PC 系列及其兼容机心脏）。

你可以将整个程序都用汇编语言来写，或把它与 Turbo C、Turbo、Pascol、Turbo Prolog、Turbo Basic 或其它语言混合编写，不论用那种方式，都可利用汇编语言写出精短的程序。

在本章中，我们将为你介绍汇编语言，并带你欣赏其独特的魅力，首先，你会看到并执行几个汇编语言程序，以便对它产生感觉并习惯使用它，接着，我们会告诉你一般计算机的基本结构以及 8086 处理器，好让你了解 8086 汇编语言的特有功能。此外，我们还会讨论有关 IBM PC 汇编语言程序设计的主题。

第四章“汇编语言程序的基本结构”可弥补本章的不足，它介绍了一个汇编语言程序的结构和基本要素，并且用一个具体而微的程序对前两章所学的概念作一个总结。

第五章“再谈 Turbo Assembler 程序设计”将继续探讨汇编语言的程序设计问题，而第十章“Turbo Assembler 的高级程序设计”会再介绍内存模式，宏指令和其他高级主题。

当然，我们无法凭几章的课程就使你成为汇编语言的程序设计高手。我们只不过为你介绍了汇编语言，好让你可以动手编写程序，我们郑重地建议你去选择一本有关汇编语言和 PC 机结构的专业书来做参考，除此之外，你还可以去找“IBM DOS 技术参考”、“BIOS 接口技术参考”和“PC XT 技术参考”等手册来参考，这些手册皆对有关汇编语言中关于 IBM PC 软硬件系统中的方面作了说明。

在你阅读其他专业书之前，你必须先看第三章“命令行参数”，以便熟悉命令行选择项。如果你尚未完成第一章所提的“开始”工作，也请你立即照办。

最后一点：汇编语言是一个复杂的课题，即使你只想编写一个相当简单的程序，也得知道许多事情。因此，我们有时会在未来的课程中为你说明每个概念。如果你任何时候对某个概念特别感兴趣，就请立即查阅“参考手册”的第三章“伪指令”。

如果你已完成“开始”工作并且把“参考手册”的第三章放在手边，那么即可开始建立第一个汇编语言程序。

§2.1　编写你的第一个 Turbo Assembler 程序

在程序设计世界里，处女作往往是一个用来显示“Hello, World”信息的程序，因为它对于初学者而言是一个良好的起步，现在进入你所熟悉的编辑程序（其输出必须是 ASCII 文件），然后键入下列内容：

```
DOSSEG
.MODEL SMALL
.STACK 100h
.DATA
```

```
Hello Message   DB"Hello, World",13,10,"$"
    .CODE
    mov ax, @DATA
    mov ds, ax                      ;初始DS指数据段
    mov ah, 9                       ;DOS功能调用号
    mov dx, OFFSETHelloMessage      ;指向"Hello, World"
    int 21h                         ;显示"Hello, World"DOS功能调用
    mov ah, 4ch                     ;DOS返回调用号
    int 21h                         ;DOS返回DOS
    END
```

当你键完后，请以 hello.asm 为名存到盘上。

如果你已熟悉 C 或 Pascal，也许会觉得上面的程序似乎长了一点，没错，汇编语言程序的确有长的倾向，因为每个汇编语言指令所能做的事情较 C 或 Pascal 指令要少，不过在另一方面，你可以依自己希望的方式来使用汇编语言指令而享有充分的自由，也就是说，汇编语言并不像 C 或 Pascal，它使你能够用程序来命令计算机做其所能胜任的工作——但是你得多键入几行程序。

§2.2 汇编你的第一个程序

即然你已存好 Hello.ASM，你一定想要执行它。在执行此程序之前，必须将其转换成可执行文件，这需要经过两个步骤，那就是汇编和连接。如图 2.1，其描述了一个程序的发展过程——包括编辑、汇编、连接和执行。

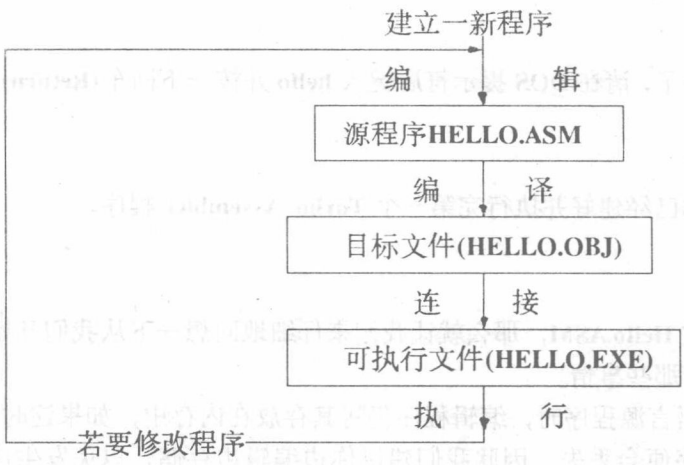

图 2.1 从编辑、编译、连接到执行的一个循环

汇编步骤是将原始程序转换成一种中介形式，称为目标模块。而连接步骤将组合一个或多个模块成为一可执行程序。你可以在命令行上进行汇编和连接工作。

为了汇编 Hello.ASM，请键入：

```
TASM hello
```

除非指明别的文件名，否则 Hello.ASM 将被译成 HELLO.OBJ（请注意不需要键入文

件扩展名，Turbo Assembler 会假定其为.ASM）。你会在屏幕上看到如下信息：

Turbo Assembler Version 1.0 Copyrinht(C)1988 by Borland International, Inc.
Assembling file:HELLO.ASM
Error messages:None
Warning message: None
Remaining mesmory:266K

如果如上键入 HELLO.ASM，则不会收到任何警告或错误信息。如果有任何警告和错误，它们都将出现在屏幕上，并且伴随着它们所在行的行号。如果收到错误，请检查你的程序并确认其是否与书上所列的完全一样，然后再汇编一次。

§2.3 连接你的第一个程序

在成功地汇编了 HELLO.ASM 之后，只差一步便可执行第一个可执行汇编程序了，这就是把刚才编译得到的目标代码连接成可执行程序，为此可使用 TLINK，请键入：

```
TLINK hello
```

同样，不用键入文件扩展名，TLINK 会假定其为.OBJ。连接完毕（最多花几秒钟），连接程序会自动产生和目标文件同名的.EXE 文件，除非你特别指定别的名字。当连接成功之后，下面的信息会出现在屏幕上：

Turbo Linber Version 2.0 copyright(C)1987,1988 by Borland Internaitonal,Ine.

在连接过程中也可能有错误产生，不过对我们这个示例而言是没有的。如果你确实收到任何连接错误(它们会出现在屏幕上)，请修改你的程序以使和书中所学的相同，然后再次汇编和连接。

§2.4 执行你的第一个程序

现在，可以执行你的程序了，请在 DOS 提示符后键入 hello 并按一下回车(Return)键，则下列信息：

```
Hello, world
```

将显示在屏幕上。到这里，你已经建好并执行完第一个 Turbo Assembler 程序。

§2.5 发生了那些事？

既然你已经建好并执行过 Hello.ASM，那么就让我们来仔细地回想一下从我们开始编辑程序到执行完毕到底发生了那些事情。

当你着手编辑你的汇编语言源程序时，编辑程序仍将其存放在内存中，如果这时 PC 机电源突然中断，那么源程序便会丢失。因此我们建议你边编辑边存储，以免发生这样的悲剧。当你将源程序存放在磁盘之后，一份程序的永久拷贝便放在 HELLO.ASM 文件内，所以即使电源突然关闭也无所谓。HELLO.ASM 是一个标准的 ASCII 文本文件，你可以键入下列命令而将其显示在屏幕上。

```
type hello.asm
```

当你汇编 HELLO.ASM 时，Turbo Assembler 会把 HELLO.ASM 内的指令转换成同等的二进制形式而存放在目标文件 HELLO.OBJ 中，HELLO.OBJ 是一个处于源文件和可执行文件间的中间文件，虽然它所包含的数据是用来将 HELLO.ASM 中的指令转化成可执行

码，但它却是从准备和其目标文件结合成单一程序的形式存在的。在第五章“再谈 Turbo Assembler 程序设计”中，你会看到当开发大型程序时它是多么有用。

再来，当你连接 HELLO.OBJ 时，TLINK 会将其转换成可执行文件 HELLO.EXE。最后在提示符下键入 hello 后，就可以执行 HELLO.EXE 了。

现在。请你再键入：

```
dir hello.*
```

从列出磁盘上不同的 HELLO 文件，你将会看到 HELLO.ASM，HELLO.OBJ，HELLO.EXE 和 HELLO.MAP。

§2.6 修改你的第一个 Turbo Assembler 程序

现在请你再进入编辑程序中去修改程序，使其能从键盘上接受一点输入。下列所示即为将修改成的程序清单：

```
    DOSSEG
    .MODE SMALL
    .STACK 100h
    .DATA
TimePrompt DB 'Is it after 12 noonn(Y/N)?$'
Good Moring Message LABEL BYTE
        DB 13,10,'Good afternoon,world!',13,10,'$'
    .CODE
    mov ax,@Data
    mov ds,ax                       ;初始化数据段
    mov dx,OFFSET TimePrompt        ;时间提示字符串
    mov ah,9                        ;DOS打印字符串功能号
    int21h                          ;调用DOS功能显字
    mov ah,1                        ;DOS取字节功能号
    int21h                          ;读键
    Cmp al,'y'                      ;比较读入的字母"y"
    jz IsAfternoon                  ;过了中午跳转
    Cmp al,'Y'                      ;比较大写字母"Y"
    jnz IsMorning                   ;还没过中午跳转
IsAfternoon:
    mov dx,OFFSET Good Afternoon Message  ;指向下午的显示信息
    jmp DisplayGreeting
IsMorning:
    mov dx,OFFSET GoodMorningMessage  ;指向正午前的显示信息
DisplayGreeting:
    mov ah,9            ;DOS打印字符串功能号
    int 21h              ;显示字符串
    mov ah,4ch          ;DOS返回功能号
```

```
    int 21h
    END
```

此刻你已经为源程序添了两项重要的新功能：输入和做决策。本程序先问你现在是不是午后，然后从键盘上接受一个字节，如果键入为大写或小写的“Y”，则程序会显示一适当的下午问候语，否则它会问你早安。一个有用程序的必备要素——由外界来的输入、数据处理、决策和送到外界的输出皆出现在本程序中了。

请将此修改过的程序存到磁盘上（这样会使新程序取代原程序），然后将其重新汇编并重新连接，你在 DOS 提示符后再度键入 hello 令程序执行后，下列信息：

```
Is it after12 noon (Y/N)?
```

会显示出来，其后跟着一闪烁的光标在等待你的回答。请按下 Y，而程序会显示：

```
Good afternoon,world!
```

作为回答，HELLO.ASM 现在变成一个具有交互性可做决策的程序了。

在学习汇编语言的过程中，你必然会在键入和程序语法上犯下各种错误。Turbo Assembler 会在汇编程序时为你找出许多错误并报告出来，所报告的错误可分为两类：警告和错误。如果 Turbo Assembler 发现某种可疑的现象（并不一定是错的）时，它会显示警告信息。但如果 Turbo Assembler 遇到确切的错误时，它会显示错误信息并停止编译，而不产生目标文件。

换句话说，警告只是示警或非致命的，不过在你可以运行程序前必须将所有的错误修改好。Turbo Assembler 可能产生的各种错误和警告信息目录在参考手册的附录 E 中。

如同其他程序语言一样，Turbo Assembler 并不能为你找出逻辑上的错误，Turbo Assembler 可以告诉你程序是否通过汇编，但却不能告诉你程序是否能如你希望的运行，这件事只有靠你自己判断了。

如果这个示例目前对你而言没有多大意义，则请别着急，即使是熟悉其他语言的程序设计员也得花些时间才能活用汇编语言。现在，你应该对汇编语言程序有些认识了。待会在本章和第四章里，我们将为你介绍汇编语言的各个基本要素。

若想把你的程序送到打印机，请参阅你的文本编辑程序手册。Turbo Assembler 源文件为标准 ASCII 文本文件，所以你可从在 DOS 提示符后键入 PRINT 命令来打印任一源文件。

§2.6.1 输出到打印机

打印机是一个放在手边的输出设备，你不仅有时需将程序文件送往打印机打印，而且偶尔也会想使程序送一些输出给它。底下是"Hello,world"程序的另一个版本，它将在打印机上输出它的结果（原来是在屏幕上）！

```
    DOSSEG
    .MODEL SMALL
    .STACK 100h
    .DATA
HelloMessage    DB 'Hello,world',13,10,12
HELLO_MESSAGE_LENGTH EQU $-HelloMessage
    .CODE
```

```
    mov ax,@Data
    mov ds,ax                ;初始数据段
    mov ah,40h               ;DOS打印功能号
    mov bx,4                 ;打印机代码
    mov cx, HELLO_MESSAGE_LENGTH;要打印的字节数
    mov dx, OFFSET HelloMessage    ;要打印的字符串
    int 21h                  ;打印
    mov ah, 4ch
    int 21h
    END
```

在这个 HELLO 程序中，你已经用一个送字符串到选择的设备和文件的 DOS 功能调用取代了原来送字符串到屏幕的 DOS 功能调用。在本例里，我们选择的设备为打印机。请建立并运行此程序，你将会发现“Hello,world”被印在一张打印纸上。你可以轻易地将程序修改成将输出送到屏幕而非打印机，这只须将：

```
    mov bx,4  ;Printer handle
```

更换成

```
    mov bx,1  ;standard output handle
```

即可。修改完之后，请于运行前再汇编和连接一次。你会注意到当输出到屏幕上时，最后显示的字符为“♀”。实际上，它是一个控制字符，是程序用来强迫打印机换页用的，因为屏幕不是打印机，它不知道“♀”是干什么的，只能将其对应字符显示出来。

§2.7 编写你的第二个 Turbo Assembler 程序

现在你可以着手建立并运行一个的确“能够做某些事”的程序了。进入你的编辑程序，将下面的程序建好，命令为 REVERSE.ASM。

```
    DOSSEG
    .MODEL SMALL
    .STACK 100h
    .DATA
  MAXIMUM_STRING_LENGTH EQU1000
  StringToReverse DB MAXIMUM_STRING_LENGTH DUP (?)
  ReverseString   DB MAXIMUM_STRING_LENGTH DUP (?)
    .CODE
    mov ax,@Data
    mov ds,ax
    mov ah,3fh               ;DOS读字符串功能号
    mov bx,0                 ;标准输入代码
    mov cx,MAXIMVM_STRING_LENGTH;最大读取字符数
    mov dx,OFFSET StringToReverse   ;把字符串存入变量中
    int 21h                  ;读字符串
    and ax,ax                ;有读入的字符吗?
```

```
    jz Done                              ;没有，转到Done
    mov cx,ax          ;将字符串长度送入CX作为计数器
    push cx                ;将字符串长度送入堆栈
    mov bx,OFFSET StringToReverse
    mov si,OFFSET ReverseString
    add si,cx
    dec si                       ;指向返回字符串缓冲区的尾部
  Reverseloop:
    mov al,[bx]                  ;取下一个字符
    mov [si],al                  ;从反向次序存储
    inc bx                       ;指向下一个字符
    dec si                       ;指向反向缓冲区前一个位置
    loop ReverseLoop             ;循环
    pop cx                       ;取回字符串长度
    mov ah,40h
    mov bx,1
    mov dx,OFFSET ReverseString      ;指向反向字符串
    int21h                           ;打印
Done:mov ah,4ch
    int21h
    END
```

你待会就会知道程序到底在做些什么。但首先你得照例将其存储起来。

§2.7.1 运行 REVERSE.ASM

若想运行 REVERSE.ASM，你必须先汇编，键入：

```
TASM reverse
```

以产生可执行文件。

在提示符后键入 reverse 运行你的程序。如果 Turbo Assevbler 告知任何错误或警告，则请你仔细地检查所建的程序与本文所示有何不同。修改完后，再试着运行看看。

当你运行程序后，光标会在屏幕闪烁。显然，程序在等待你键入一些东西，请试着键入：

```
A B C D E F G
```

然后按下回车键，程序会显示

```
G F E D C B A
```

然后结束。请在命令行再键入 rerense 令程序运行，这次你键入：

```
0 1 2 3 4 5 6 7 8 9
```

再按回车键，程序会显示

```
9 8 7 6 5 4 3 2 1 0
```

现在你该很清楚 REVERSE.ASM 在做什么了。它可以将你所键入的任何字符串颠倒过来。对字符及字符串快速处理是汇编语言胜过其它语言的优点之一，至于其他优点我们

会在往后几章为你介绍。

恭喜你！你已经运行过几个汇编语言程序了，并且也认识了汇编语言程序设计的基本要素——输入、处理和输出。

如果你不想要目标文件但需要列表文件，或者你想要调试信息文件但不需列表文件，那么你可以指明空设备 NUL 作为文件名，举例来说：

```
TASM FILE1, NUL,
```

将 FILE1.ASM 汇编成目标文件 FILE1.OBJ，但并不产生列表文件。不过它会产生一交叉参考文件 FILE1.XRF。

现在你准备开始学习汇编语言程序设计的基本要素。我们将告诉你究竟是什么让汇编语言在计算机世界里是独一无二的。

§2.8 计算机的结构

早先我们曾说汇编语言是计算机的专用语言。为了使你明白这句话的含义，你得先知道计算机是什么。

从深一层来看，计算机只不过是一个单纯的设备。它可以把数据从一处转移到另一处，有时会用各种逻辑和算术方法转换数据。就我们的目的而言，将计算机视为一个含有五个功能子系统的系统较为妥当。此五个子系统为输入、控制、算术和逻辑运算、存储和输出。请参看图 2.2。

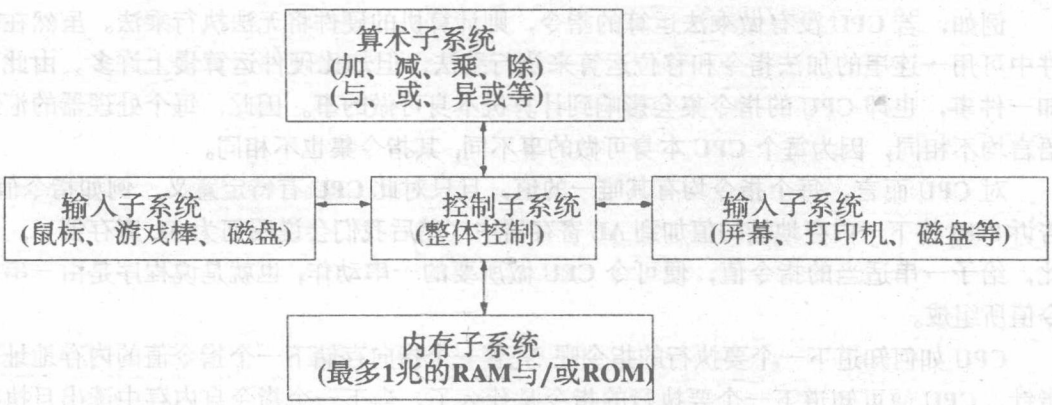

图 2.2 五个子系统

计算机算术子系统为大多数人所想像的计算机面貌。但事实上，大多数计算机只花很少的时间在处理数字运算上，而把大部分时间用在处理字符、字符串以及输入输出。那么可能会问：一部文字处理器需要算术功能作什么？告诉你，算术功能仍然是十分重要的，因为它不仅负责加减乘除，还负责逻辑运算（象 and,or 和 xor 等）。

现在你知道计算机执行算术运算没有问题了，但运算符的值从何处来呢？别担心，计算机的内存子系统就是担负这个任务的。它提供立即存取的内存供数以千计的字符和数字使用。计算机还有软盘和硬盘让我们存放庞大的数据，只不过它们是属于输入／输出设备而非内存子系统的一部分。