第 3 章　汇编语言和汇编软件

处理器依靠机器指令工作，但机器指令从形式上看都是一些没有规律的数字，难以书写、阅读和理解，这样就发明了汇编语言。本章的目标是：

1．了解汇编语言的作用和“汇编”一词的由来。

2．下载 NASM 编译器，并学会使用它来编译汇编语言源程序。

3.1　汇编语言简介

在前面的章节里，我们讲到了处理器，也讲了处理器是如何进行算术逻辑运算的。为了实现自动计算，处理器必须从内存中取得指令，并执行这些指令。

指令和被指令引用的数据在内存中都是一些或高或低的电平，每一个电平都可以看成是一个二进制位（0 或者 1），8 个二进制位形成一字节。

要解读内存中的东西，最好的办法就是将它们按字节转换成数字的形式。比如，下面这些数字就是存放在内存中的 INTEL8086 指令，我们用的是十六进制：

```
B8  3F  00  01  C3  01  C1
```

对于大多数人来说，他们很难想象上面那一排数字对应着下面几条 8086 指令：

```
将立即数 003FH 传送到寄存器 AX；
将寄存器 BX 的内容和寄存器 AX 的内容相加，结果在 BX 中；
将寄存器 CX 的内容和寄存器 AX 的内容相加，结果在 CX 中。
```

即使是很有经验的技术人员，要想用这种方式来编写指令，也是很困难的，而且很容易出错。所以，在第一个处理器诞生之后不久，如何使指令的编写变得更容易，就提上了日程。

为了克服机器指令难以书写和理解的缺点，人们想到可以用一些容易理解和记忆的符号，也就是助记符，来描述指令的功能和操作数的类型，这就产生了汇编语言（Assembly Language）。这样，上面那些指令就可以写成：

```
mov ax,3FH
add bx,ax
add cx,ax
```

对于那些有点英语基础的人来说，理解这些汇编语言指令并不困难。比如这句

```
mov ax,3FH
```

首先，mov 是 move 的简化形式，意思是“移动”或者“传送”。至于“ax”，很明显，指的就是 AX 寄存器。传送指令需要两个操作数，分别是目的操作数和源操作数，它们之间要用逗号隔开。在这里，AX 是目的操作数，源操作数是 3FH。汇编语言对指令的大小写没有特别的要求。所以你完全可以这样写：

```
MOV AX,3FH
```

```
mov ax,3fh
MOV ax,3FH
mov AX,3fh
```

在很多高级语言中，如果要指示一个数是十六进制数，通常不采用在后面加“H”的做法，而是为它添加一个“0x”前缀。像这样：

```
mov ax,0x3f
```

你可能想问一下，为什么会是这样，为什么会是“0x”？答案是不知道，不知道在什么时候，为什么就这样用了。这不得不让人怀疑，它肯定是一个非常随意的决定，并在以后形成了惯例。如果你知道确切的答案，不妨写封电子邮件告诉我。注意，为了方便，我们将在本书中采用这种形式。

在汇编语言中，使用十进制数是最自然的。因为 3FH 等于十进制数 63，所以你可以直接这样写：

```
mov ax,63
```

当然，如果你喜欢，也可以使用二进制数来这样写：

```
mov ax,00111111B
```

一定要看清楚，在那串“0”和“1”的组合后面，跟着字母“B”，以表明它是一个二进制数。

至于这句：

```
add bx,ax
```

情况也是一样。add 的意思是把一个数和另一个数相加。在这里，是把 BX 寄存器的内容和 AX 寄存器的内容相加。相加的结果在 BX 中，但 AX 的内容并不改变。

像上面那样，用汇编语言提供的符号书写的文本，叫做汇编语言源程序。为此，你需要一个字处理器软件，比如 Windows 记事本，来编辑这些内容。如图 3-1 所示，相信这些软件的使用都是你已经非常熟悉的。

图 3-1　用 Windows 记事本来书写汇编语言源程序

有了汇编语言所提供的符号，这只是方便了你自己。相反地，对人类来说通俗易懂的东西，处理器是无法识别的。所以，还需要将汇编语言源程序转换成机器指令，这个过程叫做编译（Compile）。在编译的时候，汇编语言编译器的作用是将 mov、add、ax、bx 等这些符号组合起来，转换成类似于数值的机器指令，这个过程叫做汇编，这就是汇编语言的由来，也有人称之为组合语言。

编译肯定还需要依靠一个软件，称为编译器，或编译软件。因为如果需要人类自己去做，还费这周折干嘛。另一方面，想想看，一个帮助人类生产软件的工具，自己居然也是一个软件，这很有意思。

从字处理器软件生成的是汇编语言源程序文件。编译软件的任务是读取这些文件，将那些符号转变成二进制形式的机器指令代码。它把这些机器代码存放到另一个文件中，叫做二进制文件或者可执行文件，比如 Windows 里以“.exe”为扩展名的文件，就是可执行文件。当需要用处理器执行的时候，再加载到内存里。

3.2 NASM 编译器

3.2.1 NASM 的下载和安装

每种处理器都可能会有自己的汇编语言编译器，而对于同一款处理器来说，针对不同的平台（比如 Windows 和 Linux），也会有不同版本的汇编语言编译器。

现存的汇编语言编译器有多种，用得比较多的有 MASM、FASM、TASM、AS86、GASM 等，每种汇编器都有自己的特色和局限性。特别是，有些还需要付费才能使用。不同于前面所列举的这些，在本书中，我们用的是另一款叫做 NASM 的汇编语言编译器。

NASM 的全称是 Netwide Assembler，它是可免费使用的开源软件。下面是它的下载地址：

```
http://sourceforge.net/projects/nasm/files/
```

通过以上地址可以找到所有平台上的 NASM 版本，比如为 16 位和 32 位 DOS、LINUX、OS/2 等操作系统开发的版本。因为本书的读者一般在 Windows 平台上工作，所以应当使用下面的链接来直接定位到 Windows 平台上的 NASM 版本：

```
http://sourceforge.net/projects/nasm/files/Win32%20binaries/
```

通过以上链接，可以显示所有 Windows 平台上的 NASM 版本，应当选择最新版本下载。这本书出版的时候，最新的 NASM 版本是 2.07。

本书不配光盘，所以书中的所有源代码连同我自己写的小工具都只有通过网络下载方能使用。这是一个压缩文件，名字叫 booktool.zip，可以通过下面这个链接来下载它：

```
http://ishare.iask.sina.com.cn/f/34697012.html
```

如果此链接不可用，可以到电子工业出版社的网站上下载，或者直接给我写信，我会以最快的时间给予支持。

3.2.2 代码的书写和编译过程

和你已经司空见惯的其他 Windows 应用程序不同，NASM 在运行之后并不会显示一个图形用户界面。相反地，它只能通过命令行使用。

比如，我们可以用 Windows 记事本编写一个汇编语言源程序，并把它保存到 NASM 工作目录下（就是在前面安装 NASM 时所用的安装文件夹），文件名为 exam.asm。作为惯例，汇编语言源程序文件的扩展名是“.asm”，不过，你当然可以使用其他扩展名。

一旦有了一个源程序，下一步就是将它的内容编译成机器代码。为此，可以从 Windows 开始菜单里找到“Netwide Assembler xxx”，其中的“xxx”取决于你安装的 NASM 版本。然后，选择其下的“Nasm Shell”，这将打开一个命令行窗口。

接着，在命令行提示符后输入“nasm –f bin exam.asm –o exam.bin”并按 Enter 键，如图 3-2 所示。

图 3-2　在命令行方式下使用 NASM 编译一个汇编源程序

NASM 需要一系列参数才能正常工作。-f 参数的作用是指定输出文件的格式（Format）。这样，-f bin 就是要求 NASM 生成的文件只包含“纯二进制”的内容。换句话说，除了处理器能够识别的机器代码外，别的任何东西都不包含。这样一来，因为缺少操作系统所需要的加载和重定位信息，它就很难在 Windows、DOS 和 Linux 上作为一个普通的应用程序运行。不过，这正是本书所需要的。

紧接着，exam.asm 是源程序的文件名，它是将要被编译的对象。

-o 参数指定编译后输出（Output）的文件名。在这里，我们要求 NASM 生成输出文件 exam.bin。

用来编写汇编语言源程序，Windows 记事本并不是一个好工具。同时，在命令行编译源程序也令很多人迷糊。毕竟，很多年轻的朋友都是用着 Windows 成长起来的，他们缺少在 DOS 和 UNIX 下工作的经历。

为了写这本书，我一直想找一个自己中意的汇编语言编辑软件。互联网是个大宝库，上面有很多这样的工具软件，但大多都包含了太多的功能，用起来自然也很复杂。我的愿望很简单，能够方便地书写汇编指令即可，同时还具有编译功能。毕竟我自己也不喜欢在命令行和图形用户界面之间来回切换。

在经历了一系列的失望之后，我决定自己写一个，于是就有了 Nasmide 这个小程序，它同样位于配书文件包中。不过遗憾的是，这个小程序却并非是用汇编语言书写的。

现在，你可以双击 nasmide.exe 来运行它。启动之后，如图 3-3 所示，Nasmide 的软件界面分为三个部分。顶端是菜单，可以用来新建文件、打开文件、保存文件或者调用 NASM 来编译当前文档。

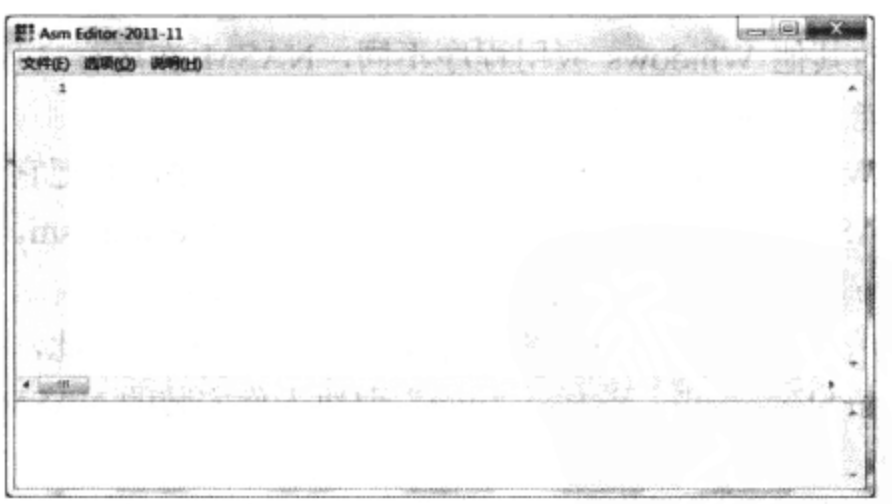

图 3-3　Nasmide 程序的基本界面

中间最大的空白区域是编辑区，用来书写汇编语言源代码。

窗口底部那个窄的区域是消息显示区。在编译当前文档时，不管是编译成功，还是发现了文档中的错误，都会显示在这里。

基本上，你现在已经可以在 Nasmide 里书写汇编语句了。不过，在此之前你最好先做一件事情。Nasmide 只是一个文本编辑工具，它自己没有编译能力。不过，它可以在后台调用 NASM 来编译当前文档，前提是它必须知道 NASM 安装在什么地方。

为此，你需要在菜单上选择“选项”→“编译环境设置”来打开如图 3-4 所示的对话框。

如图 3-4 所示，这个路径就是你在前面安装 NASM 时，指定的安装路径，包括可执行文件名 nasm.exe。

不同于其他汇编语言编译器，NASM 最让我喜欢的一个特点是允许在源程序中只包含指令，如图 3-5 所示。用过微软公司 MASM 的人都知道，在真正开始书写汇编指令前，先要穿靴戴帽，在源程序中定义很多东西，比如代码段和数据段等，弄了半天，实际上连一条指令还没开始写呢。

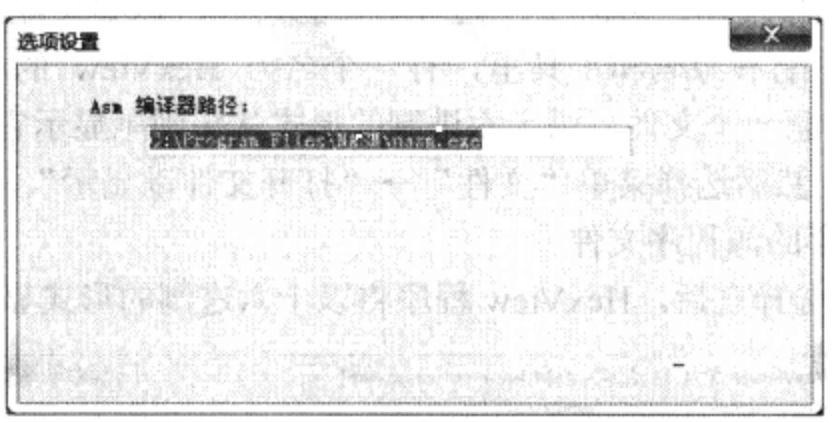

图 3-4 为 Nasmide 指定 NASM 编译器所在路径

图 3-5 NASM 允许在源文件中只包含指令

如图 3-5 所示，用 Nasmide 程序编辑源程序时，它会自动在每一行内容的左边显示行号。对于初学者来说，一开始可能会误以为行号也会出现在源程序中。不要误会，行号并非源程序的一部分，当保存源程序的时候，也不会出现在文件内容中。

让 Nasmide 显示行号，这是一个聪明的决定。一方面，我在书中讲解源程序时，可以说第几行到第几行是做什么用的；另一方面，当编译源程序的时候，如果发现了错误，错误信息中也会说明是第几行有错。这样，因为 Nasmide 显示了行号，这就很容易快速找到出错的那一行。

在汇编源程序中，可以为每一行添加注释。注释的作用是说明某条指令或者某个符号的含义和作用。注释也是源程序的组成部分，但在编译的时候会被编译器忽略。如图 3-5 所示，为了告诉编译器注释是从哪里开始的，注释需要以英文字母的分号“;”开始。

当源程序书写完毕之后，就可以进行编译了，方法是在 Nasmide 中选择菜单“文件”→“编译本文档”。这时，Nasmide 将会在后台调用 NASM 来完成整个编译过程，不需要你额外操心。如

图3-5所示，即使只有三行的程序也能通过编译。编译完成后，会在窗口底部显示一条消息。

◆ **检测点3.1**

1. 在你的计算机中启动 Nasmide 程序，输入图 3-10 中的三行代码，然后编译它们。看看消息显示区是否有编译成功的提示。
2. 选择填空：指令 mov ax,0xf5fc 中，“mov”指示这是一条（　　）指令，0xf5fc 是（　　）。指令执行后，寄存器 AX 中的内容是（　　）。
   A.立即数　B.传送　C.0xf5fc　D.加法　E.0xfcf5　F.寄存器

3.2.3 用 HexView 观察编译后的机器代码

编译成功完成之后，Nasmide 会在编辑窗口的底部显示相应的消息，同时显示了源文件名称和编译之后的文件名称（含路径）。

尽管我们强调源文件和编译之后的文件具有不同的内容，但如果能用工具看一看，相信印象更为深刻。在前面下载的配书源码和工具里，有一个名为 HexView 的小程序，可以实现这个愿望。HexView 用于打开任意一个文件，以十六进制的形式从头到尾显示它每个字节的内容。

双击启动 HexView，然后选择菜单“文件”→“打开文件以显示”，在文件选择对话框里找到你在 3.2.4 节里编辑并保存的源程序文件。

如图 3-6 所示，文件选择之后，HexView 程序将以十六进制的形式显示刚刚选择的文件。

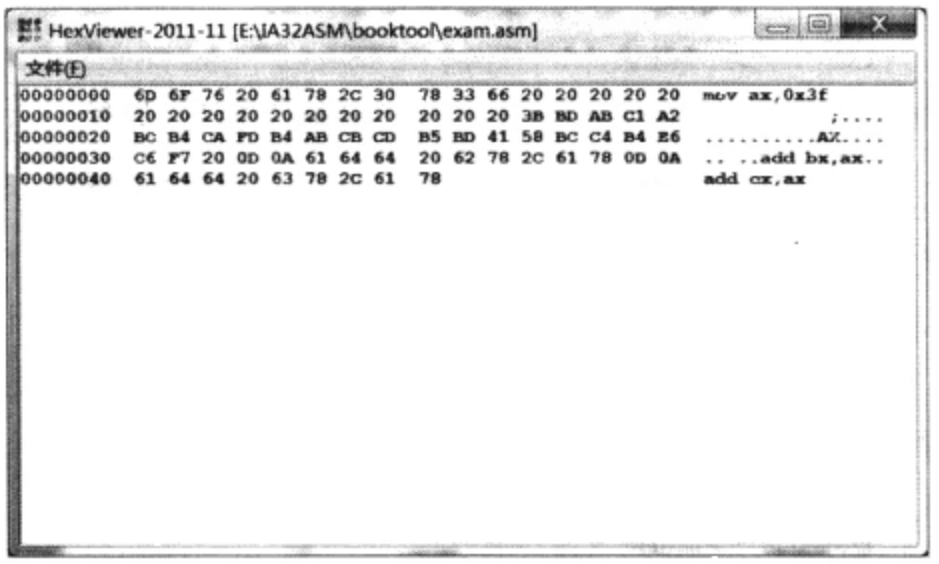

图 3-6　用 HexView 程序显示源程序文件的内容

在 HexView 中，文件的内容以十六进制的形式显示在窗口中间，以 16 个字节为一行，字节之间以空白分隔，所以看起来很稀疏。如果文件较大的话，则会分成很多行。

作为对照，每个字节还会以字符的形式显示在窗口右侧，如果它确实可显示为一个字符的话。如果该字节并非一个可以显示的字符，则显示一个替代的字符“.”。因为源程序中还有汉字注释，所以，如果细心的话，从图中可以算出每个汉字的编码是两个字节，比如“将”字的编码是 0xBD 0xAB。由于 HexView 以单字节的形式来显示每个字符，所以无法显示汉字。

左边的数字，是每一行第一个字节相对于文件头部的距离（偏移），也是以十六进制数显示的。字母“m”是整个源程序文件内的第 1 个字符，因此，它的偏移量是 00000000（H），其他字符以此类推，最后一个字符“x”的偏移量是 00000048（H）。

源程序很长，但是，编译之后的机器指令却很简短。

如图 3-7 所示，编译之后的文件只有 7 个字节，这才是处理器可以识别并执行的机器指令。

图 3-7 用 HexView 显示编译之后的文件内容

本 章 习 题

如图 3-6 所示，请问：

（1）源程序共有 3 行，每一行第一个字符在文件内的偏移量分别是多少？

（2）该源程序文件的大小是多少字节？