

```

// 返回两个参数的平均值的函数
double Average(double a, double b) { (1)
    return (a + b) / 2;
}

// 程序运行起始位置的函数
int WINAPI WinMain(HINSTANCE h, HINSTANCE d, LPSTR s, int m)
{
    double ave;                // 保存平均值的变量
    char buff[80];             // 保存字符串的变量

    // 求解 123,456 的平均值
    ave = Average(123, 456);

    // 编写显示在消息框中的字符串
    sprintf(buff, "平均值 = %f", ave); (3) (2)

    // 打开消息框
    MessageBox(NULL, buff, title, MB_OK); (4)

    return 0;
}

```



图 8-1 代码清单 8-1 的执行结果

类似于代码清单 8-1 这样，用某种编程语言编写的程序就称为**源代码**^①，保存源代码的文件称为**源文件**。用 C 语言编写的源文件的扩展名通常是“.c”，因此，这里我们就把代码清单 8-1 的文件命名为 Sample1.c。

① 这里的“源代码”用英文表示是“source code”。source 有“原始的”的意思，因此所谓源代码，就是原始的代码。源代码有时也称为源程序。

因为源文件是简单的文本文件，所以用 Windows 自带的记事本等文本编辑器就可以编写。

代码清单 8-1 的源代码是无法直接运行的。这是因为，CPU 能直接解析并运行的不是源代码而是本地代码的程序。作为计算机大脑的 Pentium 等 CPU，也只能解释已经转换成本地代码的程序内容。

本地（native）这个术语有“母语的”意思。对 CPU 来说，母语就是机器语言，而转换成机器语言的程序就是本地代码。用任何编程语言编写的源代码，最后都要翻译成本地代码（图 8-2），否则 CPU 就不能理解。也就是说，即使是用不同编程语言编写的代码，转换成本地代码后，也都变成用同一种语言（机器语言）来表示了。

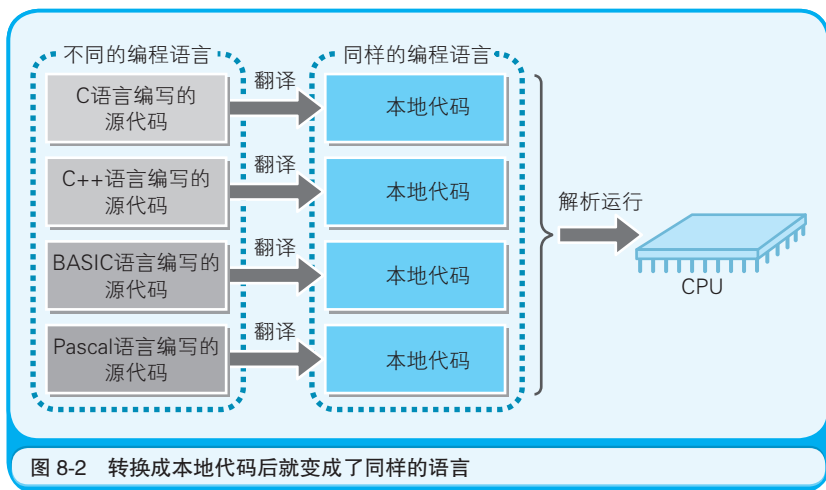


图 8-2 转换成本地代码后就变成了同样的语言



8.2 本地代码的内容

Windows 中 EXE 文件的程序内容，使用的就是本地代码。正所谓“百闻不如一见”，接下来就让我们来看一下本地代码的内容吧。

用记事本打开由代码清单 8-1 的内容转换成本地代码得到的 EXE 文件 (Sample1.exe)，页面显示情况如图 8-3 所示。据此我们应该可以看出，本地代码的内容是人类无法理解的。也正是因为如此，才有了用人类容易理解的 C 语言等编程语言来编写源代码，然后再将源代码转换成本地代码这一方法。

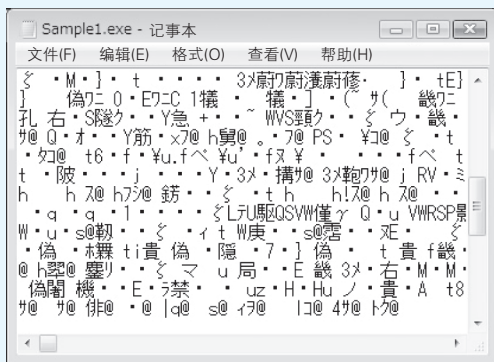


图 8-3 用记事本打开 EXE 文件后出现了无法理解的文字

接下来，我们把刚才的 EXE 文件的内容 Dump 一下。Dump 是指把文件的内容，每个字节用 2 位十六进制数来表示的方式。本地代码的内容就是各种数值的罗列，这一点想必大家都了解。而这些数值就是本地代码的真面目。每个数值都表示某一个命令或数据 (图 8-4)。这里我们用的是原始的 Dump 程序。

而计算机就是把所有的信息作为数值的集合来处理的。例如，A 这个字符数据就是用十六进制数 41 来表示的。与此相同，计算机指令也是数值的罗列。这就是本地代码。

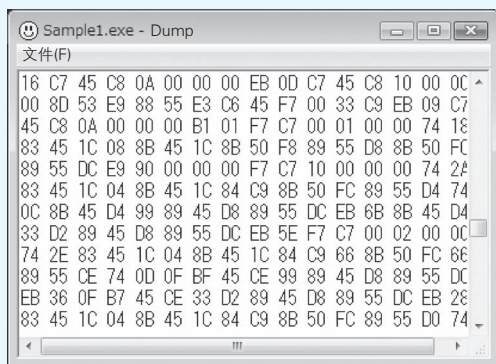


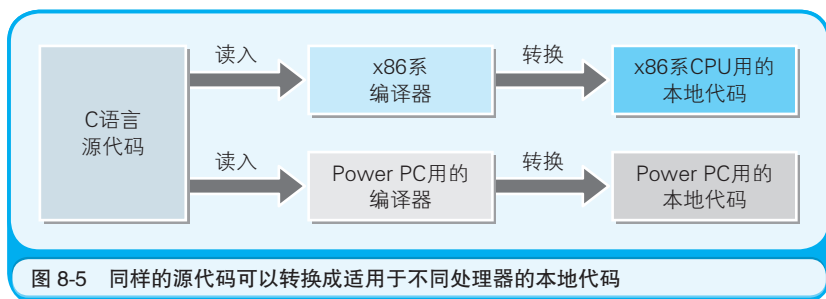
图 8-4 本地代码的真面目是数值的罗列

8.3 编译器负责转换源代码

能够把 C 语言等高级编程语言编写的源代码转换成本地代码的程序称为**编译器**。每个编写源代码的编程语言都需要其专用的编译器。将 C 语言编写的源代码转换成本地代码的编译器称为 C 编译器。

编译器首先读入代码的内容，然后再把源代码转换成本地代码。编译器中就好像有一个源代码同本地代码的对应表。但实际上，仅仅靠对应表是无法生成本地代码的。读入的源代码还要经过语法解析、句法解析、语义解析等，才能生成本地代码。

根据 CPU 类型的不同，本地代码的类型也不同。因而，编译器不仅和编程语言的种类有关，和 CPU 的类型也是相关的。例如，Pentium 等 x86 系列 CPU 用的 C 编译器，同 PowerPC 这种 CPU 用的 C 编译器就不同。从另一个方面来看，这其实是非常方便的。因为这样一来，同样的源代码就可以翻译成适用于不同 CPU 的本地代码了（图 8-5）。



因为编译器本身也是程序的一种，所以也需要运行环境。例如，有 Windows 用的 C 编译器、Linux 用的 C 编译器等。此外，还有一种交叉编译器，它生成的是和运行环境中的 CPU 不同的 CPU 所使用的本地代码。例如，在 Pentium 系列 CPU 的 Windows 这一运行环境下，也可以作成 SH^① 及 MIPS 等 CPU 用的 Windows CE^② 程序，而这就是通过使用交叉编译器来实现的。

读到这里大家可能稍微有一些混乱，不妨让我们来梳理一下。大家在计算机软件商店等处购买编译器时，可能会跟店员说明 3 点：“想要买的是何种编程语言用的编译器”“编译器生成的本地代码是用于哪种 CPU 的”以及“该编译器是在什么环境下使用的”（图 8-6）。而实际上，通常只要说明产品名及版本就可以了^③。

-
- ① SH (SuperH) 是日立制作所和三菱电机共同成立的瑞萨技术开发的 CPU。该 CPU 有多种类型，在手机、车载 GPS、PDA、游戏机等设备上均有使用。
- ② Windows CE 是采用了 MIPS、SH 等 CPU 的 PDA 及嵌入式开发领域广泛使用的操作系统。
- ③ 现在编译器基本上不需要购买，都已经默认集成到开发 IDE 中了。

——译者注



8.4 仅靠编译是无法得到可执行文件的

编译器转换源代码后，就会生成本地文件。不过，本地文件是无法直接运行的。为了得到可以运行的 EXE 文件，编译之后还需要进行“链接”处理。下面，就让我们使用 Borland C++ Compiler 5.5（以下称为 Borland C++）来看一下编译和链接是如何进行的。

Borland C++ 的编译器是 `bcc32.exe` 这个命令行工具。在 Windows 的命令提示符^①中，运行下列命令后，由 C 语言编写的源文件 `Sample1.c` 就会被编译^②。

```
bcc32 -W -c Sample1.c
```

“-W-c”是用来指定编译 Windows 用的程序的选项。选项是对编译器的指示。有时也称为“开关”。

① 命令行工具指的是在 Windows 的命令提示符下使用的 CUI 程序。

② 编译 `Sample1.c` 后，可能会出现 `WinMain` 的参数没有被用到的警告提示，不过这不会造成什么影响。由于警告并不是出错，因而也可以生成目标文件。

编译后生成的不是 EXE 文件，而是扩展名为 “.obj” 的目标文件^①。Sample1.c 编译后，就生成了 Sample1.obj 目标文件。虽然目标文件的内容是本地代码，但却无法直接运行。那么这是为什么呢？原因就是当前程序还处于未完成状态。

让我们再来看一遍代码清单 8-1 中的源代码。(1) 围起来的函数 Average() 同 (2) 围起来的函数 WinMain() 是程序员自己作成的，处理内容记述在源代码中。Average() 是用来返回两个参数数值的平均值的函数，Winmain() 是程序的运行起始函数。除此之外，还有 (3) 指出的 sprintf() 函数和 (4) 指出的 MessageBox() 函数。sprintf() 是通过指定格式把数值转换成字符串的函数，MessageBox() 是消息框函数，不过源代码中都没有记述这些函数的处理内容。因此，这时就必须将存储着 sprintf() 和 MessageBox() 的处理内容的目标文件同 Sample1.obj 结合，否则处理就不完整，EXE 文件也就无法完成。

把多个目标文件结合，生成 1 个 EXE 文件的处理就是链接，运行连接的程序就称为链接器 (linkage editor 或连接器)。Borland C++ 的链接器就是 ilink32.exe 的命令行工具。在 Windows 命令提示符下运行以下命令后，程序所需的目标文件就会被全部链接生成 Sample1.exe 这个 EXE 文件。

```
ilink32 -Tpe -c -x -aa c0w32.obj Sample1.obj, Sample1.exe,,  
import32.lib cw32.lib
```

① 目标文件 (object file) 中的 object 一词，指的是编译器生成结果的意思。和面向对象编程 (object oriented programming) 的 object 没有任何关系。面向对象编程的对象指的是数据和处理的集合体。

8.5 启动及库文件

链接选项“-Tpe-c-x-aa”是指定生成 Windows 用的 EXE 文件的选项。在这些选项之后，会指定结合的目标文件。而该命令行中就指定了 c0w32.obj、Sample1.obj 这两个目标文件，这点相信大家都能看得出来。Sample1.obj 是 Sample1.c 编译后得到的目标文件。c0w32.obj 这个目标文件记述的是同所有程序起始位置相结合的处理内容，称为程序的**启动**。因而，即使程序不调用其他目标文件的函数，也必须要进行链接，并和启动结合起来。c0w32.obj 是由 Borland C++ 提供的。如果 C：盘中安装有 Borland C++ 的话，文件夹 C:\Borland\bcc55\lib 中就会有 c0w32.obj 这个文件。

那么，大家可能会有这样一个疑问：“链接时不指定 sprintf() 和 MessageBox() 的目标文件也没问题么？”这个担心是多余的。在链接的命令行末尾，存在着扩展名是“.lib”的 import32.lib 和 cw32.lib 这两个文件。这是因为 sprintf() 的目标文件在 cw32.lib 中，MessageBox() 的目标文件在 import32.lib 中（实际上，MessageBox() 的目标文件在 user32.dll 这个 DLL 文件中。关于这一点，我们会在后面进行说明）。

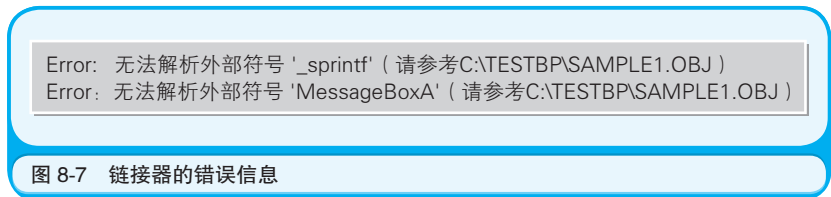
像 import32.lib 及 cw32.lib 这样的文件称为库文件。**库文件**指的是把多个目标文件集成保存到一个文件中的形式。链接器指定库文件后，就会从中把需要的目标文件抽取出来，并同其他目标文件结合生成 EXE 文件。

Sample1.obj 是尚未完成的本地代码，这个在前面已经进行了说明。这是因为，Sample1.obj 文件中包含有“链接时请结合 sprintf() 及 MessageBox()”这样的信息。意思是如果不存在其他函数的话，程序就无法运行。下面，我们就来做一下尝试，看看在不指定这两个库文件

的情况下进行链接会发生什么。

```
ilink32 -Tpe -c -x -aa c0w32.obj Sample1.obj, Sample1.exe
```

在命令提示符上运行上述命令后，链接器就会出现如图 8-7 所示的错误消息（实际上显示的错误消息更多，这里对其进行了省略）。



```
Error: 无法解析外部符号 '_sprintf' (请参考C:\TESTBP\SAMPLE1.OBJ)
Error: 无法解析外部符号 'MessageBoxA' (请参考C:\TESTBP\SAMPLE1.OBJ)
```

图 8-7 链接器的错误信息

该错误消息表示的是无法解析 Sample1.obj 参照的外部符号。外部符号是指其他目标文件中的变量或函数。sprintf 及 MessageBoxA 是目标文件中 sprintf() 及 MessageBox() 的名称。代码中记述的函数名同目标文件中的函数名有一些差异，不过大家只需把它理解成这是 C 编译器的规定即可。错误消息“无法解析的外部符号”表示的是无法找到记述着目的变量及函数的目标文件，因而无法进行链接的意思。

sprintf() 等函数，不是通过源代码形式而是通过库文件形式和编译器一起提供的。这样的函数称为**标准函数**。之所以使用库文件，是为了简化为链接器的参数指定多个目标文件这一过程。例如，在链接调用了数百个标准函数的程序时，就要在链接器的命令行中指定数百个目标文件，这样就太繁琐了。而利用存储着多个目标文件的库文件的话，则只需在链接器的命令行中指定几个库文件就可以了。

通过以目标文件的形式或集合多个目标文件的库文件形式来提供函数，就可以不用公开标准函数的源代码内容。由于标准函数的源代码是编译器厂商的贵重财产，因此若被其他公司任意转用的话，可能

会造成一些损失。

8.6 DLL 文件及导入库

Windows 以函数的形式为应用提供了各种功能。这些形式的函数称为 API (Application Programming Interface, 应用程序接口)。例如, Sample1.c 中调用的 MessageBox(), 它并不是 C 语言的标准函数, 而是 Windows 提供的 API 的一种。MessageBox() 提供了显示消息框的功能。

Windows 中, API 的目标文件, 并不是存储在通常的库文件中, 而是存储在名为 DLL (Dynamic Link Library) 文件的特殊库文件中。就如 Dynamic 这一名称所表示的那样, DLL 文件是程序运行时动态结合的文件。在前面的介绍中, 我们提到 MessageBox() 的目标文件是存储在 import32.lib 中的。实际上, import32.lib 中仅仅存储着两个信息, 一是 MessageBox() 在 user32.dll 这个 DLL 文件中, 另一个是存储着 DLL 文件的文件夹信息, MessageBox() 的目标文件的实体实际上并不存在。我们把类似于 import32.lib 这样的库文件称为**导入库**。

与此相反, 存储着目标文件的实体, 并直接和 EXE 文件结合的库文件形式称为**静态链接库**。静态 (static = 静态的) 同动态 (dynamic = 动态的) 是相反的意思。存储着 sprintf() 的目标文件的 cw32lib 就是静态链接库。sprintf() 提供了通过指定格式把数值转换成字符串的功能。

通过结合导入库文件, 执行时从 DLL 文件中调出的 MessageBox() 函数这一信息就会和 EXE 文件进行结合。这样, 链接器链接时就不会再出现错误消息, 从而就可以顺利编写 EXE 文件。

至此, 我们总结一下 Windows 中的编译及链接机制, 如图 8-8 所示。