目录

第 19 章 上机操作

本书所介绍的例子在开发和运行时，使用的 CPU 为 Intel Core i7，是一种基于 x64 位的处理器；操作系统为 Windows10，为 64 位的操作系统；开发工具为 Visual Studio 2019 社区版（community）。这是一款免费的产品，可在网站 https://visualstudio.microsoft.com/zh-hans/ 上下载。本章介绍在这样的环境下开发和调试汇编语言程序的操作方法。当然，我们以前在 Win 7 等操作系统和 Intel 系列其他 CPU 的环境下，使用 VS 2010、VS2013、VS2015、VS2017 等开发平台也开发过汇编语言程序。它们的操作方法本质上差别不大。本章介绍 VS2019 开发环境的使用方法。

19．1 创建工程和生成可执行文件

使用 VS2019 开发汇编源程序与开发 C 语言程序的操作步骤是类似的，最关键的差别是在开发汇编语言程序时要设置编译方法。特别要注意下面的第③步要先于第④步。

① 新建一个空项目

在 VS2019 的主界面上，单击“创建新项目”，在创建新项目的页面上，选择“空项目（使用 C++ for Windows 从头开始操作，不提供基础文件）”。然后单击“下一步”出现“配置新项目”的页面。

注意：若操作界面上未出现上述选项，则注意界面上有三个下拉列表框。其中语言选择“C++”；平台选择“Windows”；项目类型选择“控制台”。这三项选择不同的条目，界面上的内容会随之发生变化。

② 配置新项目

在此页面上，要输入项目的名称，如 Huibian_Test，选择项目存放的位置，之后单击“创建”即可。此时，会在 VS2019 的窗口中出现“解决方案资源管理器”，如图 19.1 所示。

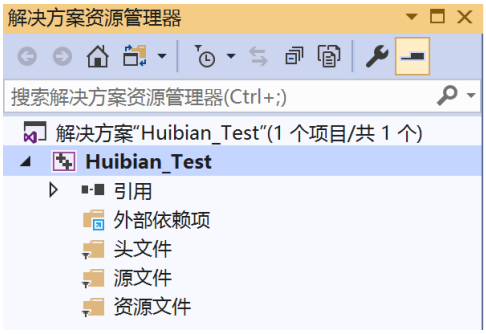

图 19.1 解决方案资源管理器

注意，在配置新项目页面上，在选择文件存放“位置”之下，有一个“解决方案名称”，它自动的与项目名称相同。实际上解决方案与项目是两个概念。一个解决方案之下可以有多个项目，它是一个更大级别的容器。可以将相关的项目放在一个解决方案中。在新建第一个项目时，同时创建了一个解决方案。之后，可以在该解决方案下添加新项目。

设置该项目的编译方法，在“生成自定义”选择 masm 项

③ 设置生成依赖项

在项目（图 19.1 中的亮条 Huibian_Test）上单击鼠标右键，会出现一个弹出式菜单，在此菜单上，单击“生成依赖项”，然后，再单击“生成自定义...”，在出现的“Visual C++ 生成自定义文件”界面中勾选“masm”,如图 19.2 所示，之后单击“确定”。

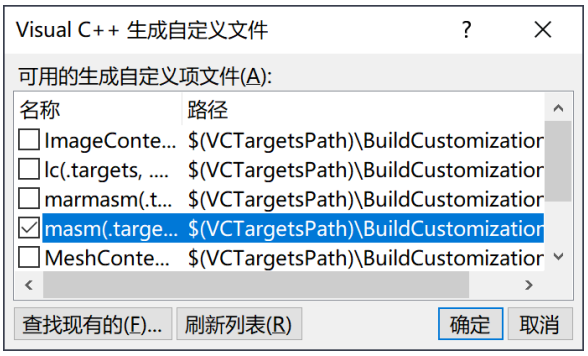

图 19.2 生成自定义界面

特别注意：设置生成依赖项一定要先于添加源文件（即第④步）。如果是向项目中先添加了源文件,然后在设置生成依赖项,则在生成执行程序时是不会对汇编源程序进行编译的。在 C 和汇编混合编程时，也应先设置生成依赖项。

④ 添加汇编源程序文件

与第③步相同，在项目上单击鼠标右键，在出现一个弹出式菜单中选择“添加”->“新建项”，在出现的“添加新项”的页面上，输入汇编语言源程序的名称，如 test.asm。之后单击“添加”即可。

除了添加 asm 文件外，用同样的方法可添加头文件、资源文件。

⑤ 编辑源程序

输入汇编源程序，完成程序的编辑工作。

⑥ 生成执行程序

在项目上单击鼠标右键，在出现一个弹出式菜单中单击“生成”。观察 VS 界面中的输出窗口，看是否出现错误提示。若有错误，则返回第⑤步修改源程序，然后在执行本步，直到生成 exe 文件。

对于本书上给出的程序基本上用上述步骤即可生成可执行程序。

19．2 程序的调试

用 VS2019 可以开发 C 语言程序、汇编语言程序。在生成执行程序之后，调试方法是相同的。只是我们在调试 C 语言程序时，一般不会观察机器层面的内容，如反汇编代码、寄存器等。而在调试汇编语言程序时，需要看这些内容。对于变量、内存单元的监视实际上没有差别的。

单击 VS2019 的菜单上的“调试”，在下拉菜单中可以看到有“开始调试”、“逐语句”。对于汇编语言程序的调试，要选择“逐语句”。此时可以看到进入了调试界面。在调试时，可以设置断点、取消断点、单步执行、继续执行直到遇到断点、逐过程执行、跳出函数或子程序到一级等等。

单击“调试”菜单中的“窗口”，可以看到出现“反汇编”、“寄存器”、“内存”、“监视”、

“调用堆栈”、“断点”等菜单项。可以单击相应的菜单项，打开对应的窗口。本节将介绍最常用的“反汇编”、“寄存器”、“内存”、“监视”窗口的基本操作方法。

1、反汇编窗口

在反汇编窗口，可以看机器指令的地址、机器指令的字节编码、反汇编指令、当前待执行的指令等等。界面如图 19.3 所示。

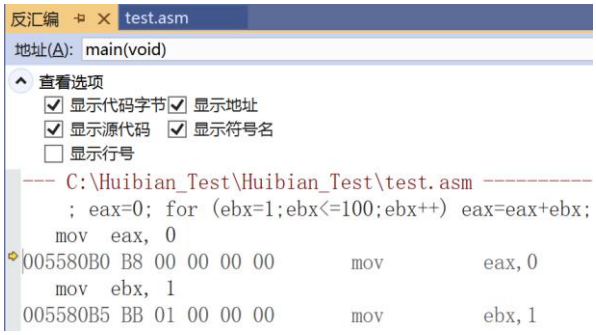

图 19.3 反汇编窗口

注意，在反汇编窗口显示的信息是可以设置的。在反汇编窗口的左上角有一个“查看选项”，单击其左边的“∨”，可以展开一个小窗口，如图 19.3 所示，选择要显示的内容，之后，可以按“∧”缩回查看选项。

在查看选项中，有一个“显示符号名”，勾选该项，则在反汇编窗口是以符号的形式显示全局变量（data 段中定义的变量）、局部变量（子程序中定义的变量）。若不勾选，则显示的是该变量对应的地址，对全局变量和局部变量显示的结果是不同的。

例如，设 x 是全局变量，有是局部变量。

对于语句 mov eax, x 和 mov eax, y，在勾选“显示符号名”时，显示的形式如下：

```
00FE80B6 A1 00 70 05 01       mov  eax,dword ptr [x (01057000h)]
00FE80BB 8B 45 FC             mov  eax,dword ptr [y]
```

在不勾选“显示符号名”时，显示的形式如下：

```
00FE80B6 A1 00 70 05 01       mov  eax,dword ptr ds:[01057000h]
00FE80BB 8B 45 FC             mov  eax,dword ptr [ebp-4]
```

2、寄存器窗口

在寄存器窗口显示各寄存器的值。但是通常只看得到几个通用寄存器的值、EIP、标志寄存器 EFL。如图 19.4 所示。

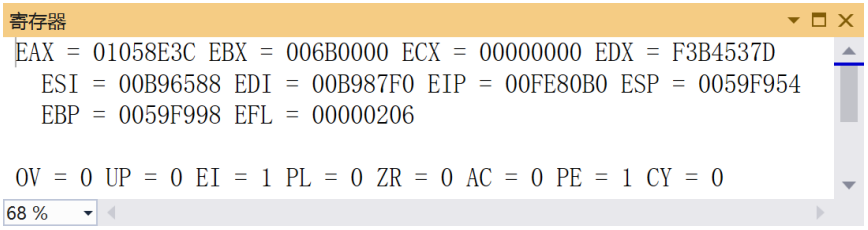

图 19.4 寄存器窗口

寄存器窗口中显示的内容是可以设置的。在寄存器窗口中，单击鼠标右键，会弹出一个菜单，在菜单中可单击希望显示的项。有勾选标志的项的内容显示在窗口中。再次单击该项，又可从显示内容中移除。

可选的内容包括；

CPU 段： 段寄存器 cs,ds,es ss,fs,gs

浮点： x87 中的寄存器，st0 - st7,crtl、stat、tags、eip、ed0

MMX ： mm0 – mm7

SSE ： xmm0 - xmm7、MXCSR

AVX ： ymm0 – ymm7

注意，若在 x64 平台上，在寄存器窗口看到的寄存器有所不同，但操作方法是相同的。

寄存器窗口上有滚动条，窗口的大小、位置等都是可以自己调整的。

3、监视窗口

在监视窗口，可以多种形式来观察一个变量，界面如图 19.5 所示。

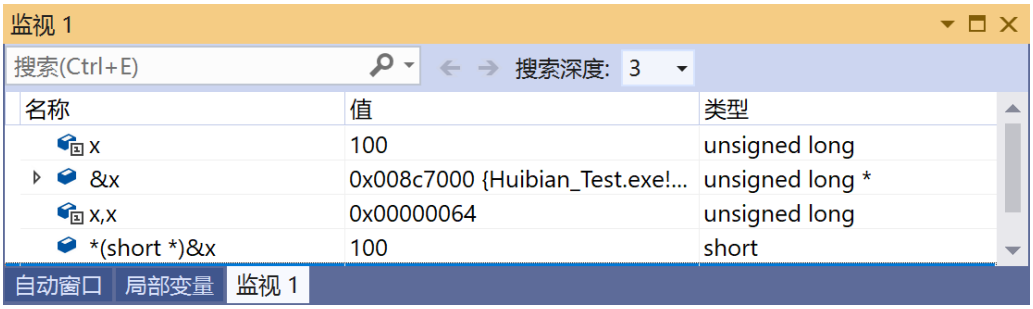

图 19.5 监视窗口

例如，对于变量 x，直接在“名称”下输入 x,可看到变量 x 的值；输入&x，看到变量 x 的地址；“x,x”是以 16 进制形式显示变量 x，在输入“x,”后，会弹出一个菜单，可从中选择是以何种形式显示相应的内容，默认的是 10 进制形式。另外，还可以进行强制地址类型转换后显示，例如“*(short *)&x”。

4、内存窗口

在内存窗口，可以观察从某地址开始的一片单元中存储的内容。界面如图 19.6 所示。

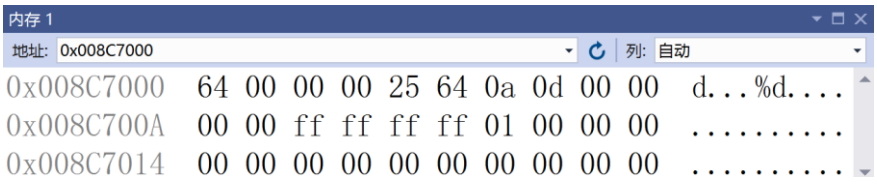

图 19.6 内存窗口

内存窗口一般分三部分，最左边的一列，是内存单元的地址；中间的列是内存单元的值，最右边的是以 ASCII 形式来显示中间列的内容。

内存窗口显示什么，以何种形式显示，也是可以设置的。在内容窗口单击鼠标右键，将弹出一个菜单项。在该菜单上可选择中间部分的显示方式：1 字节整数，2 字节整数，...，8 字节整数，32 位浮点，64 位浮点，也可以选择“没有数据”，从而不显示中间部分。另外，也可以控制以何种进制显示，是否显示 ASCII 部分等等。

19．3 编译链接器的设置

在 19.1 节中，给出了汇编语言程序开发时生成执行程序的操作步骤。本节介绍在生成执行程序前对编译器和链接器可做的一些配置。此外，在学习汇编语言程序设计的时候，一种有效的手段的编写 C 语言程序，然后调试生成的机器语言程序。本节重点介绍 C 语言程序开发时的一些配置，设置不同的开发平台、编译选项生成的执行程序是不一样的。

在“解决方案资源管理器”中，右键单击项目名称，在弹出的菜单上单击“属性”，会出现项目属性页，在该页面上可以做各种配置。

1、 平台配置

在平台配置中，可以选择 Win32，也可以选择 x64。注意，不同的平台所采用的编译器并不相同。在 Win32 下编译通过的程序，在 x64 下不一定能编译通过。

此外，可以在工具栏上快捷的进行解决方案平台的配置，或者在 Debug 版本和 Release 版本之间切换；或者在 Win32 和 64 等不同平台上切换。操作界面如图 19.7 所示。

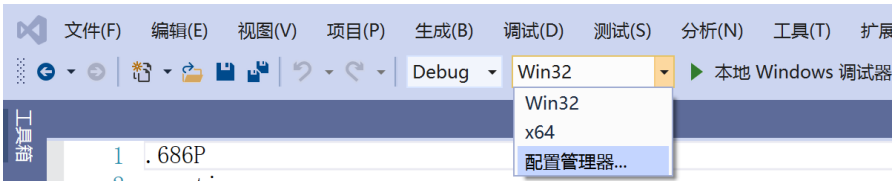

图 19.7 解决方案平台的配置或切换

2、 微软宏编译器配置

对于汇编程序开发项目，在“项目属性页”中有“Microsoft Macro Assembler”。展开该项，可以看到“General”、“Command Line”、“Listing File”等条目。在“Command Line”中，可以看到对汇编源程序进行编译时，实际执行的命令。如果希望在编译时，生成列表文件，可以在“Listing File”中设置。一般情况下，可以不对“Microsoft Macro Assembler”做任何设置，采用默认值即可。

3、 C/C++编译器配置

对于 C 程序开发项目，在“项目属性页”中有“C/C++”，该条目用于设置编译开关。展开该条目，可以看到“常规”、“优化”、“代码生成”、“输出文件”等等子项。每一子项下面又有更多的小项。一般情况下，也可以采用默认值。但下面几项对汇编语言学习是有用的。

① 生成汇编语言程序

在“输出文件”->“汇编程序输出”中选择“带源代码的程序集(/FAs)”。编译后，会生成与源文件名同名但后缀为 asm 的文件。这是一个文本文件，可以打开该文件，可观察生成的汇编语言程序。

② 变量的空间分配

调试 C 语言程序时，会发现变量之间似乎有“间隙”，即它们的地址间距比所需要的空间大，这是由于在编译设置造成的，其目的在于调试时能快速发现访问越界等问题。在“代码生成”->“基本运行时检查”中，设置为“默认值”，则在变量之间不会留大的间隙，自然边界对齐留出的空间除外。

③ 结构成员对齐

默认情况下，结构变量中各字段的起始地址也是采用自然边界对齐的方式。若要采用紧凑模式，可以在“代码生成”->“结构成员对齐”中设置为 1 字节。

④ 启用增强指令集

对同一个程序，在生成代码时，可以选用不同的指令集。在“代码生成”->“启用增强指令集”可以选择:“流式处理 SIMD 扩展 (/arch:SSE) ”、“高级矢量扩展 (/arch:AVX) ”、“无增强指令 (/arch:IA32) ”等等。

在 C/C++中的设置会影响最后执行的命令。在 C/C++的命令行中可以看到执行编译时所用到的配置参数

3、链接器配置

对 C 语言程序和汇编语言程序开发都有“链接器”。默认情况下，也不需要配置链接器，但是在使用到其他库时，还是要一些手动配置。链接器下的配置项有:“常规”、“系统”、“高级”等项。

① 附加依赖项

在“输入”->“附加依赖项”中，输入库的名字。在汇编语言程序中也可以直接使用 includelib 语句添加需要的库。

② 附加库的目录

在“常规”->“附加库目录下”，添加库文件所在的目录。也即指明在何处去寻找附加的库文件。

③ 子系统

在“系统”->“子系统”中可以选择“控制台 (/SUBSYSTEM:CONSOLE) ”、“窗口 (/SUBSYSTEM:WINDOWS) ”等，指明程序的类型。

④ 入口点

在“高级”->“入口点”中输入要执行的程序的第一条指令的地址，通常程序中的一个标号，或者子程序的名字。

此外，可以在“VC++目录”中设置“包含目录”，即使用的各种头文件的目录。当然，在汇编源程序中也可以直接写出头文件所在的目录。可以在“VC++目录”中设置“库目录”。

在链接器中的设置会影响最后执行的命令。在链接器的命令行中可以看到执行链接时所用到的配置参数。

19．4 其他操作

1、多项目的管理

在一个解决方案下可以包括多个项目。例如，将一章中多个例子项目放在一个解决方案中。目的是便于集中管理这些项目，虽然这些项目有各自的存储目录，但是它们一般是解决方案的子目录。

方法 1：首先创建一个“空白的解决方案”。在“创建新建项目”时，选择 “空白解决方案”。

方法 2:“创建新项目”时，自动生成解决方案。

在创建解决方案后，右键单击“解决方案”，在弹出的菜单上选择“添加”->“新建项目”，可以创建一个新的项目。多次进行上述操作，就可以生成含有多个项目的解决方案。

在多个项目中，设置一个项目为启动项目。

在某个项目上，按鼠标右键，在弹出菜单中点击：“设为启动项目”，即将当前项目设为启动项目。后面就是对该项目进行编译和调试。直到采用同样的操作方法切换到一个新的项目。

2、调试工具栏

除了使用“调试”菜单下的项目进行调试操作外，还可以使用快捷的调试工具。

在 VS 工具栏的空白处，按鼠标右键，会弹出菜单。其中有一项为“调试”，若该项前面有“√”，表示显示了“调试工具栏”，否则就没有显示“调试工具栏”。 单击“调试”，可以切换“调试工具栏”的显示状态。

可以增加、删除调试工具栏中的工具。在鼠标停在“调试工具栏”的“下三角箭”，可看到出现“调试工具栏选项”的提示。单击该按钮，可出现“添加或移除按钮”。其右边框中有“√”的，表示该按钮出现在了工具栏中。使用“自定义”可调整工具栏上显示的工具。

，与存储模型说明中的语言类型一致，因而可以省略 printf 说明中的语言类型，它等同于“printf proto c :vararg”。注意，如果函数说明与函数实现体使用的语言类型不一致，则在链接时会出现错误“无法解析的外部符号……”

“ExitProcess proto stdcall :dword”和“printf proto :vararg”是程序中调用的两个函数的原型说明。在 C 语言程序中，一般用“#include”来包含函数原型说明的头文件”.h”，也可以直接在程序中说明使用的外部函数。在汇编语言程序中，同样可以将函数说明放在头文件中，然后用 include 来包含头文件，也可以直接在程序中说明。

在早期的 VS 版本（VS2010,VS2013）中，实现 printf 的库是 msvcrt.lib。在 VS2019 下不在使用该库。当然，在 VS2019 下还是可以使用 msvcrt.lib 的，不过要自己找到相应的老版本（在第 13 中介绍的 masm32 软件包中），并设置相应的库目录。

函数 ExitProcess 的实现在库 kernel32.lib 中。该库文件在创建工程时一般都已自动添加，无需使用“includelib　kernel32.lib”

若没有包含实现 printf 的库文件，在对程序进行汇编时是正常的，但在链接时就报错“无法解析的外部符号_printf”。

提示：并不是一定要在程序中有 includelib。可以通过在开发平台（Visual Studio）中配置项目属性来解决链接库的问题。在“项目属性->配置属性->链接器->输入->附加依赖项”中包含库。当然，如有必要，还需要指定库文件所在的目录，可以在配置属性->VC++目录->库目录中设置。Visual Studio 平台对汇编语言源程序与 C 语言程序的编译、链接和调试是相同的，并没有什么差别。