**汇编作业五**

2151644 周一韬

Github代码仓库地址：https://github.com/FDWJTZSBJEP/2023TJ\_Assembly

**一 嵌入式方式实现c语言和汇编混合编程**

混合编程是指在同一个程序中同时使用C语言和汇编语言编写代码。这种做法通常是为了充分发挥C语言的高级特性，同时利用汇编语言的底层控制和直接操作硬件的能力。

这里我们先使用在c语言中嵌入使用汇编的方法，即内联汇编。内联汇编是将汇编代码嵌入到C语言代码中的一种方式。在C语言源代码中，使用asm、asm或\_\_asm等关键字，将汇编代码直接插入到C代码的某个位置。这使得程序员可以在C语言的框架下嵌入底层的汇编指令，实现更精细的控制和性能优化。

下面是一个使用嵌入式方式实现混合编程的示例（hw5-2.c）

|  |
| --- |
| Assembly language int main() {  int a = 10, result;  // 嵌入式使用asm汇编  asm (  "movl %[input\_a], %[output\_result]" // 汇编代码  : [output\_result] "=r" (result) // 输出操作数  : [input\_a] "r" (a) // 输入操作数  : // 使用到的寄存器  );  printf("Result: %d\n", result);  return 0; } |

需要说明的是本次作业的运行环境是vscode，使用gcc编译器，asm关键字引导了一段使用AT&T语法的内联汇编代码。这段代码使用了内联汇编，通过汇编指令movl将C变量 a 的值移动到 result 中，并通过printf函数输出结果。在内联汇编中，[output\_result] "=r" (result)表示输出约束，将result 的值写入一个通用寄存器，[input\_a] "r" (a)表示输入约束，从通用寄存器中读取 a 的值。

**二 多文件方式实现c语言和汇编混合编程**

这种方式的实现类似于上一次多文件实现一个大汇编程序，在多文件方式实现C语言和汇编混合编程时，通常会将C和汇编代码分别写在不同的文件中，然后通过链接这些文件生成可执行程序。以下是一个简单的示例，演示了如何使用多文件方式进行混合编程。

同样以上一题的作业为例，在此基础上，我们需要将汇编代码部分和c语言代码部分分开处理，分别放在两个文件中，然后通过编译连接运行。

c语言文件如下（hw5\_1-1.c）

|  |
| --- |
| Assembly language extern int my\_asm\_function(int a);  int main() {  int a = 10;  int result = my\_asm\_function(a);   printf("Result: %d\n", result);   return 0; } |

汇编语言文件如下（hw5\_1-2.S）

|  |
| --- |
| Assembly language .global my\_asm\_function  my\_asm\_function:  movl %edi, %eax  ret |

这个汇编文件包含了一个全局的函数 my\_asm\_function，它将传入的参数 %edi（这里表示C语言中的整型参数）移动到 %eax 中并返回。在C模块中，通过 extern 声明了汇编模块中的 my\_asm\_function 函数，以便在C代码中调用。

编译连接两个文件

gcc -c hw5\_1-2.S -o hw5\_1-1.o

gcc -c hw5\_1-1.c -o hw5\_1-1.o

gcc hw5\_1-1.o hw5\_1-1.o -o hw5\_1

最后生成运行文件hw5\_1.exe

**作业心得**

本次作业的代码量不大，实现的逻辑也很简单，但是重点在于让我认识到了汇编和c语言的混合使用。C语言和汇编混合编程具有一些优点，这些优点使得在某些情况下选择混合编程成为一种有效的策略。但是目前看来，c语言主流的编译器功能已经足够强大，这种方法的使用并不能很好地体现出其优点。