**汇编语言程序设计**

**大作业**

深入理解c语言

学院：计算机科学与技术学院

专业：计算机科学与技术

班级：1536101班

学号：1150310607

姓名：张宁

2016年11月19日

1. 比较C与汇编的语言要素

1.C语言的语言要素：

|  |  |  |
| --- | --- | --- |
| 类型 | 标识符 | 字节数 |
| 短整型 | Short | 2 |
| 整形 | Int | 4 |
| 长整形 | Long | 4 |
| 超长整形 | Long long | 8 |
| 无符号整型 | Unsigned int | 4 |
| 单精度实型 | Float | 4 |
| 双精度实型 | Double | 8 |
| 字符型 | Char | 1 |
| 布尔型 | Bool | 1 |
| 指针型 | \* | 4 |
| 数组 |  |  |
| 多维数组 |  |  |
| 结构体 | Struct |  |
| 共用体 | Union |  |
| 枚举类型 | Enum |  |

汇编语言数据类型：

|  |  |  |
| --- | --- | --- |
| 8位无符号数 | Byte | 1 |
| 8位有符号数 | Sbyte | 1 |
| 16位无符号数 | Word | 2 |
| 16位有符号数 | Sword | 2 |
| 32位无符号数 | DWORD | 4 |
| 32位有符号数 | Sdword | 4 |
| 48位整数 | Fword | 6 |
| 64位整数 | Qword | 8 |
| 80位整数 | Tbyte | 10 |

C中数据类型int char short long long，等数据类型在汇编语言中有相应的数据byte，dword 等类型与之对应。Char 对应的是byte，int对应的是DWORD，二者在内存中的长度是相同的，汇编语言中不同的数据类型就是不同的长度的内存，c语言中不同的数据类型可以执行的操作不同。

2.

C语言和汇编语言的常量和常量表达式在编译的时候，会计算出数值，用数值进行替换，

C语言和汇编语言中存在变量，只不过变量的类型时使用方法不同，

C语言中存在变量表达式，条件表达式和关系表达式，但是在汇编语言中不存在这些，

关于条件表达式和关系表达式，汇编语言中用cmp语句和各种不同的跳转语句来实现了这个操作 。

变量表达式在 汇编语言中不存在，但是汇编语言会用很多条汇编语句来实现变量表达式的功能。

3．C语言赋初值采用简单的一个等号加上数据就行了即可，汇编采用mov语句加上源操作数和目的操作数。

C语言的赋值语句只需采用一个=号，左右两边写上变量名字就行了，汇编语言由于不能直接对内存中两个地址进行赋值操作，所以 当给内存中的变量赋值时，汇编语言的mov语句的源操作数一定是寄存器或者立即数，当给寄存器赋值的时候，mov语句的源操作数可以是 寄存器，内存，立即数。此外，汇编语言访问全局变量可以通过变量名，但是访问局部变量的时候，只能通过地址来访问。

C语言的算术运算有加减乘除，求余，取模等，逻辑运算包括 与或非。

汇编语言的算术运算除了c语言之外，还包含很多

C语言的分支转移语句十分的灵活，if else 语句的使用可以嵌套，switch语句的使用也会带来很多的方便，相反汇编语言没有这些语句，汇编语言通过简单的判断和跳转语句来实现这些分支转移指令，所以操作起来很麻烦。

C语言的循环语句是for循环，while 和do while，同样，汇编语言没有循环语句，汇编通过cmp语句以及跳转语句来实现循环操作。

4．

C语言中子函数的声明十分容易，定义一个函数头，把函数体实现了之后 就可以使用这个函数了，

但是汇编语言的函数声明有三种模式，一种是最麻烦的

一种是 proto invoke 模式，一种是

C语言的函数调用很容易，只需要传入参数就行了，

但是汇编语言的函数调用，需要先push参数，再调用函数，最后考虑堆栈平衡。

C语言的函数返回值是在eax里面。汇编语言的返回值可以在不同的寄存器里面。

C语言的参数传递是在函数调用的时候，写在一起 的，参数只能存在堆栈段里面。

但是汇编语言的参数传递需要先将参数push 进堆栈里面，然后调用函数来进行参数传递。

5.

汇编语言中的伪指令是区别于硬指令的，伪指令并不被cpu执行，伪指令挥被编译器进行处理。

C语言中宏有三种，第一种是引入库函数头文件的include指令，第二种是条件包含，第三种是宏展开：define语句。和汇编语言一样，c语言程序在 调用宏的地方，在编译阶段，替换成相对应的代码。

汇编语言中宏汇编，重复汇编，条件汇编统称为宏结构。宏结构的使用十分灵活，可以减少源代码，但是不能减少目标代码。宏的功能强大，采用”宏名macro【形参表 】宏定义体endm”的形式来定义，可以起到类似与函数的作用，但是同时却不会增加代码的执行时间，宏的使用可以增加代码的模块性，并且=使得程序更容易修改。7，向外部提供变量和子程序，使用外部的变量和子程序

6.

C语言向外部提供变量和子程序的话，需要在main函数之前，用extern“C”声明，并放在同一个大括号内，括号外部不再声明，在使用外部的函数的时候，extern 函数头/变量声明

汇编语言向外部提供的 函数和变量写成public C函数名/变量名，汇编语言使用外部的变量的话，用extern C 变量名 数据类型。如果是子程序的话，要声明为 函数名PROTO C:参数的类型。

总之 ，二者在 使用或提供函数或者变量的时候，只是形式不同而已。

8．头文件的内容区别，格式区别

C语言头文件的内容主要是函数的定义，宏的定义，引入其它的头文件，结构体的定义等。

汇编语言的头文件是宏的定义，以及介绍程序的运行环境，指定存储模式，指定最低兼容的cpu，定义堆栈段的大小，声明函数，和提供给外部的内容。

9．源程序模板的主要区别

C语言的注释是//和/\*\*/而汇编语言的注释是；宏注释是；；

C语言程序入口是一个main函数，函数可以没有返回值，函数可以调用函数，但是函数在使用一定要在函数的声明之后。

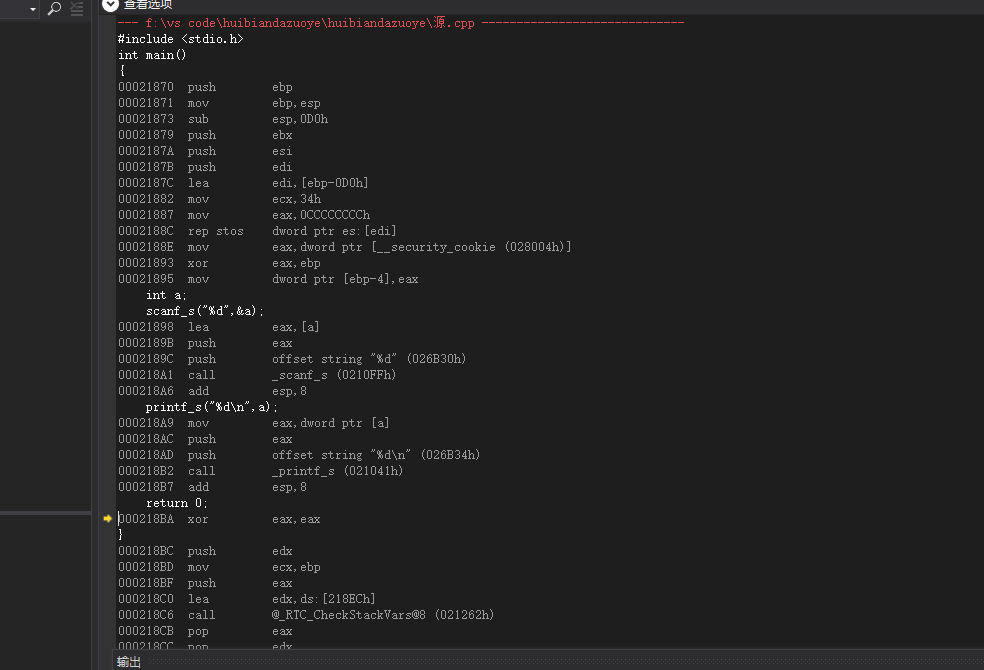
C语言的每一句以；结尾 汇编语言每一句没有结尾

10.第一条语句标识的 区别

C语言的入口是main函数，汇编语言的入口可以是任何一个函数。

二，自编C程序,对其执行文件反汇编，深入分析并阐述C语言各要素的底层实现

1．



这个例子里面，我编写了一个简单的，输入数字，打印数字的功能，里面根据反汇编的结构，可以看到汇编语言是怎么实现的，在进入main函数之后，进行了一系列堆栈的操作，目的就是为了保证程序可以正常的返回，scanf这个函数的汇编语言实现是通过，将变量push到 堆栈里面，然后调用c的scanf进行操作，最后进行 堆栈平衡，add esp 8。这个例子说明了主函数也和子程序一样是在堆栈中，进入主函数要进行push ebp，move ebp esp 等操作，和子程序一样。

这个例子可以 很好的说明c程序是怎样在计算机里面运行的，之前我们就已经知道c源代码通过编译器的编译和链接，生成真正的可执行文件，然后系统在调用这个文件的时候，将代码拷贝到内存中，为程序分配数据段，附加段，分配堆栈，然后设置cs，ds，ss，es，设置ip，接下来根据cpu的工作原理，cpu找到的下一条指令就是我们的程序代码，这样我们的程序就真正的开始运行了。

2.我在 内存中申请了几个变量，并且对其进行了一些 初始化的操作，我发现了 一些 很有意思的 东西。

当一个变量没有初始化的时候，在汇编语言中是没有语句与之对应，所以这个时候，cpu根本没有执行任何操作，但是一旦进行初始化的话，相应的汇编语言代码就会出现。

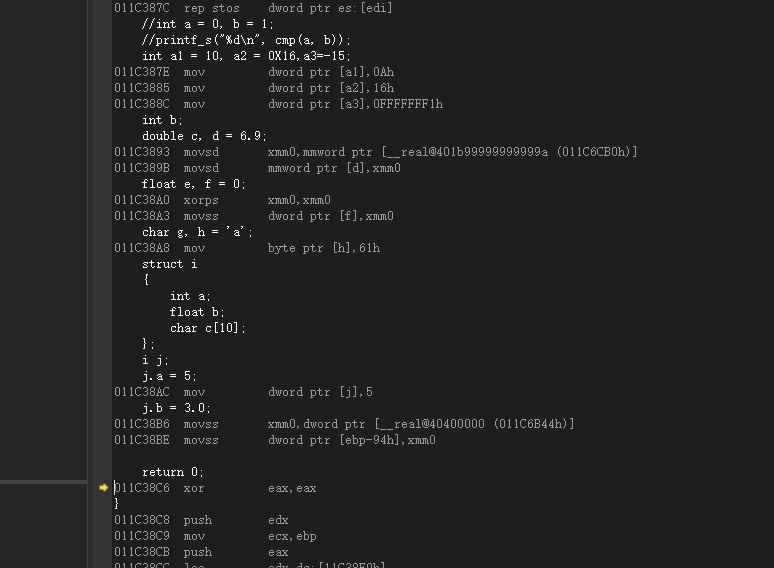
对于int型变量，汇编语言用dword来存储，double用mmword存储。

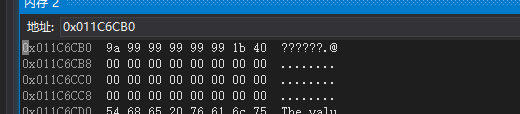
对变量赋初值为0的时候，是先将一个寄存器与自己进行异或操作，接下来使用mov语句进行赋值。

在对整形数据进行初始化的时候，数据直接就在代码段里面，但是对浮点型数据进行初始化的时候，数据的地址与代码的地址并不在一起，所以应该是存在了数据段里面。

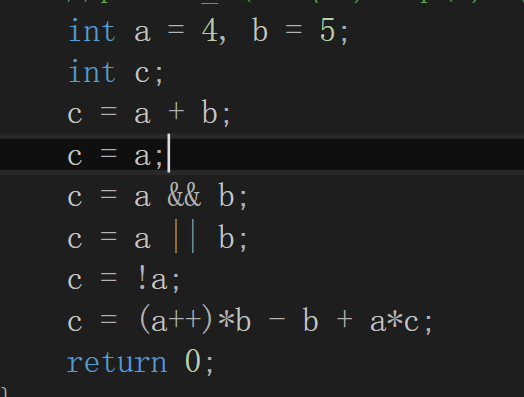
对于不同类型的数据，编码方式不同，对于int型数据，无论是16进制，还是十进制数据，在内存中都是16进制的数据，int型数据的负数采用反码表示，所以-15的二进制实际是一个非常大的二进制数据。。

在结构体的定义中，情况也类似，声明结构体的时候，不对应汇编语言代码，当对结构体的对象进行赋值的时候 ，实际上是对结构体在内存中的地址进行操作。



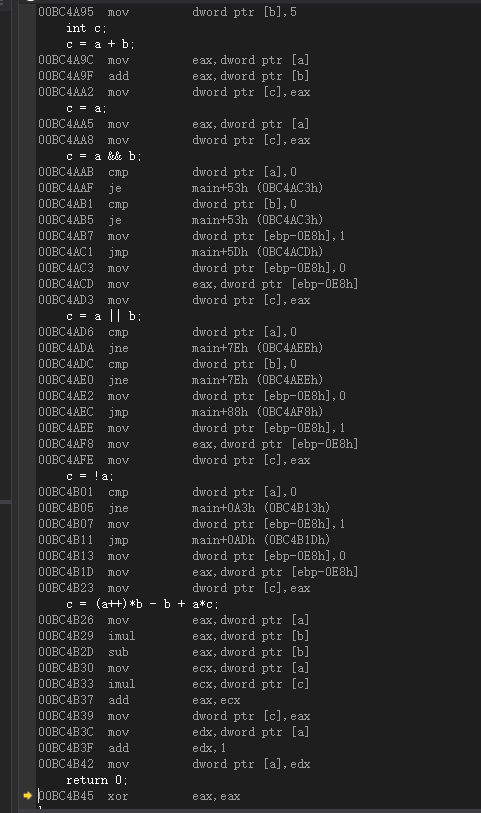


3.



这个是我写的主函数，在函数体里面进行了一些很基本的简单操作，下面是反汇编的结果，当执行加法操作的时候，就是用三条汇编语言语句进行的，赋值语句就是借助于寄存器，将数据先读到cpu里面，然后再把数据写入到目的地址里面。进行比较复杂的操作的时候，汇编语言就是 按照运算的优先级，一次一次的执行。c = (a++)\*b - b + a\*c

下面以与操作为例，介绍一下与或非操作的实现，执行与操作的时候，汇编语言依次判断两个数据是不是0，如果存在0的 时候，跳转到相应的语句，使得结果是0，否则的话，将结果赋值成1，然后跳转，将结果赋值给c变量。

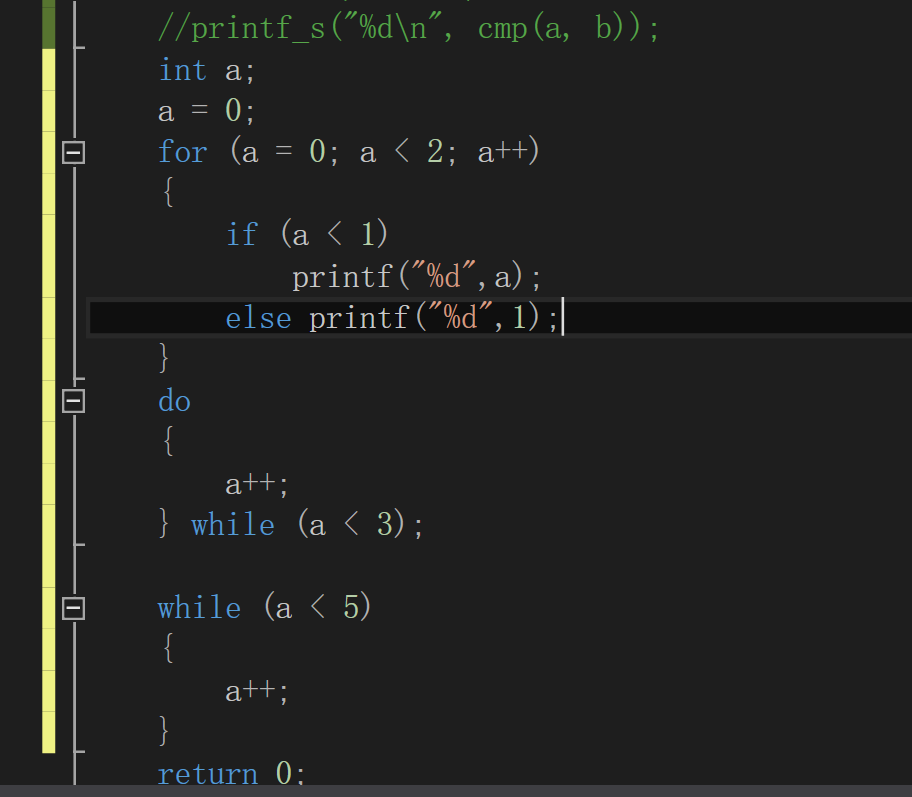


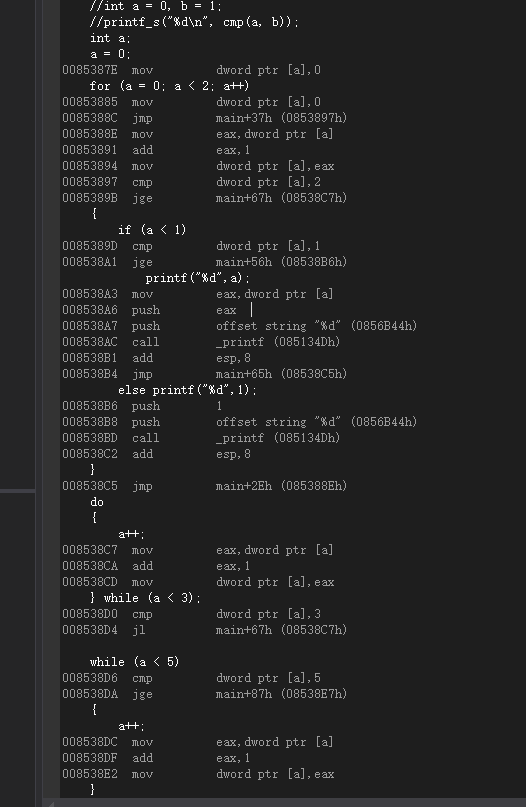
4.在这个程序里面 ，我编写了各种结构，有 if else的判断结构，有三种循环结构，下面分别是源码和反汇编的结构 ，下面我简要 分析一下。

在第一部分的for循环和if判断之中，汇编语言是通过一系列的判断和 跳转语句来实现循环和判断结构的， 在每一个for循环中，先进行a与2的 比较，如果大于等于的话，跳入下一行执行语句，否则的话，进入函数体，在函数体结束的地方返回，然后执行a的更新操作，接下来继续判断，循环往复，直到不满足循环函数。

For语句的执行时，就是进行判断，然后跳转到需要执行操作的地方。

Do while 和while do循环的原理类似。



5.编写了一个非常简单的函数，目的是测试函数的调用。

在 主函数里面，调用函数的时候 ，先将参数压栈，然后调用函数，最后进行堆栈平衡。

在 子函数里面，先进行一些规范化的操作（push ebp； MOV ebp esp）

然后进行寄存器压栈，保护现场，然后执行这些语句：

lea edi,[ebp-0C0h]

mov ecx,30h

mov eax,0CCCCCCCCh

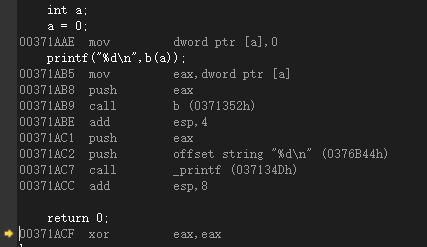
rep stos dword ptr es:[edi]

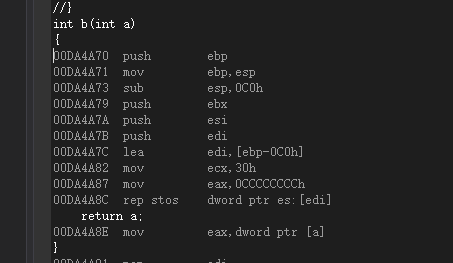
rep指令的目的是重复其上面的指令.ECX的值是重复的次数.

STOS指令的作用是将eax中的值拷贝到ES:EDI指向的地址.

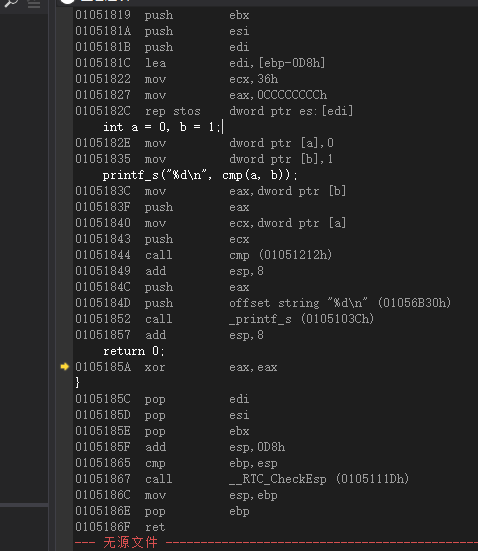
这些语句的目的是初始化函数的堆栈空间，将所有的内容赋初值0CCCCCCCCh

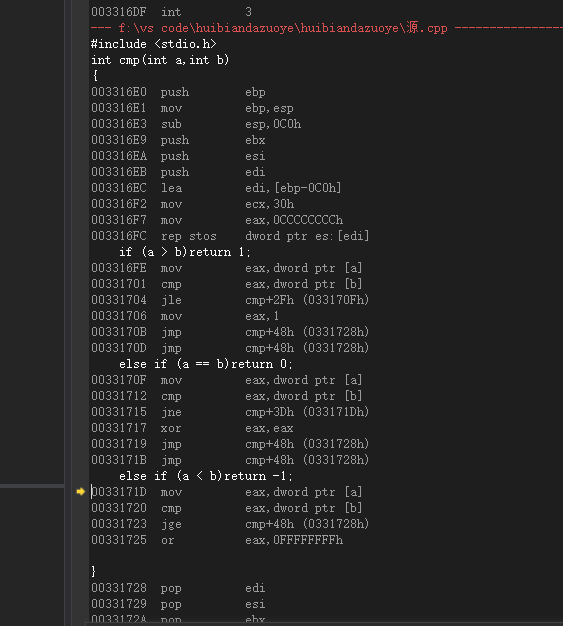
然后接下来 就是 函数体里面的操作 。





6.在这个程序里面，我在主程序里面写了一个驱动程序，然后调用了 一个cmp的函数，下面是反汇编的结果。

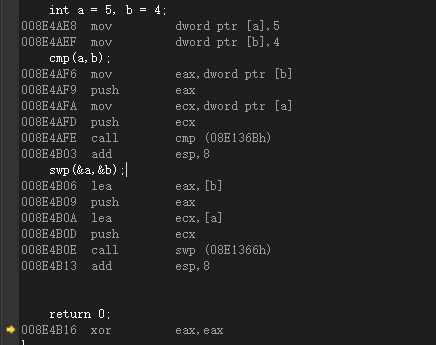


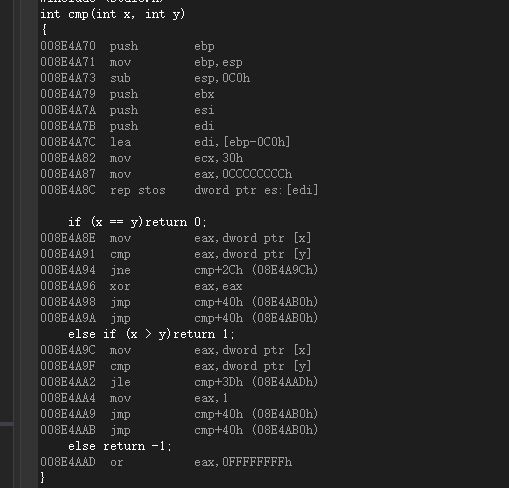


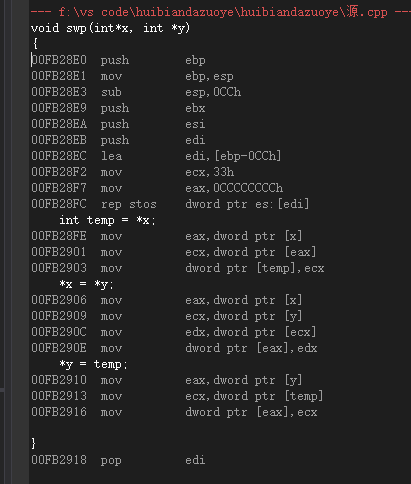
7.

实现了一个简单的比较函数和交换两个值的函数，通过反汇编可以看出，传值的时候，汇编语言是将值的副本放到了堆栈段里面，传地址的时候是将变量的地址取出，放到了堆栈里面。

在访问这个变量的时候，可以根据变量名进行访问。当传进去的是指针的时候 ，可以根据变量名访问变量的地址，找到 地址之后，就可以进行读写操作。

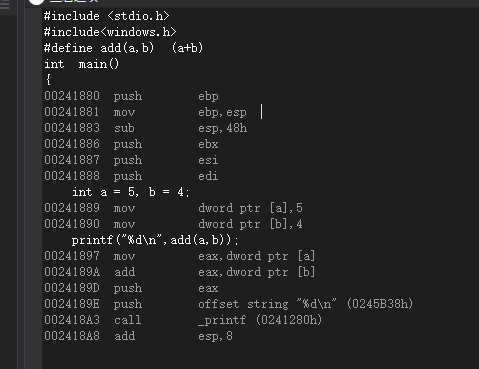






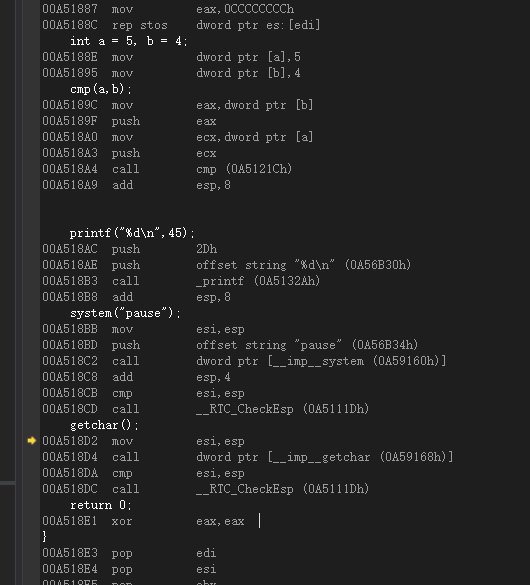
8.宏的本质

我写了一个宏定义的语句add（x,y）所以当编译器遇到add语句的时候，就会将其编译成定义好的语句，x+y，所以对应的汇编语言语句就是x+y的语句。所以宏的本质就是替换，遇到有宏的地方，就找到相对应的语句，执行。



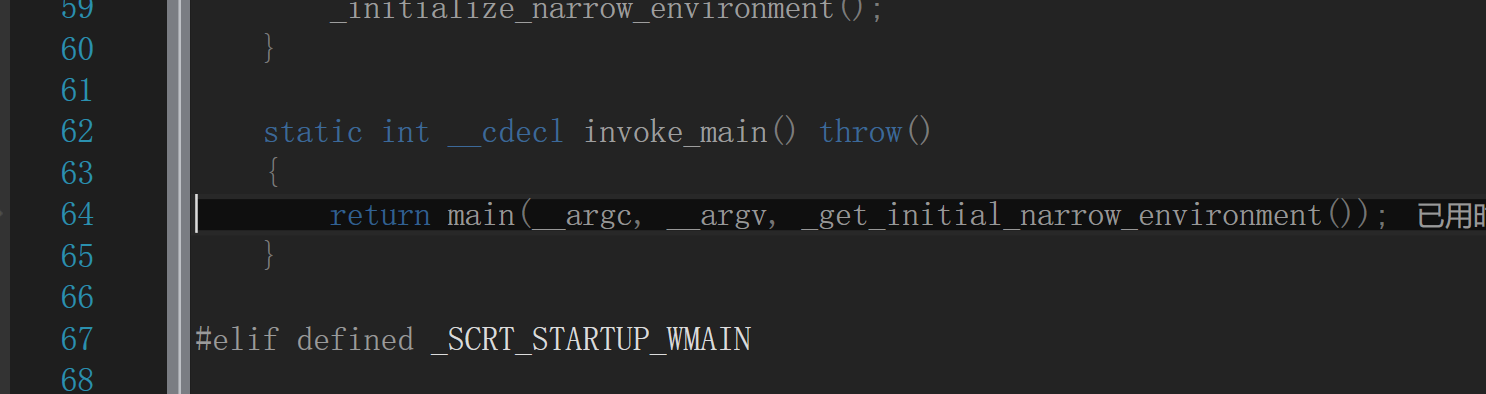
9.

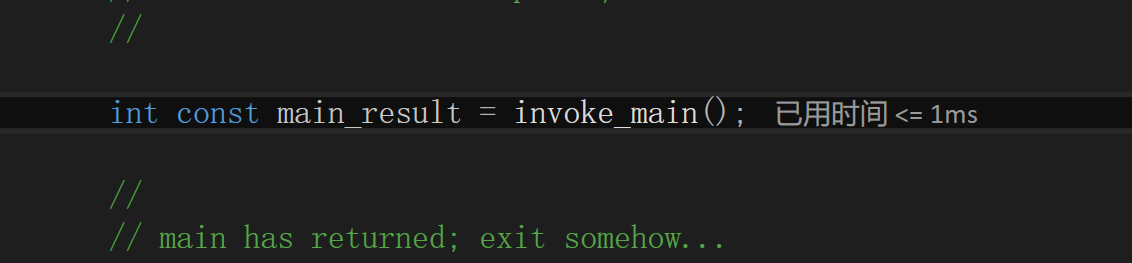
编写一个主函数，调用一个外部函数，调用一个c语言库函数，调用一个系统函数，最终发现，调用的过程没有什么太大 的区别。都是在调用之前，将变量push进去，然后调用函数，最后add esp 4\*变量数，平衡一下堆栈就行了。

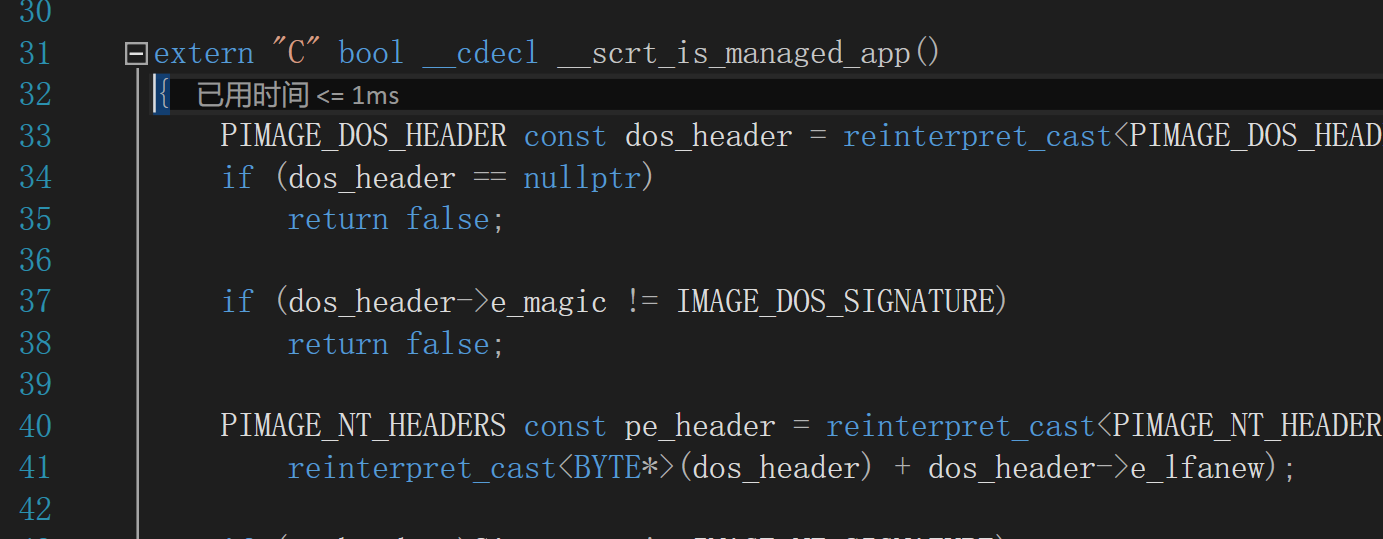


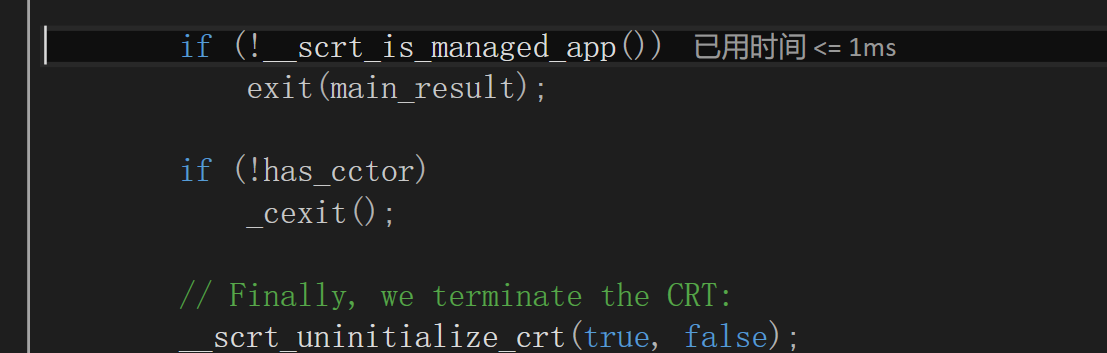
9.

在main函数最后一句return 0语句单步调试，我发现一个invoke\_main函数调用了main函数,函数的返回值给了main\_result这个 变量。然后返回到\_scrt\_is\_managed\_app函数，最后执行到exit函数，程序结束。









三、比较C与汇编的优缺点，适应场合

C语言的相比于汇编，“高级”了很多，书写起来很容易，很多事情由编译器来帮我们做，我们只需要关注我们需要关注的算法数据结构那一部分就行了，但是正是因为如此，一个简单的c语言程序，经过反汇编之后得到的 汇编语言程序就让人觉得云里雾里，摸不着头脑，这是因为编译器帮我们干的事情，干的并不好，干的不高效，但是这个事情是无法避免的。但是要是我们自己写一个实现相同功能的汇编语言程序的话，确实可以得到执行效率很高的代码，但是同时会花掉很多的时间，而且一旦更换机器，汇编语言就要重新写。

所以，综上，c语言的执行效率不如汇编，代码清晰易读，但是开发效率要高很多，并且代码可具有可移植性。相反，c语言的缺点就是汇编语言的优点，汇编语言的执行效率非常高，直接对cpu进行编程，这是其它语言可望而不可即的，但是同时带来了不好的事情，程序难以阅读，程序开发起来很困难，难以编写大规模程序，而且代码难以移植，并且要求 程序员对计算机的底层了解的非常清楚，否则是写不好汇编语言代码的。

根据这些优缺点就可以判断他们的适用场合了，显然，汇编语言代码适合对速度要求非常高的情况，在某些特殊的场合，必须采用汇编语言编程，比如编写驱动程序。但是对于开发速度要求高，但是对执行效率要求没有那么高的情况下，可以采用c语言进行编写。我们在编程的时候，可以进行c和汇编联合编程，某些执行大量计算的语句可以采用汇编来写，其他的采用c语言进行编写。这样的话，就可以结合二者的长处，避免二者的短处。

四、总结与分析，顿悟与畅想

通过学习 这门课程，我的收获还是很大的，首先通过前几章节的学习，我对汇编语言的语法和特性有了一些认识，后来学习了数据如何在机器中存储，加上我之前学习的数字逻辑方面的知识，使得我对于计算机的工作原理有了更深层次的认识，再后来学习了计算机的组成，介绍了汇编语言的语法，和汇编语言的变量，常量，表达式，程序结构，反汇编方面的知识，使得我对汇编语言有了一个较为深入的认识，最后老师又介绍了不同平台下的汇编语言，以及汇编语言的代码优化问题，我没怎么听明白。

在学习汇编之前，一直不清楚cpu到底是如何工作的，通过这节课的学习，了解了一下cpu的工作原理，使得我明白了电脑到底是怎样的一个系统。各个部分之间是如何协调统一进行工作的。

以后如果不从事硬件方面的工作的话，汇编语言对我们的主要作用就在于我们可以更加深层次的 了解我们的程序，帮助我们提升执行效率，或者我们有时候会进行一些联合编程，这样的话，可以大幅提升程序的执行效率。

五、本课程你的其他收获及希望

我根据我自身的经历来说，汇编语言的学时太少了，时间过于紧张，所以我建议老师开放一个网上的mooc平台，这样的话，我们可以随时随地的学习，现在我觉得，汇编语言的难度 确实很大，要想学好，真的要花很多时间。所以希望老师可以争取尽量多的学时。