**汇编语言大作业－C语言的秘密**

**耿 飞**



院 （系）：计算机科学与技术学院 专 业：计算机科学与技术

学　　号：1140310413 指导教师：史先俊

**2015年11月**

**目 录**

**第1章 比较C与汇编的语言要素**………………………………………………………………….1

1.1 数据类型比较……………………………………………………………………………………….1

1.2 常量及变量及其表达式的比较………………………..…………………………………….1

1.3 赋值及算数运算等的比较………………………..…………………………………….2

1.4 子程序（函数）的相关…………………..………………………..………………………………….3

1.5伪指令与宏……………………………….………………………..…………………………………….4

1.7向外部提供变量或子程序 ，使用外部变量和子程序………………………………….5

1.8头文件：.inc 与 .h的内容区别，格式区别………..…………………………………….6

1.9源程序模板………………………………..………………………..…………………………………….6

**第2章 C语言各要素的底层实现秘密**…………………………………..……………………….8

2.1 C程序是怎么在计算机运行的…………….……………………………………………8

2.2类型等相关的本质………………….………………………..…………………………………9

2.3变量的本质……………………………….………………………..…………………………………10

2.4操作的本质……………………………….………………………..…………………………………10

2.5程序结构的本质………………….………………………..…………………………………11

2.6函数的本质……………………………….………………………..…………………………………12

2.7参数的本质……………………………….………………………..…………………………………14

2.8宏的本质……………………………….………………………..…………………………………15

**第3章 C与汇编的优缺点及其适应场合** …………………………………………………16

**第4章 总结与分析，顿悟与畅想** ……………………………………………………………17

**第5章 本课程的其他收获及希望** ……………………………………………………………18

**一．比较C与汇编的语言要素**

1.1

1.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编语言数据类型 | C语言数据类型 | 表示范围 | 所占字节数 |
| BYTE | char | 0~255 | 1 |
| SBYTE | unsigned char | -128~127 | 1 |
| WORD | short | -32768~32767 | 2 |
| SWORD | unsigned short | 0~65535 | 2 |
| DWORD | int | -2147483648~2147483647 | 4 |
| SDWORD | unsigned int | 0~4294967295 | 4 |
| QWORD | long long | -9223372036854775808~9223372036854775807 | 8 |
| REAL4 |  | 1.18E-38~3.40E38 | 4 |
|  | float | -3.40E-38~3.40E38 | 4 |
| REAL8 |  | 2.23E-308~1.7E308 | 8 |
|  | double | -1.7E-308~1.7E308 | 8 |
| REAL10 |  | 3.37E-4932~1.18E4932 | 10 |
|  | long double |  | 取决于编译器 |

在以上的数据类型中，在整数方面，我们可以找到c语言和汇编的很好的对应，但是在浮点数方面，汇编与c语言有较大差别

1.1.2

指针：在c语言中，我们利用int \*p来定义一个指向类型为整数的指针，然而在汇编中，没有这种方式，但是表示地址的方式有很多，比如[],寄存器,标号等等

多维数组：c语言中直接可以定义多维数组，例如:int a[5][5]，即定义了一个5\*5的二维数组，而在汇编语言中，采用基址+变址寻址即可实现二维数组

而像在c语言中，类、结构、联合、自定义类型等这些复杂类型，若要在汇编语言中实现，将及其复杂。

1.2

1.2.1常量

汇编语言中，常量大致有三种形式：

其中，数字常量通常指立即数，即在代码段中直接使用的数字

例如：MOV AX 08H，08H便是常量，可以是多种形式，2进制，十进制，十六进制，也可以是字符串等。

也可以是符号常量，如经过EQU，＝等伪指令定义过后的字符

也可以是数值返回算符，例如OFFEST产生的数字都是立即数

而在c语言中，常量通常也有三种，与汇编语言类似，但是在符号常量的定义上面，和变量类似，只是前面在定义时加上了const，例如const int a=9就定义了一个值为9的int型常量。

1.2.2

常量表达式，是一个算式，算式的组成部分均为常量

1.2.3变量

在汇编语言中，变量在.data代码段定义，而在c语言中，变量的定义非常灵活，可以在任意位置，随用随时定义，甚至可以在for循环的过程当中定义，例如，for (int i=1;i<=10;i++)，i即为一个只在当前循环起作用的变量。

其中，在.data定义的的变量为全局变量，c中定义全局变量的方法为只要是在非结构或过程中定义的变量均为全局变量。

汇编语言中，寄存器是特殊的全局变量，可以供所有的子程序使用。

两者的局部变量类似，都是在子过程中定义的变量，只能在子过程的使用，子程序外无法调用。

C语言中，有静态变量一说，就是在变量的定义前面加上static即可。

1.2.4 变量表达式

这是汇编与c语言差别非常大的地方，在汇编语言中，变量表达式只能通过寄存器来实现，即在寄存器基址＋变址寻址的时候，可以有变量表达式，内存变量是不能够直接参与运算的，而在c语言中，就灵活的多，变量表达式多种多样，和我们正常的数学书写类似，且变量的组成也可以是多种多样，而且即使变量的类型不一致，c语言也会进行强制类型转换使其满足运算规则。

1.2.5条件表达式

在汇编语言中，条件伪指令和c语言中的条件表达式类似，几乎没有分别，而在汇编中普通的条件判断语句则很有限，例如 je ax等。而在c语言中，条件表达式的组成多种多样，变量表达式也可以用于条件表达式，其中如果变量表达式的值为0，则结果为假，否则为真。

1.3

1.3.1在汇编语言中，.data中可以进行变量的初始化，例如n dword 0,如果不需要初始值可以用？代替，数组的初始值，buf byte 16 dup (?)，在c语言中，初始值可以直接用=，例如：int a=10;数组的初始值，int a[10]={0,1,2};则将a数组的前三个值分别赋为0，1，2。

1.3.2在赋值当中，c语言和都可以用等号进行赋值，汇编语言中的等号赋值为伪指令，全局变量和局部变量都类似

1.3.3在算数运算中，汇编语言可以有和c语言类似的伪指令操作，而标准的汇编语言算数运算比较繁琐，例如ADD AX，10，而对这些指令，汇编语言都有一些要求比如对于mov指令：

mov reg , imm/reg/mm/sreg

mov mm , reg,sreg,imm

mov sreg , reg/mm

其他的一些指令都有自己的要求。

而在c语言中，算数运算则比较自由，没有太多的要求，和我们正常理解的数学运算比较相似。

1.3.4逻辑指令

|  |  |  |
| --- | --- | --- |
|  | c语言 | 汇编语言 |
| 与运算 | && | and |
| 或运算 | || | or |
| 非运算 | ! | not |
| 异或运算 | ^ | xor |

1.3.5分支转移指令

在汇编语言中的操作伪指令类似与c语言中的分支判断

if （）

else if（）

else if（）

else

而在汇编语言中为.IF 条件语句

.ELSE

.END IF

汇编语言中判断条件无需加括号

1.3.6循环语句

汇编语言和c语言的循环语句差别较大，c语言的实现：

for (i=1;i<=10;i++)需要有三个部分控制整个循环，第一部分为循环变量的初始值，第二个部分为循环的执行条件，第三个为循环变量的改变

而在汇编语言中，除了伪指令只能采用标号跳转的方式实现。

1.4子程序

以下是一个汇编语言的例子：

incp proc c public uses esi edi ebx ecx, p : ptr dword

mov ebx, p

mov eax, [ebx]

add eax, 1

mov cl, 16

div cl

mov cl, ah

xor eax, eax

mov al, cl

mov[ebx], eax

ret

incp endp

下面是一个c语言的例子：

bool can(int a,int b,int k)

{

if (a+x[k]<1||a+x[k]>n) return false;//行出界判断

if (b+y[k]<1||b+y[k]>m) return false;//列出界判断

if (s[a+x[k]][b+y[k]]=='#') return false;//障碍判断

if (flag[a+x[k]][b+y[k]]) return false;//重复判断

return true;

}

在汇编语言中，子程序有多种形式，proto invoke，proc call等等，在这个例子中，为invoke形式的子程序，在调用的时候

mov edi, [ebp + 8]

mov esi, [ebp + 12]

add edi, [esi]

mov eax, [ebp + 16]

mov[edi], al

invoke incp, esi

需要将使用的值传入寄存器中，来实现了参数的传递，返回时，需要的值也在程序员自己设定的寄存器中，在这种模式中不需要进行堆栈框架的建立与拆除，是编译器自动进行的，而在其他模式中，我们还需要对现场进行保护，对堆栈框架的建立和拆除。

在c语言中，调用时if (can(dl[l].x,dl[l].y,i))，整个函数表达式作为返回值可以直接使用。

在c语言中，这个例子就是传值，大多数时候c语言函数中参数传递都是按照值传参的，少数情形下，如果需要直接使原址的值改变的话，可以采用地址传参，例如：void swap(int \*a,int \*b)

在调用的时候，swap(&a,&b);

这样就可以实现地址传递参数了。

1.5

汇编语言和c语言中的内联函数类似，不是真正的函数，而是在源代码级别进行的文本替换，在编译的过程当中，都会有宏展开和内联函数的还原，然而c语言的内联函数更为灵活，与函数的定义仅仅有一个区别

例如int gcd(int a,int b) 和 inline int gcd(int a,int b)

只在正常函数的前面添加inline 就可以实现内联函数的定义，在操作等和正常的函数几乎没有区别。

宏的声明：  
 宏名 MACRO [形参表]

宏定义体

ENDM

宏名是一个标识符，互相之间不能重名，也不能和变量、标号、段名等标识符重名。

宏的声明仅仅是来告诉程序，将来宏调用的时候复制的对象是什么，宏中的程序段并不是程序的一部分，也就是说，如果程序中定义了一个宏而没有去调用它，编译器将会忽略宏。

宏可以写在程序的任何地方，但习惯将宏声明写在程序最前面。

宏的调用：

在宏后，就可以在源程序中调用宏了。宏调用的方式是在源程序中需要复制宏体的地方写宏的名字。宏名单独占一行，当源程序被汇编时，汇编程序将对宏调用进行宏体复制，并取代宏名，这种复制操作称为宏展开。

参数的传递与返回：

由于汇编的宏是伪操作，所以在参数传递和返回方面，并没有真正的传递和返回参数，而是通过宏展开而实现，参数并没有真正的传递，传递的实际上是宏的代码。

与宏有关的伪指令

1. 局部标号伪指令LOCAL：

如果宏定义体采用标号，可以使用局部标号伪指令LOCAL加以说明，格式为 LOCAL 标号列表

例： 求绝对值

absol macro oprd

local next

cmp oprd,0

jge next

neg oprd

next:

endm

（2）宏定义删除伪指令PURGE

当不需要某个宏定义的时候，可以把它删除，删除的格式为

PURGE 宏名表

期中宏名表是由逗号分隔的需要删除的宏名。一经删除，后续程序代码部分就不能够调用该宏了。

1.7c语言和汇编语言的相互调用

方式一般有：嵌入式，obj或函数库方式，c与asm源程序级别的联合编程。

虽然c语言和汇编语言可以相互调用，但是大多数情况下都是c语言作为主程序去调用汇编子程序。

c调用asm的变量：

extern “c” 变量类型： 变量名；

c调用汇编函数：

extern “c”函数声明

返回类型 函数名 （参数表）

c为汇编提供函数或变量的方式：

extern “c”｛

……

｝

汇编调用c变量

extern C 变量名：类型

汇编调用c函数

proto 声明

汇编为c提供变量

.data

变量名，类型，初值

public C 变量名，表

汇编为c提供函数

PROTO 声明

Public C 函数名

下面是一些例子：

printf PROTO C : dword, : vararg

scanf PROTO C : dword, : vararg

extern "C" int \_cdecl pq(BYTE\*, int, int);

extern "C" BYTE buffer[16];

extern "C" int ip;

extern "C" void incp(int \*p)

{

\*p = (\*p + 1) % 16;

}

1.8.inc和.h的一些关系

首先，.inc是汇编的头文件，其中内容用汇编语言编写，.h是c的头文件其中内容用c语言编写。

其次，汇编语言无法直接调用.h格式的头文件，而c语言是可以调用汇编语言的头文件的。

在c语言中，.h文件一般是对函数、变量等的声明，在汇编语言中，内容和c语言的头文件类似，其内容是对过程（子程序等）、变量、宏等的声明。

1.9汇编语言和c语言的程序模板

汇编语言的程序模板：

TITLE MASM Template (main.asm)

; 程序描述

; 作者

; 创建日期

; 修改日期

; …

includelib msvcrt.lib

.data

;定义变量

.code

main PROC

;可执行代码

exit

main ENDP

;插入其他子程序

END main

C语言程序模板：

//可以写一些程序说明

//创建时间

//作者

//等等

#include <stdio.h>

//其他需要的头文件

//全局变量的定义

//子程序

int main()

{

/\*

在此插入可执行代码

\*/

return 0;

}

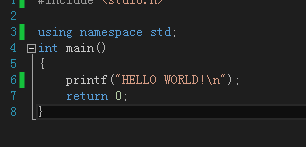
从以上模板可以看出来，汇编语言和c语言的程序模板有类似的地方，整体结构类似，在主要代码的书写之前，都需要做一些准备和说明工作。

在注释方面，汇编语言的注释方法为；单行注释，而c语言有两种注释方式，一个是//可以进行和汇编语言功能相同的单行注释，另外一个可以进行/\*开始，\*/结束的整段注释，很方便，可以用于备用子程序的书写，debug等。

**二．C语言的底层实现**

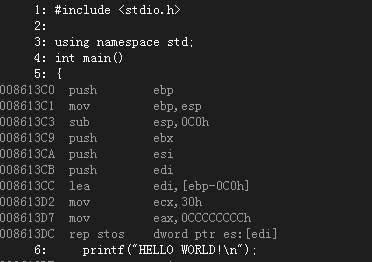
2.1 C程序是怎么在计算机运行的

首先，一个程序要想能够在计算机上运行，那么就必须最后转换为机器语言才可以，c的编译器把我们编写的c语言代码转换为可以在硬件上面执行的可执行代码的过程，就是编译链接过程，比如接下来这个最简单的例子：hello world！

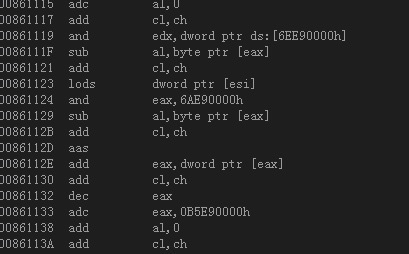
这是一个最最简单的c语言程序了，

实际上只有一行代码是有效的。

现在我们来看一看它的反汇编结果：



在mian函数开始之前，反汇编做了一些准备工作，然而，在我们编写的程序的前面，还有非常长的代码段，这是什么呢：



其实，这些就是编译链接产生的结果，我们调用了一个头文件“stdio.h”

这个头文件中包含了printf 和 scanf等我们经常使用的用于输入输出信息的函数，在编译链接的过程当中，编译器就会将我们用到的头文件中的函数加载到我们的程序的最前面，这样我们就可以调用这些函数了。

编译成功之后，我们就可以运行我们编译成功的程序了，在运行程序的之前，我们的程序代码等是存储在硬盘中的，计算机要想能够运行这个程序，就要就核心的进程，加载到内存当中，这样cpu就可以更快的执行一些指令，我们的程序就可以飞快得在我们的电脑上运行起来了。

2.2数据类型

一个程序若想运行，势必要占用一些系统资源，比如程序在运行时，必须要申请一些内存空间，来为程序的运行提供运算的空间，任意一个数据类型，存储是都会占用空间，这就是类型存储的本质。

所有数据最后存储在电脑中的时候，都是以01的磁性存储的，然而我们的计算机，我们的c语言有如此多的数据类型，有无符号数，不同的进制类型，这些依靠的都是我们自己所制定的编码规则：

首先最基本的规则就是，电脑上面直接存储的数据都是二进制，称为这个数据的机器数，我们用机器数的最高位来表示数据的正负，0为正，1为负，这些都是约定俗成的。比如一个8位的机器数，－3的表示就是10000011。

原码，补码，反码就是计算机最基本的编码方式，这里我们就不加以赘述了。

而对于复杂类型，比如下面这个结构体：

struct mult

{

int x, y;

char str[10];

double v;

};

复杂类型的本质就是通过组合封装已有的数据类型，来实现一个新的数据类型。

2.3 变量

变量储存在内存当中，在变量声明的时候系统会为其申请内存空间，其中线性的存储结构申请的是一段连续的空间，每一个变量都对应一个地址，表示该变量存与内存中的哪个位置，方便与我们调用这个变量，如果想要得到 一个变量的内存，可以用&取地址符号来得到一个变量的地址，例如：

int a; int \*b=&a;

b中存储的便是a的地址。B是一个指针变量，作为一个变量，b也有他自己的地址，只不过b存储的内容是一个int型变量的地址。

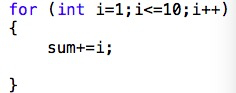
字符串的连续存储的例子：





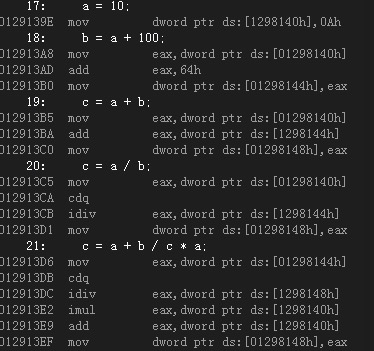
在内存中是连续存储的。

对于全局变量，在整个程序都可以调用，而对于局部变量，只有声明的那一部分可以调用。例如：



i就是一个局部变量，只能在循环体内部起作用，在外部则没有i这个变量。

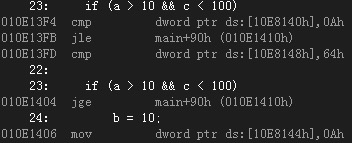
2.4赋值，表达式等操作的本质



从上图我们可以看出来，在c语言中，我们的很简单的赋值语句，在底层实现的实质其实是将不同地址的数据之间进行内容传递。2.5

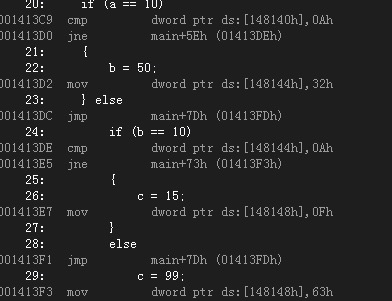
而对于一个复杂表达式，编译器则会将其按照运算的优先级拆分成若干二元运算来进行操作。

对于逻辑运算也是一样的道理：



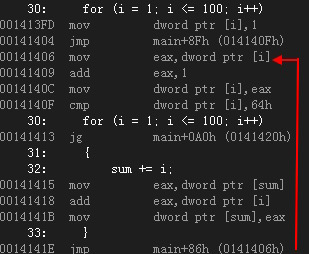
2.5结构的本质

对于顺序操作，实际上与我们的理解也是一致的，是按照顺序一步一步执行的，我们这里就不多加解释了。

对于分支转移指令，我们来看一下下面这个例子：  


从上面我们可以看出来，分支转移指令的实现，是通过jmp跳转实现的。

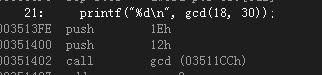
对于循环语句：



同样是利用jmp语句进行跳转实现，在jmp和jmp跳转到的地方之间，便是循环体。

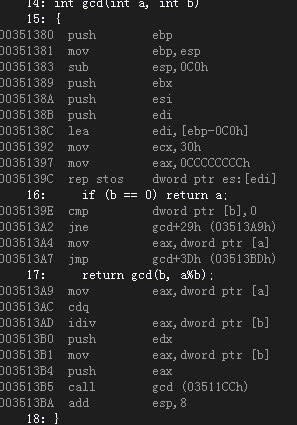
2.6函数的本质

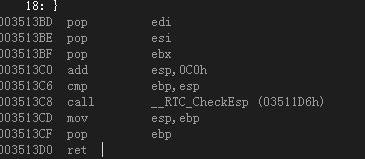
下面便是一个最简单的例子，求两个数的最大公约数：

首先是调用：

我们可以直到，我们所编写的函数，在c语言编译器下是采用call模式的实现的。

下面让我们进入函数的内部：





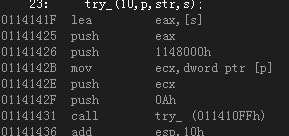
短短的两行代码，在反汇编之后竟然有30多行，原因便是，在函数子程序的执行之前，需要进行堆栈框架的建立和现场的保护，在结束返回之前也同样要进行堆栈框架的拆除和现场的还原。并且这个求最大公约数的函数采用了递归的方法，也让底层实现变得复杂起来。在这里堆栈框架的作用便是进行参数传递和现场的保护与恢复。

2.7参数的本质

首先我们来看这样一个例子，我们定义一个这样的函数

void try\_(int a, int \*b, char \*s, int re[][10])

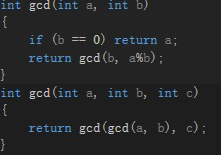
其中分别传递了int型的值，指针，字符串，和一个二维数组，现在我们来看看c语言具体是如何实现这个参数传递的过程的：



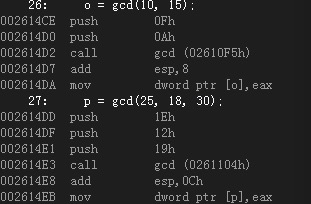
首先参数传递的顺序是从右至左的，对于二维数组s的传递方式为将s的首地址赋给eax，然后通过将eax入栈，其他参量类似，分别按顺序压入堆栈。传送值时，直接将值传送，传送地址时，将原来数据的地址传入，这样在子程序修改这个值的时候，原址的值也会被改变，然而传值则不会。对于字符串，数组等这些变量较多的参量，也是通过传递地址实现的。

下面我们来看一个可变参数函数的例子：

我们仍旧以最大公约数为例，下面是一个2参数或3参数的最大公约数：



让我们来看看分别调用他们的时候，编译器是如何处理的：



我们从上图可以看出来，编译器可以通过参数数量的不同或者参数类别的不同来自动分别出你所调用的函数到底是同名函数中的哪一个，这样就间接实现了可变参数函数。

2.8宏的本质

首先c语言的宏替换有两种，一种是define宏替换，例如#define LL long long

相当于给long long 起了一个别名，功能上和typedef类似，define还可以实现一些简单的简化，比如#define For for ( int i = 0 ; i < n ; i++ ) 这样可以起到简化for循环的作用。另外一个宏替换,内联函数，首先，内联函数在源代码来看，是具有函数结构的，然而在编译之后却不具有，内联函数的本质是宏替换，和汇编语言中的宏替换是类似的，编译后都会在调用宏的位置进行宏展开，从而实现代码的完整性，整个代码都会还原到调用函数的地方，这样以来参数也无需传递，就间接实现了参数传递。

三.c语言与汇编语言的优缺点比较

c语言：

优点：

c语言作为世界上使用最为广泛的高级语言之一，其优点有很多，c语言较之汇编语言的优点，大致也就是高级语言较之汇编语言的优点，使用方便灵活、数据处理能力强、编程简单，而c语言较之其他的高级语言，其编译后的程序运行速度也是比较快的，但当然和汇编语言不能相提并论。

缺点：

在一些对速度要求很高的情况，或者需要对特定硬件进行操作的情况下，

c语言的功能就捉襟见肘了。

汇编语言：

优点：

执行速度非常快，由于汇编语言更靠近底层，执行效率也就非常高。其次，汇编语言可以针对特定的硬件进行编程，能够充分利用硬件优势，发挥硬件的性能。

缺点：

其代码编写非常繁琐，开发的工作量很大，程序的可读性也很低。

其适应场合：

汇编：面对硬件的编程一般采用的是汇编语言，例如单片机，虽然c语言也可以进行芯片级别的编程，但较之汇编语言，还是汇编语言的优势更为明显。还有一种场合就是需要执行速度非常快的场合，汇编语言利用其执行速度非常快的这个特点也可以很好的胜任。

C语言：既可以用于大型软件，大型项目的编写，也可以用于子程序的编写，其功能齐全，执行效率高，可以用于3D游戏，数据库，开发操作系统等。

**四．总结与分析，顿悟与畅想**

通过详细比较了c语言和汇编语言的语言要素，底层实现，和各自的优缺点和使用场合，使我充分了解了计算机语言多样性的必要性，每一种语言都有其存在的合理性和价值，在不同的场合使用不同的编程语言。作为一个程序员，掌握多种编程语言，也是一项不可或缺的技能，如果只在一个方面死守，没有跨平台，跨领域的学习与研究，那么一个程序员的发展空间也会很大。只有广泛涉猎，才能找到自己真正愿意用心钻研，自己真正喜欢的某一个具体的领域。

C语言，作为目前来说使用最为广泛的语言，是一门十分优秀的语言。而汇编语言也在计算机编程方面有着不可代替的作用，通过c与asm的联合编程，我们可以将这两种语言的优势结合，将一个程序写到极致。

**五．本课程的其他收获和希望**

汇编语言程序设计这门课，带给我的收获很多，作为大学第一门靠近计算机底层的课程，汇编语言让我更好的了解了计算机的内部机理，让我更深刻的理解了计算机的内涵。而让我感触更多的，则是汇编老师的负责与认真，可以负责任的说，从上大学到现在上的每一门课，史老师是我见过最负责任也是在课后付出时间最多的老师，在qq群中，每天晚上，老师都会悉心解答每一个同学的每一个问题，真正让我感受到了人文关怀，老师的风趣幽默也让略显死板的汇编课程生动有趣了起来。半个学期以来，收获很大，私有一建议，就是实验和作业的提交时间上面，我觉得可以适当延长一些，可能很多同学都有和我一样的感受，因为这个学期的课业压力比较大，所以课余时间就比较少，而老师安排的实验和作业提交时间都不是特别宽松，也就导致了同学们在做作业的过程当中，往往都是到作业提交的deadline才开始着手做，也就导致了对实验和作业的内容的理解程度不到位，交上去的作业也就草草了事，并没有真正掌握。