

**本科实验报告**

**实验名称：汇编上机作业**

|  |  |  |  |
| --- | --- | --- | --- |
| 课程名称： | 汇编语言与接口设计 | 实验时间： | 2020/4/27 |
| 任课教师： | 张华平 |
| 实验教师： | 张华平 | 实验类型： | √原理验证  □综合设计  □自主创新 |
| 组长姓名： | 董斌 |
| 学号/班级： | 1120173585/07111706 | 专业： | 计算机科学与技术 |
| 学院： | 计算机学院 |



目录

[大数相乘 1](#_Toc40962993)

[1. 实验目的 1](#_Toc40962994)

[2. 实验内容 1](#_Toc40962995)

[3. 实验原理 1](#_Toc40962996)

[4. 实验环境 1](#_Toc40962997)

[5. 实验过程 2](#_Toc40962998)

[5.1 环境配置 2](#_Toc40962999)

[5.2 代码编写 4](#_Toc40963000)

[6. 实验结果 7](#_Toc40963001)

[7. 实验感想 7](#_Toc40963002)

# 大数相乘

## 实验目的

1. 学习汇编编程思想；
2. 了解接口设计过程。

## 实验内容

要求实现两个十进制大整数的相乘（100位以上），输出乘法运算的结果。

## 实验原理

当数值的大小超过了范围限制，就智能通过字符数组进行存储，将其中的数字按照字符的形式获取。

算法流程大致如下：

1. 将数字字符全部转化为整型数组存储起来，方便之后的计算；
2. 处理完之后使用栈对其进行反转，也就是低位在数组的起始位置；
3. 模拟乘法手算过程；
4. 处理进位，然后反转输出结果

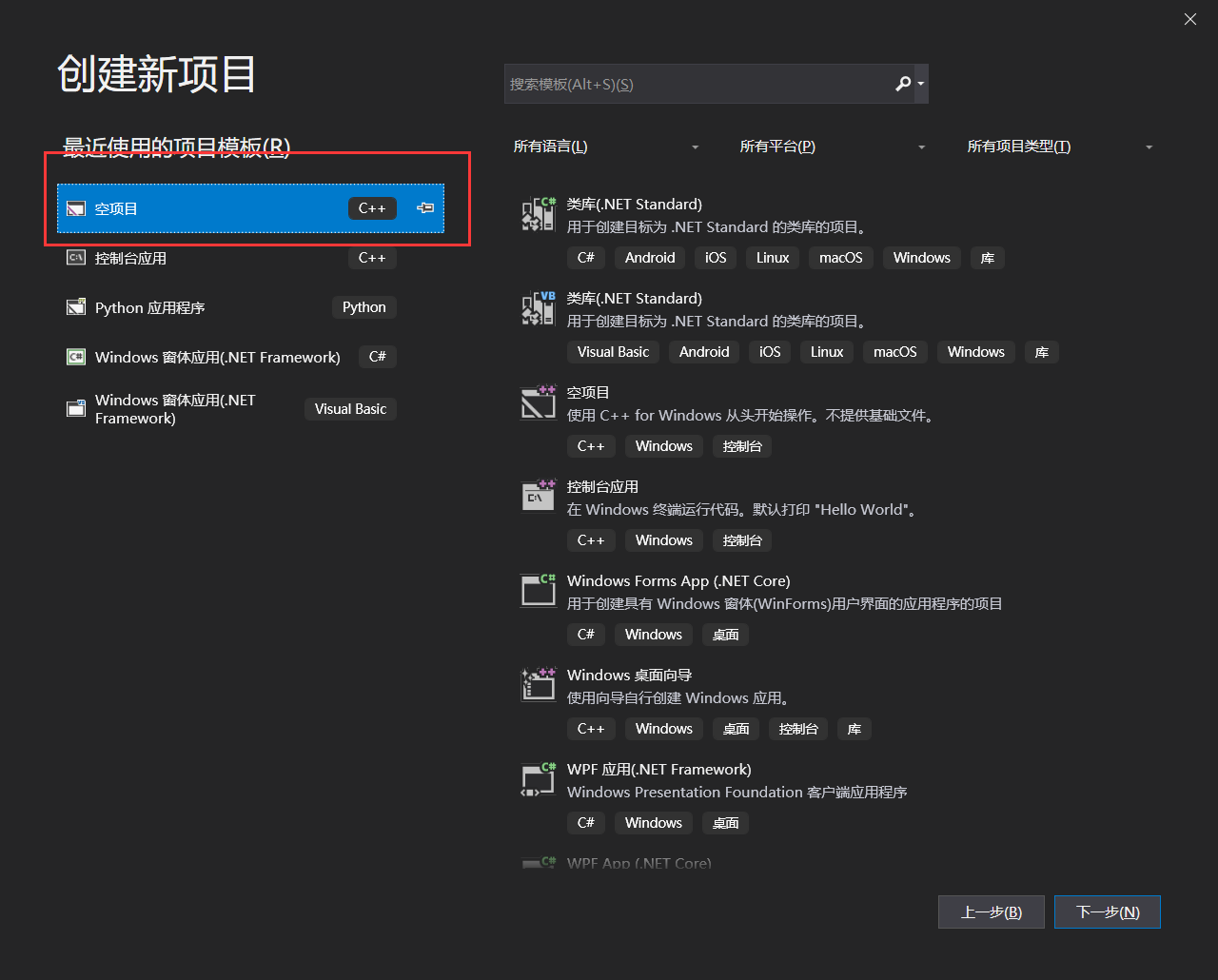
|  |  |
| --- | --- |
| 指标 | 参数 |
| OS  Tool | Windows 10 专业版  Visual studio 2019 |

## 实验环境

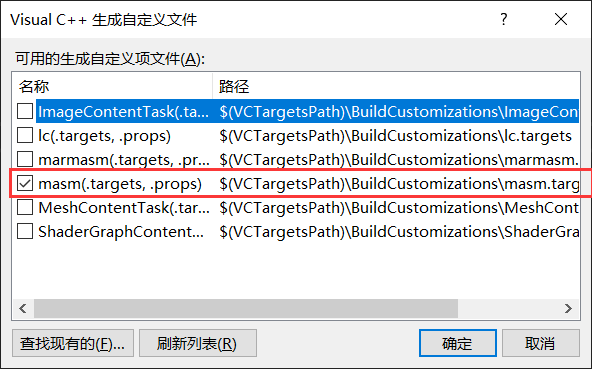
## 实验过程

### 环境配置

1. 创建一个空项目

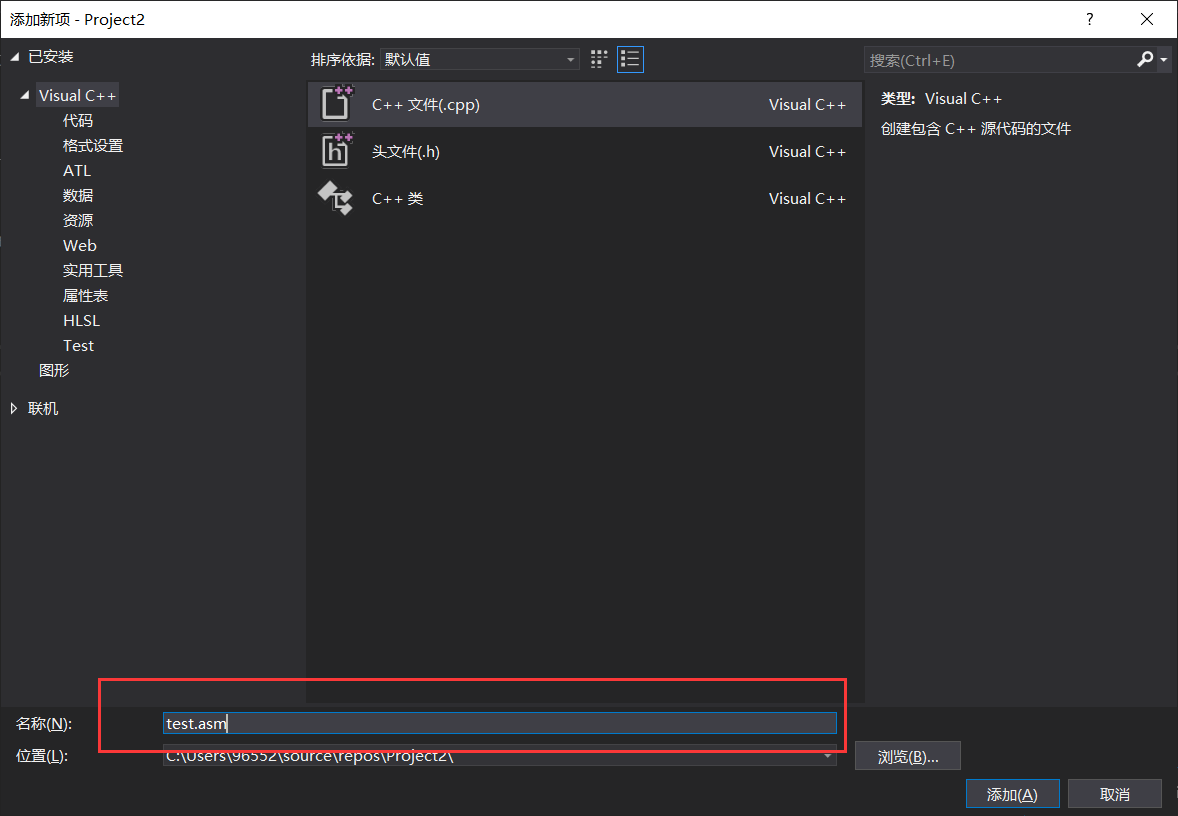


1. 生成MASM规则
   1. 项目右键→生成自定义→勾选masm选项

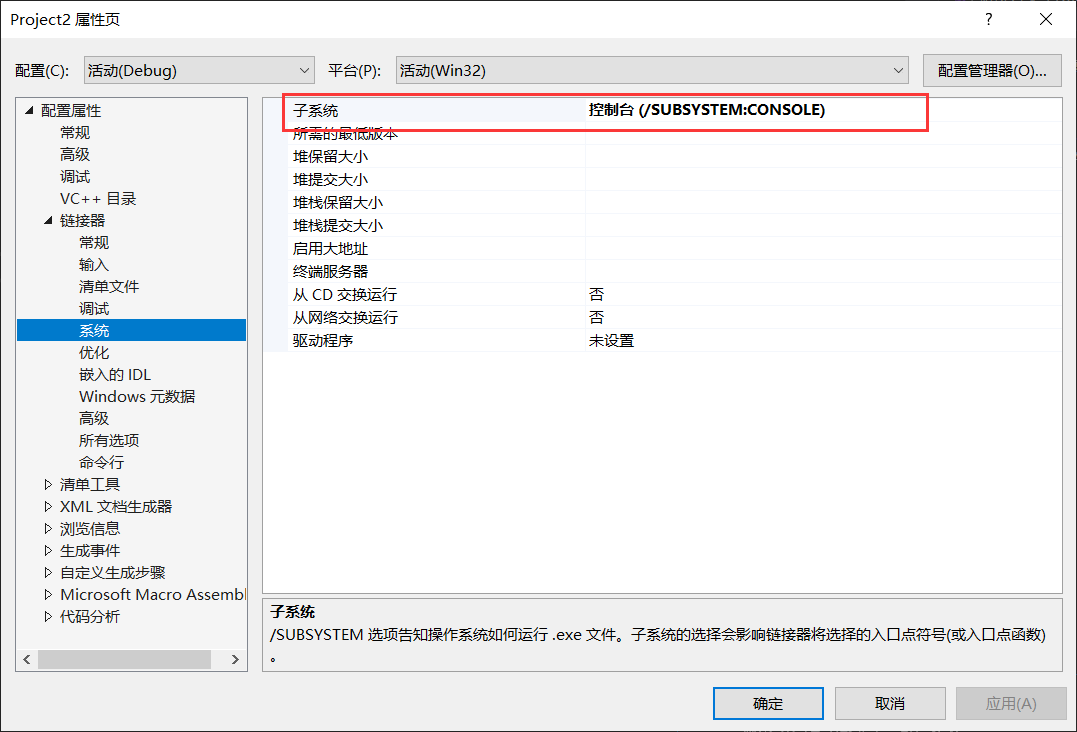


* 1. 添加源文件

选择源文件右键→添加→新建项，新建时将文件的后缀名修改为.asm。遇到的一个坑就是必须要要下面的页面对后缀名进行更改。



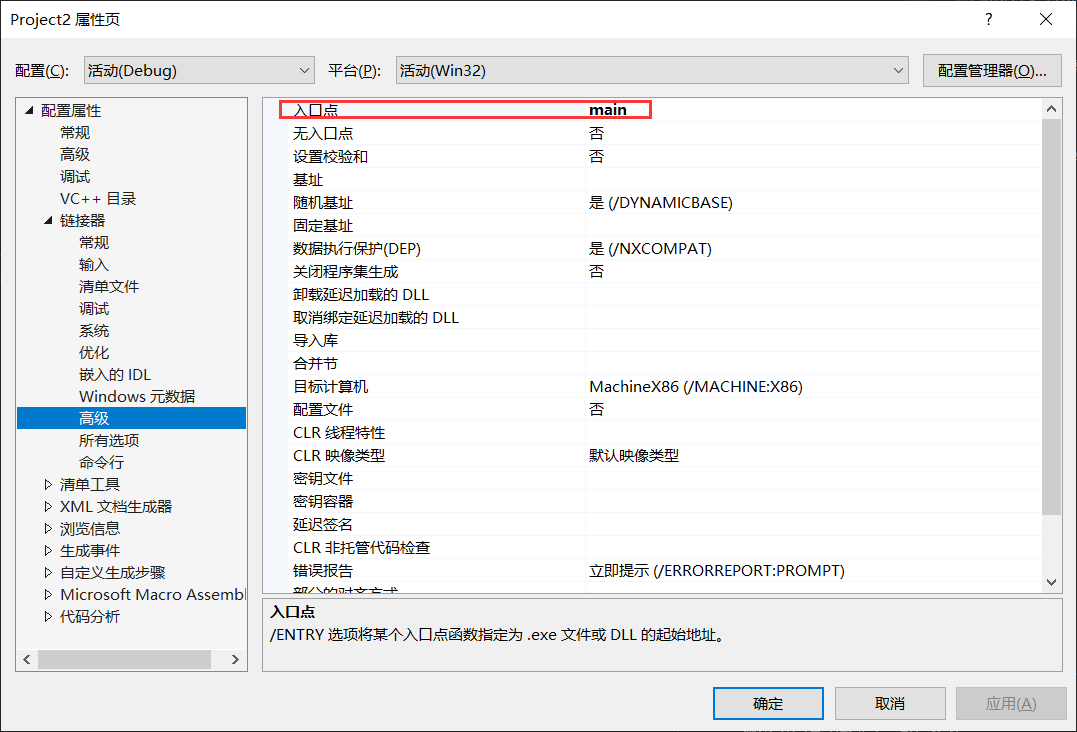
1. 设置属性
   1. 选中项目右键→属性→链接器→系统→子系统→”控制台(SUBSYSTEM:CONSOLE)”→应用



* 1. 设置项目入口点

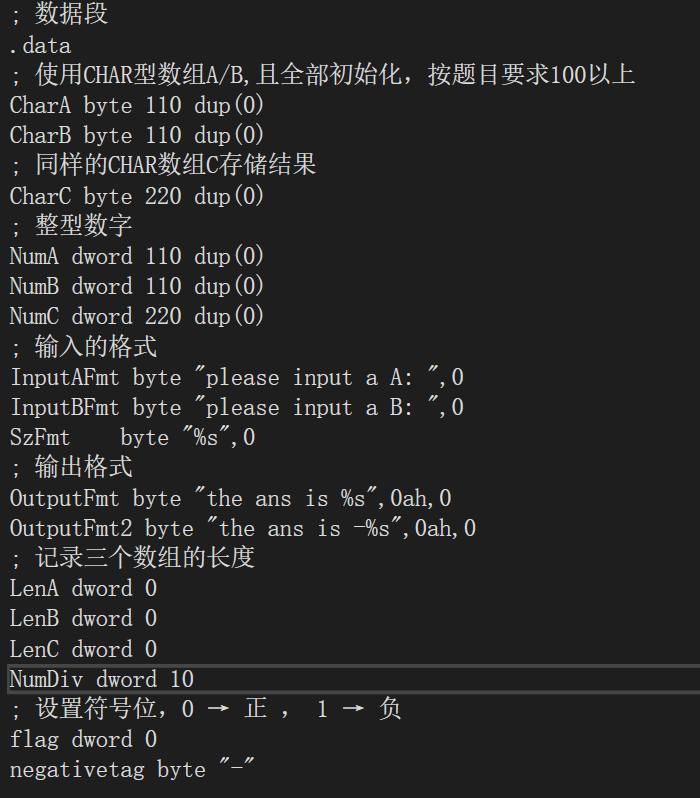
选中项目右键→属性→链接器→高级→入口点→输入”main@0”→应用

注意此项设置完之后，主程序开始程序的关键字只能是Main



### 代码编写

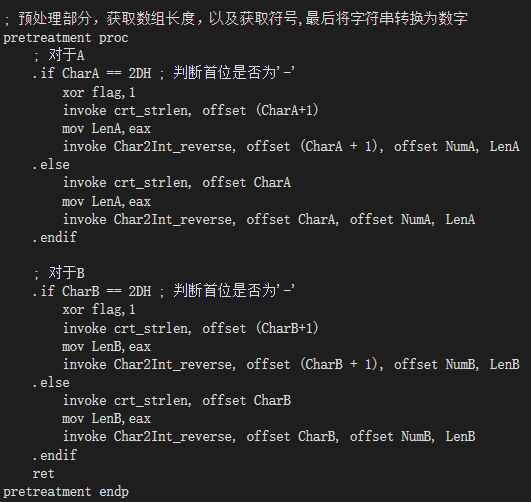
1. 数据段



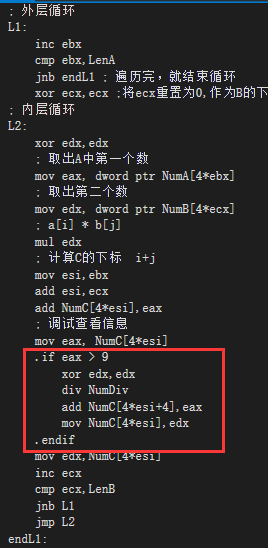
1. 程序入口main函数



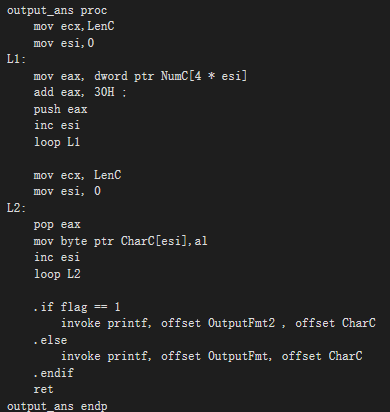
首先就是调用printf、scanf函数获取大数，对其进行预处理，在预处理过程中，首先会将其转化为整型数据并反转输入到整型数组中去，同时在这个过程中判断出乘积后的符号。



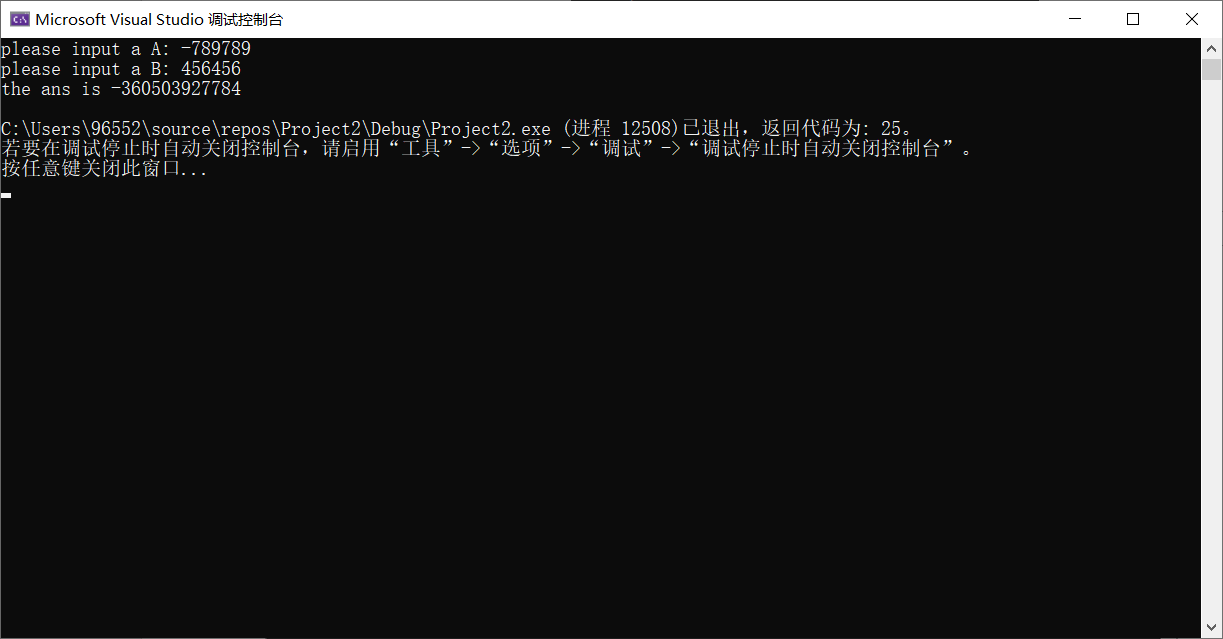
然后就是调用大数乘法，在这个过程中就是通过模拟手算的过程完成逐个的乘法运算。下面仅展示片段（圈住位置是处理进位）：

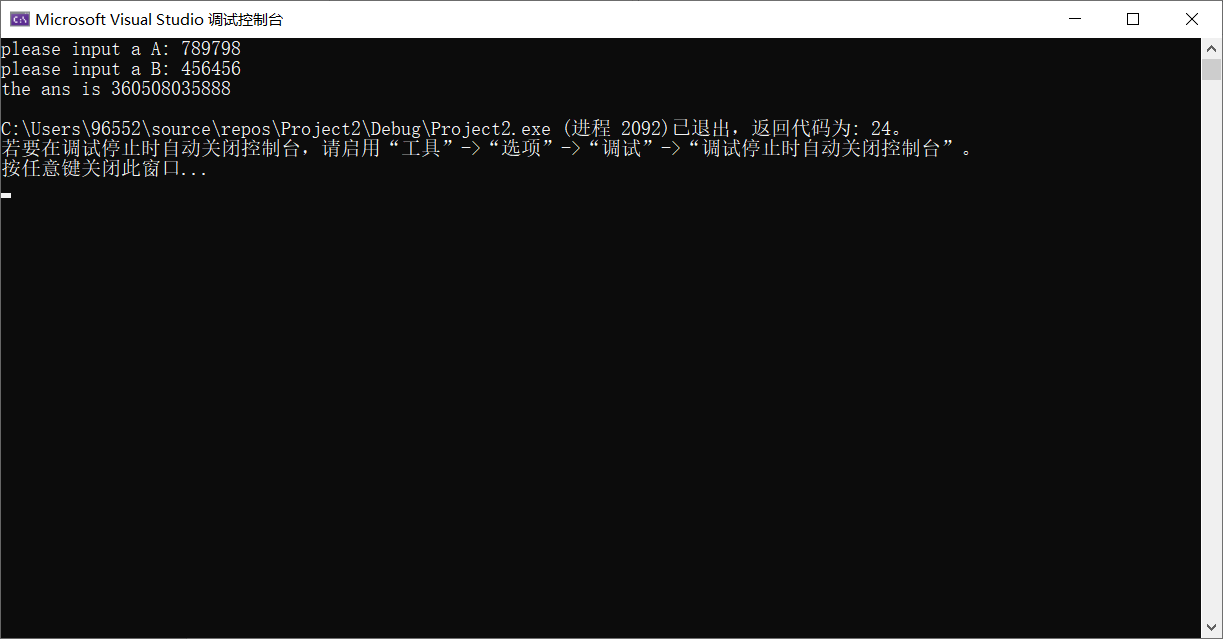


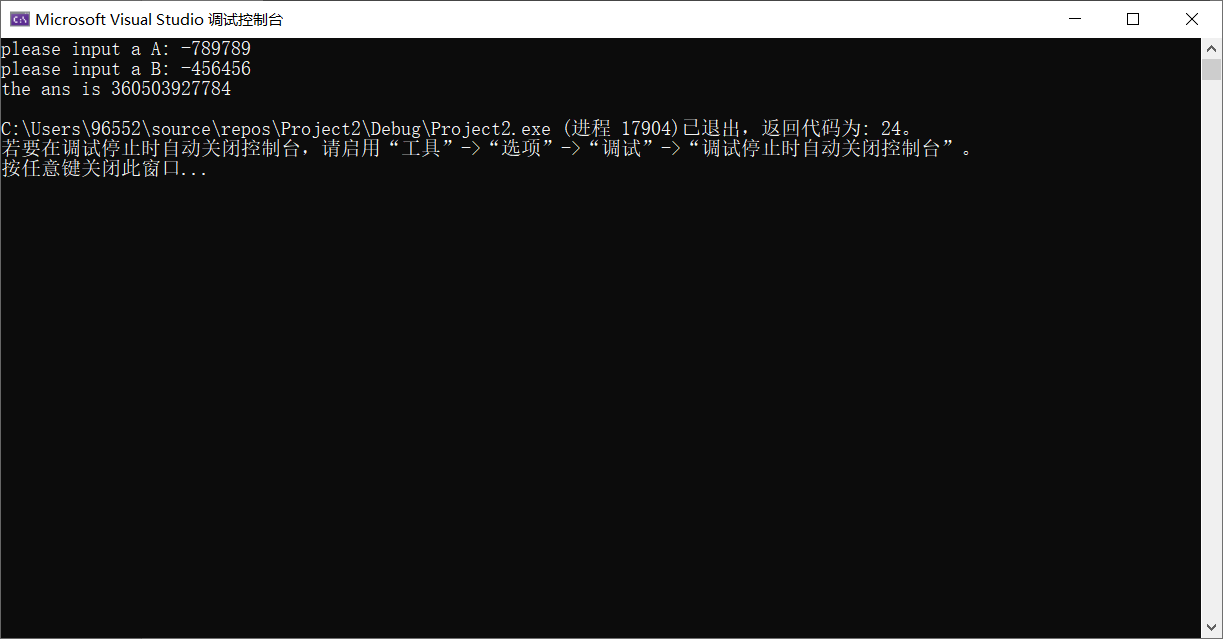
之后就是重新倒置字符，以字符串的形式输出



## 实验结果







## 实验感想

这次实验的思路还是沿用着之前用C语言的大数乘法的过程，在思路上还是比较流畅的，只是在具体操作过程中，遇到很多汇编代码编写的问题，比较如何计算字符串数组长度以及如何使用双层for循环嵌套。

最后在网上查阅相关资料，终于实现了老师的基本要求，通过这次实验，我初步了解汇编语言的编写过程，相信对之后的实验也会有一定的帮助。

# 多重循环程序

## 实验目的

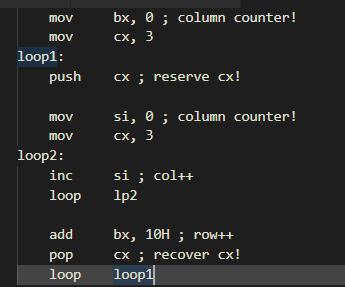
1. 了解循环程序在汇编中实现的过程；
2. 学会节省寄存器使用的方法；
3. 了解汇编中接口的实现。

## 实验内容

C语言编写多重循环程序（大于3重），查看其反汇编码，分析各条语句功能（分析情况需要写入实验报告），并采用汇编语言重写相同功能程序。

## 实验原理

在汇编中实现循环需要使用到loop，涉及到的寄存器有ecx或cx，为了节省寄存器资源的使用，这里使用栈将当前的ecx值保存，压栈和弹栈的顺序刚好符号循环嵌套的顺序，并且只需要push和pop操作，无需程序员位置栈中区域。实例如下：



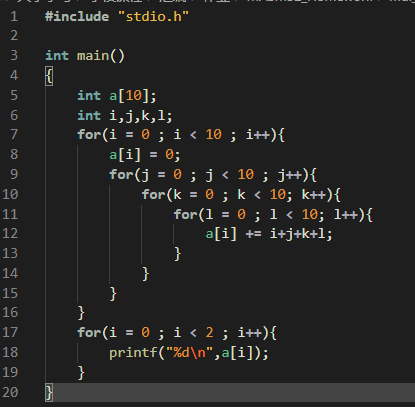
## 实验环境

|  |  |
| --- | --- |
| 指标 | 参数 |
| OS  Tool | Windows 10 专业版  Visual studio 2019 |

## 实验过程

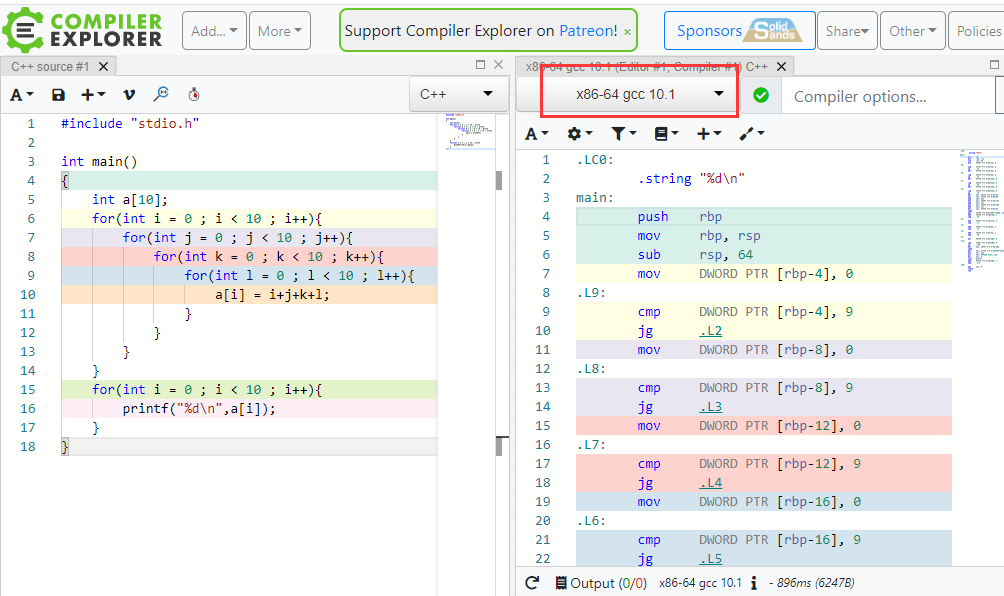
### C语言程序

由于此次实验的目的是为了熟悉汇编中的多重循环嵌套，并且学会使用栈来解决寄存器不足的问题，所以简单设计一个四层for循环，帮助完成实验内容，具体如下：

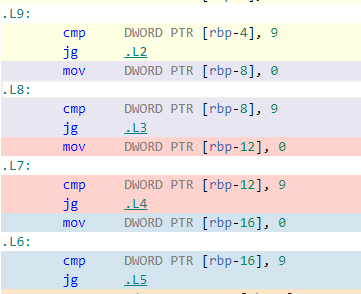


### 反汇编代码分析

这里使用[Compile explorer](https://godbolt.org/?tdsourcetag=s_pctim_aiomsg)进行反汇编，具体操作就是将C语言代码放入网页左边，随后右边即可出现汇编代码，使用的编译器是x86-64 gcc 10.1。



1. 四个for循环



可以看到四个循环的主要部分由cmp，jg，mov三条指令组成，下面对其主要部分分别进行解释：

* cmp

比较指令，将左边的数与右边的数进行比较，该指令影响标志寄存器，当指令cmp ax,bx执行后,会出现如下情况：

* + ax = bx，则ZF = 1，否则，ZF = 0。
  + ax < bx，则SF = 1，否则，SF = 0。

其他相关指令对标志寄存器进行判断即可进行下一步操作。

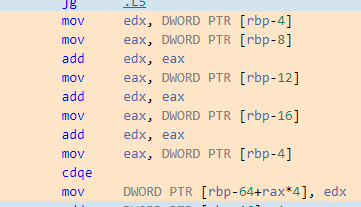
* jg

跳转指令，通过判断上面的标志寄存器来判断是否需要跳转。可以简单理解为当ZF = 1时就会发生跳转，即一个循环的计数位等于9。

知道这些就可以明白，这些指令就相当于判断循环的计数位小于9，则继续执行下一步，如果大于或等于9，则跳转该for循环。对于每个循环的计数位，该编译器使用的是内存的方法，通过rbp给循环的计数位分配内存，然后进行操作，也就是对应的rbp-4等，可以简单理解为[rbp-4] = i，[rbp-8] = j，[rbp-12] = k，[rbp-16] = l；

1. 循环体部分

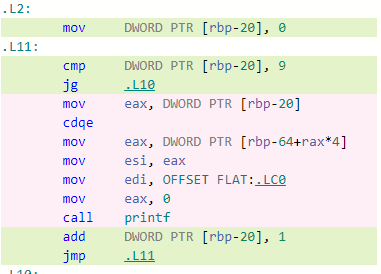
这里为了简化实验，将循环体仅设置为a[i] += i + j + k + l，即每个循环的计数位的和，一个简单的累加的过程，其对应反汇编代码如下：



通过mov和add指令，实现了a[i] += i + j + k + l。

1. 输入部分

从C语言代码上看，输出是一个一层的for循环，其反汇编代码如下：



代码逻辑也是与之前类似，使用cmp判断[rbp-20]的值，循环体中的语句即是将数组中的数据调用printf进行输出。

### 汇编代码实现