一、实验目标

1. 使用 MASM32+Visualstudio 写汇编程序入门
2. 了解汇编语言中对字符串的操作

二、实验要求

用 x86 汇编语言实现以下 C 语言函数功能

unsigned int strlen(char *s);	计算给定字符串的长度，不包括'\0'在内
char *strchr(const char* s,char c);	查找字符串 s 中首次出现字符 c 的位置，返回首次出现 c 的位置的指针，如果 s 中不存在 c 则返回 0
int strcmp(const char *s1,const char *s2);	当 s1<s2 时，返回为-1； 当 s1==s2 时，返回值= 0； 当 s1>s2 时，返回 1。
char *strset(char *s, char c);	把字符串 s 中的所有字符都设置成字符 c

三、实验过程

1. 配置 VS2017+MASM32 环境

1.1 下载安装 MASM32：

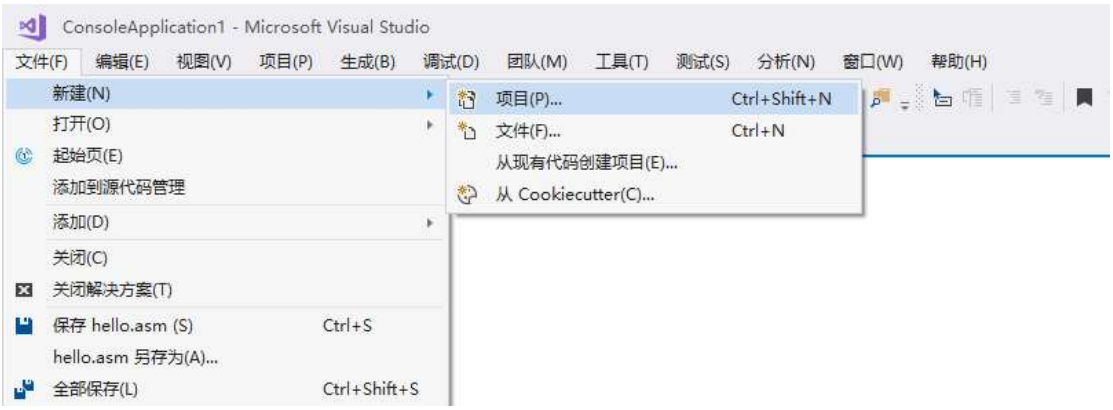
在 http://www.masm32.com/官网下载并安装

1.2 安装 VS2017：

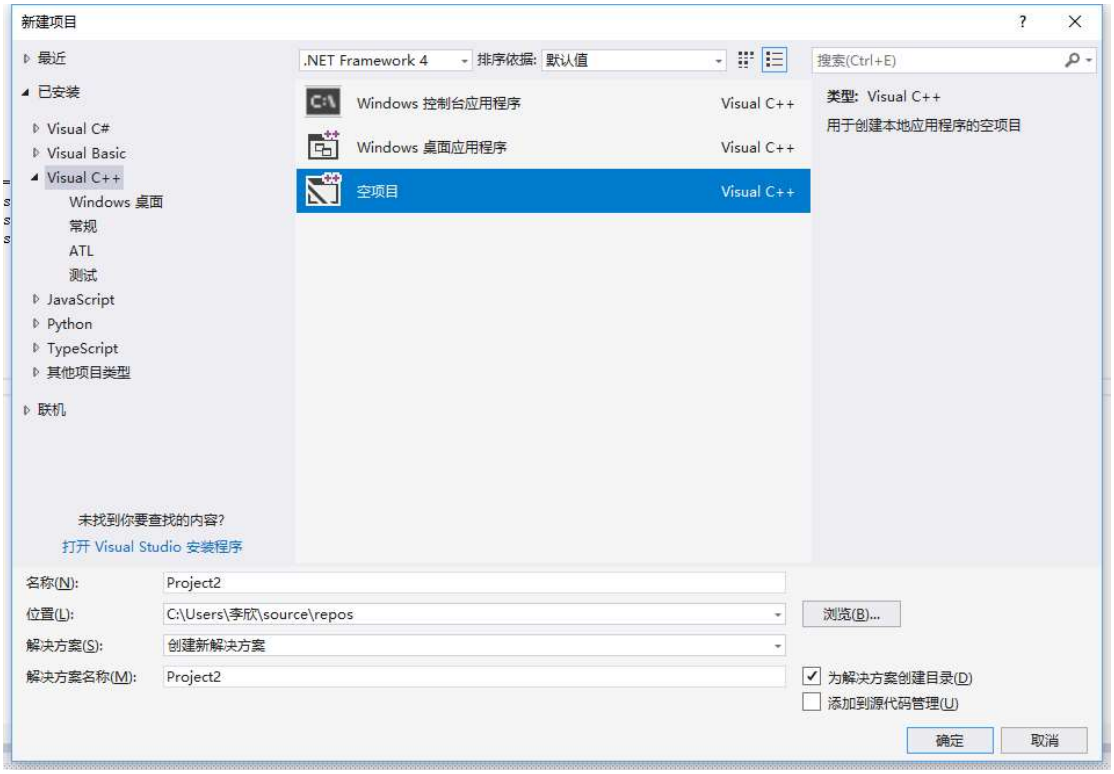
由于我的电脑之前装了 VS2017，所以在此不再赘述安装过程。给的教程是 VS2010，太老了，新版也支持汇编，所以干脆就直接用 VS2017 了。

1.3 新建一个汇编项目：

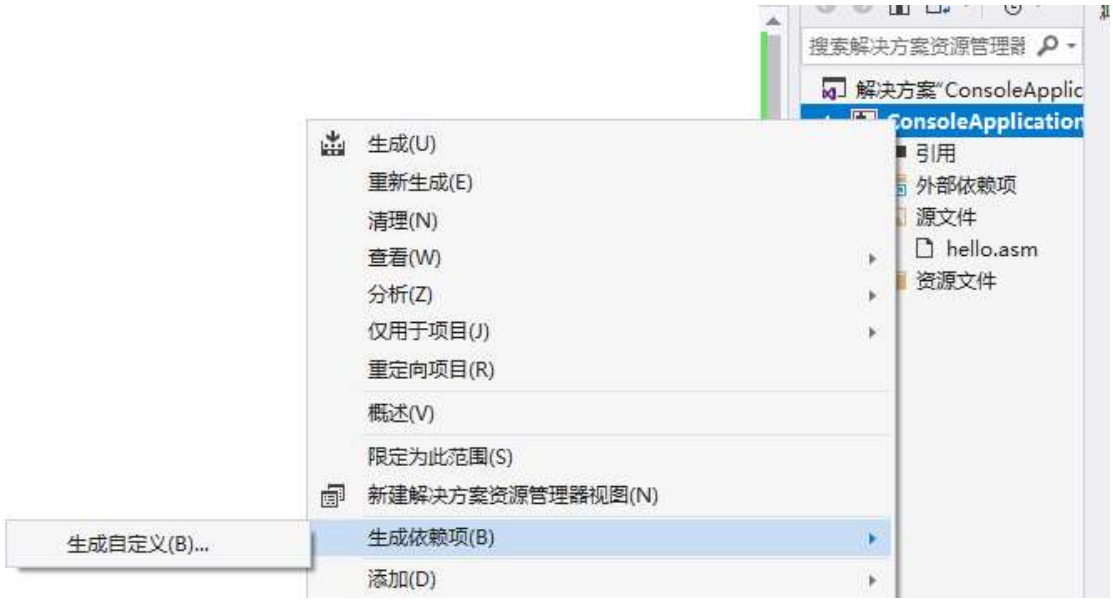
打开 VS2017：



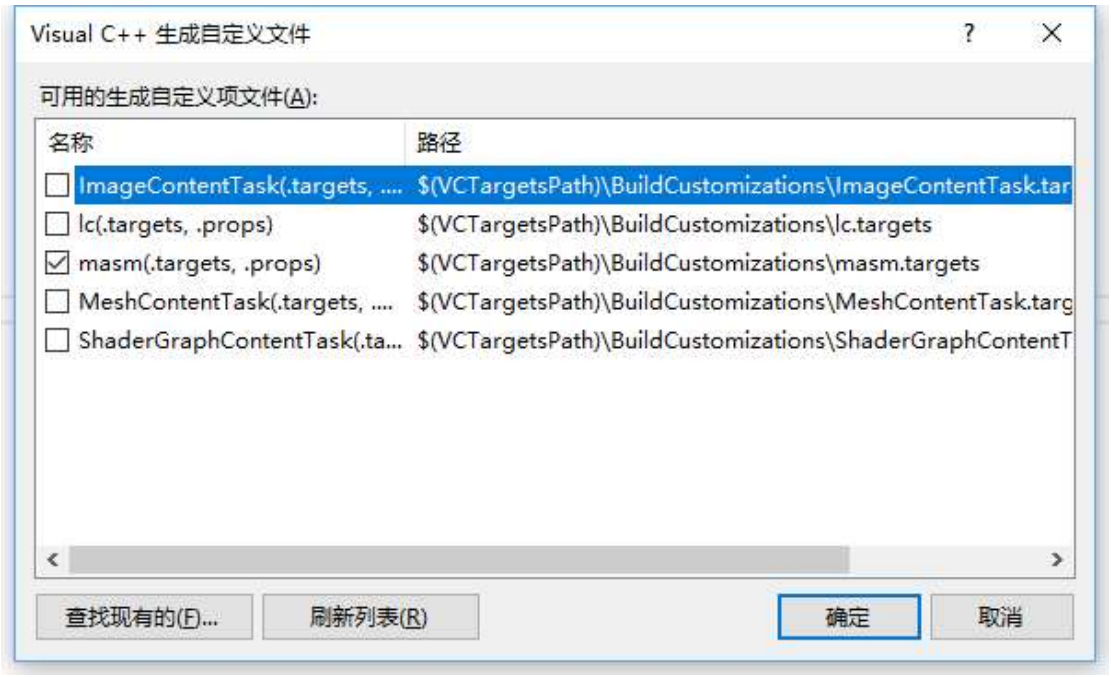
新建一个 C/C++项目：



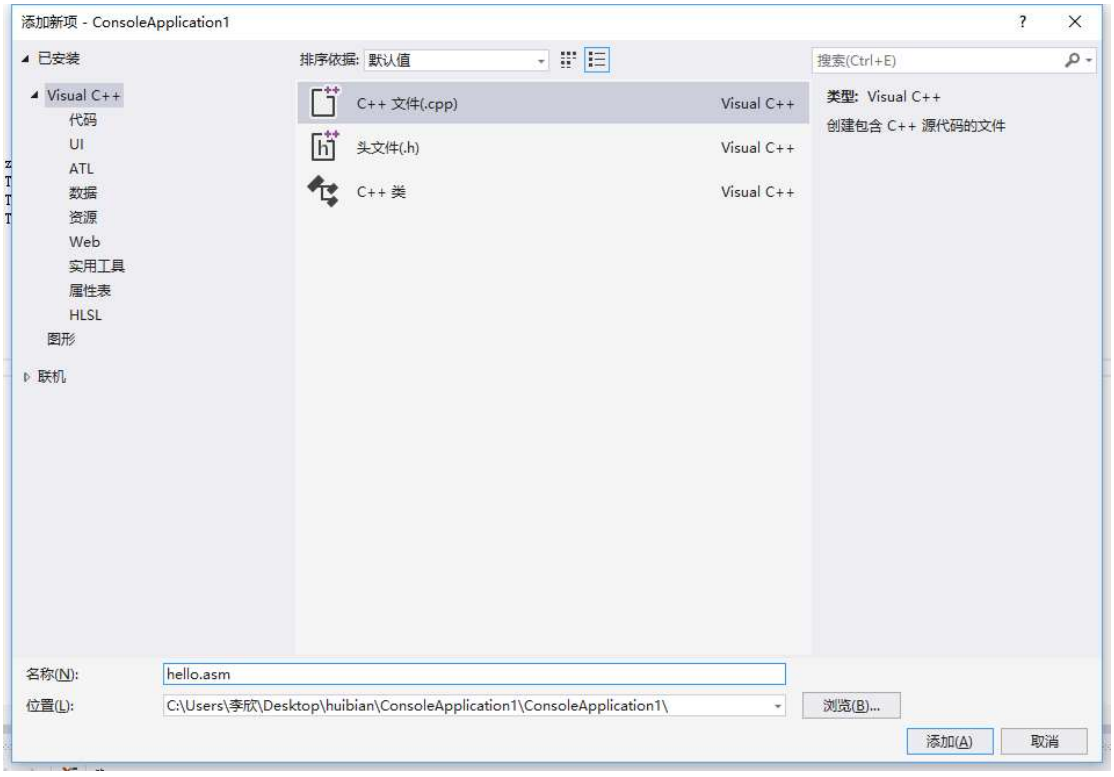
在项目下点右键，生成依赖项，生成自定义：



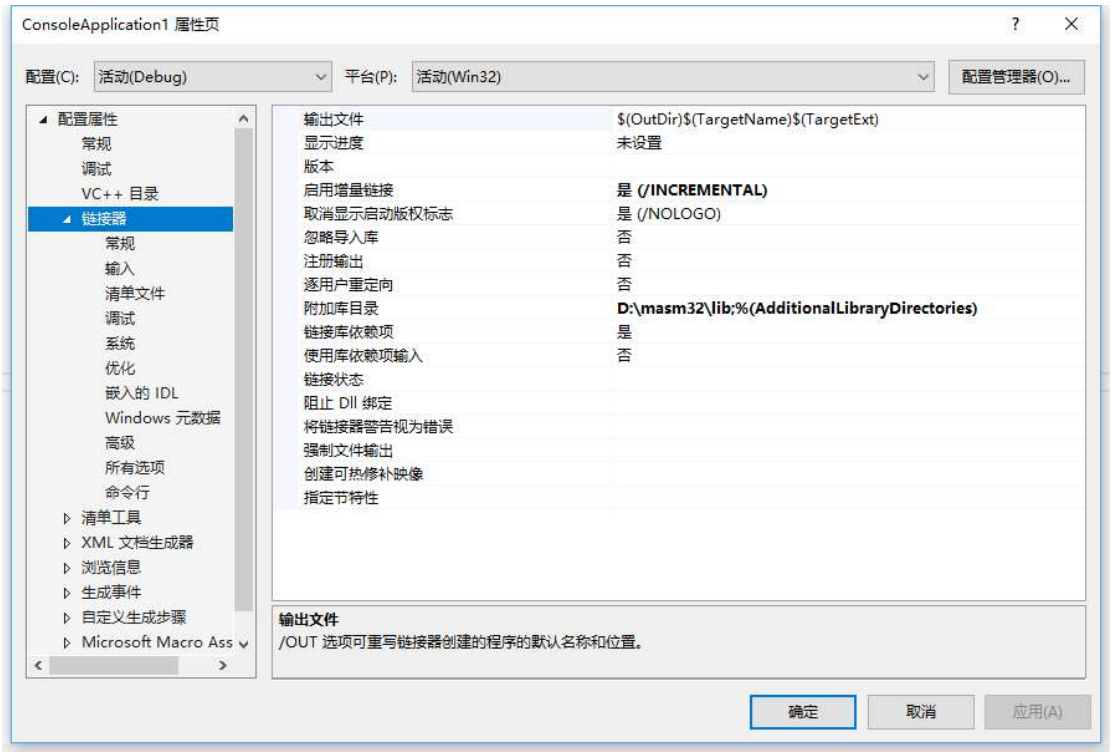
选择 masm，确定：



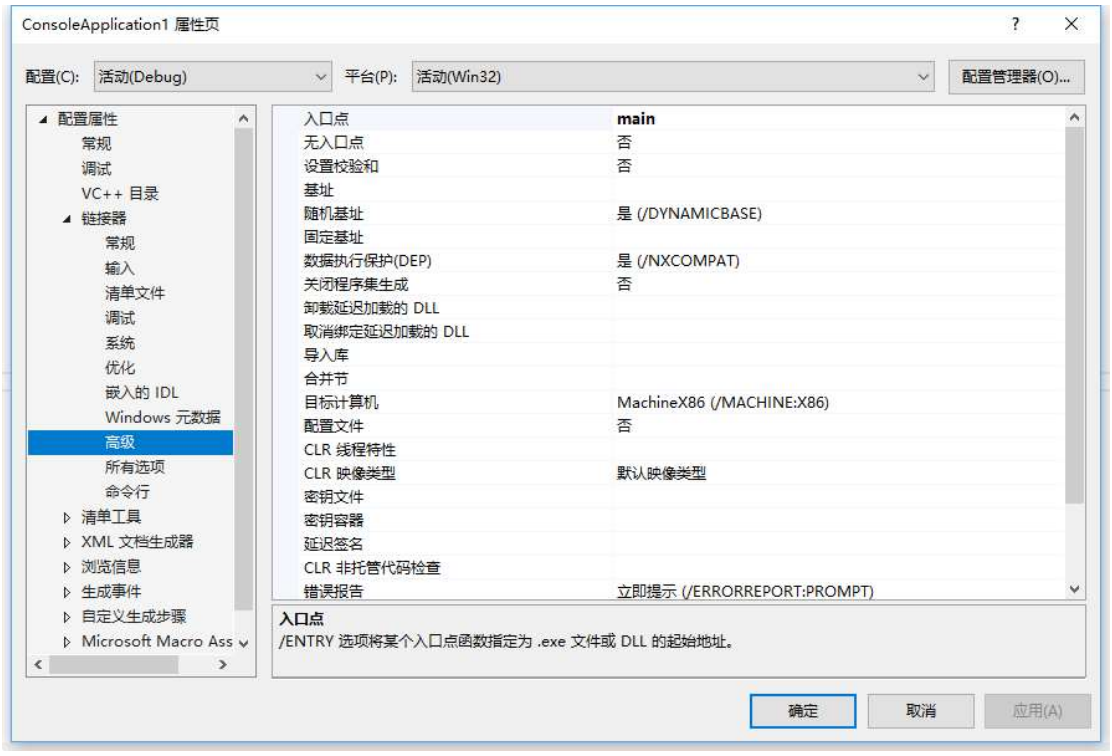
右键点击项目，新建一个 asm 文件：



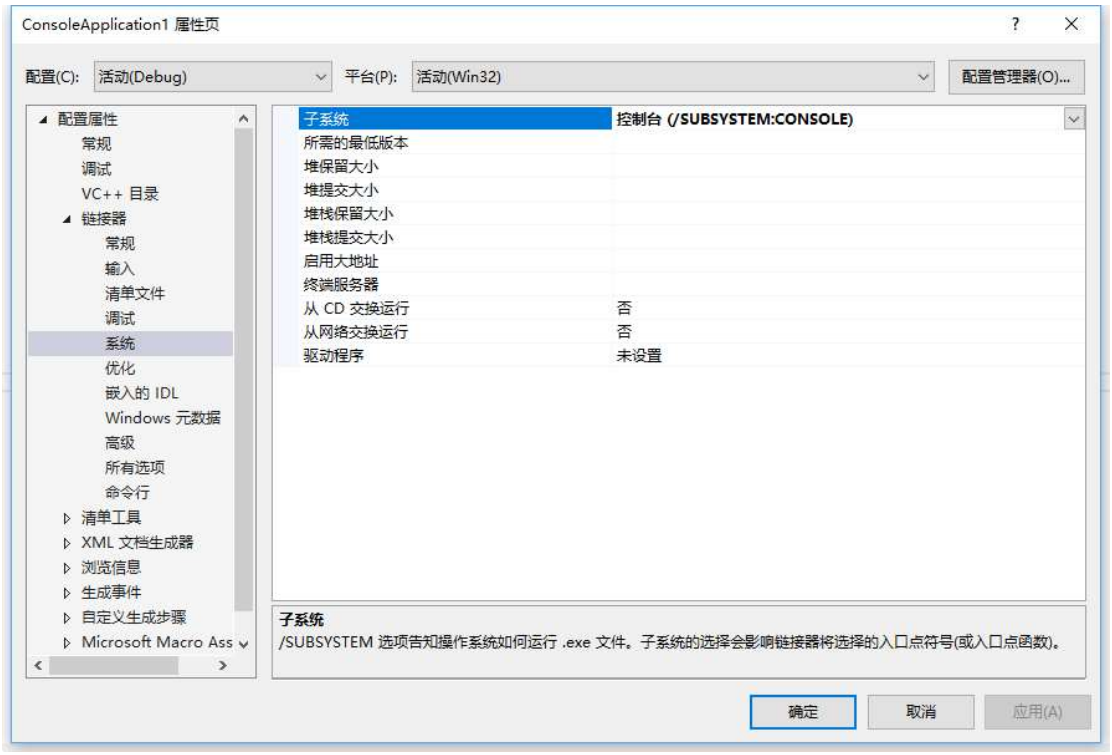
继续点击项目，右键，属性，修改附加库目录为 masm 安装目录的 lib 文件夹：



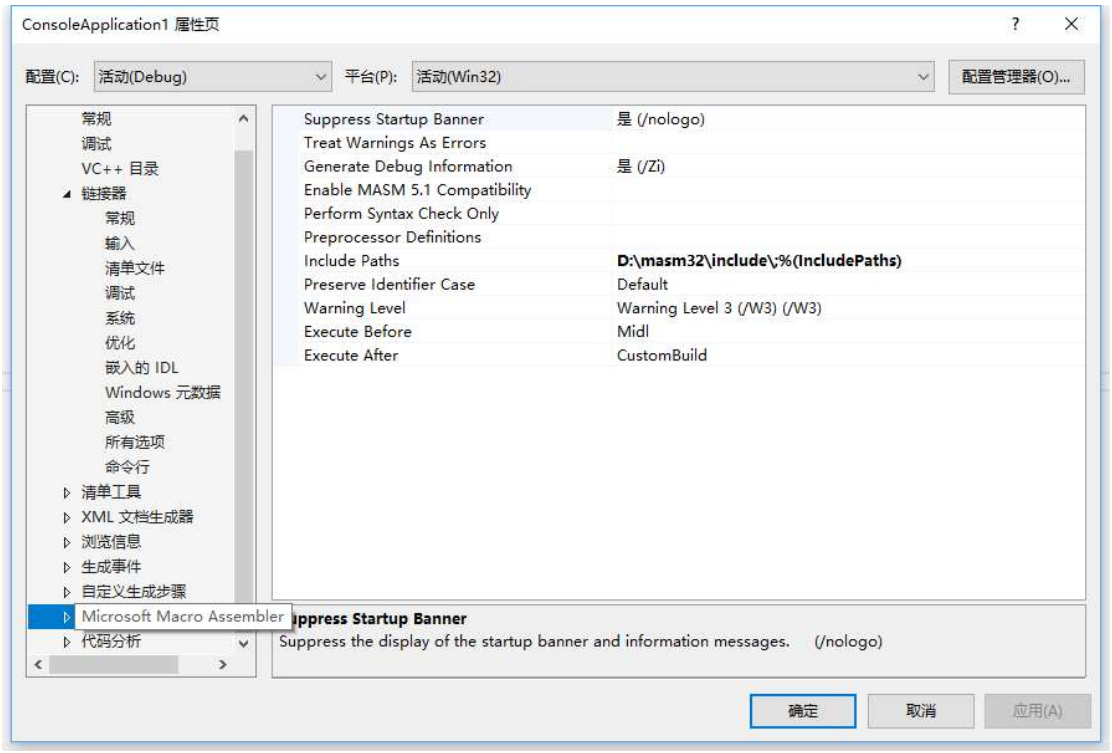
高级，入口点改为 main：



子系统改为控制台：



选择 Microsoft Macro Assembler，修改 Include Paths，然后点击确定：



然后点开 hello.asm 就可以写代码了，点击开始调试可以运行代码。

2.编写实验代码

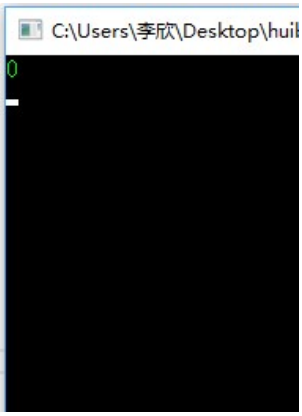
2.1 strcmp：

```
;write by chujian
.386
.model flat, stdcall

include kernel32.inc
includelib kernel32.lib

include msvcrt.inc
includelib msvcrt.lib

.data
format      db  "%d", 0AH, 0
szText      db  "Reverse Engineering", 0
szText2     db  "Reverse Engineering", 0    ;szText==szText2
szText3     db  "Reverse Eng", 0            ;szText>szText3
szText4     db  "Reverse Engj", 0           ;szText<szText4
szText5     db  "Reverse Engh", 0           ;szText>szText5

.code

main PROC
    LEA ESI, szText
    LEA EDI, szText2    ;result=0
    ;LEA EDI, szText3   ;result=1
    ;LEA EDI, szText4   ;result=-1
    ;LEA EDI, szText5   ;result=1
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;strcmp逻辑
    MOV ECX,20
    REPE CMPSB
    CMP ECX,0
    Jecxz equal
    MOV ECX,1
equal:
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    INVOKE crt_printf, addr format, ECX

    INVOKE crt_getchar
    INVOKE ExitProcess, 0
main ENDP

END main
```

关键的是使用 CMPSB，字符串比较指令，并用 REPR 指令重复执行。

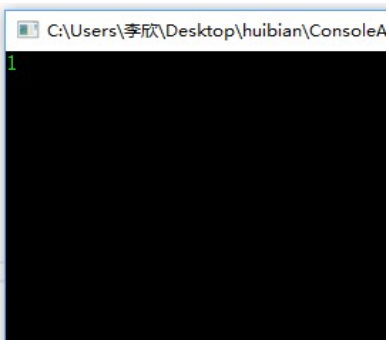
字符串相等情况，szText 与 szText2 比较，运行结果：

```
.data
format      db  "%d", 0AH, 0
szText      db  "Reverse Engineering", 0
szText2     db  "Reverse Engineering", 0    ;szText==szText2
szText3     db  "Reverse Eng", 0            ;szText>szText3
szText4     db  "Reverse Engj", 0           ;szText<szText4
szText5     db  "Reverse Engh", 0           ;szText>szText5

.code

main PROC
    LEA ESI, szText
    LEA EDI, szText2    ;result=0
    ;LEA EDI, szText3   ;result=1
    ;LEA EDI, szText4   ;result=-1
    ;LEA EDI, szText5   ;result=1
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```



字符串不相等情况，szText 与 szText3 比较，修改注释并运行：

```
.data
format      db  "%d", 0AH, 0
szText      db  "Reverse Engineering", 0
szText2     db  "Reverse Engineering", 0    ;szText==szText2
szText3     db  "Reverse Eng", 0            ;szText>szText3
szText4     db  "Reverse Engj", 0           ;szText<szText4
szText5     db  "Reverse Engh", 0           ;szText>szText5

.code

main PROC
    LEA ESI, szText
    ;LEA EDI, szText2   ;result=0
    LEA EDI, szText3    ;result=1
    ;LEA EDI, szText4   ;result=-1
    ;LEA EDI, szText5   ;result=1
```



2.2 strlen：

```
;write by chujian
.386
.model flat, stdcall

include kernel32.inc
includelib kernel32.lib

include msvcrt.inc
includelib msvcrt.lib

.data
szText  db  "Reverse Engineering", 0
format  db  "length = %d", 0AH, 0

.code

main PROC
    LEA EDI, szText
    MOV ECX,0FFFFFFFFH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;strlen逻辑
    XOR AL,AL
    MOV EBX,EDI
    REPNE SCASB
    SUB EDI,EBX
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    INVOKE crt_printf, addr format, EDI

    INVOKE crt_getchar
    INVOKE ExitProcess, 0
main ENDP

END main
```

先将 AL 寄存器置零，复制一份字符串首地址到 EBX，比较 AL 与字符串中的字符，不相等就自增/自减，最后与原地址相减即可得到字符串长度。

字符串长度计算运行结果：

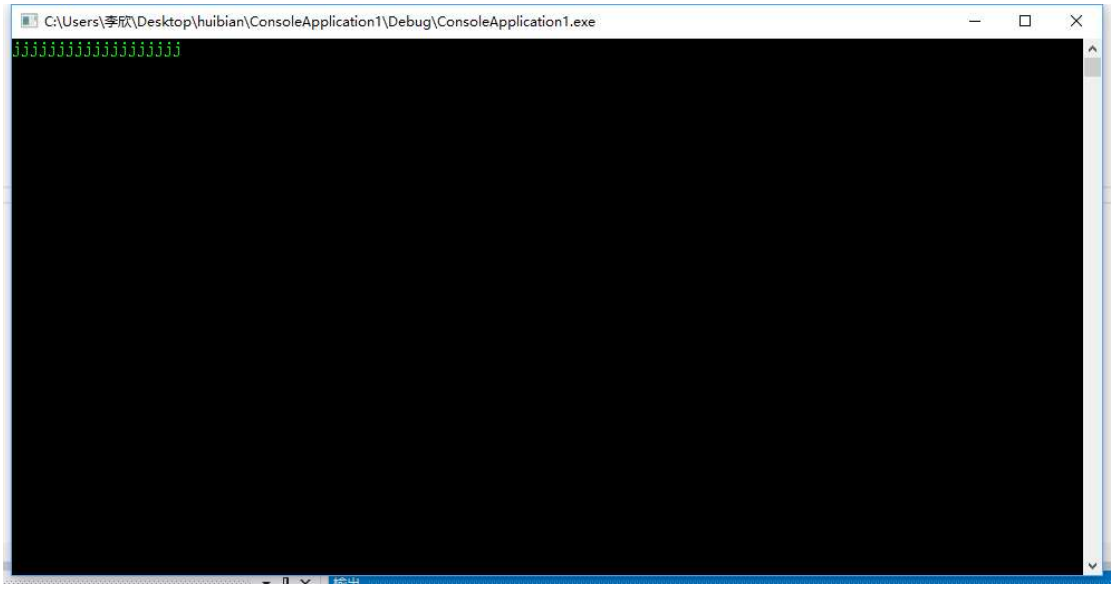
```
----------  ----------  ---

include msvcrt.inc
includelib msvcrt.lib

.data
szText  db  "Reverse Engineering", 0
format  db  "length = %d", 0AH, 0

.code

main PROC
    LEA EDI, szText
    MOV ECX,0FFFFFFFFH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

2.3 strset：

```
;write by chujian
.386
.model flat, stdcall

include kernel32.inc
includelib kernel32.lib

include msvcrt.inc
includelib msvcrt.lib

.data
szText  db  "Reverse Engineering", 0
chr     db  'j'

.code

main PROC
    LEA EDI, szText
    MOV ECX,0FFFFFFFFH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;strset逻辑
    PUSH 'j'
    POP EAX
    PUSH 19
    POP ECX
    REP STOSB
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    INVOKE crt_printf, addr szText

    INVOKE crt_getchar
    INVOKE ExitProcess, 0
main ENDP

END main
```

使用 STOSB 指令将’ j’重复写入字符串 19 次即可

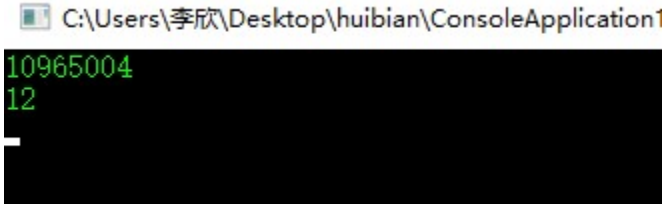
运行结果：



2.4 strchr：

```
;write by chujian
.386
.model flat, stdcall

include kernel32.inc
includelib kernel32.lib

include msvcrt.inc
includelib msvcrt.lib

.data
szText  db  "Reverse Engineering", 0
chr     db  'i'
format  db  "%d", 0AH, 0

.code

main PROC
    LEA EDI, szText
    MOV ECX,0FFFFFFFFH
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;strchr逻辑
    MOV EBX,EDI
    MOV AL,[chr]
    REPNE SCASB
    INVOKE crt_printf, addr format, EDI
    SUB EDI,EBX
    INVOKE crt_printf, addr format, EDI
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    INVOKE crt_getchar
    INVOKE ExitProcess, 0
main ENDP

END main
```

使用 SCASB 指令进行字符与字符串比较，最终显示字符串实际地址和相对地址。
运行结果：



i 在字符串中实际地址为 10965004，相对字符串地址为 12。

四、实验总结

通过本实验，我学会了使用 MASM32+VS2017 环境运行 x86 汇编程序，掌握了基本的汇编程序编写方法，熟悉了汇编语言中对字符串的操作指令。中间配置环境的时候遇到很多问题，均通过百度谷歌自行解决，指令使用不会就查询文档，最终还是成功完成了本次实验。