****

**汇编语言程序设计课程设计**

****

|  |  |
| --- | --- |
| **题目** | **编制完整的汇编语言源程序** |
| **学院** | **计算机学院** |
| **专业** | **计算机科学与技术** |
| **学号** |  |
| **姓名** |  |
| **指导老师** | **余波老师** |
| **实验时间** | **2022年5月10日** |

# 第三次实验：编制完整的汇编语言源程序

学号: 姓名:

## 一、实验目的：

1、掌握编写完整汇编语言源程序的方法，掌握汇编、连接等将源程序转换为可执行代码的必要步骤。

2、在理解逻辑运算指令原理的基础上，按照实验要求的功能编写完整程序，生成可执行文件后，使用DEBUG对程序进行调试。

## 二、实验环境（硬件，软件环境）：

硬件环境： MACOS系统

软件环境：DOSBox v0.74-3-3

## 三、实验内容及要求（算法，程序，步骤，方法）：

1、实验内容

（1）在实验报告中说明汇编语言源程序框架的基本原理。

（2）说明生成可执行文件的必要步骤。

（3）调试并观察程序的运行结果，并解释其合理性。

2、实验要求

编写一个完整源程序完成如下功能：

1）设定字节型变量VAR1的内容为35H，编写程序将VAR1的高4位与低4位分离，分别作为低4位保存在RES1和RES2中,这两个变量高4位置0。

2）设定32位（二进制）数据0E2597455H（注：最高位为E）保存在变量VAR2中，将该变量中的内容算术左移2位，所得结果仍然保存在VAR2中。

（1）对VAR1处理过程中所使用的逻辑运算指令，其分别的作用需在报告中解释；各逻辑运算指令的执行结果（包括寄存器、标志位受到的影响）需作分析。

（2）针对VAR2的处理，对移位操作进行原理分析，对所使用的各类移位指令在程序中作用进行解释，对各移位指令的执行结果进行分析。

（3）结合标志位，对VAR2移位后的结果作溢出分析。

3、实验步骤

3.1 汇编语言源程序框架的基本原理

（1）框架结构

一个完整的汇编语言源程序通常由若干个逻辑段(Segment)组成, 包括数据段、附加段、堆栈段和代码段,它们分别映射到存储器中的物理段上。每个逻辑段以 SEGMENT语句开始, 以 ENDS语句结束,整个源程序用END语句结尾。

代码段中存放源程序的所有指令码, 数据、变量等则放在数据段和附加段中。程序中可以定义堆栈段，也可以直接利用系统中的堆栈段。具体一个源程序中要定义多少个段应根据实际需要来定。但一般来说，一个源程序中可以有多个代码段, 也可以有多个数据段、附加段及堆栈段, 但一个源程序模块只可以有一个代码段、一个数据段、一个附加段和一个堆栈段。将源程序以分段形式组织是为了在程序汇编后，能将指令码和数据分别装入存储器的相应物理段中。

下面给出本题要求源程序来作为一个完整的汇编语言源程序示例，三个结构有标注。（无附加段的情况）

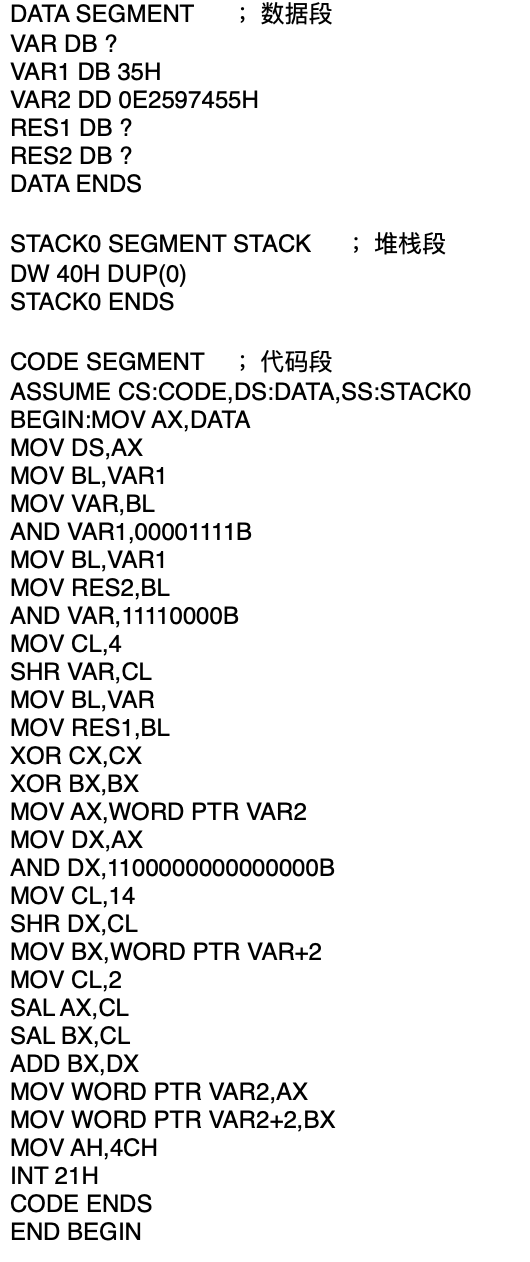


图1.1 完整程序段示例（以本题要求为例）

1. 编译原理

从完成程序，到机器执行程序的过程一般是：编辑，汇编（masm）链接（link）、加载（command）、调试（debug）。编译生成目标文件、和其他目标文件或库进行链接生成可执行文件，最后被加载到内存中执行。

1. 将源码保存为.asm格式的文件并放在DOS系统目录下

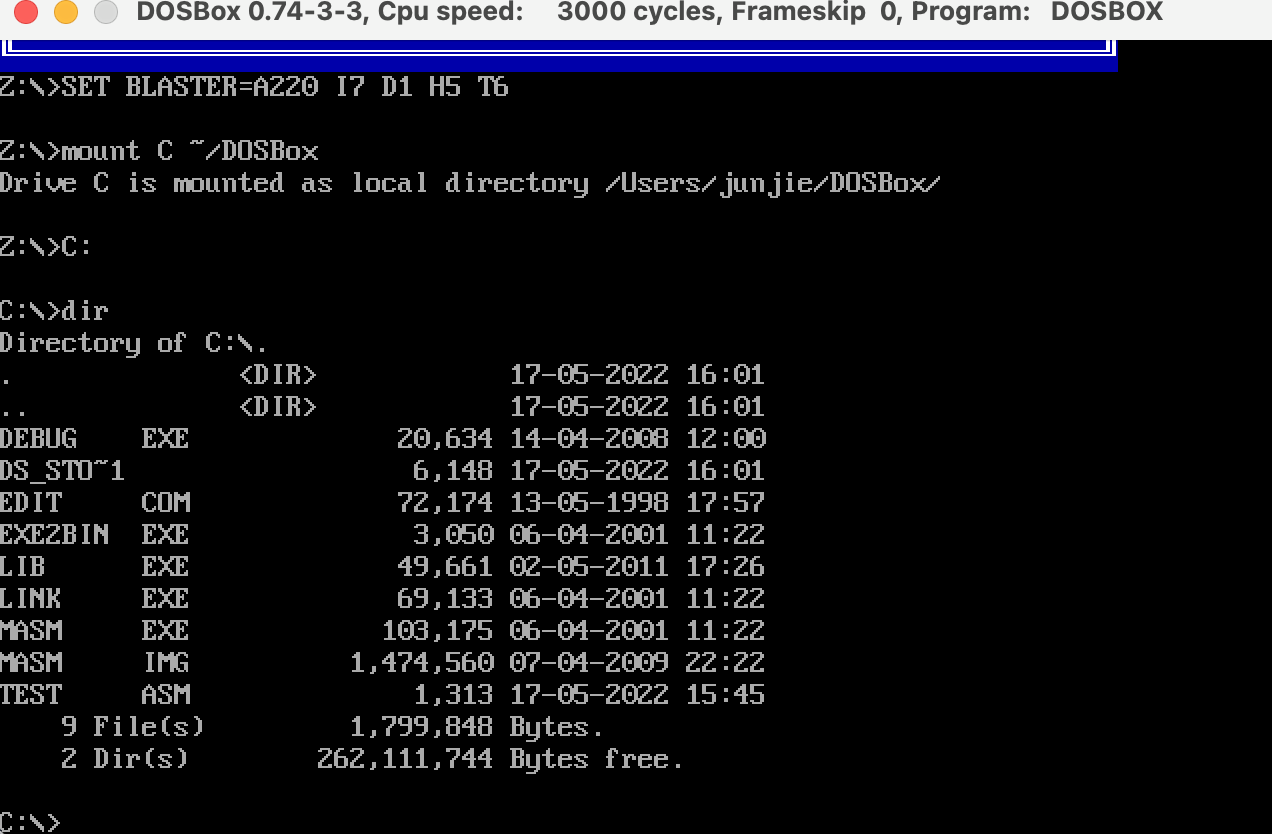


图1.2 查看源码文件是否在系统目录下

1. 使用masm对源文件编译：MASM FILENAME.ASM（文件路径，目标文件文件名，列表文件文件名都作为默认即可，则三个选项回车跳过）

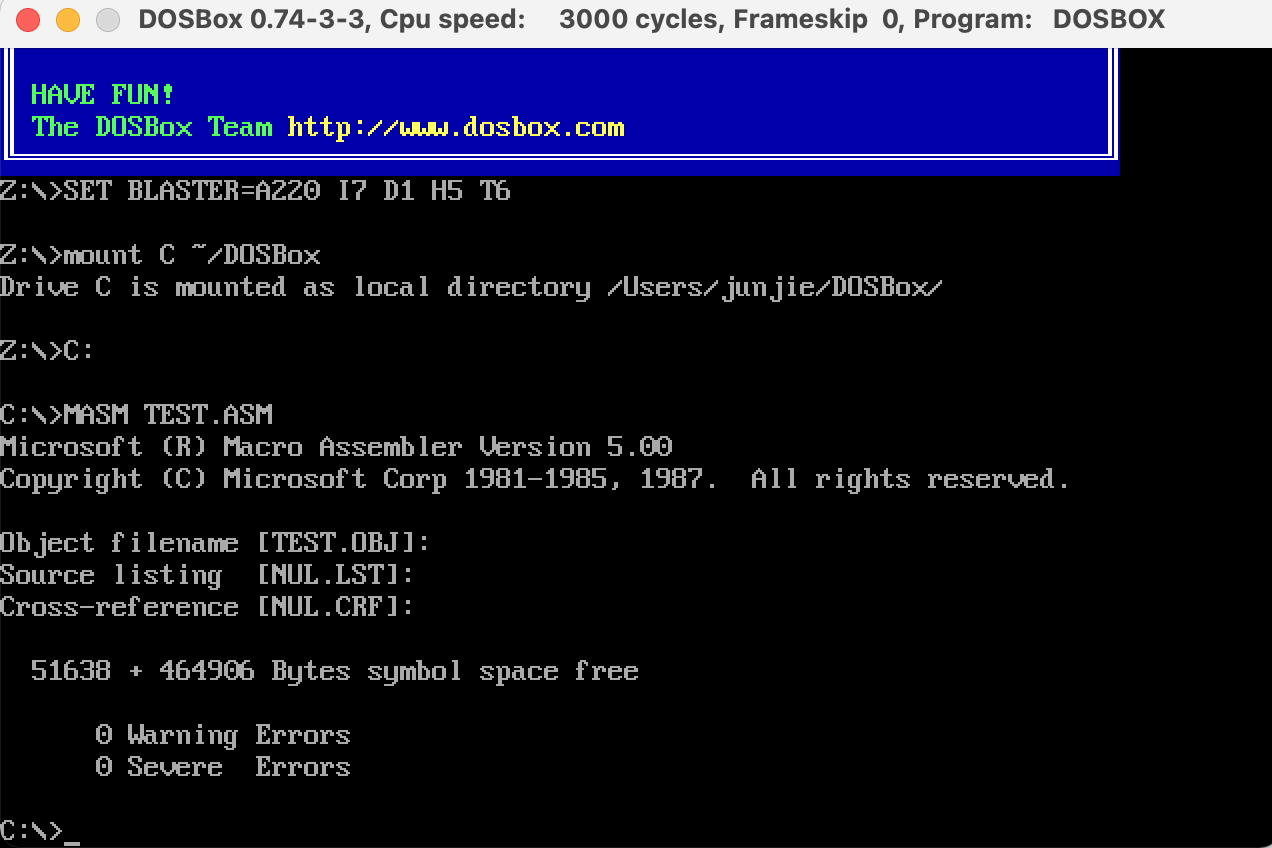


图1.3 masm编译

1. 使用link对目标文件链接（文件路径，目标文件文件名，列表文件文件名都作为默认即可，则三个选项回车跳过）

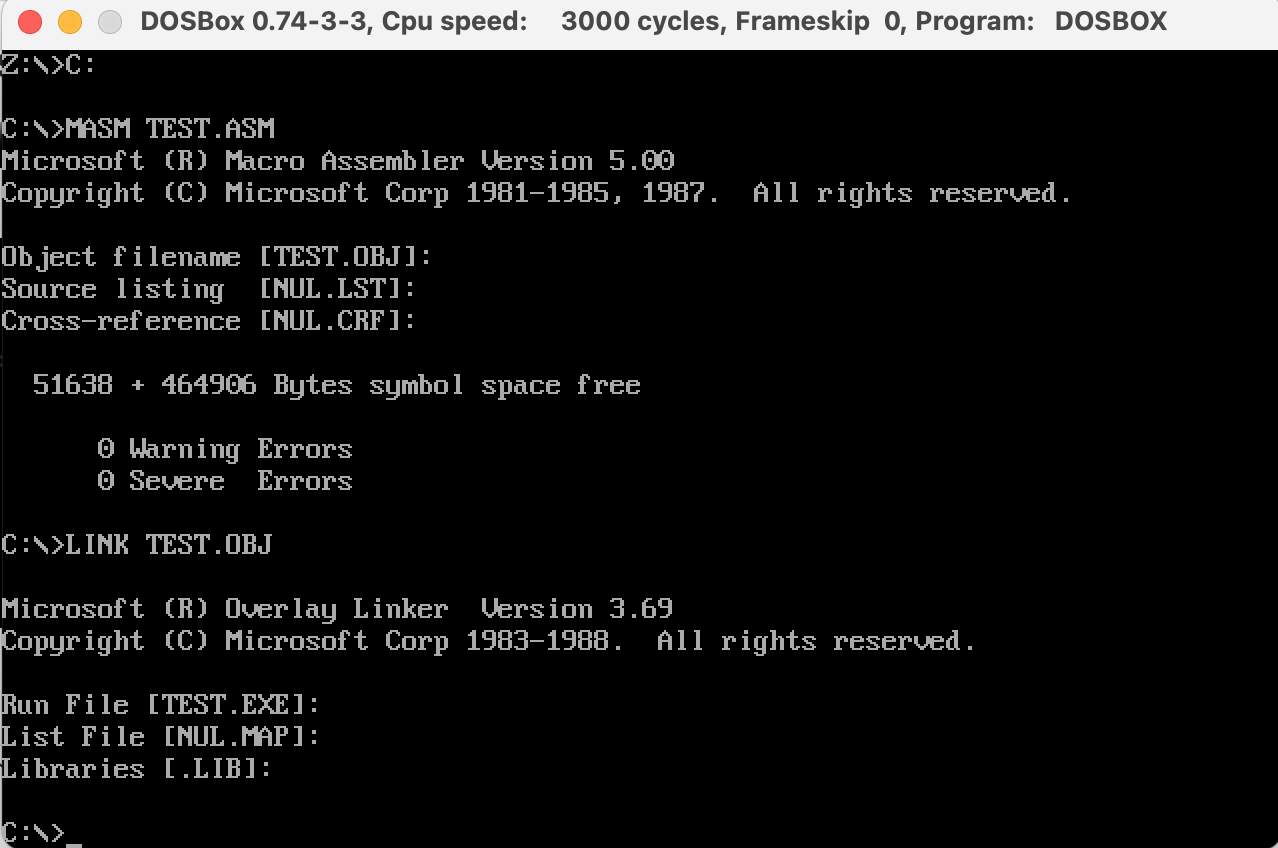
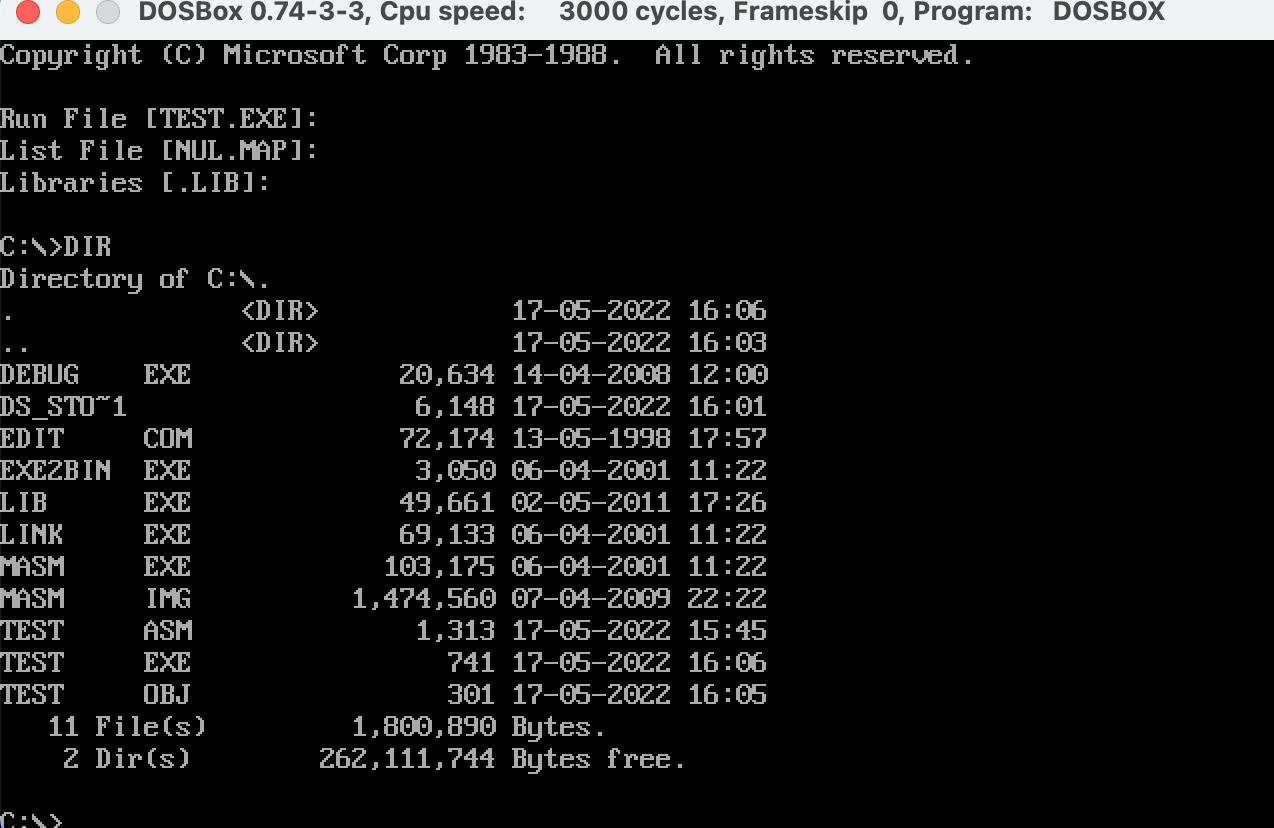


图1.4 link链接



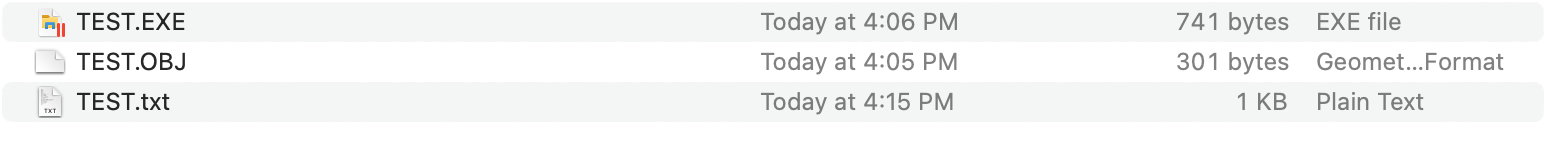


图1.5 产生的obj与exe文件

3.2 源程序的调试及分析（源码在第4节）

1. 首先运行DOSBox，键入-DEBUG TEST.EXE进入debug模式

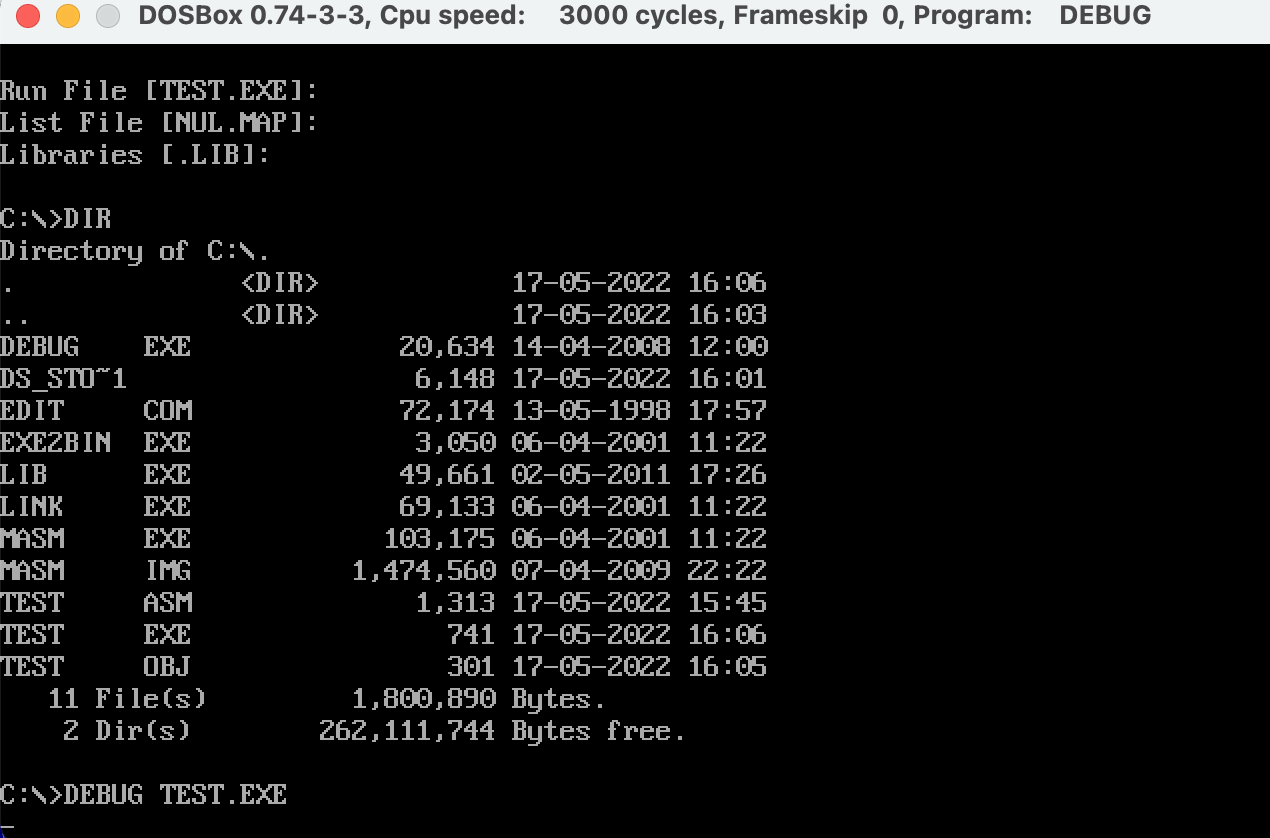
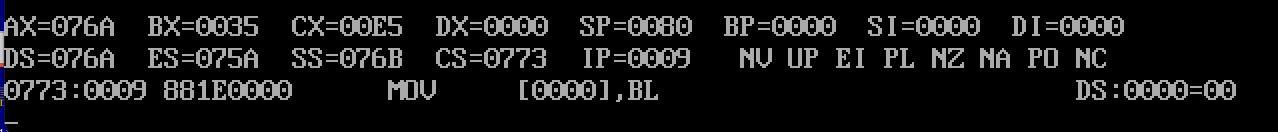


图2.1 进入debug模式

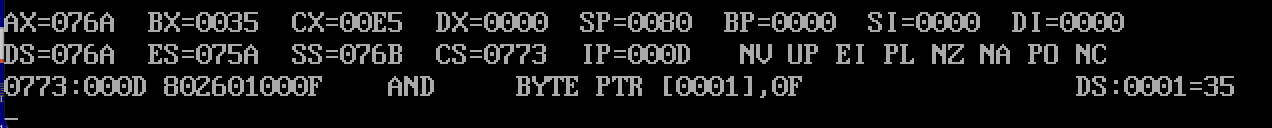
1. 下面对VAR1处理板块逐步运行分析，其中会以逻辑运算指令AND重点分析（其分别的作用；各逻辑运算指令的执行结果（包括寄存器、标志位受到的影响））

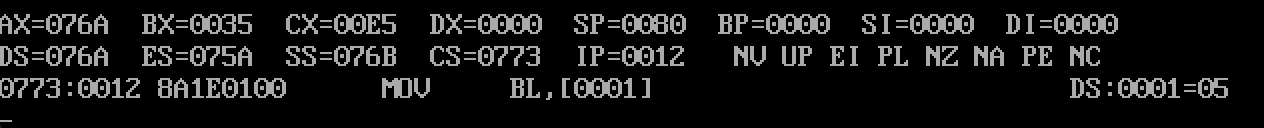
这里给出的对应图片为起始语句，下一张图片为运行结果。

A.将VAR1赋值给BL

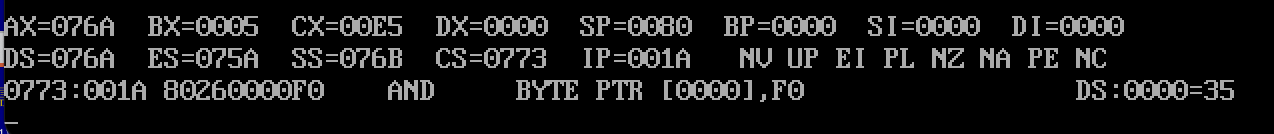
 B.由于之后会对BL和VAR1进行运算，故将BL赋给VAR，用于临时保存

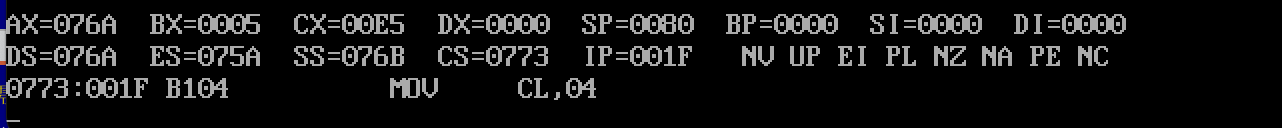
C.运用AND指令00001111B取VAR1低四位（作用），VAR1只剩下低四位（执行结果），其中CF,OF强行置0，AF不确定，SF=0，ZF=0，PF=1与算术运算解释保持一致。

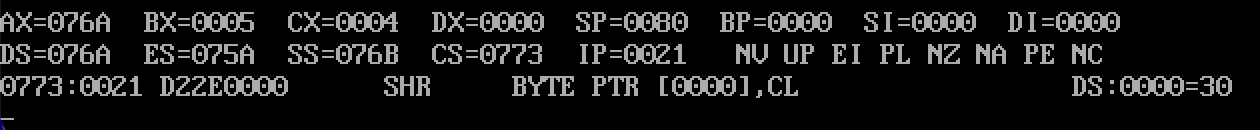
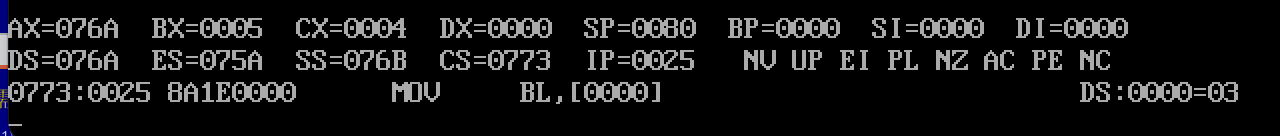
 D.将处理后的VAR1赋给BL



E.将处理后的BL赋给RES2。

F.对用于备份的VAR（值为VAR1）处理，取高四位（作用）。即AND指令11110000B，VAR只剩下高四位（结果），其中CF,OF强行置0，AF不确定，SF=0，ZF=0，PF=1与算术运算解释保持一致。

G.由于大于1时移位运算必须由寄存器过度，将4赋值给CL。

H.VAR右移四位，将保留的高四位移到低四位。 I.将处理后的VAR赋给BL。

J.结果赋给RES1。

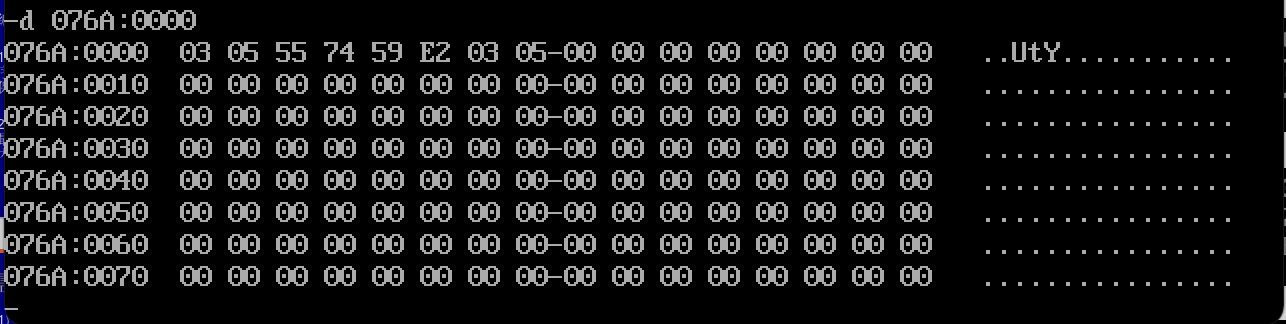
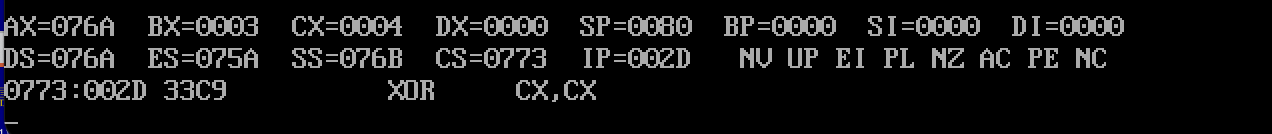
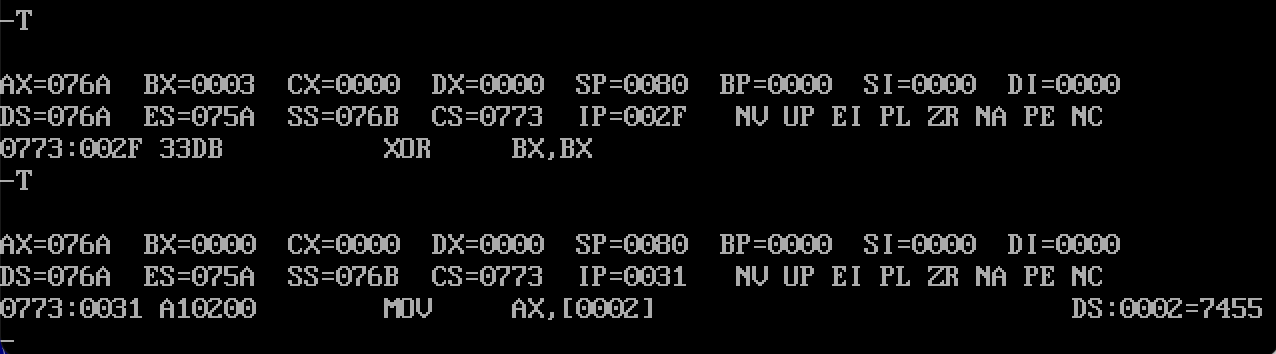
K.运行结果

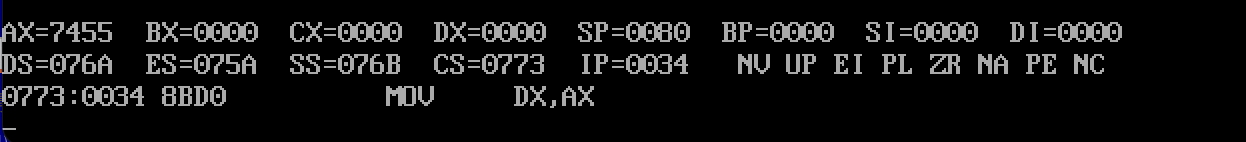
图2.2 对VAR1处理结果

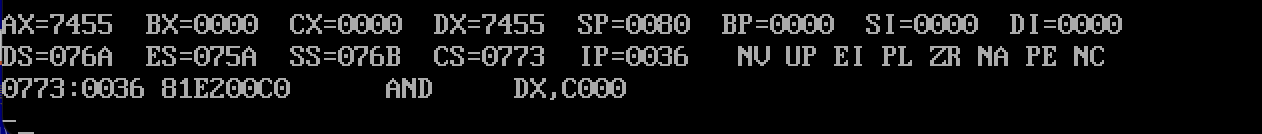
1. 下面对VAR2处理板块逐步运行分析，其中会以各类移位指令重点分析（对移位操作进行原理分析，对所使用的各类移位指令在程序中作用进行解释，对各移位指令的执行结果进行分析。）

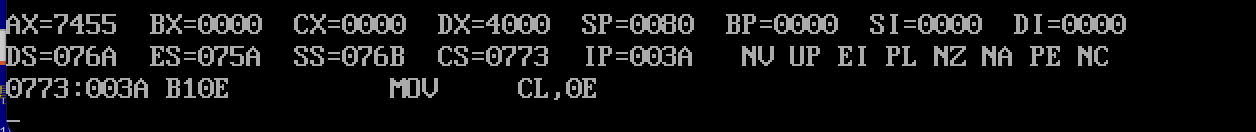
此时对应图片为执行序号命令后的结果。

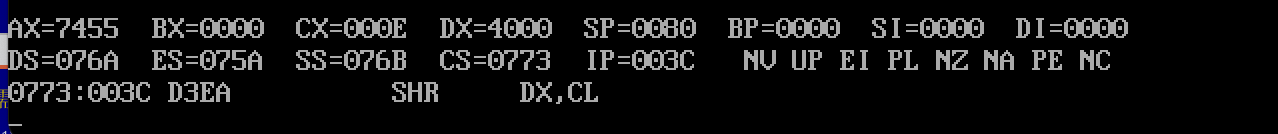
A.清空CX，BX

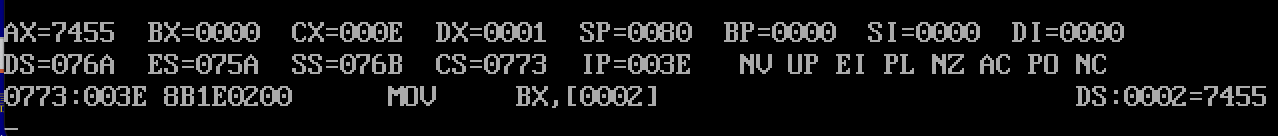


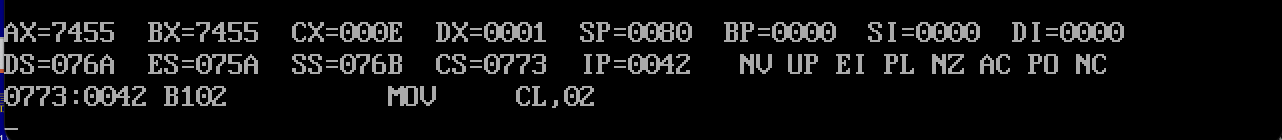
B.将VAR2低十六位取出给AX。

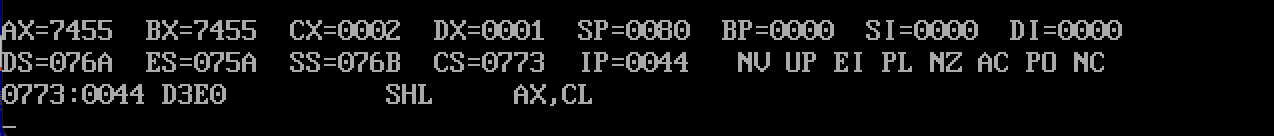
C.将低十六位给DX（由于本题要求用算术运算符，所以不用循环移位，用手动加上丢失的两位的方式来运算，比如高位1111左移为1100，低位1111左移1100，丢失了11两位，此时手动给高位加上11即可）

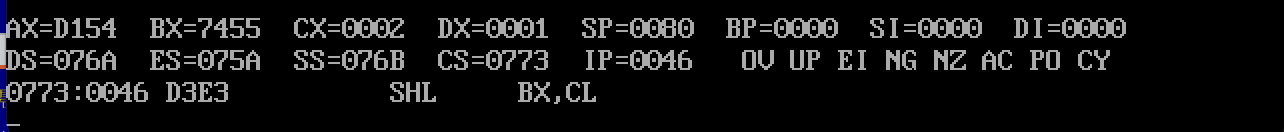
D.取出DX最高的两位（防止溢出丢掉，手动在之后移位时加上）

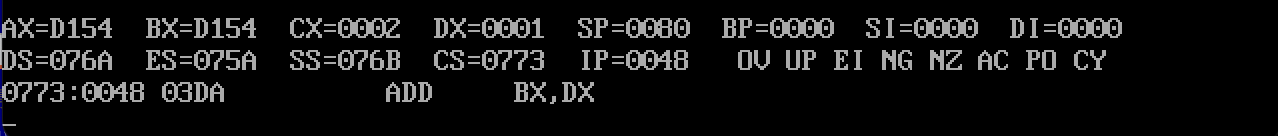
E.大于1时算术移位运算需要用寄存器过渡

F.得到DX高两位

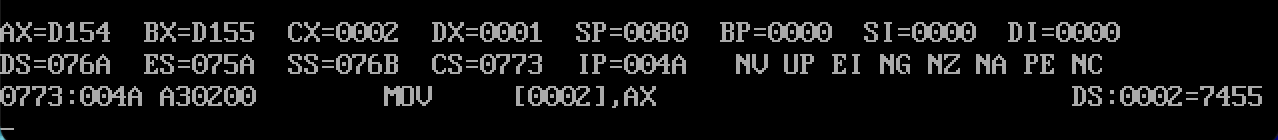
G.将VAR2高十六位取出给BX。

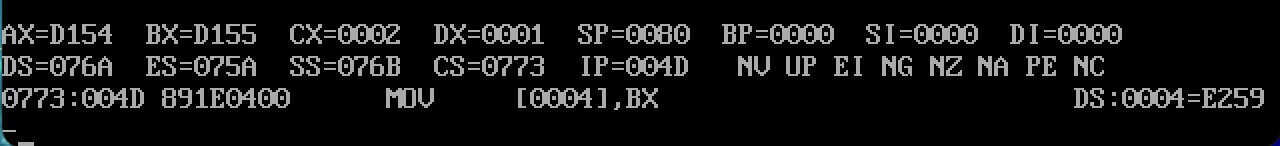
H.AX开始左移两位（低十六位）此时进位CF=1，移动后1的个数为奇数PF=0，产生第四位辅助进位AF=1，运算结果不为0即ZF=0，最高位为SF=1，同时符号位改变OF=1，结果判定为溢出。（溢出分析）

I.BX开始左移两位（高十六位）此时进位CF=1，移动后1的个数为奇数PF=0，产生第四位辅助进位AF=1，运算结果不为0即ZF=0，最高位为SF=1，同时符号位改变OF=1，结果判定为溢出。（溢出分析）

J.加上之前溢出丢失的低两位

K.低十六位



L.高十六位

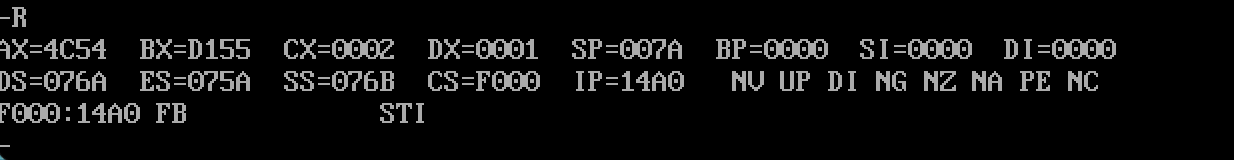
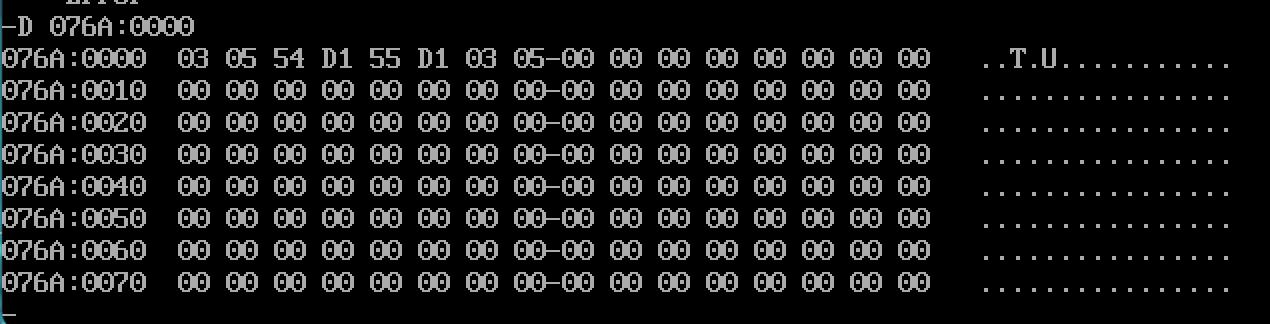
M.实验结果

图2.3 对VAR2处理运行结果

4、实验源代码

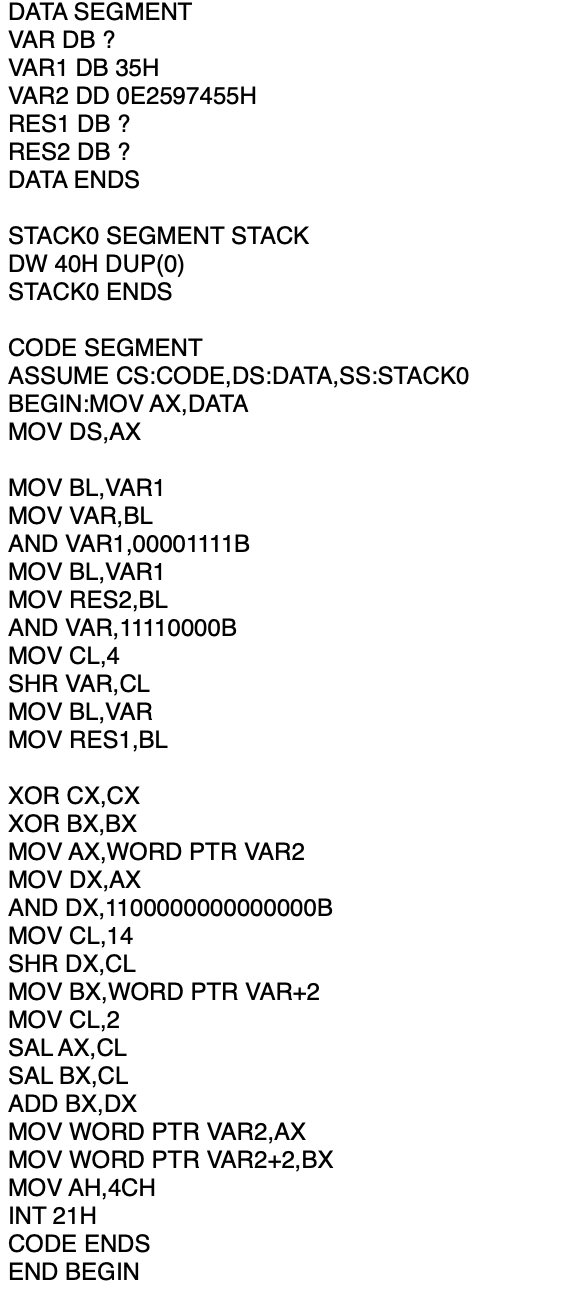


图3.1 源代码

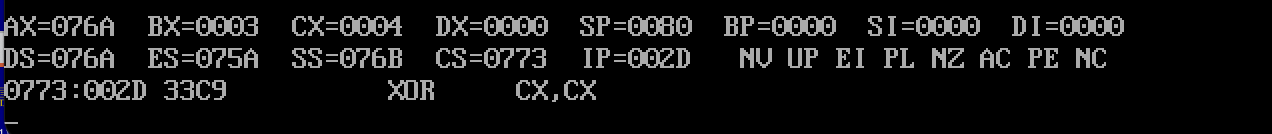
为方便读者阅读，下给出源代码注释版本

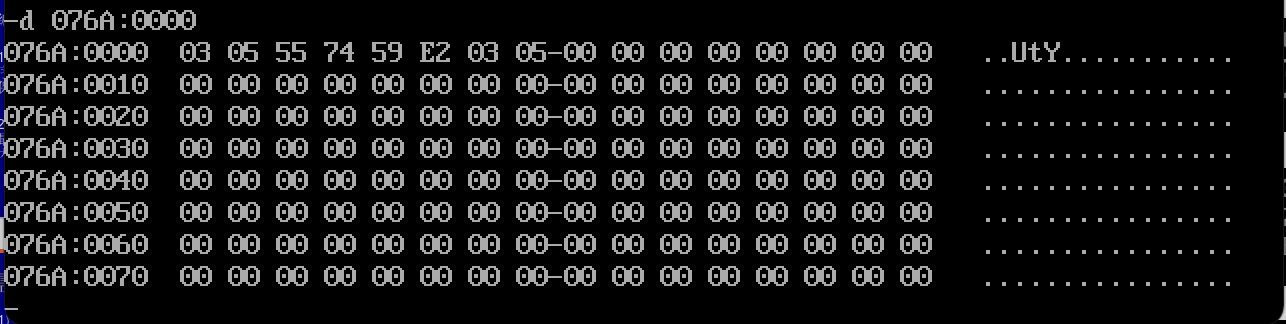


图3.2 源代码（含注释）

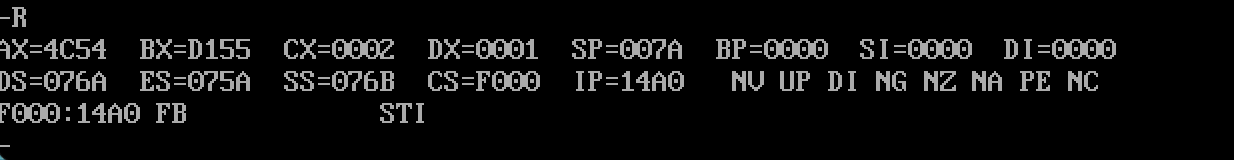
## 实验结果分析：

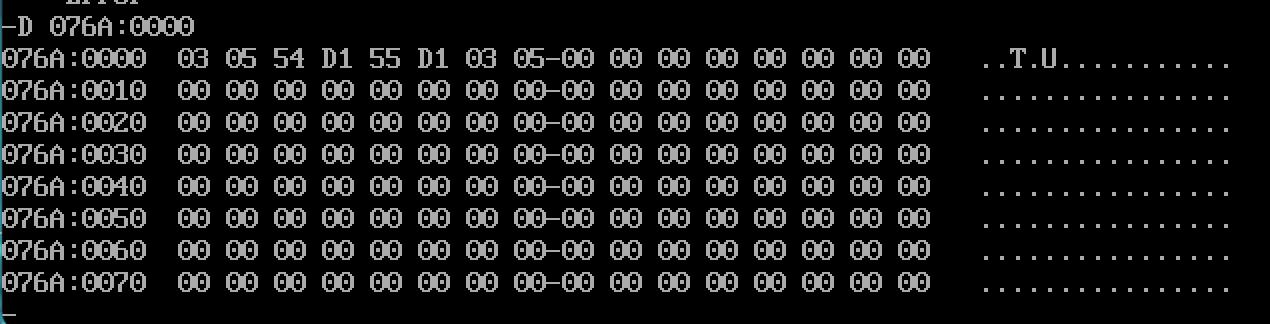
对于VAR1：设定字节型变量VAR1的内容为35H，编写程序将VAR1的高4位与低4位分离，分别作为低4位保存在RES1和RES2中,这两个变量高4位置0。





对于VAR2：设定32位（二进制）数据0E2597455H（注：最高位为E）保存在变量VAR2中，将该变量中的内容算术左移2位，所得结果仍然保存在VAR2中。





实验结果符合预期功能设计，实验成功！

## 5、实验体会：

实验过程应注意：

1. 在调试上述程序片段时，应使用r命令先修改DS、SS段寄存器，使之与CS指向不同位置。建议将DS提供的原始段基值加0100H，将SS提供的原始段基值加0200H。避免数据段、堆栈段内的操作覆盖代码段中已录入的指令，影响重复调试。
2. 编写源代码时要大致设计出框图，明确每一步运算结果，编写的程序段必须完整才能运行！

3.调试时要有耐心，认真分析每一步结果。

4.PUSH指令出栈会恢复。

通过本次实验，了解如何编写一个完整的源程序，实验成功！