

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 编程基础**

**实验时间： 2018-3-30，18:30-21:50 实验地点： 南一楼804室12号实验台**

**指导教师： 李海波 专业班级：计算机科学与技术201607班**

**学 号： U201614700 姓 名： 王亚宁**

**同组学生： 无 报告日期： 2018年3月30日**

**原创性声明**

  本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名：

日期：

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                    日期：

**目录**

[实验目的与要求 1](#_Toc510204616)

[实验内容 1](#_Toc510204617)

[实验过程 1](#_Toc510204618)

[任务1 1](#_Toc510204619)

[1. 设计思想 1](#_Toc510204620)

[2. 流程图 2](#_Toc510204621)

[3. 实验步骤 2](#_Toc510204622)

[4. 实验记录 2](#_Toc510204623)

[任务二 9](#_Toc510204624)

[4. 设计思想 9](#_Toc510204625)

[5. 流程图 10](#_Toc510204626)

[6. 源程序 10](#_Toc510204627)

[7. 实验步骤 12](#_Toc510204628)

[8. 实验记录 12](#_Toc510204629)

[任务三 16](#_Toc510204630)

[1. 设计思想 16](#_Toc510204631)

[2. 流程图 16](#_Toc510204632)

[3. 源程序 16](#_Toc510204633)

[4. 实验步骤 17](#_Toc510204634)

[5. 实验记录 17](#_Toc510204635)

[体会 18](#_Toc510204636)

[参考文献 18](#_Toc510204637)

# 实验目的与要求

1. 掌握汇编源程序编辑工具、汇编程序、连接程序、调试工具TD的使用；
2. 理解数、符号、寻址方式等在计算机内的表现形式；
3. 理解指令执行与标志位改变之间的关系；
4. 熟悉常用的DOS功能调用；
5. 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；
6. 加深对转移指令及一些常用的汇编指令的理解。

# 实验内容

任务1. 《80X86汇编语言程序设计》教材中 P31的 1.14题。

任务2. 《80X86汇编语言程序设计》教材中 P45的 2.3题。

任务3. 《80X86汇编语言程序设计》教材中 P45的 2.4题的改写。

# 实验过程

## 任务1

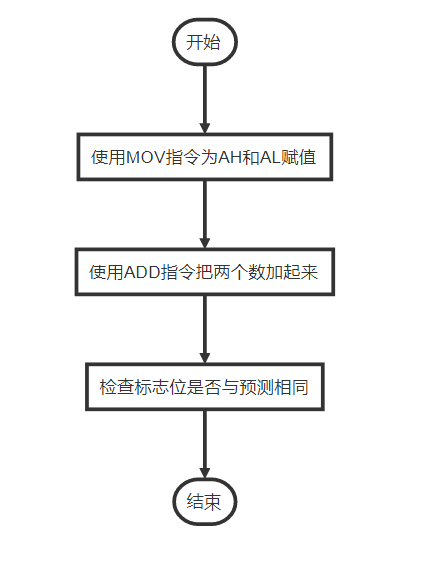
任务详细内容：已知二进制数x1和x2的值，求x1和x2的补码的和，并指出符号，判断是否发生了溢出和进位。

### 设计思想

设计思路：任务一的内容是对两个数进行加减运算，指出结果的符号，可以使用AX寄存器的高低位存放两个数的值，运算即可。

寄存器分配：AH存放x1，AL存放x2。

### 流程图



### 实验步骤

1. 打开dosbox，将masm的等文件目录挂载为c，输入c:切换到c盘。
2. 输入td打开td的窗口。
3. 直接编写要执行的汇编代码：首先把数据x1和x2的补码使用MOV指令到AX寄存器的高位和低位。
4. 使用ADD AH, AL命令计算AL加AH并将结果存放在AH中。
5. 观察此时的标志位SF, OF, CF, ZF，记录下来和自己实验前预测的对比并分析原因。

### 实验记录

实验环境条件：i5 6200U，2.8GHz，8GB内存；dosbox命令行窗口；visual studio 2017；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。

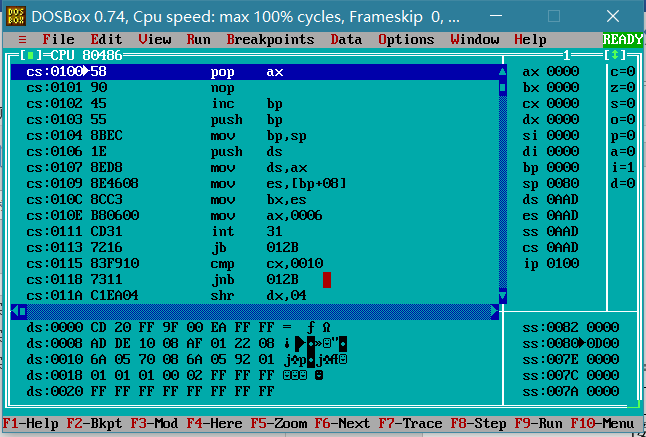
首先是实验使用的数据x1，x2，它们的补码表示，如表1-1所示：

先计算求和的标志位：

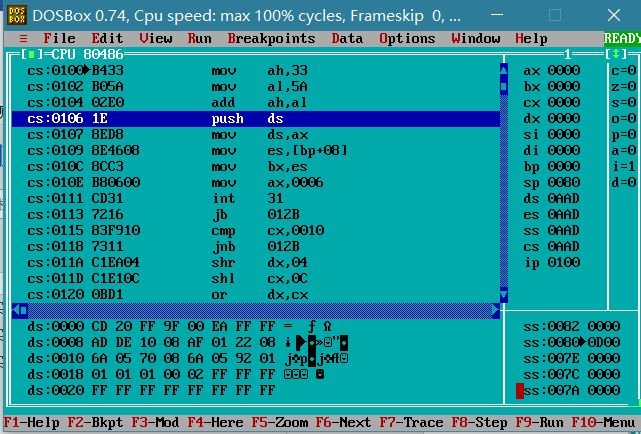
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | 和 | SF | OF | CF | ZF |
| 33H | 5AH | 8DH | 1 | 1 | 0 | 0 |
| D7H | A3H | 7AH | 0 | 1 | 1 | 0 |
| 65H | A3H | 08H | 0 | 0 | 1 | 0 |

接下来测试在CPU上执行汇编代码的真实情况。

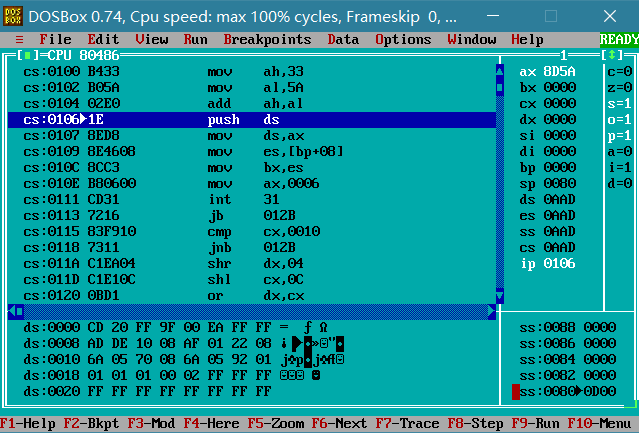
计算第一个表达式的值，先打开dosbox，挂载磁盘，切换磁盘，输入td打开程序窗口，界面如图所示：



此时直接将光标移到箭头所指处，直接输入指令MOV AH, 33H和MOV AL, 5AH，输入指令如图所示：

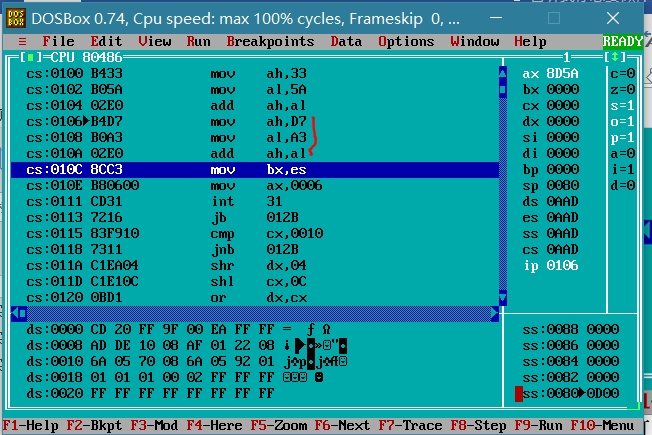


按F8单步运行指令，记录四个指示标志位，结果如图所示：

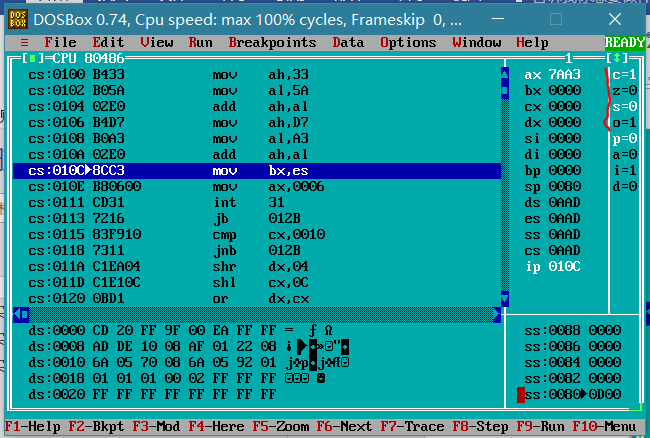


可见SF为1，表示结果是负数，ZF为0表示结果不为0，OF为1表示发生了溢出，CF为0表示没有发生进位或者借位，与预测相同。

测试第二个表达式，指令如图所示：

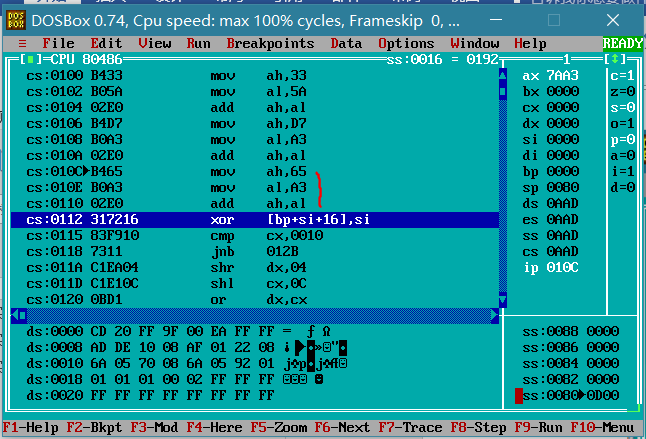


运行指令如图所示：

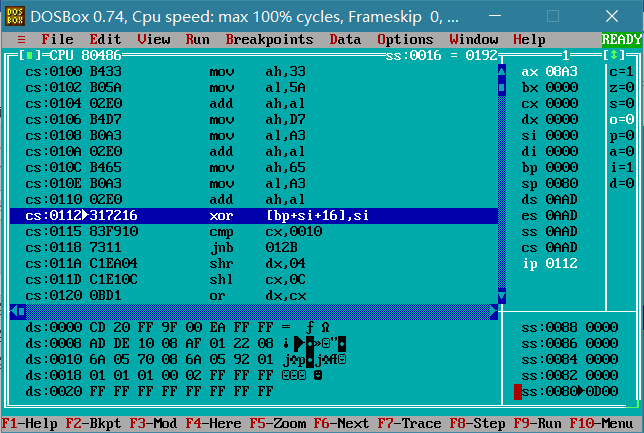


可见CF为1表示产生了进位，OF表示发生了溢出，ZF为0表示结果不为0，SF为0表示结果为整数，与预测结果相同。

输入第三个表达式的指令，如图所示：



运行指令如图所示：

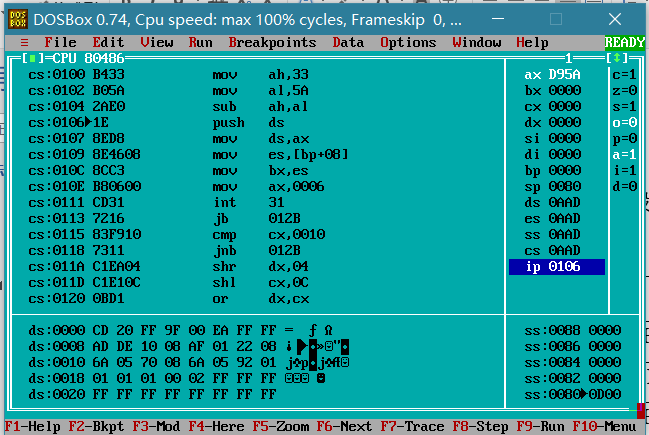


CF为1表示发生了进位，ZF为0表示结果不为0，SF为0表示结果为正数，OF为0表示运算没有发生溢出，预测正确。

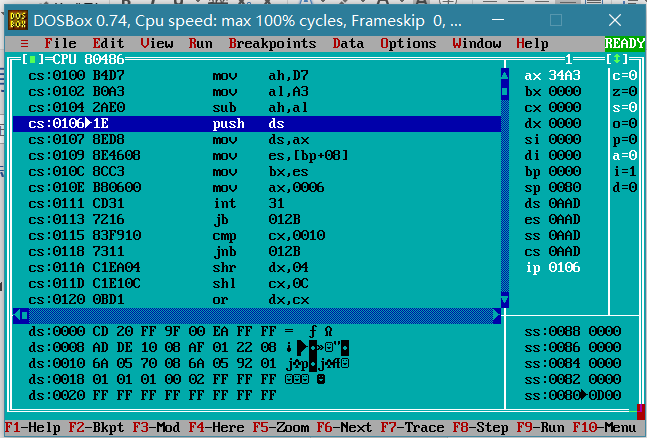
在进行减法运算时结果的预测：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 | X2 | 差 | SF | OF | CF | ZF |
| 33H | 5AH | D9H | 1 | 0 | 1 | 0 |
| D7H | A3H | 34H | 0 | 0 | 0 | 0 |
| 65H | A3H | C2H | 1 | 1 | 1 | 0 |

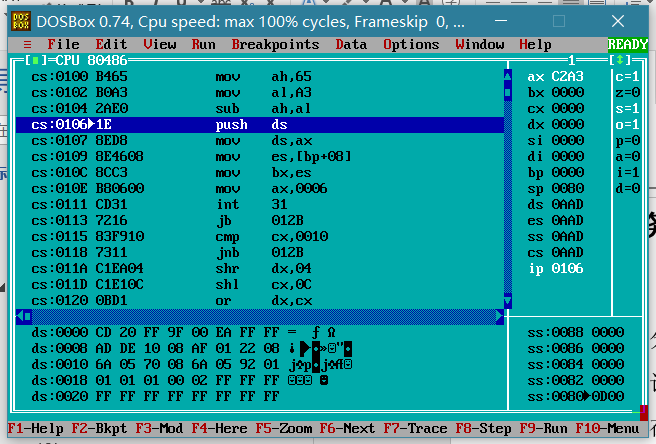
进行第一组数据：



第二组数据：



第三组数据：



标志位的改变和预测的相同。

## 任务二

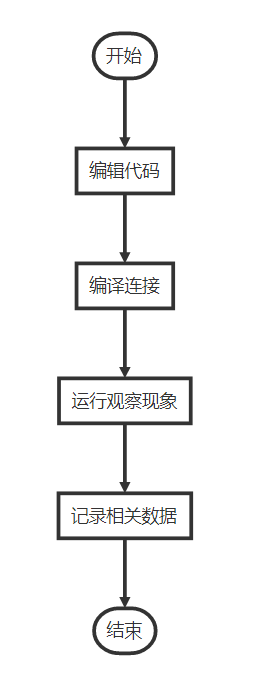
任务二是运行课本上45页题2.3的汇编代码，观察现象。

1. 分别记录执行到“MOV CX，10”和“INT 21H”之前的(BX), (BP),(SI),(DI)各是多少。
2. 记录程序执行到退出之前数据段开始40个字节的内容，指出程序运行结果是否与设想的一致。
3. 在标号LOPA前加上一段程序，实现新的功能：先显示提示信息“Press any key to begin!”, 然后，在按了一个键之后继续执行LOPA处的程序。

### 设计思想

由于课本上已经给出了代码，只需要将代码写入文本文件保存并使用ml编译链接，使用td打开程序进行运行。

### 流程图



### 源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:DATA, SS:DATA

START:

MOV AX, DATA

MOV DS, AX

; code after here

MOV SI, OFFSET BUF1

MOV DI, OFFSET BUF2

MOV BX, OFFSET BUF3

MOV BP, OFFSET BUF4

MOV CX, 10; count number

LOPA: MOV AL, [SI]; let ax = BUF1[0]

MOV [DI], AL

INC AL; add 1

MOV [BX], AL

ADD AL, 3; add 3

MOV DS:[BP], AL

INC SI

INC DI

INC BP

INC BX

DEC CX

JNZ LOPA; check jmp or not

;stop here

MOV AH, 4CH

INT 21H

CODE ENDS

END START

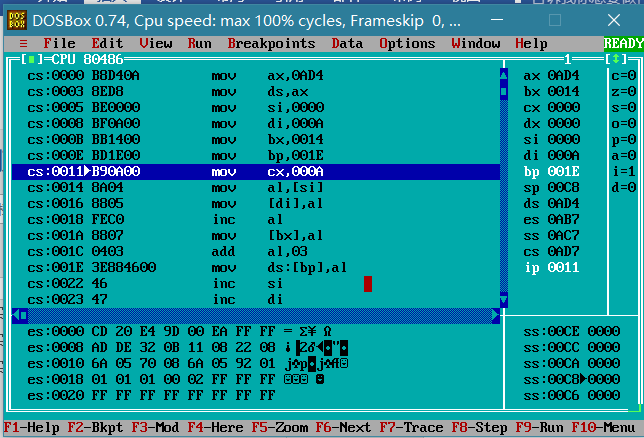
### 实验步骤

1. 编写代码：使用visual studio编写汇编代码并保存为asm文件。
2. 打开dosbox，挂载d:\github\asm\asm为c盘。
3. 输入c:切换到c盘，输入ml mission2.asm完成编译和连接。
4. 输入td mission2运行程序。
5. 单步运行程序，分别记录执行到“MOV CX，10”和“INT 21H”之前的(BX), (BP),(SI),(DI)各是多少，记录程序执行到退出之前数据段开始40个字节的内容，指出 程序运行结果是否与设想的一致。
6. 修改程序：在标号LOPA前加上一段程序，实现新的功能：先显示提示信息“Press any key to begin!”, 然后在按了一个键之后继续执行LOPA处的程序。

### 实验记录

使用visual studio编写代码并保存到d:\github\asm\asm文件夹，运行dosbox执行编译连接后运行程序。记录如下。

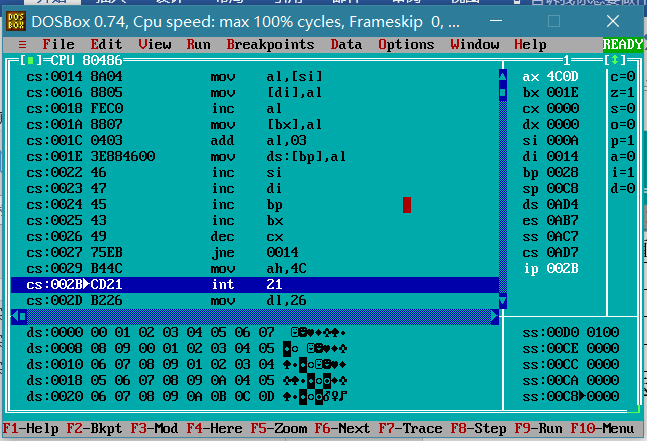
程序执行到MOV CX, 10之前的BX，BP，SI，DI如图所示：



值分别为：

|  |  |
| --- | --- |
| BX | 0014H |
| BP | 001EH |
| SI | 0000H |
| DI | 000AH |

运行程序到INT 21H，记录此时的BX，BP，SI，DI如图所示：



数据：

|  |  |
| --- | --- |
| BX | 001EH |
| BP | 0028H |
| SI | 000AH |
| DI | 0014H |

阅读汇编代码可知代码的功能是将BUF1中的值进行操作后存储到BUF2,BUF3,BUF4中，BUF2中的为原值，BUF3中为加1后的值，BUF4为加上4之后的值，而且是顺序写入，数据应该是：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BUF1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 9BUF2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| BUF3 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A |
| BUF4 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D |

参考上方截图中的数据可知程序正确。

修改代码让运行LOPA前输出一个字符串并等待用户按键以继续，本功能需要使用BIOS的功能，主要为9号调用和1号调用，需要注意的是输出的字符串必须是’$’结尾的，我尝试了一下如果不是’$’结尾的字符串将会一直输出直到遇到内存中一个’$’字符。修改后的代码：

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

WAIT\_STR DB "Press any key to begin!$"; dolar in the end

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:DATA, SS:DATA

START:

MOV AX, DATA

MOV DS, AX

; code after here

MOV SI, OFFSET BUF1

MOV DI, OFFSET BUF2

MOV BX, OFFSET BUF3

MOV BP, OFFSET BUF4

MOV CX, 10; count number

LEA DX, OFFSET WAIT\_STR;load offset

MOV AH, 9;set function

INT 21H ; call function

; wait for key

MOV AH, 1

INT 21H

; begin loop

LOPA: MOV AL, [SI]; let ax = BUF1[0]

MOV [DI], AL

INC AL; add 1

MOV [BX], AL

ADD AL, 3; add 3

MOV DS:[BP], AL

INC SI

INC DI

INC BP

INC BX

DEC CX

JNZ LOPA; check jmp or not

;stop here

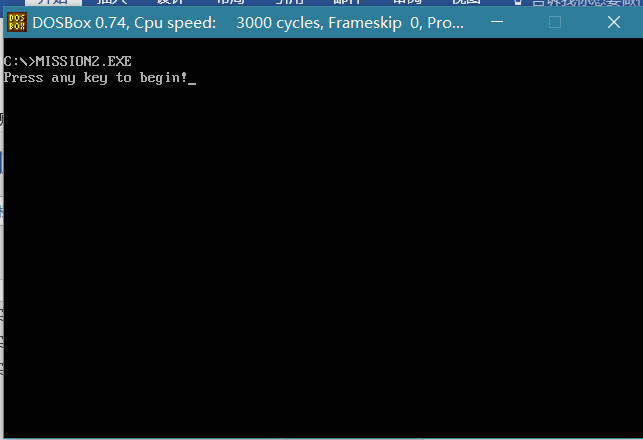
MOV AH, 4CH

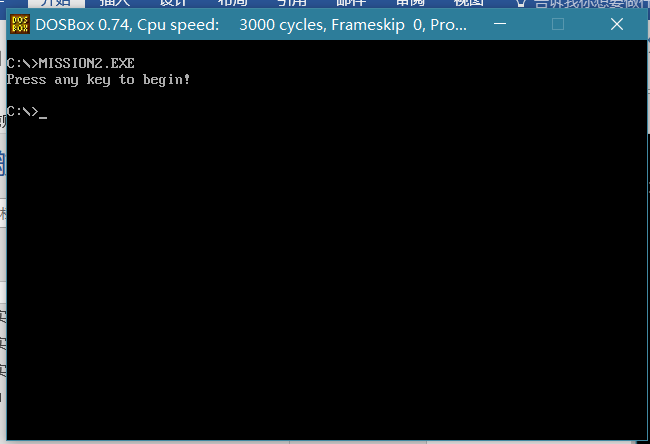
INT 21H

CODE ENDS

END START

运行截图：





修改正确。

## 任务三

对教材45页2.4题代码的改写，实现使用32位寄存器变址寻址方式运行，记录数据区前40个字节的内容。

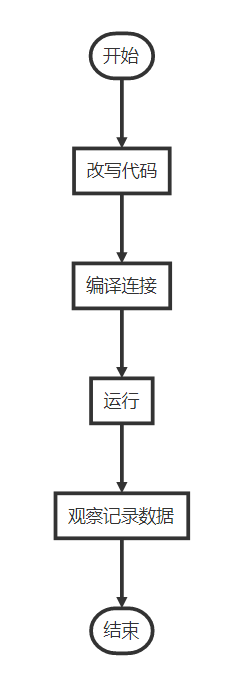
### 设计思想

本任务为使用32位寄存器变址寻址运行程序，记录数据段前40个字节的内容，所以只需要将4个寻址使用的寄存器替换为32位的即可。

变址寻址的实现：使用EBX存放BUF1的地址，根据他们在内存中的位置可以知道BUF2的地址位EBX+10，BUF3地址为EBX+20，BUF4地址为EBX+30，由此就可以改写为变址寻址。

寄存器和内存空间的分配：由于只是修改代码使其能够使用32位寄存器，还有就是使用变址寻址，那么我直接将两个功能结合到一起，使用32位寄存器对数据进行变址寻址，只使用一个ESI寄存器存放地址，CX计数。

### 流程图



### 源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF1 DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

DATA ENDS

CODE SEGMENT USE16

ASSUME CS:CODE, DS:DATA, SS:DATA

START:

MOV AX, DATA

MOV DS, AX

; code after here

MOV ESI, OFFSET BUF1

MOV CX, 10

LOPA: MOV AL, [ESI]

MOV [ESI + 10], AL; according memory order

INC AL

MOV [ESI + 20], AL

ADD AL, 3

MOV [ESI + 30], AL

INC ESI

DEC CX

JNZ LOPA

;stop here

MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 实验步骤

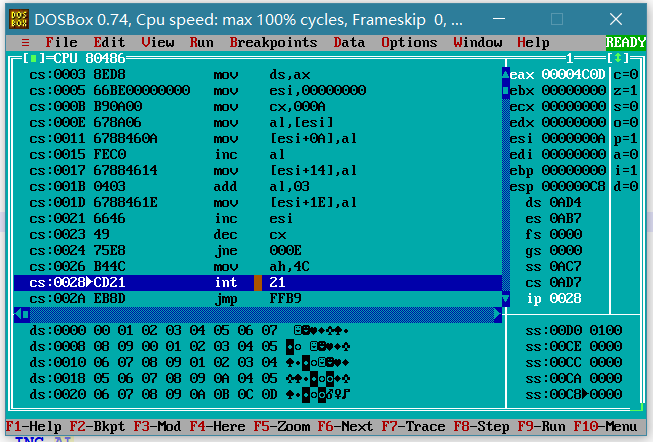
1. 实验步骤和任务二相同。

### 实验记录

记录主要是记录代码运行前后数据区前40个字节的内容。

编译连接没有报错，开始使用td运行程序。

单步运行程序直到INT 21H，记录截图如下：



运行结果与使用4个16寄存器表示地址时的结果相同。

# 体会

这次的汇编实验总体来说由于我的准备的比较充分，任务书上说的也比较清楚，实验时没有遇到什么较大的困难。通过这次的汇编实验我知道了在内存中代码和数据属于同一个地位，都只是内存中的0101而已，只是被不同的CPU部件读取而导致意义不同，这也就是冯·诺依曼架构的计算机的同一个概念，数据与指令没有区别。这次实验还见识到了计算机对于数据表示并没有什么正负之分，都是存储的补码，加就是直接加，减就是加补码，对我理解计算机系统有很大帮助。

# 参考文献

[1] 王元珍等.80x86汇编语言程序设计.版本(第1版)

[2] 王晓虹等.汇编语言程序设计教程.版本(第1版)