**计算机科学与工程学院实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验课程名称** | | **汇编语言程序设计实验** | | | **实验总成绩** |  |
| **专业** | **计算机科学与技术** | | **班级** | **计算机2001** | **指导教师签字** |  |
| **学号** | **20205962** | | **姓名** | **袁子宸** | **实验报告批改时间** |  |
| **实验报告分项成绩**   |  |  |  | | --- | --- | --- | | **序号** | **实验项目** | **成绩** | | **1** |  |  | | **2** |  |  | | **3** |  |  | | **4** |  |  | | **5** |  |  | | | | | | | |
| **实验课程总结**  从以下方面总结：1.实验体现知识应用和初步研究能力；2. 反映基本观察、发现问题和分析问题能力；3. 实验项目内容或者实验课程是否存在问题及下一年度改进意见）4.其他方面  通过本次实验，我掌握了对于子程序设计的基本要点，明白了子程序设计的基本流程，了解了汇编语言循环子程序定义伪指令PROC具体使用方法，对于使用汇编语言解决基本问题积累了一定的经验。在实验过程中，我遇到了若干问题，在经过查阅资料，询问同学的过程中，对这些问题有了更深刻地认知，最终成功地解决。汇编语言实验这门课程设置的非常科学，在课上学完基础知识后，亟需实验课程来加深印象，实验课老师十分负责，对于学生提出的问题都能给予详细解答，总体来讲，是一门非常不错的课程，我在这门课程中收获了很多。 | | | | | | |

## 实验五

## （一）、实验目的

汇编语言程序设计实验(上机练习)是必不可少的学习环节之一,也是检验掌握汇编语言程序设计知识的手段之一。

一方面通过上机练习，可加深对课程内容的理解和掌握。 另一方面，汇编语言编制的程序上机调试、运行是检验程序设计是否正确的唯一途径。

要求学生要熟悉上机练习的四个过程：编辑、汇编、连接和调试。熟悉和掌握编辑、汇编、连接和调试四个实用程序的使用方法，掌握调试程序中的几个常用命令的使用方法。

## （二）、实验要求

求一个多字节数据的绝对值及求两个多字节数据的和的功能分别用子程序来实现；

分别用三组数据调试程序，检验程序的正确性：

    ① 两个多字节数据都是正数；

    ② 两个多字节数据都是负数；

    ③ 两个多字节数据一正一负。

## （三）、实验内容（各题目的题干）

已知两个长度相等的带符号的多字节数据分别存放在内存DATA1和DATA2开始的连续单元中，数据长度存放在LEN单元。编制程序，计算两个数据的绝对值之和，将结果存放在SUM开始的连续单元中。

## （四）、实验环境及实验原理

#### 4.1实验环境

环境：DosBox

汇编：MASM

链接：LINK

调试：DEBUG

编写：Visual Studio Code

#### 4.2实验原理

1. 子程序定义伪指令：

PN PROC［NEAR］/［FAR］ ；说明过程开始

: ；过程体:

PN ENDP ；说明过程结束

2. 取补指令：

NEG xx

功能： 对给定数据取其补码。

3. 带进位加法指令：

ADC xx，xx

功能：将两操作数相加，并加上状态标志位CF

4. 带借位减法

SBB xx,xx

功能：将两操作数相减，并减去状态标志位CF

5. 算法描述

（1）、多字节数据取绝对值函数：

首先判断数据正负，若为正数，则绝对值为其本身；若为负数，则绝对值为其原码。

对多字节负数求其原码的算法：对负数求其原码等同于对其补码再取补，即可得到正数原码。首先，对其最低字节取补，取补后会影响状态标志位CF，对CF取反；后所有该字节数据均取反加上CF最终即可得到负数原码。此外，亦可每位高字节取补，再减去CF，也可得到相同效果。

（2）、多字节数据相加函数：

先对最低位字节求和，求和后会影响状态标志位CF，其余高字节做带进位加法即可求得多字节数据之和。

#### 4.3 程序流程图

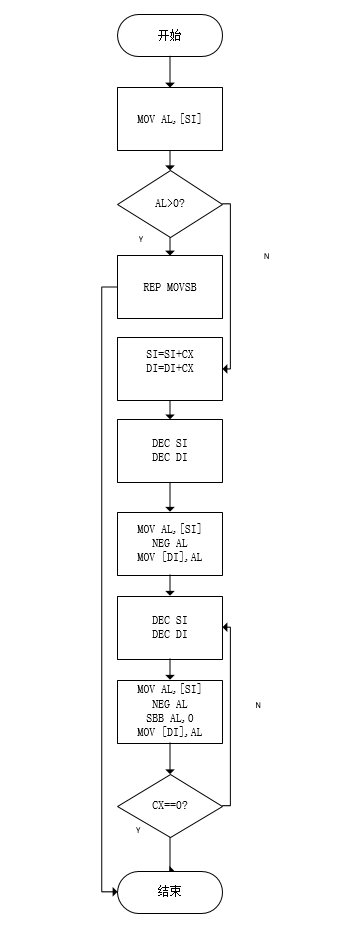


图1求绝对值函数流程图

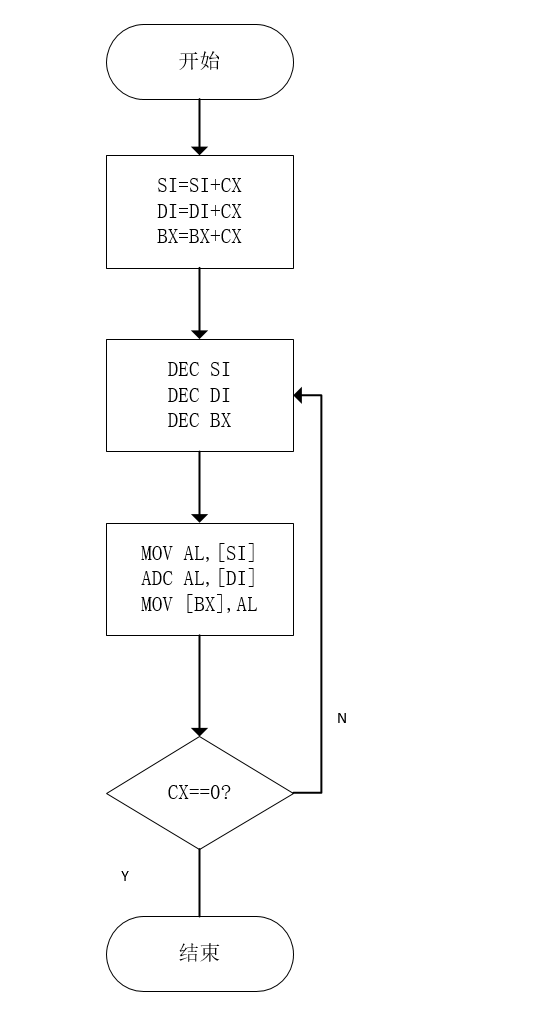


图2求多字节加法函数流程图

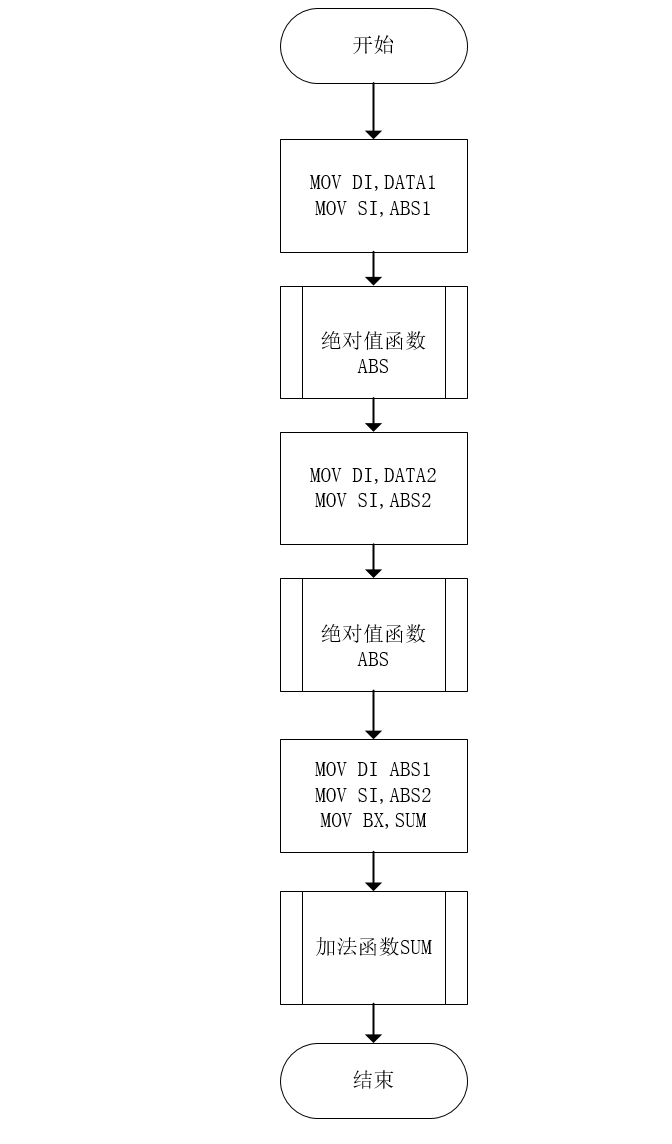


图3主程序流程图

## （五）、源程序（加注释）

DATAS SEGMENT

DATA1 DB  88H,89H,90H;被加数

DATA2 DB  12H,34H,56H;加数

COUNT DW 3

ABS1 DB 3 DUP(0);绝对值1

ABS2 DB 3 DUP(0);绝对值2

SUM DB 3 DUP(0);和

DATAS ENDS

STACKS SEGMENT

    STK DW 20 DUP(0)

STACKS ENDS

CODES SEGMENT

    ASSUME CS:CODES,DS:DATAS,SS:STACKS

;-------------------------------------------绝对值函数

 ABS  PROC NEAR;受影响的寄存器：AX，SI，DI，CX

        MOV AL,[SI]

        OR AL,AL;判断正负，负数去补，正数保持

        JS BELLOW

        CLD

        REP MOVSB

        JMP FINISH

 BELLOW:

        STD

        ADD SI,CX

        ADD DI,CX

        DEC SI

        DEC DI

        LODSB

        NEG AL;处理最低字节

        STOSB

        DEC CX

        JZ FINISH

AGAIN:  ;对高字节去补减去cf，或者取反加上非cf，二者等效

        LODSB

        NEG AL;取补

        SBB AL,0;减去1

        STOSB

        LOOP AGAIN

 FINISH:

        RET

 ABS    ENDP

 ;--------------------------------------加法函数

 SUMFUN   PROC NEAR

        ADD SI,CX

        ADD DI,CX

        ADD BX,CX

        CLC

AGAIN1:

        DEC SI;被加数地址

        DEC DI;加数地址

        DEC BX;和地址

        MOV AL,[SI]

        ADC AL,0

        ADD AL,[DI]

        MOV [BX],AL

        LOOP AGAIN1

        RET

SUMFUN     ENDP

START:

    MOV AX,DATAS

    MOV DS,AX

    MOV ES,AX

    MOV CX,COUNT

    MOV SI,OFFSET DATA1;源操作数首地址

    MOV DI,OFFSET ABS1;绝对值首地址

    PUSH CX

    CALL ABS

    POP CX

    MOV SI,OFFSET DATA2;源操作数首地址

    MOV DI,OFFSET ABS2;绝对值首地址

    PUSH CX

    CALL ABS

    POP CX

    MOV SI,OFFSET  ABS1;加数首地址

    MOV DI,OFFSET ABS2;被加数首地址

    MOV BX,OFFSET SUM;和的首地址

    PUSH CX

    CALL SUMFUN

    POP CX

    MOV AH,4CH

    INT 21H

CODES ENDS

    END START

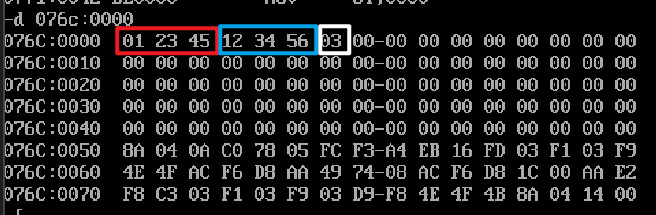
## （六）、运行结果（贴图）需要有debug截屏记录每次调试程序所输入的数据以及显示器上输出的结果

### 6.1第一组数据：  两个多字节数据都是正数

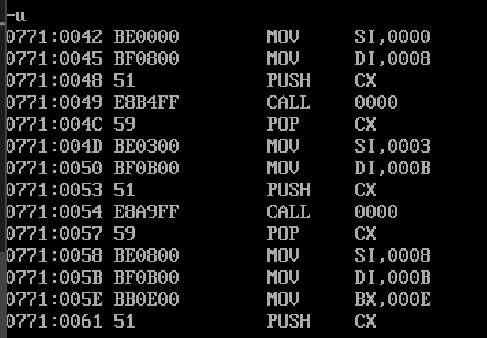
DATA1 DB  01H,23H,45H;被加数

DATA2 DB  12H,34H,56H;加数

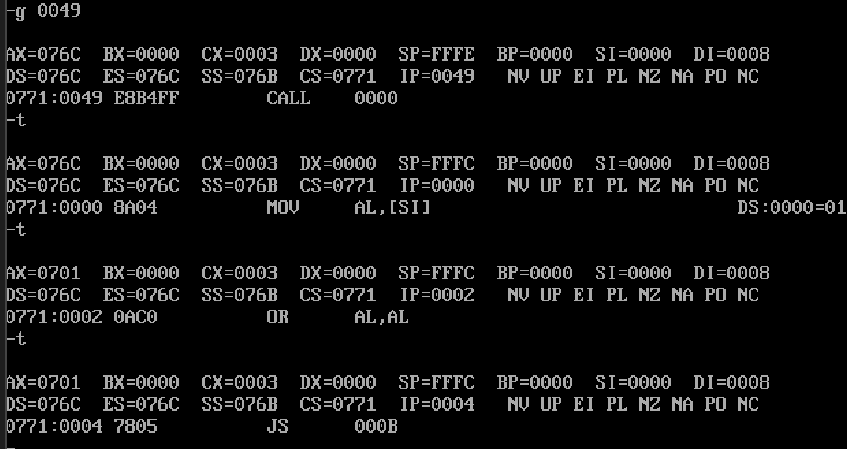
1. D命令观察数据段中数据，红框内为初始DATA1数据, 蓝框内为初始DATA2数据；白框内为数据字节长度3个。



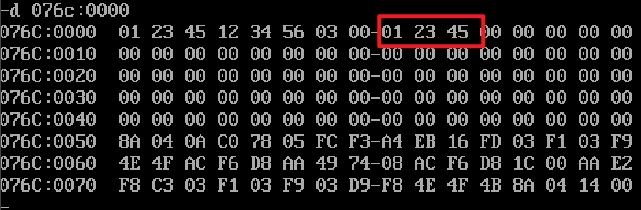
2. U命令反汇编，G命令运行置CALL命令处，即0049处，调用一次求绝对值函数，观察内存状态。



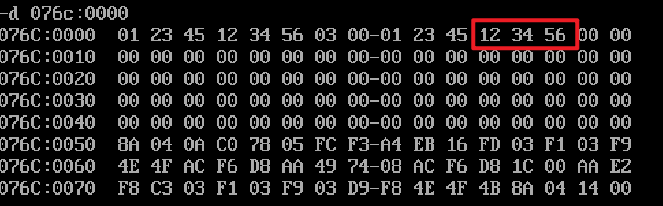
执行CALL命令后，程序成功跳转至子程序内部。



子程序执行结束后，DATA1的绝对值被存放于ABS1处，即红框处所选中的数据。



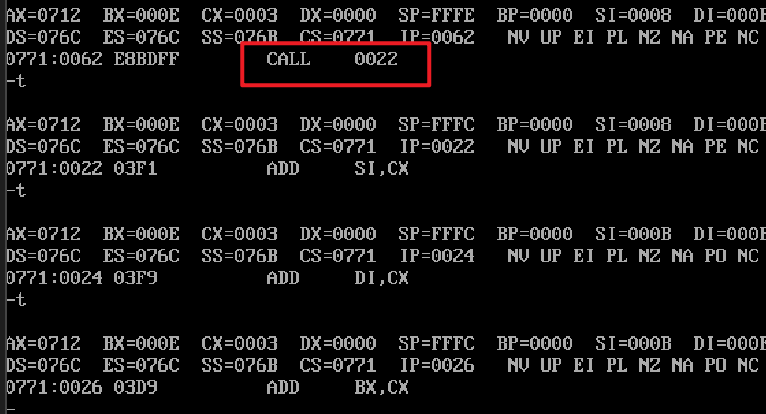
继续执行主程序，再次调用CALL指令，求出DATA2的绝对值存于ABS2处。



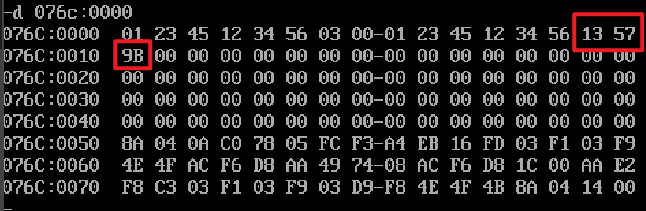
可以观察到，两个数据绝对值完全正确。

3.跳转至加法子程序处，计算两绝对值之和。

进入加法子程序



程序结束观察结果，完全正确。

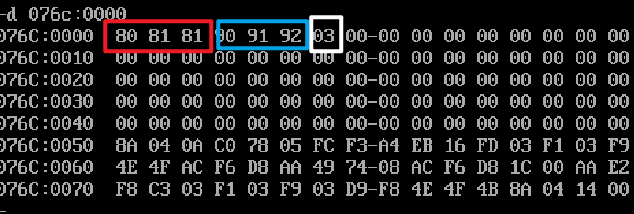


### 6.2第二组数据：全部负数

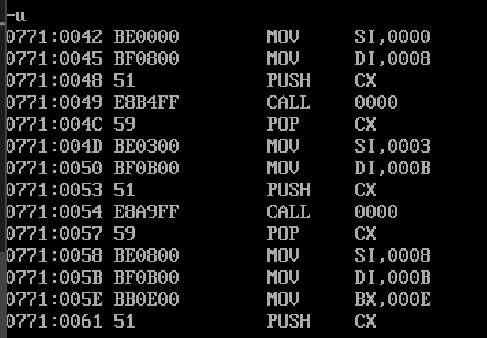
DATA1 DB  80H,81H,81H;被加数

DATA2 DB  90H,91H,92H;加数

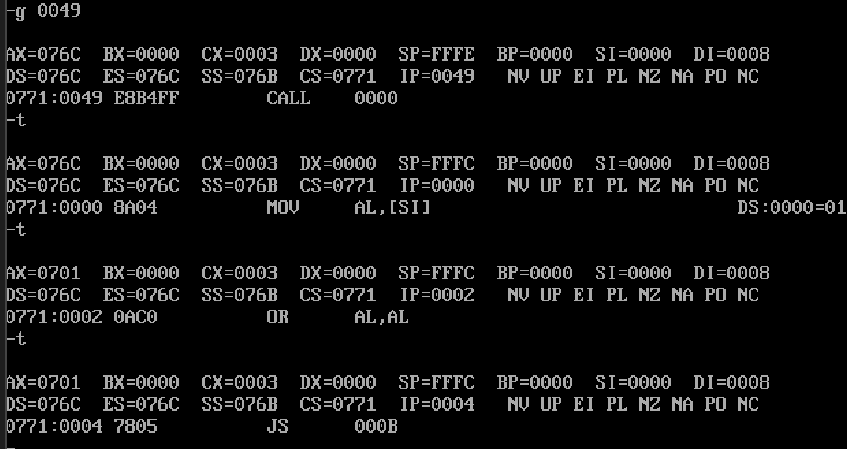
1. D命令观察数据段中数据，红框内为初始DATA1数据, 蓝框内为初始DATA2数据；白框内为数据字节长度3个。



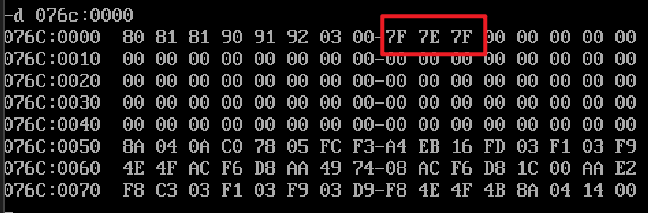
2. U命令反汇编，G命令运行置CALL命令处，即0049处，调用一次求绝对值函数，观察内存状态。



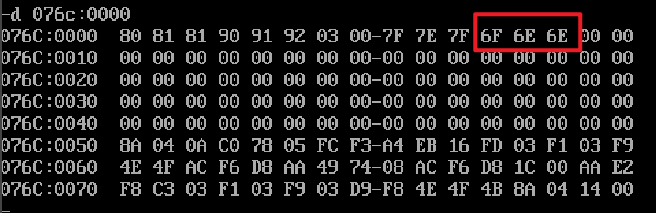
执行CALL命令后，程序成功跳转至子程序内部。



子程序执行结束后，DATA1的绝对值被存放于ABS1处，即红框处所选中的数据。



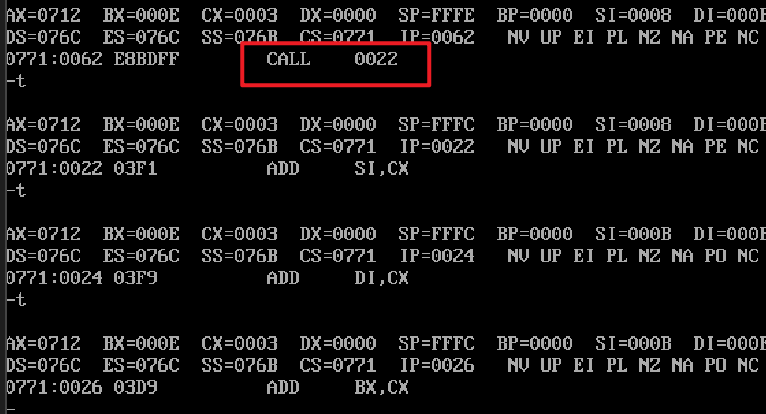
继续执行主程序，再次调用CALL指令，求出DATA2的绝对值存于ABS2处。



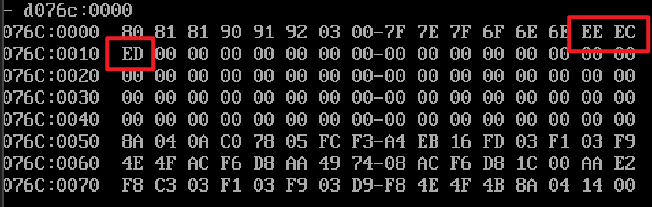
可以观察到，两个数据绝对值完全正确。

3.跳转至加法子程序处，计算两绝对值之和。

进入加法子程序



程序结束观察结果，完全正确。

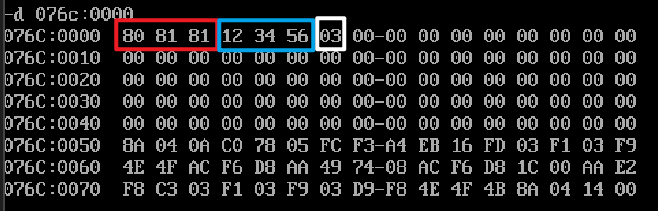


### 6.3第三组数据：两个多字节数据一正一负

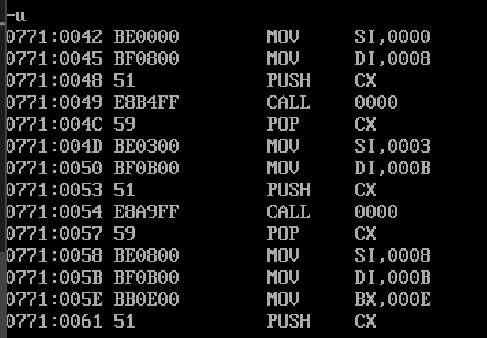
DATA1 DB  80H,81H,81H;被加数

DATA2 DB  12H,34H,56H;加数

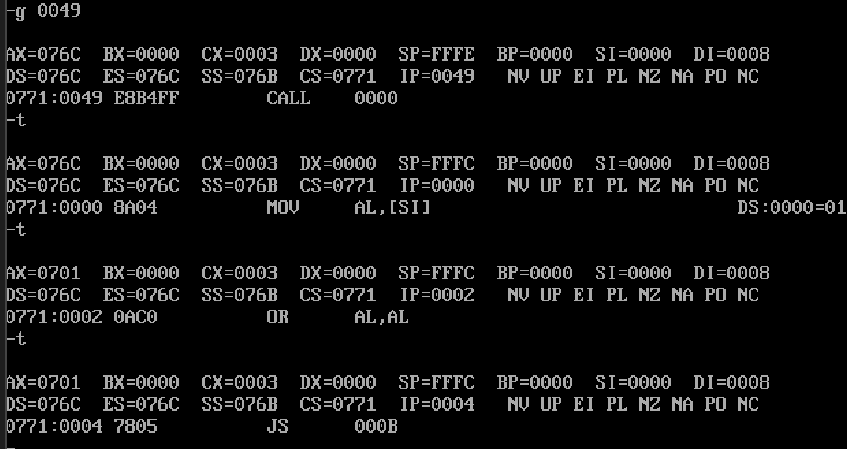
1. D命令观察数据段中数据，红框内为初始DATA1数据, 蓝框内为初始DATA2数据；白框内为数据字节长度3个。



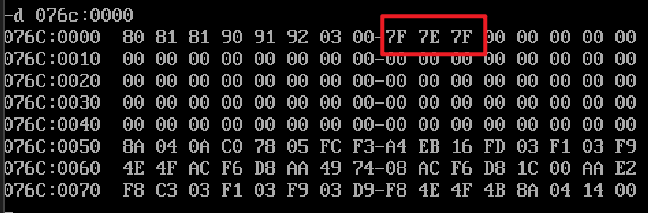
2. U命令反汇编，G命令运行置CALL命令处，即0049处，调用一次求绝对值函数，观察内存状态。



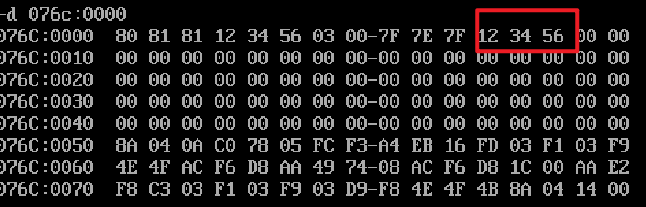
执行CALL命令后，程序成功跳转至子程序内部。



子程序执行结束后，DATA1的绝对值被存放于ABS1处，即红框处所选中的数据。



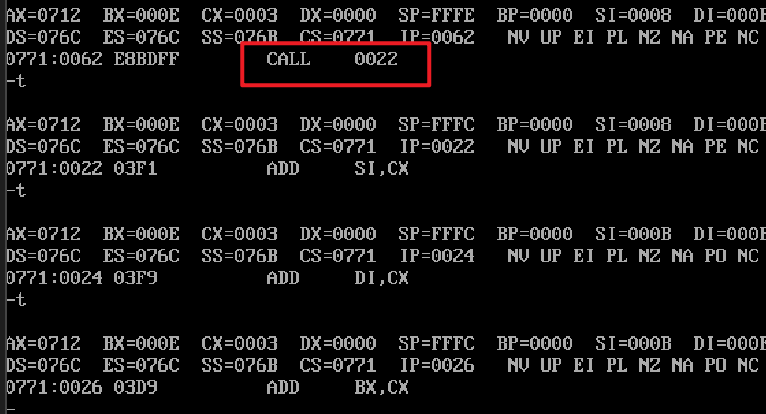
继续执行主程序，再次调用CALL指令，求出DATA2的绝对值存于ABS2处。



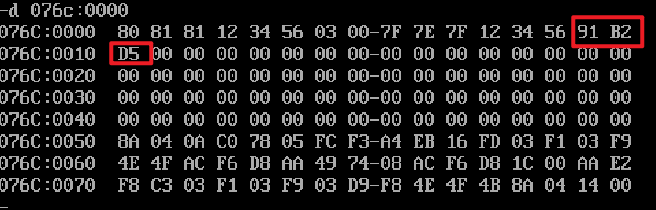
可以观察到，两个数据绝对值完全正确。

3.跳转至加法子程序处，计算两绝对值之和。

进入加法子程序



程序结束观察结果，完全正确。



## (七)、心得体会：学到了什么；遇到的问题及解决方法等

1. 通过本次实验，我对与汇编语言的子程序设计有了更深刻地认知，对于如何设计子程序以及调用子程序解决实际问题有了更多的经验。

2. 掌握了汇编语言车子程序定义伪指令PROC与ENDP，以及子程序调用指令CALL的使用方法。

3.掌握了 MOVSB语句的功能：即将SI所指向的单元存入到DI所知单元中，同时改变DI，SI。

4. 在debug的过程中，我发现输出结果与期望结果不同，分析原因应为数据存储出现问题。在检查之后，成功解决此问题。

5. 懂得了如何去实现对多字节数据求其绝对值。

6. 懂得了多字节数据的加法实现原理，并能成功编写一个多字节加法程序。