[redacted]

汇编语言程序设计

（上机大作业报告）

学　号　：[redacted]

姓　名　：[redacted]

专业/年级：[redacted]

所在院系　：[redacted]

任课教师　：[redacted]

完成日期　：[redacted]

（正文格式：宋体，小四，首行缩进 2 汉字，单倍行距）

一、 设计题目

从键盘输入一个简单的表达式，如“S=8+6*9-2+7/5”，按回车结束输入，则屏幕显示 S=61.4，小数点保留 1 位，四舍五入。假设输入的表达式中只含个位十进制数和“+”、“-”、“*”、“/”运算符，且同一运算符最多出现 2 次。

思考：如果取消上述只能输入个位数的限制，允许输入 10000 以下的任意十进制数，程序如何完成？

二、 设计说明

1. 正确描述整个程序的功能，完成什么样的工作。

该程序完成的功能是完成只含个位十进制数的加减乘除四则运算，结果以四舍五入保留一位小数的方式显示在屏幕上。该程序可以实现算式以字符串的形式输入，对字符串中的数字、字母和运算符进行处理，按照表达式进行计算，并以十进制的形式，四舍五入保留一位小数正确显示。

2. 把整个工作划分成多个任务（子程序），并说明调用关系。

1）字符串的输入
2）字符串的预处理
3）乘 100 处理
4）表达式中乘除法的优先计算
5）表达式中加减法的运算
6）十六进制转化为十进制
7）结果的输出
8）回车宏定义

（各个子任务顺序执行，输入部分调用回车宏定义）

3.确切地定义每个子程序的功能，它与其它子程序之间的参数传递说明。

1）字符串输入部分

在数据段输入缓冲区 BUFFER，存放输入的字符，使用 CRLF 宏回车表示结束。

2）字符串的预处理部分

在数据段创建一个数组 COPY，主要工作是将 BUFFER 数组中的字节类型元素转化成字元素，以便后期运算和对数字的存储；并将输入字符串的数字字符通过 ASCII 码转化为数字，以便后期运算。所有转化后的字符都以字的形式存储在 COPY 数组内。

3）乘 100 处理部分

在加减运算过程中，将加数、减数和被减数分别进行乘 100 处理，将乘数中的一个和被除数也进行乘 100 处理。因为后期要进行保留一位小数以及四舍五入的操作，所以先对数字进行类似移位的处理，结果依然存储在 COPY 数组内。

4）表达式中乘除法的优先计算部分

利用计数循环识别出乘数、被除数和除数，并根据符号进行计算。乘除法运算结果都存在 DX:AX 中，并再挪回 COPY 数组中后一个乘数或除数的位置，并将前一个乘数或被除数、乘号或除号置零，一遍后续的计算。结果存在 COPY 数组内。

5）表达式中加减法的运算部分

在本程序中，加减法的运算通过计数循环识别出加号和减号。这部分利用 BX 作为标志，当 BX 置为 0 时作加法计算，BX 置为 1 的时候作减法计算。在前面对字符串的处理后，所有因式都已经乘 100，并已经完成乘除法的计算，各个无用的符号以及数字都已经转化为 0。此时就根据 BX 寄存器的值来判断是进行加法还是减法操作，无用的符号和数字因为已经置零所以加减对结果不会造成影响。所有结果加起来，存在 SUM 变量中。

6）十六进制转化为十进制部分

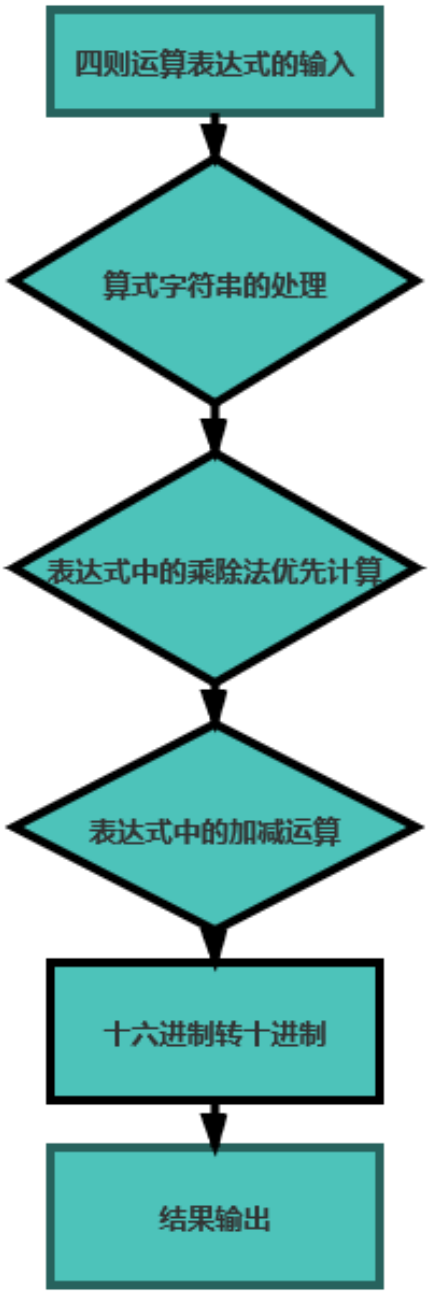
SUM 变量内已经是乘了 100 倍的正确答案，此部分要做的是进行数制转换。因为 DOS 系统中，输出是以 ASCII 码形式输出，需要单字符逐个输出，数字又默认为十六进制，所以需要进行十六进制转化为十进制数。通过求余的方式算出每一位的十进制数值，并以字的形式依次存储在 NUMS 数组中。

7）结果的输出部分

结果输出依赖上一步中 NUMS 数组，逐个输出并加上小数点，根据小数点后两位的数值四舍五入。

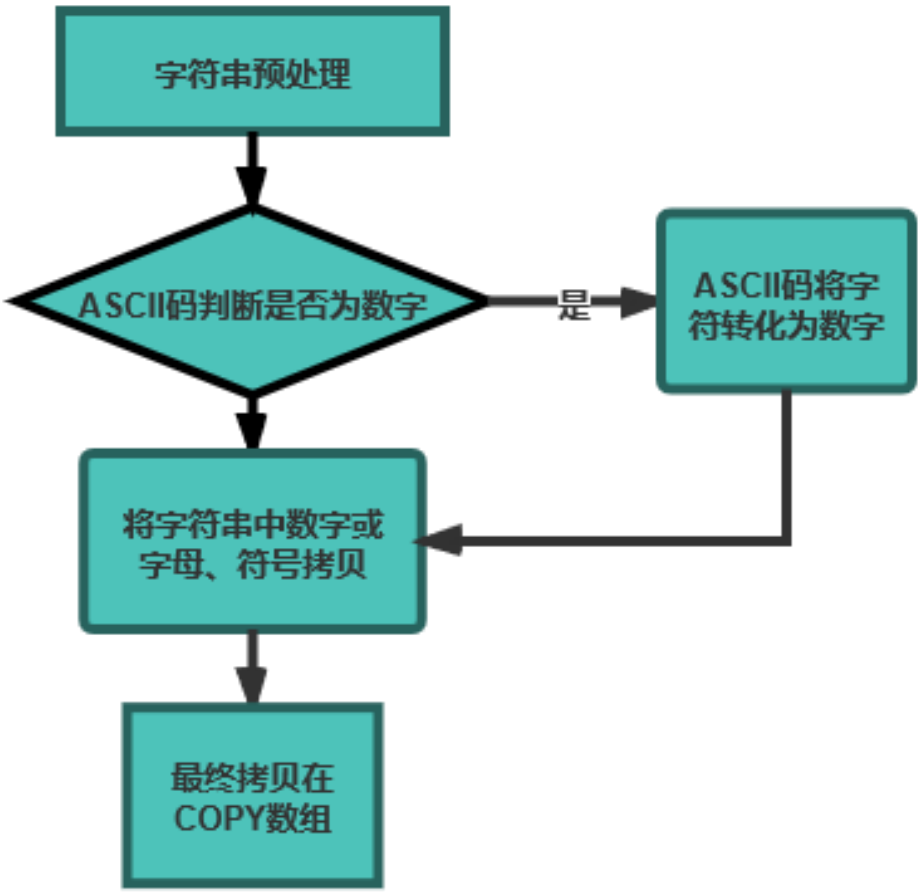
4.程序框图

整个程序的程序框图如下：

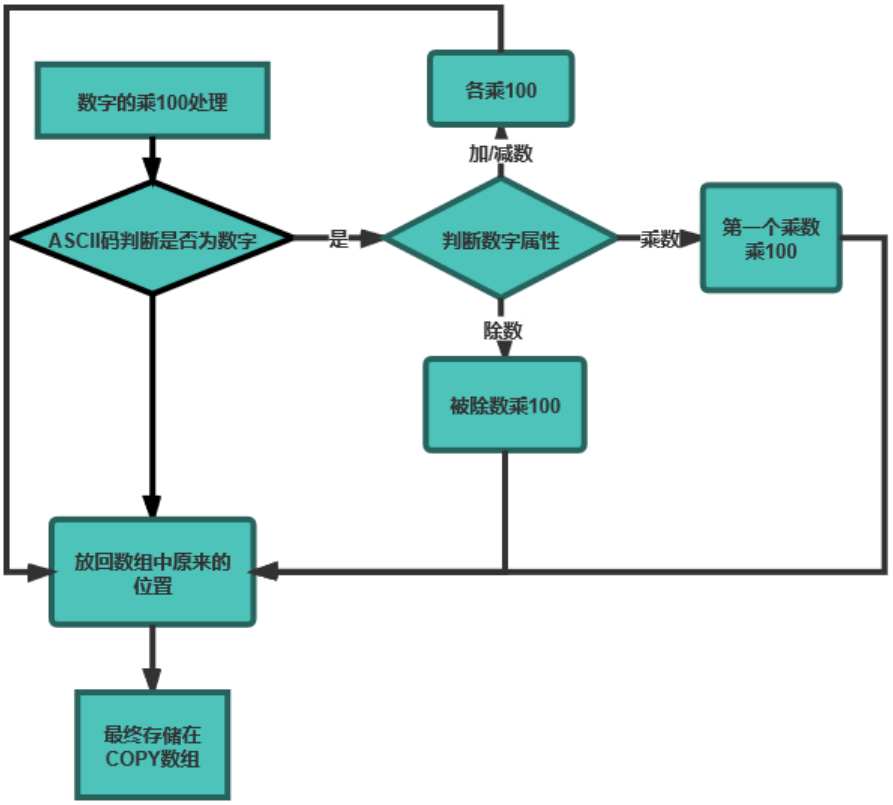


5.子程序流程图

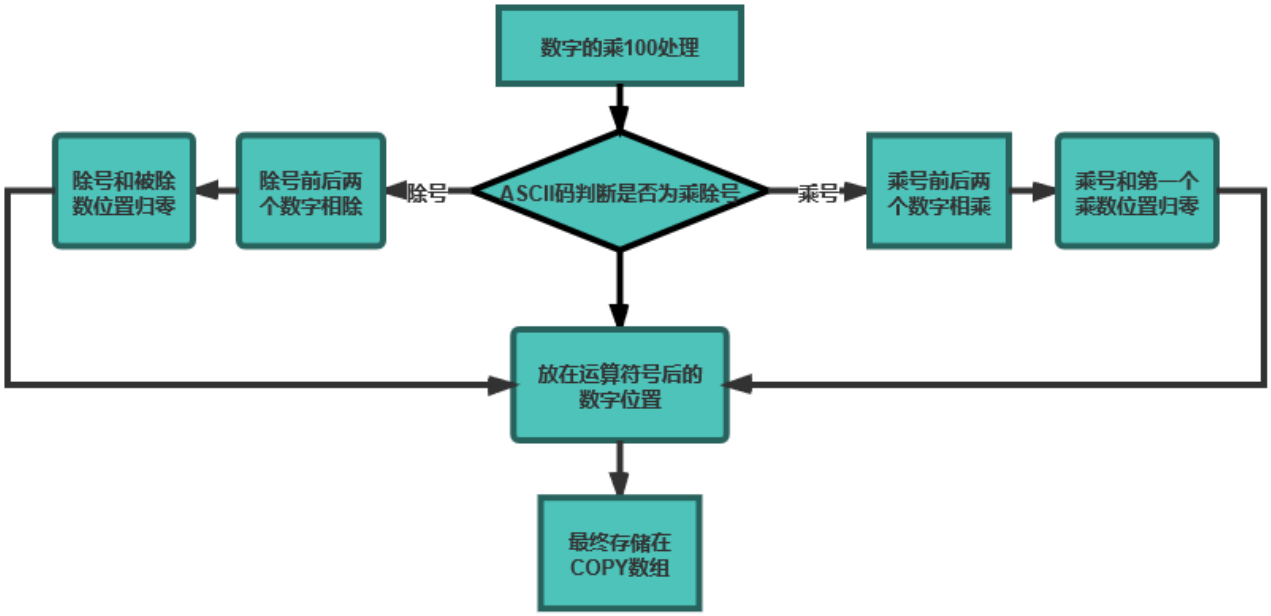
字符串预处理：



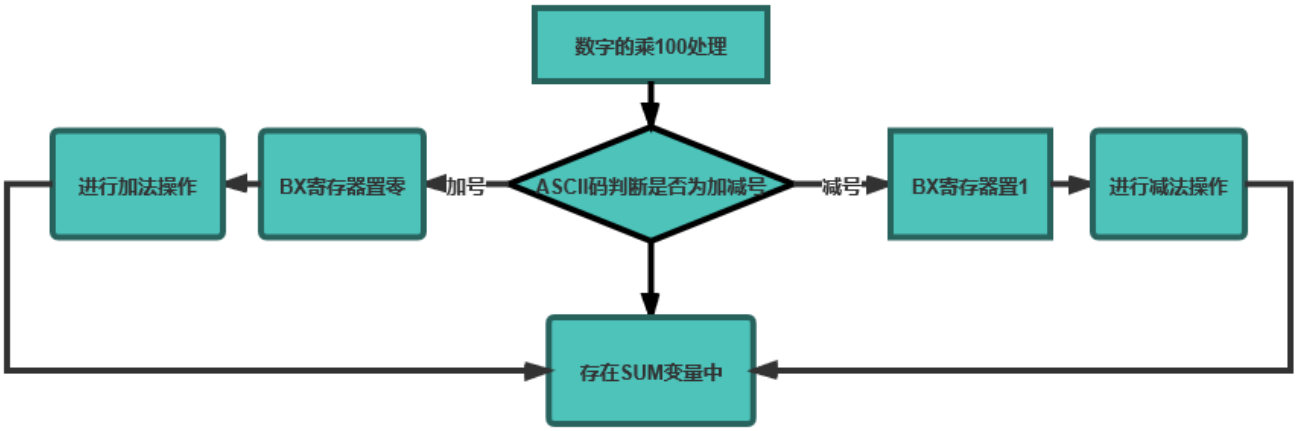
数字的乘 100 处理：



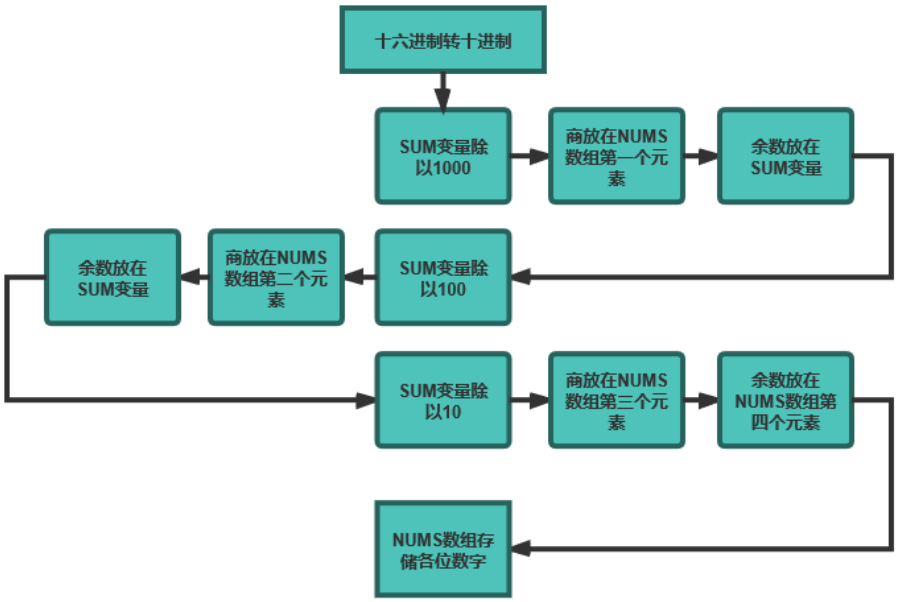
乘除法优先计算：



加减法的计算：



十六进制转十进制：



6.程序清单见报告末尾

三、 调试说明

1.调试情况

上机时遇到的问题：

1）加减乘除运算先后问题

字符串输入的时候各个字符没有优先级的区分，因此很难对乘除优先计算并利用计算后结果。

解决方法：优先识别乘除号，先进行乘除运算，并将运算结果直接放入字符串数组中。

2）连乘连除以及连续乘除问题

原来的想法是进行乘除以后将结果置于乘号或除号的位置，可是后来又要考虑到如果乘除式子后依然有乘号或者除号，前一个数字和运算符号之间又要相隔一个空位，不利于后续的乘除计算。

解决方法：修改运算结果在数组中的位置，将乘除后结果放在后一个数字的位置上，这样后面可以接上乘号或者除号，继续进行连乘或连除计算。修改后的代码如图所示。

```
MOV     WORD PTR[SI-2],0
MOV     WORD PTR[SI],0
```

3）小数计算和四舍五入的问题

原本的方案是直接进行运算，并将整数和余数部分分别挑出来进行处理。这造成了两大困难：第一是小数部分加减实现较为繁琐，第二是利用余数不容易进行十六进制转十进制的操作。

解决方法：对整个式子进行乘 100 处理，再挪动小数点，实现小数计算和四舍五入。处理代码如图所示。

```
        LEA     SI,COPY+4           ;直接从数字开始
        MOV     CL,COUNT            ;数字和运算符个数
        SUB     CL,2
        MOV     DX,100              ;准备乘100
COMP0:  MOV     AX,[SI]     ;元素放入AX
        MOV     BX,[SI-2]
        CMP     AX,0
        JB      JUMP
        CMP     AX,9
        JA      JUMP        ;判断AX中是否为数字
        CMP     BX,'*'      ;若作为乘数无需多次乘100
        JE      JUMP
        CMP     BX,'/'      ;若作为除数无需乘100
        JE      JUMP
        MUL     DL
        MOV     [SI],AX
JUMP:   ADD     SI,2
        LOOP    COMP0
```

4）调试过程中除法问题

在上机的过程中，本来选择 AX 寄存器中的值除以 8 位十六进制数字来进行除法运算，但是由于除以 8 位十六进制数时，结果直接默认存储在 AL 寄存器当中，则可能会导致溢出等问题，于是改用 DX:AX 寄存器除以 16 位十六进制数字的方法。可是随后出现了问题，即在调试过程中进入了死循环，结果没有办法正确输

出。（由图可知，输入的表达式是 s=7/4，然而在调试过程中，在进行到除法操作的时候，程序进入了死循环）

```
AX=0A6A  BX=0000  CX=01E7  DX=0000  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=14A1   NV UP EI PL NZ NA PO NC
F000:14A1 FE38          ???     [BX+SI]                             DS:0000=12
-t
s=7/4
AX=0A6A  BX=0000  CX=01E7  DX=0000  SP=FFFA  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=14A5   NV UP EI PL NZ NA PO NC
F000:14A5 CF            IRET
-
```

```
AX=02BC  BX=0004  CX=0002  DX=0064  SP=FFF4  BP=0000  SI=003A  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=1060   NV UP DI PL ZR NA PE NC
F000:1060 FE38          ???     [BX+SI]                             DS:003E=00
-t

AX=02BC  BX=0004  CX=0002  DX=0064  SP=FFF4  BP=0000  SI=003A  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=1064   NV UP DI PL ZR NA PE NC
F000:1064 CF            IRET
-t

AX=02BC  BX=0004  CX=0002  DX=0064  SP=FFFA  BP=0000  SI=003A  DI=0000
DS=076A  ES=075A  SS=0769  CS=0771  IP=00AA   NV UP EI PL ZR NA PE NC
0771:00AA F7F3          DIV     BX
-t

AX=02BC  BX=0004  CX=0002  DX=0064  SP=FFF4  BP=0000  SI=003A  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=1060   NV UP DI PL ZR NA PE NC
F000:1060 FE38          ???     [BX+SI]                             DS:003E=00
-t

AX=02BC  BX=0004  CX=0002  DX=0064  SP=FFF4  BP=0000  SI=003A  DI=0000
DS=076A  ES=075A  SS=0769  CS=F000  IP=1064   NV UP DI PL ZR NA PE NC
F000:1064 CF            IRET
-
```

```
C:\MASM>k1953463
Please input an expression:
s=7/4
```

可见，也无法运行出正常结果

解决方法：在进行了一番思考与尝试过后，发现问题在一个非常微不足道但是极为重要的点。在用DX:AX寄存器进行除法运算的时候，首先要将DX寄存器置零，否则不但结果可能会有错误，甚至还会出现无法输出、进入死循环的现象。

2.连接的要求说明

对[redacted].ASM 进行汇编，并对 [redacted].OBJ 进行连接。如图所示。

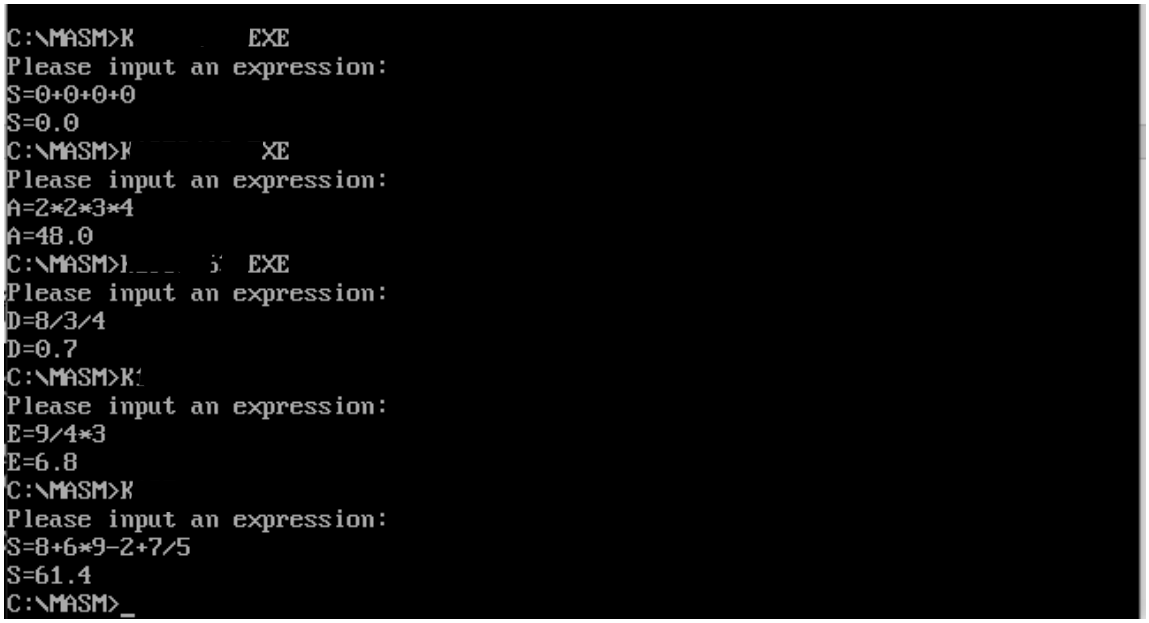
```
C:\MASM>LINK K

Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserved.

Run File [K       .EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\MASM>
```

3．运行结果及分析



运行结果用 6 组测试数据来进行测试。

第一个使用连续加法，并且每一个加数都是 0，结果正确。

第二个式子使用连乘的式子测试，结果正确。

第三个式子使用连除的式子测试，结果正确。

第四个式子使用乘除混合的式子测试，结果正确。

第五个式子使用加减乘除的混合运算进行测试，结果正确。

第六个式子使用运算过程中含有负数的式子进行测试，结果正确。

四、使用说明

1.软硬件环境：DOS 系统或 DOSBOX。8086CPU。

2. 对 K1953463.ASM 进行汇编，并对 K1953463.OBJ 进行连接。运行[redacted].EXE，屏幕显示提示“Please input an expression: ”，在该提示下方输入一个算式，为一位数加减乘除的四则运算，输入按照“字母+等号+算式”的格式，并以回车结束。

3.支持结果为 100 内正数的一位数四则运算,结果以保留一位小数的四舍五入形式表示。

五、课程总结

1、课程总结

由于疫情，我这次长达两星期的汇编语言程序设计课程是在家里通过网课进行的。对于这次暑期小学期的课程总结我主要从以下几个方面来谈：课堂体验、习题情况和上机成果。

就课堂体验而言，我觉得李文根老师上课非常认真仔细。对于汇编我本来没有多少了解，只是在考试结束和小学期之间的一星期里预习了一些汇编的内容，所以基本是一个汇编入门的小白。而李文根老师上课的时候也非常充分地考虑了初学者的情况，用通俗易懂的语言让我对汇编语言有了非常深刻的了解。而且课堂上老师放映的 ppt 也非常详细，基本概念讲得也非常清楚，还有丰富的示例程序可供学习，对我的作业都颇有帮助。

对于习题情况来说，习题中还是有一些颇有难度的题目。题目类型还是挺丰富的，从选择题、填空题到程序分析题都涉及到了，还有自己设计程序的题目。在做老师布置的习题的

过程中我也遇到了一些困难，但是绝大多数问题都能通过搜查资料等方式解决。所以个人觉得习题的难度还是适中的。

上机题目对我而言有一些难度。有的上机题附上了示例程序，可以为我的程序设计提供一些启发。而有的上机题也没有示例，对我来说如果思路不够清晰，会有一些困难。但是好在平时上课的时候老师讲的示例程序比较多，自己也尝试了很多，最终上机题都完成了，我的程序也很好地完成了该有的功能。实验报告的撰写方面，我觉得我的截图还是有点多，文字方面还是有点缺乏，但是基本能对我的程序进行简短的说明。

回顾这两个星期的表现，我觉得相比我学 C 语言、C++的时候，我已经有了很大的进步。无论是程序编写的速度还是思维等方面都有相当大的提升。在汇编语言程序当中，我跳转、循环等方面的知识掌握得还不错，但是也能明显感觉到在子程序方面有的时候对参数传递等还不是很熟悉，还需要多加练习。

2、大作业总结

在这个暑假，我做大作业的时间安排及自学内容如下：

7 月 25 日-7 月 29 日：上网搜索资料并查阅相关书籍，对汇编的浮点操作、汇编的四则运算以及进制转换等内容进行了学习

7 月 30 日-8 月 2 日：对程序进行编写

8 月 3 日-8 月 4 日：撰写实验报告

参考资料：

《汇编语言第 3 版》（王爽著）

课上课件

https://blog.csdn.net/q_l_s/article/details/54909328

https://blog.csdn.net/simpleact/article/details/24124859

完成情况：

除了在网上查找以及书本上的资料以外，全程独立完成。

这次的暑假大作业我花费了大量的精力。这个题目对于 C 语言、C++等高级语言来说是非常简单的，但是对于汇编来说难度就相对比较大。因为第一，高级语言可以直接通过输入数字形式而非字符的形式，这就意味着无需 ASCII 码转换，可以直接进行计算；第二，高级语言可以通过创建各种变量对变量直接进行操作，比较方便快捷，而不是像汇编，需要寄存器作为媒介进行计算；第三，高级语言的判断、循环等语言更加简洁，并且排版更清晰，汇编相比之下思路更加复杂。所以在写程序之前我先查找了相关的资料，有了自己的思路以后才着手编程。

在大作业完成的过程中，我的汇编程序设计能力有了很大进步，具体体现在如下几个方面：

第一，在刚开始接触汇编的时候，对于汇编的循环和条件等内容我非常不熟悉，使用起来非常生疏，常常会因为编写过程中头脑混乱而编写错，从而导致结果错误或者进入死循环等。究其原因，可能是我当时并没有熟练掌握汇编的思维方式，没有形成汇编特有的思考模式。但是现在我已经较好掌握了循环和条件跳转等内容，写程序的时候思维非常清晰，不会产生混乱。

第二，刚刚开始进行编程学习的时候，我对输入输出等方面还是有些糊涂。经常在编写程序的时候，要么就是输入内容以后程序崩溃，要么就是无法输出自己想要的结果。归根结底是没有明白不同功能号对应的输入输出方式的区别以及具体的步骤。在大作业完成过程中，因为要进行多次输入和输出操作，这方面我已经非常熟练了。

在进行多次的调试和修改以后，我的程序已经基本可以完成一位数的加减乘除计算了，报告也花了非常大的精力进行对我的程序的阐释，并经过多次修改。但是依然有一些小遗憾，具体体现在三个方面：

第一，我在前期参考的资料当中有看到浮点指令集，即.387 指令集，并尝试着进行了学习。可惜由于自己学习能力实在有限，没有学会，所以使用了先乘 100 后加小数点的迂回方式。

第二，正负号的问题。我的程序在运算过程中是可以出现负号的，但是对于最后输出的结果，如果是负数将显示错误。我的设想是在加减部分利用标志位判断正负，并用一个变量来存储正负状态，并在十六进制转十进制部分根据是否是负数来判断是否需要求补后加负号的操作。但是在编程过程中，我没能够得以实现。

第三，我对子程序依然没有熟练掌握，以至于在完成大作业的过程中遇到了很大障碍，最终为了实现功能不得不抛弃子程序，而使用主程序任务的顺序结构。

总结起来，这个暑假小学期的过程中我学到了很多关于汇编语言程序设计的知识和技巧。在今后的学习中我将继续进步，如果有机会希望可以在校图书馆看更多有关汇编语言程序设计的书籍，熟练掌握.387 指令集和子程序设计部分。

最后感谢老师这几个星期来的教导和耐心解答，老师辛苦了！

程序清单：

1）表达式输入部分：

```
    MOV AH,9
    MOV DX,OFFSET MSG
    INT 21H      ;输出提示信息
    LEA DX,BUFFER
    MOV AH,0AH
    INT 21H      ;输入表达式
    CRLF
```

2）字符串的预处理部分：

```
    MOV CL,BUFFER+1
    MOV CH,0
    MOV COUNT,CL
    LEA SI,BUFFER+2
    LEA BX,COPY
TRANS: MOV AH,0
    MOV AL,[SI]          ;将元素放进 AL 中
    CMP AL,'0'       ;判断是否为数字
    JB  NEXTO        ;若 ASCII 小于'0',说明不是数字,不做额外处理
NUM:    CMP AL,'9'
    JA  NEXTO        ;同理,若 ASCII 大于'9',说明不是数字,不做额外处理
    SUB AL,30H       ;筛选下来的数字进行转换
NEXTO: MOV [BX],AX      ;将该字符复制到拷贝数组
    INC SI
    ADD BX,2
    LOOP    TRANS
    ;此部分进行字符串的预处理,将所有表示数字的字符都转化为数字本身
```

3）字符串的乘 100 处理：

```
    LEA SI,COPY+4   ;直接从数字开始
```

```
    MOV CL,COUNT    ;数字和运算符个数
    SUB CL,2
    MOV DX,100      ;准备乘 100
COMP0: MOV AX,[SI]     ;元素放入 AX
    MOV BX,[SI-2]
    CMP AX,0
    JB  JUMP
    CMP AX,9
    JA  JUMP    ;判断 AX 中是否为数字
    CMP BX,'*' ;若作为乘数无需多次乘 100
    JE  JUMP
    CMP BX,'/' ;若作为除数无需乘 100
    JE  JUMP
    MUL DL
    MOV [SI],AX
JUMP:   ADD SI,2
LOOP    COMP0
```

4）乘除法优先计算部分

```
    LEA SI,COPY
    MOV CL,COUNT
COMP1: MOV AX,[SI]     ;将元素放进 AX 中
    CMP AX,'*' ;判断符号是否为乘号
    JNZ DIVI    ;不是乘号，跳转，判断是否为除号
    MOV BX,[SI-2]  ;将乘号前的数字放进 BX 中
    MOV AX,[SI+2]  ;将乘号后的数字放进 AX 中
    MOV DX,0
    MUL BX  ;两数相乘放入 DX:AX
    MOV [SI+2],AX  ;AX 中数值放入下一元素
    MOV WORD PTR[SI-2],0  ;第一个数的位置置零
    MOV WORD PTR[SI],0    ;乘号位置置零
DIVI:   CMP AX,'/' ;与除号相比较
    JNZ NEXT    ;不是除号，直接进入下一个循环
    PUSH    BX
    PUSH    CX
    PUSH    DX  ;保护数据
    MOV AX,[SI-2]  ;将被除数放进 AL 中
    MOV BX,[SI+2]  ;将除数放进 BL 中
    MOV DX,0
    DIV BX  ;除法运算
    MOV [SI+2],AX  ;并将结果放入除数位置
    MOV WORD PTR[SI],0    ;将字符串的除号部分置零
    MOV WORD PTR[SI-2],0  ;将字符串的被除数部分置零
    POP DX
    POP CX
    POP BX  ;恢复数据
NEXT:   ADD SI,2    ;指针指向下一个元素
    LOOP    COMP1   ;循环
```

;此部分的作用就是进行乘除优先计算

5）加减法计算部分

```
    LEA SI,COPY+4
    MOV CL,COUNT
    SUB CL,2
    MOV BX,0     ;先将 BX 寄存器初始化为 0 状态,这样第一个数字可以自动默认为加
ADDMIN:     MOV AX,[SI]      ;第一个元素放入 AX
    CMP AX,'+'  ;判断是否为加号
    JE  PLUS
    CMP AX,'-'  ;判断是否为减号
    JE  MINUS
COMP2: CMP BX,0     ;判断 BX 寄存器中的值
    JE  NPLUS   ;作加法运算
    CMP BX,1
    JE  NMINUS  ;作减法运算
NPLUS: ADD SUM,AX  ;加
    JMP LOP
NMINUS:     SUB SUM,AX  ;减去整数部分
    JMP LOP
PLUS:   MOV BX,0     ;如果出现的是加号就将 BX 置为 0
    JMP LOP
MINUS: MOV BX,1     ;如果出现的是减号就将 BX 置为 1
    JMP LOP
LOP:    ADD SI,2     ;指针指向下一个元素
    LOOP    ADDMIN
```

6）十六进制转十进制

```
    MOV AX,SUM
    MOV DX,0
    MOV BX,1000
    DIV BX  ;取第一位数
    MOV NUMS,AL
    MOV AX,DX
    MOV DX,0
    MOV BX,100
    DIV BX  ;取第二位数
    MOV NUMS+1,AL
    MOV AX,DX
    MOV DX,0
    MOV BX,10
    DIV BX  ;取第三位数
    MOV NUMS+2,AL
    MOV NUMS+3,DL;第四位数
```

7）输出部分

```
    MOV DL,BUFFER+2
    MOV AH,2
```

```
    INT 21H ;输出字母
    MOV DL,'='
    MOV AH,2
    INT 21H ;输出等号
    MOV DL,NUMS
    CMP DL,0
    JE  OUTPUT ;判断第一位是否为 0，若为 0，不显示
    ADD DL,30H
    MOV AH,2
    INT 21H ;第一位不是 0，显示
OUTPUT:     MOV DL,NUMS+1
    ADD DL,30H
    MOV AH,2
    INT 21H ;输出第二位数
    MOV DL,'.'
    MOV AH,2
    INT 21H ;输出小数点
    MOV DL,NUMS+2
    MOV AL,NUMS+3
    CMP AL,5    ;判断第四位数的值是否大于 5，进行四舍五入操作
    JL  PLUS1
    ADD DL,1
PLUS1: ADD DL,30H
    MOV AH,2
    INT 21H ;输出第三位数
```