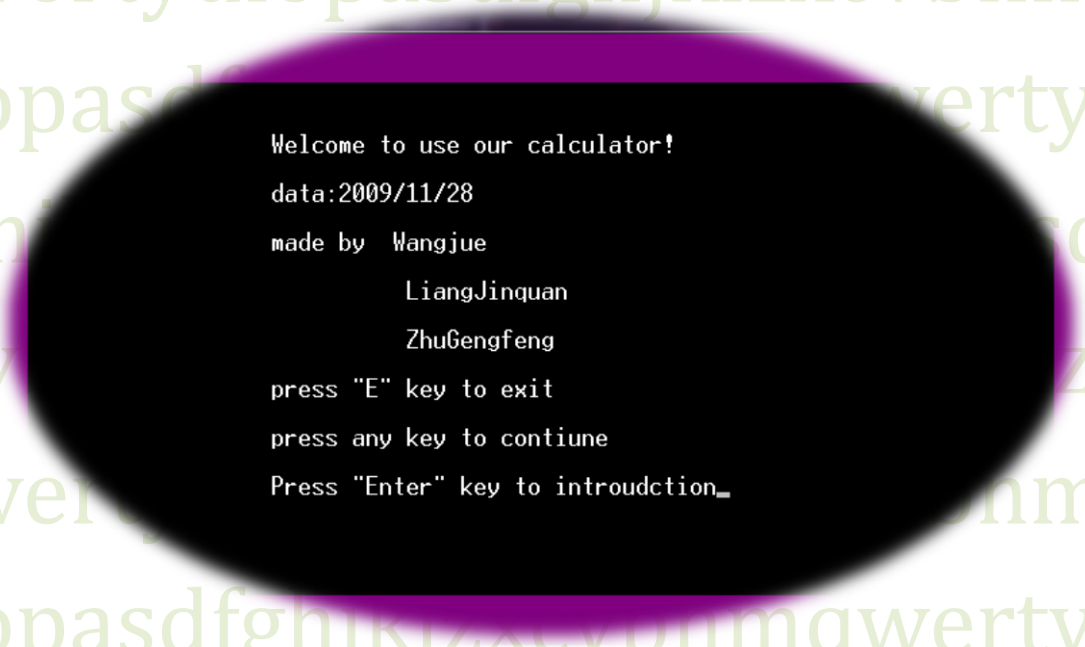
8086 汇编语言程序设计

简易计算器

2009/11/29

王珏.梁锦全.朱耿锋



8086 汇编语言程序设计

------简易计算器

目录

一、 实验题目与目的

1. 题目

使用 8086 汇编语言设计一个支持小数位及带括号的混合运算的计算器程序。

主要功能：

包括基本的四则运算、括号运算、负数运算、并且扩展 8086 的 16 位上限功能。

2. 学习目的

(1) 综合运用所学的微机汇编语言程序设计的知识。

(2) 进一步巩固在 PC 上建立、汇编、链接、调试和运行汇编语言程序的过程。

一、 分析与设计

1. 系统分析

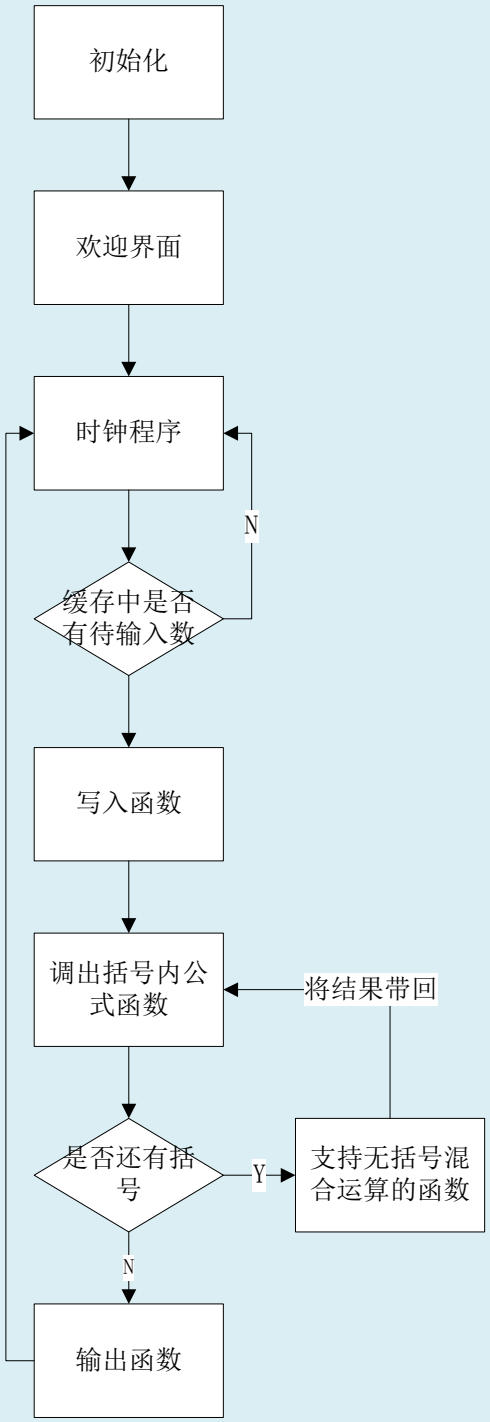
首先在 8086 的程序汇编语言当中介入一个数字时，是以 ascll 从键盘介入,那么我们就必需将它转换成我们方便运算的形式存入计算机内存。

其次在 8086 的程序汇编语言当中没有对各种运算符的优先级及括号的判断，所以必须人为的设计一个判断方式。

同时 8086 的寄存器仅有 16 位，2^{16}=65536。那么做一个正负数，数的上限就缩小两倍变为 2^{15}=32768，再代入两位定点小数的话，就又至少缩小 100 倍变为 327 的整数上限。这样在一般情况下，做起乘法运算时，都相当容易溢出。

8086 汇编语言程序设计

2. 系统设计



通过函数及宏的方法，将整个程序分成几个大的模块。这样方便分工合作，也方便错误查找。

实验的目标是完成一个支持负数、两位定点小数、带括号的混合运算、整数位能在-2^{15}~2^{15}之间，并且做到一定美化界面的计算器。

3. 功能分析

i. 写入函数：输入数是采用字符串输入，还是字符输入？输入后的数用什么方式存储到内存中，ascll，BCD，二进制？小数位的进位如何处理？。。。

ii. 去括号函数：只要能完成去像（（3+4）+（1+2））这样的等式，就没问题了。

iii. 四则混合运算函数：乘除法优先计算完，加减法次之

iv. 四则运算：由于存数方式不同所以要新建新的加减乘除运算宏
v. 时钟：如何调用计算机时间，又如何让时间能不断更新
vi. ….

二、 各子函数、宏名称及简介

1. Give macro x,y ;将 DD 型的两个数 X,Y 其中 Y 的值赋给 X
2. Judge macro x,y ; 判断 DD 型 x,y 的正负利用 fhx、fhy 两个常数记录下并将他们都化为正数方便运算
3. change macro x,y ; 根据介入 y 值的 0，1 将 x 结果转为正或负
4. carry macro x ; 对小数部分除以 100，将商进位，余数补回小数，用于对运算后的结果进位处理
5. newadd macro x,y;支持新存储方式的加法
6. newsub macro x,y;支持新存储方式的减法
7. newmul macro x,y;支持新存储方式的乘法
8. newdiv macro x,y;支持新存储方式的除法
9. curse macro cury,curx; 设置光标的位置
10. menu macro op1,op2,op3 ;将 op3 的内容显示到 op1,op2 的位置
11. scroll macro n,ulr,ulc,lrr,lrc,att;清屏或加色
12. write proc near 写入函数
13. loopcount proc near 去括号函数
14. si_ze proc near 支持不带括号的四则运算
15. output proc near 输出结果函数
16. time proc near 时间输出函数
17. BCDASC PROC NEAR 时间数值转换成 ASCII 码字符子程序

三、 各常数说明

1. fhx dw 0;记录 x 值的正负
2. fhy dw 0;记录 y 值的正负
3. fhdx dw 0;记录除法运算中的 x 值正负
4. fhdy dw 0;记录除法运算中的 y 值正负
5. divn4 dd 0;除法中帮助暂存 number4

```
number0 db 100;计入键盘输入字符串
        db 0
        db 100 dup(0);
number dw 200 dup(0);存入输入公式处
number2 dw 200 dup(0);读取去括号后的公式暂存处
number3 dd 0;存储用于计算的 x 数
number4 dd 0;存储用于计算的 y 数
crx    db 20;记录光标列 1
cry    db 10;记录光标行 1
crx2 db 2;记录光标列 2
cry2 db 2;记录光标行 2
memb dw 0;记录 bx 中的值是否储存过
memx db 0;记录是否应该进入小数存储部分
memcl db 0;记录已经存储到小数的第几位
rsi dw 0
begain db 'Welcome to use our calculator!','$'
begain1 db 'data:2009/11/28','$'
begain2 db 'made by ： Wangjue','$'
begain3 db '                LiangJinquan','$'
begain4 db '                ZhuGengfeng','$'
begain5 db 'press "E" key to exit','$'
begain6 db 'press any key to contiune','$'
begain7 db 'Press "Enter" key to introudction','$'
help      db 'Confine:32512.99~(-32768.00)','$'
help1    db 'Format: 1.32*99+(98.43/(-34))= ','$'
help2    db 'Notice 1:negative must like (-x)','$'
help3    db '            2:only can abet double decimal fraction as 567.33','$'
help4    db 'press any key go on!','$'
error1 db ' Error!',13,10,'$' ;报错信息
DBUFFER DB 8 DUP (':'),12 DUP (' ');时间输出的称底“:::::”
```

8086 汇编语言程序设计

四、 程序模块系统说明书

1. 输入函数

i. 说明

数据格式：定义一个 DD 双字数组，双字的第一个字存数的整数部分，第二个字存数的小数部分。但是为了最后的输出方便，以及乘法运算的小数部分运算后向整数部分进位，小数部分采用 100 进一制

存数范围：十进制 32512.99~(-32767.00),十六进制 7F00.63~8001.00(补码)

存数算法：整数部分 正数 ai=ai*10+a(i+1),负数对正数求补码

小数部分 正数 bi=b1*10+b2,负数对正数求补码

存数说明：采用单个字符一个一个输入的方式,字符为 0~9 时进入存数算法,当输入是运算符号时结束存数算法,并且开始存数及存运算符.但是这里要考虑一个问题不能每次有运算符号进入时都存数,比方 A+((X)),如果每次都存,A 就会被存 3 次.所以这里我们定义了一个常数 remb 记录每个数存入情况,方便存数时判断

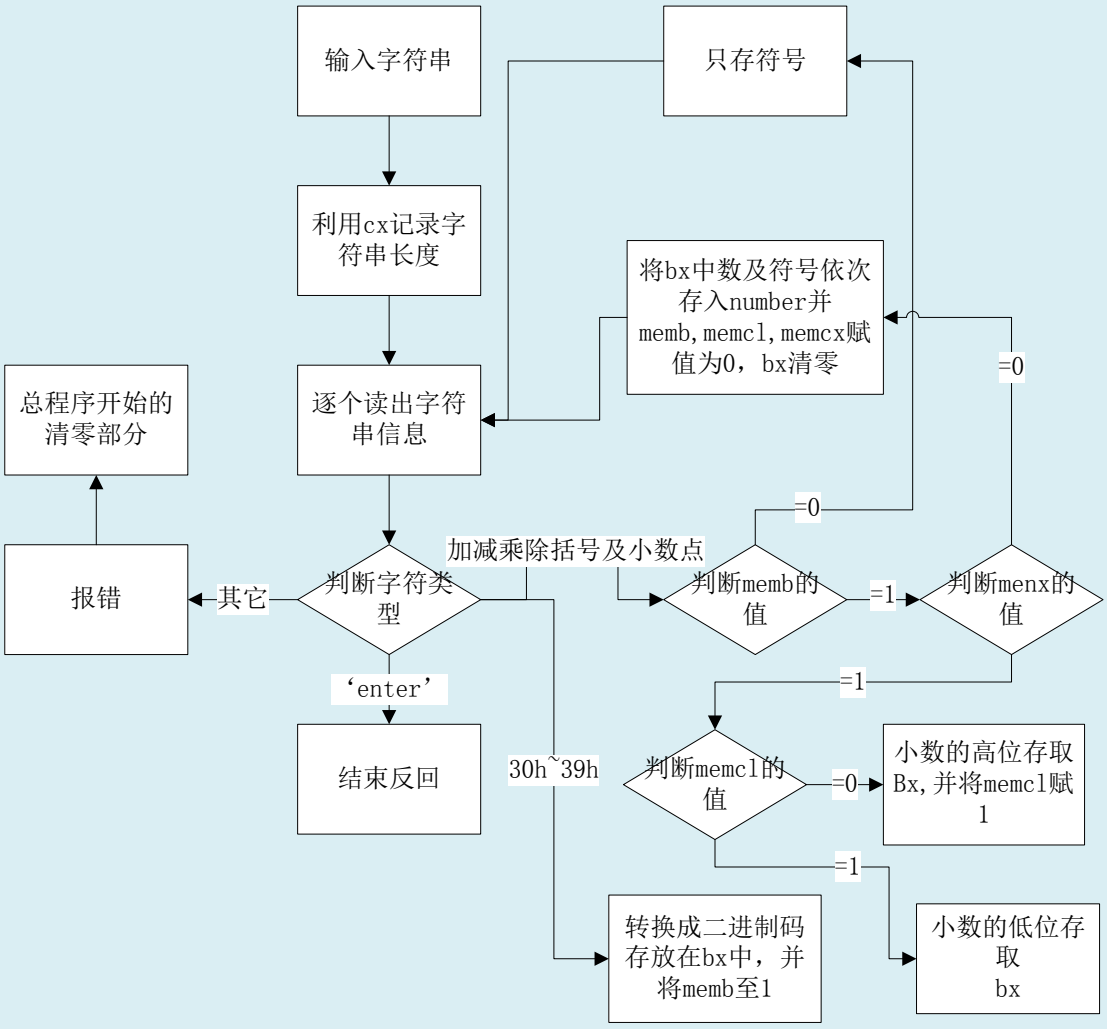
负数处理：负数的介入要求格式为(-X),因为在读取部分,会依次去括号,这样就变为了-X,所以只需要判断一串公式开头位'-'号,就知道介入的为负数,再直接对 X 求补。(注：这步处理放在支持混合运算的 si_ze 宏部分)

小数处理: 通过介入'.'来判断是否开始存入小数部分

特别说明： 1：由于运算符号的 ASCLL 较小,会与介入的数字相等,所以把运算符的 ASCLL 码加上 7F00H 在存入,这样也就要把 7F00H~7FFFH 预留给运算符号,同时这也会导致存数范围的缩小,不过个人觉得是值

输入格式: 按照平时正常等式的输入方式输入,以等号结尾.例如：11*(1-3)/(-3)=

ii. 流程图



iii. 源程序代码：

```
write proc near
;--写入函数主部分----
startw:
    lea dx,number0
    mov ah,0ah
    int 21h
    xor cx,cx
    mov cl,number0[1]
    mov si,cx
    add si,2
    mov rsi,si
    mov si,0
    mov di,2
```

```
        mov bx,0
startw2:
        mov al,number0[di]
        inc di
        inc crx
        mov ah,0
        cmp al,45h;'E'  用于程序退出
        jz    exit
        cmp al,2ah;'*'
        jz    memory;转入存数及存运算符
        cmp al,2fh;'/'
        jz    memory
        cmp al,2bh;'/'
        jz    memory
        cmp al,2dh;'-'
        jz    memory
        cmp al,29h;')'
        jz    memory
        cmp al,28h;'('
        jz    memory
        cmp al,2eh;'.'
        jz    memoryx;转入存入小数程序
        cmp al,3dh;'='
        jz    memory
        cmp al,0dh;'CR'
        jz     endwrite
        sub al,30h;为存数算法做准备,让'1'真正变成二进制码的 1
        cmp al,0
        jl    error;报错程序
        cmp al,9
        ja     error
        cmp memx,2
        je     error
        cmp memx,1
```

```
    je    arithmeticx;小数存数算法程序
    jmp arithmetic;整数存数算法程序
;---memory 存入部分---
memory:
    cmp memb,0;remb 位 0 时表示数已经数存储过；注：开始为 remb 赋值时
                ;也一定要为 0,因为这是为了防止'(1+3)*4'这样直接以运
               ;算符开始的等式
    je    memory1
    mov number[si],bx;存入数
    cmp memx,1;
    je    memory2;存小数部分程序
    add si,4
   gomemory1:
    add ax,7f00h;将运算符转换到 7F00~7FFF 之间
    mov number[si],ax;存入运算符
    mov memb,0;说明数已经存储过
    mov memx,0
    mov memcl,0
    mov bx,0
    add si,4
   loop startw2
   jmp endwrite
 memory1:;只存运算符号
    add ax,7f00h
    mov number[si],ax
    add si,4
    loop startw2
    jmp endwrite
 memory2:
     add si,2
    jmp gomemory1
;-memoryx 小数存入部分-
memoryx:
    mov memx,1;当 remx 为 1 时表示开始准备小数存入
```

```
        mov number[si],bx
        mov bx,0
        add si,2;没有加四,应为这下是要转入存小数的下个字节
        loop startw2
        jmp endwrite
;-arithmetic 存数算法--
arithmetic:
         ;算法 bx=bx*10+ax
        push ax
        mov ax,bx
        push cx
        mov cx,10
        mul cx
        pop cx
        pop bx
        add ax,bx
        mov bx,ax
        mov memb,1;说明开始存数
        cmp bx,7f00h;保证数在合适范围内
        jae error
        loop startw2
        jmp endwrite
;-arithmeticx 存数算法--
arithmeticx:
         cmp memcl,1
        je    arithmetix1
        push cx
        mov cx,10
        mul cx
        add bx,ax
        pop cx
        mov memb,1
        inc memcl
        loop startw2
```

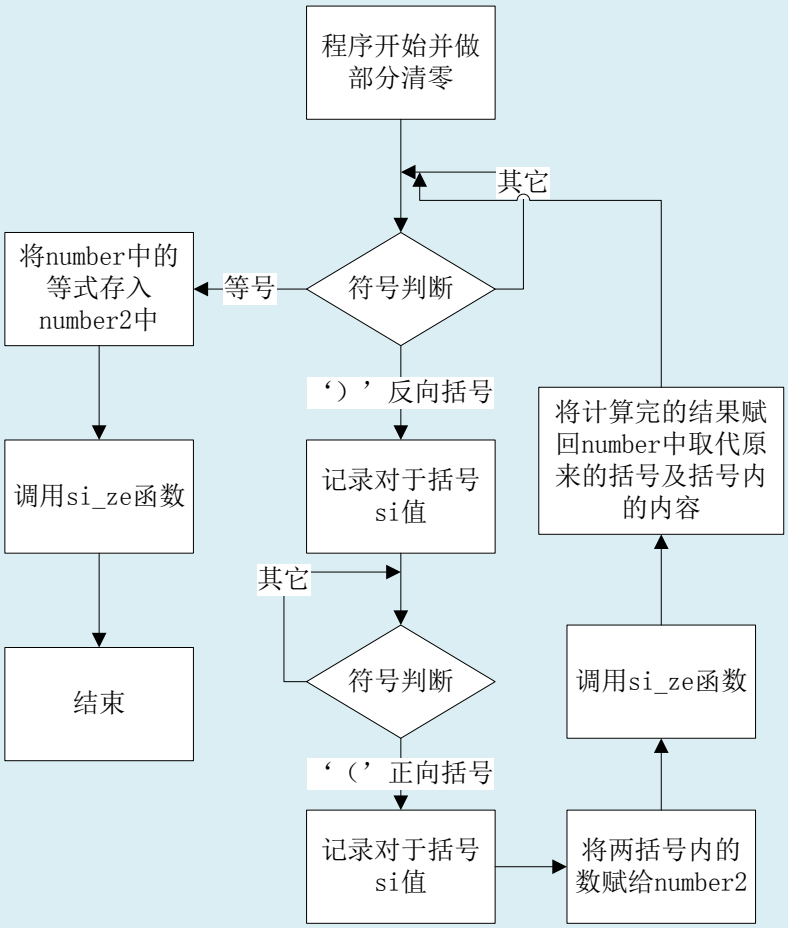
```
        jmp endwrite
    arithmetix1:
        add bx,ax
        inc memcl
        loop startw2
        jmp endwrite
;---error 报错部分----
error:
        curse cry,crx
        lea dx,error1
        mov ah,9
        int 21h
        jmp start1;重新运行程序
endwrite:
        mov di,0
        ret
write endp
;***写入函数结束***
```

2. 去括号函数

i. 说明

比方'11+(3+99)='这个等式,先从 number 的首位开始每隔两个字取出数与')'的现 ASCLL 码 7f29h 比较从而找到第一个')'的位置,再到回去找到最近的'('的位置,从而将中间没括号的等式读出到 number3 中,接着利用 si_ze 函数算出这个等式的值,并且赋值回 number 中,以替代原来的括号及括号能的等式

ii. 流程图



iii. 源程序代码：

```
loopcount proc near
    startlp:
        mov bp,0
        mov di,0
        mov bx,0
        mov si,0
    startl:
        mov ax,number[si]
        cmp ax,7f29h
        je  rsee
        add si,4
        cmp ax,7f3dh
        je  lastl
        jmp startl
    rsee:
        sub si,4
```

```
        mov ax,number[si]
        cmp ax,7f28h
        je rwrite
        jmp rsee
rwrite:
      push si
        ; mov lct2,si
        mov di,0
      add si,2
    rwrite1:
        add si,2
        mov ax,number[si]
        mov number2[di],ax
        cmp ax,7f29h
        je    rcount
        add di,2
        jmp rwrite1
rcount:
      push si
        ; mov lct1,si
        call si_ze
      pop ax
      pop di
      push ax
        ; mov di,lct2
        mov ax,number3
        mov number[di],ax
      mov number3,0
        mov ax,number3[2]
        add di,2
        mov number[di],ax
      mov number3[2],0
      pop si
        ; mov si,lct1
```

```
            add si,2
          rcount1:
                add si,2
                add di,2
                mov ax,number[si]
                mov number[di],ax
                cmp ax,7f3dh
                je    startlp
                jmp rcount1

 lastl:
            mov si,0
       lastll:
            mov ax,number[si]
            mov number2[si],ax
            add si,2
            cmp ax,7f3dh
            je    endl
            jmp lastll

 endl:
        call si_ze
            ret
loopcount endp
;**去括号函数结束** loopcount proc near
```

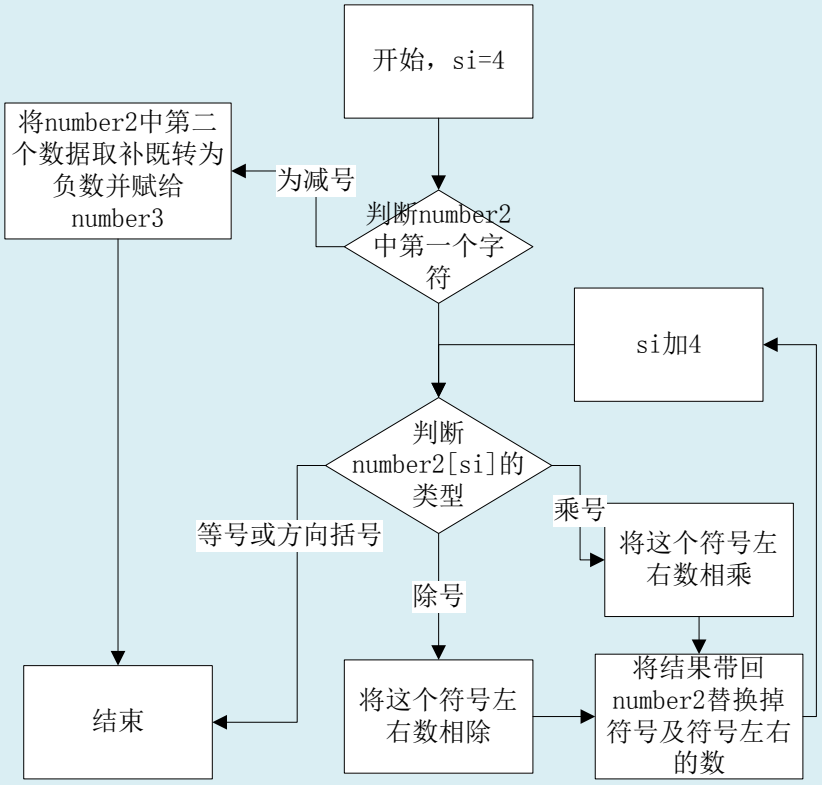
3. 混合四则运算函数

i. 说明

通过先不处理加减，先把乘除运算进行，将算出的结果带回原来的地方取代。最终得到一个只有加减的公式。来方便运算

特别说明： 本运算还承贷着将（-1）转为 ffffh 存入 number 中,既负数输入

ii. 流程图



iii. 源程序代码：

```
si_ze proc near
        mov si,4
        mov ax,number2
        cmp ax,7f2dh;'-'
        jne sstart
        give number3,number2[4]
        mov fhx,1
        change number3,fhx
        mov fhx,0
        jmp end4ze
sstart:
        mov ax,number2[si]
         cmp ax,7f2ah;'*'
         je mull;计算乘法
         cmp ax,7f2fh;'/'
         je divv
         cmp ax,7f3dh;'='
         je sznext;去完乘除后跳转
```

```
        cmp ax,7f29h;')'
        je sznext
       add si,4
       jmp sstart
mull:
       sub si,4
       give number3,number2[si]
       add si,8
       give number4,number2[si]
       newmul number3,number4
       sub si,8
       give number2[si],number3
       call sloop
       jmp sstart
divv:
       sub si,4
       give number3,number2[si]
       add si,8
       give number4,number2[si]
       newdiv number3,number4
       sub si,8
       give number2[si],number3
       call sloop
       jmp sstart
sznext:
       give number3,number2
       mov si,0
sznext1:
       add si,4
       mov ax,number2[si]
       cmp ax,7f2bh
       je    addd
       cmp ax,7f2dh
       je    subb
```

```
        cmp ax,7f3dh
        je end4ze
        cmp ax,7f29h
        je end4ze
        jmp sznext1
addd:
        add si,4
        give number4,number2[si]
        newadd number3,number4
        jmp sznext1
subb:
        add si,4
        give number4,number2[si]
        newsub number3,number4
        jmp sznext1
end4ze:
          RET
si_ze endp

sloop proc near;用于循环赋值
        mov di,si
        add si,8
    sloop1:
        add di,4
        add si,4
        mov ax,number2[si]
        push ax
        give number2[di],number2[si]
        pop ax
        cmp ax,7f3dh;等号
        je    endsloop
        cmp ax,7f29h;括号
        je    endsloop
        jmp sloop1
```

```
endsloop:
    ret
sloop endp
;**混合四则运算结束**
```

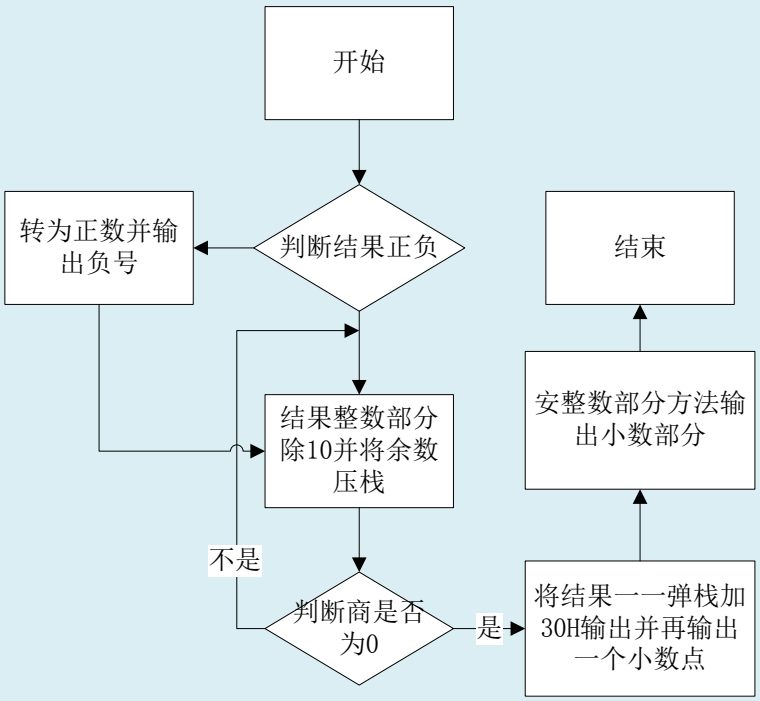
4. 结果输出函数

i. 说明

通过除十取余的方法，将数从低位到高位逐一取出再加 30h 转为 ascll 输出。

注：先判断数的正负来考虑是否输出负号，接着输出整数部分再输出一个“.”小数点最后输出小数部分

ii. 流程图



iii. 源程序代码：

```
output proc near
    mov si,rsi
    judge number3,number4
    cmp fhx,1
    jne output1
    mov number0[si],2dh
    inc si
```

```
        mov dx,2dh
        mov ah,02h
        int 21h
output1:
        mov ax,number3
        mov bx,10
        mov cx,0
        call intoutput
        mov number0[si],2eh
        inc si
        mov dx,2eh;输出小数点
        mov ah,02h
        int 21h
        mov ax,number3[2]
        mov bx,10
        mov cx,0
        call intoutput
        mov number0[si],'$'
        ret
output endp

intoutput proc near
output2:
        mov dx,0
        div bx
        push dx
        inc cx
        cmp ax,0
        je    output3
        jmp output2
output3:
        pop ax
        add ax,30h
        mov number0[si],al
```

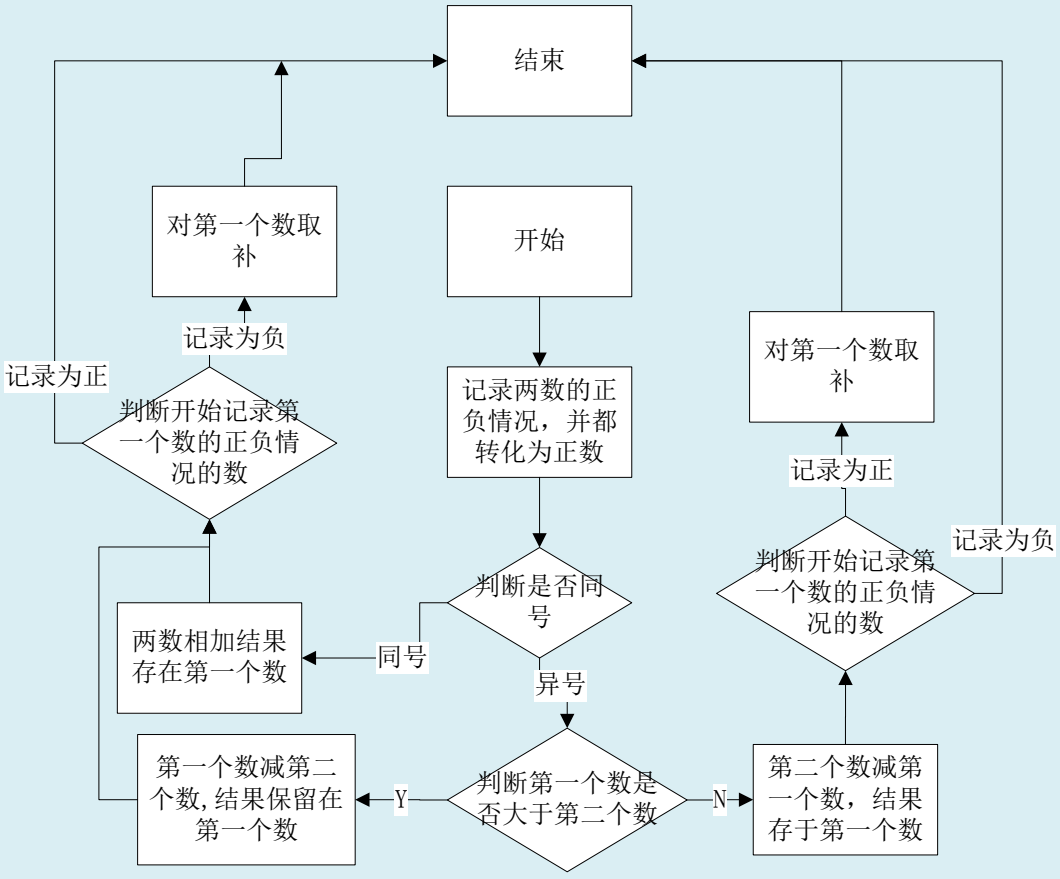
```
    inc si
    mov dx,ax
    mov ah,02
    int 21h
    loop output3
    ret
intoutput endp
;**输出函数**
```

5. 新的加法宏

i. 算法

先将用 fhx,fhy 记入正负情况并将两个数都转为正数，根据同号相加，异号大数减小数。得到结果后，通过 fhx 与 fhy 及运算过程将运算结果转为正或负。

ii. 流程图



iii. 源程序代码：

```
newadd macro x,y
    local
```

```
subsub,endnewadd,returnadd1,xbig,endadd1,endadd2,endadd3,endadda,endaddc
    judge number3,number4;;不能直接代用 x,y.
                ;;x 其实既 number3,y 既 number4。原因，下面一条注释
    mov ax,fhx
    cmp fhy,ax
    jne  subsub
;;------两个数符号相等则，直接两部分相加
    mov ax,y
    add x,ax
    mov ax,y[2]
    add x[2],ax
    carry number3
    change number3,fhx
    jmp endnewadd
;;---如果符号相反则，大数减去小数
subsub:
    mov ax,y
    cmp x,ax
    ja    xbig
    jne    endadda
    mov ax,y[2]
    cmp x[2],ax
    ja    xbig
endadda:
    mov ax,x
    sub y,ax
    add y[2],100
    mov ax,x[2]
    sub y[2],ax
    cmp y[2],100
    jnb    endadd1
    sub y,1
    jmp endaddc
endadd1:
```

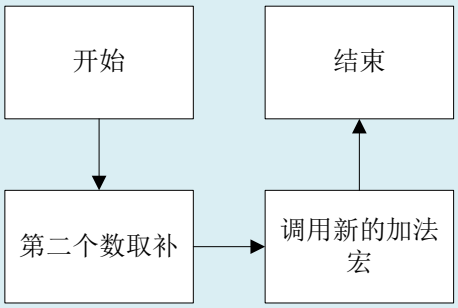
```
        sub y[2],100
endaddc:
        give number3,number4
        change number3,fhy
        jmp endnewadd
xbig:
        mov ax,y
        sub x,ax
        add x[2],100
        mov ax,y[2]
        sub x[2],ax
        cmp x[2],100
        jnb    endadd2
        sub x,1
        jmp endadd3
endadd2:
        sub x[2],100
endadd3:
        change number3,fhx
        jmp endnewadd
endnewadd:
        mov fhx,0
        mov fhy,0
endm
;**新的加法宏结束**
```

6. 新的减法宏

i. 算法

将第二数取补，再调用新的加法程序即可

ii. 流程图



iii. 源程序代码：

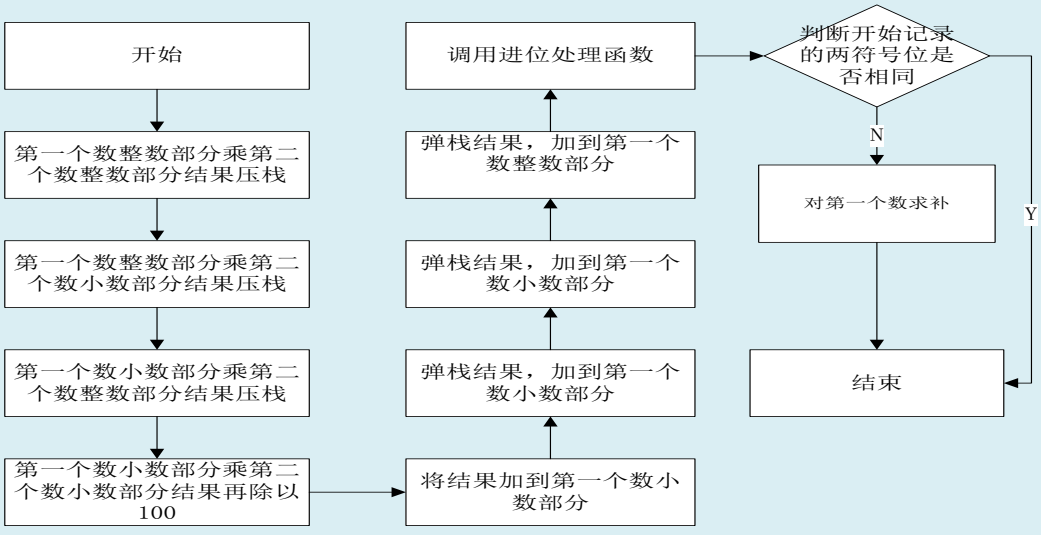
```
newsub macro x,y
      mov fhx,1
      change number4,fhx
      mov fhx,0
      newadd number3,number4
endm
;**新的减法宏*
```

7. 新的乘法宏

i. 算法

(a1+b1)*(a2+b2)=a1*a2+a1*b2+a2*1+b1*b2　a 代表正数部分，b 代表小数部分

ii. 流程图



iii. 源程序代码：

```
newmul macro x,y
      judge number3,number4
```

```
push bx
push dx
mov bx,y
mov ax,x
mul bx
push ax;;压栈用于后面的加法
mov ax,x[2]
mul bx
push ax;;正数部分乘小数部分的结果可以直接加到小数部分
mov bx,y[2]
mov ax,x
mul bx
push ax
mov ax,x[2]
mul bx
mov dx,0
mov bx,100;;小数部分乘小数部分的结果必需再缩小 100 倍，才
        ;;能再加回小数位
div bx
mov x[2],ax
pop ax
add x[2],ax
pop ax
add x[2],ax
pop ax
mov x,ax
carry number3
mov ax,fhy
xor fhx,ax
change number3,fhx
mov fhx,0
mov fhy,0
pop dx
pop bx
```

endm

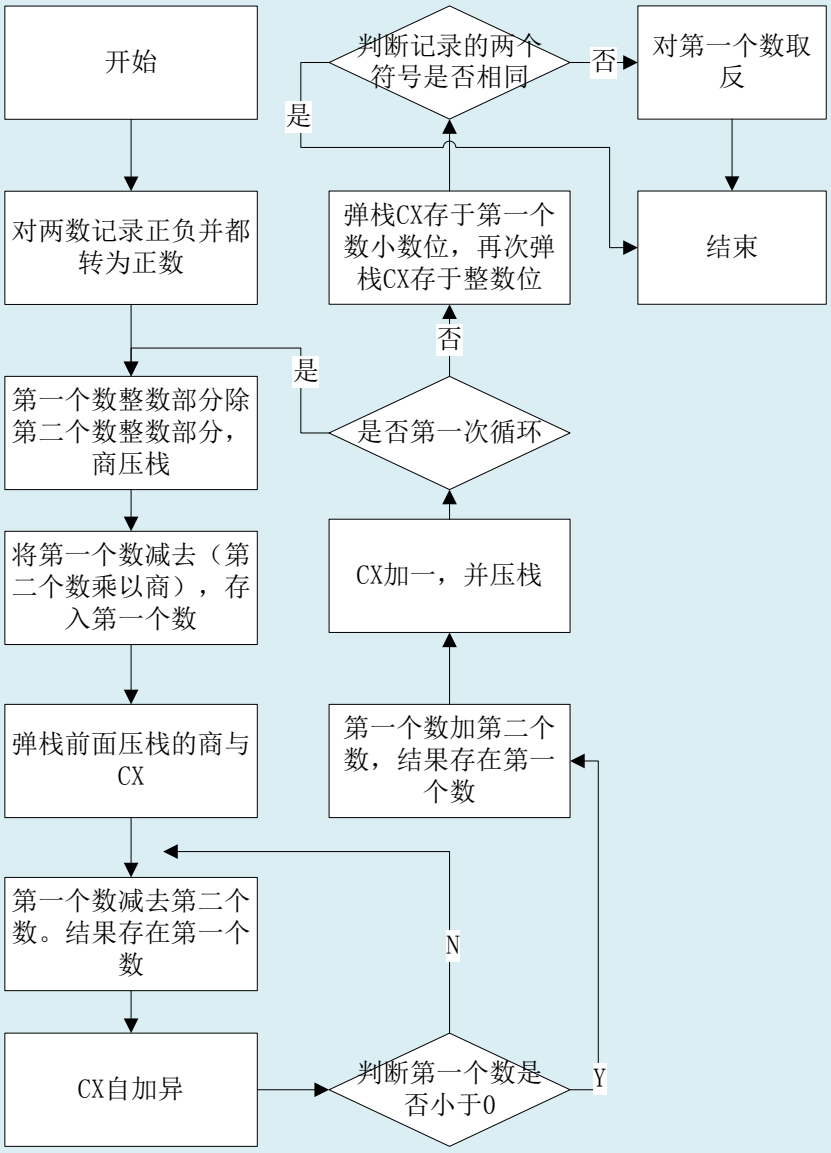
;**新的乘法宏**

8. 新的除法宏

i. 算法

利用减法完成除法运送，但是为了减少逐减次数，所以先用 a1/(a2+1)得要一个商，这个商一定不会大于逐减，所以就可以从（a1+b1）-商*（a2+b2)开始逐减，直到减出负数后，回加一个（a2+b2)得（a3+b3)。这时候的次数，就是结果的整数部分。最后将(a3+b3)*100 按前面的方法,就可以得到，结果的两位小数部分。

ii. 流程图



iii. 源程序代码：

```
newdiv macro x,y
local endnewdiv
        push bx
        push dx
        push cx
        judge number3,number4
        give fhdx,fhx
        mov fhx,0
        mov fhy,0
        give divn4,number4
        intdiv number3,number4;;求结果的整数部分
        mov bx,100;;将减完的剩余数扩大 100 倍
        mov ax,number3
        mul bx
        mov number3,ax
        mov ax,number3[2];;小数位扩大 100 倍，就等于直接进入整数位
        add number3,ax
        mov number3[2],0
        give number4,divn4
        intdiv number3,number4;;求结果的小数部分
        pop number3[2]
        pop number3
endnewdiv:
        mov ax,fhdy
        xor fhdx,ax
        change number3,fhdx
        mov fhdx,0
        mov fhdy,0
        pop cx
        pop dx
        pop bx
endm
;**新的除法宏**
```

```
;==除法的逐减宏==
intdiv macro x,y
local intdiv1,intdiv2,endintdiv;;内嵌宏，用于逐减
    mov ax,x
    mov bx,y
    add bx,1
    mov dx,0
    div bx
    push ax
    mov bx,y
    mul bx
    mov y,ax
    pop ax
    push ax
    mov bx,y[2]
    mul bx
    mov y[2],ax
    carry number4
    pop cx
    newsub number3,number4
    give number4,divn4;防止减法运算后改变的了 number4 的值
intdiv1:
    newsub number3,number4
    give number4,divn4
    inc cx
    cmp number3,0
    jl endintdiv
    je intdiv2
    jmp intdiv1
intdiv2:
    cmp number3[2],0
    jl endintdiv
    jmp intdiv1
```

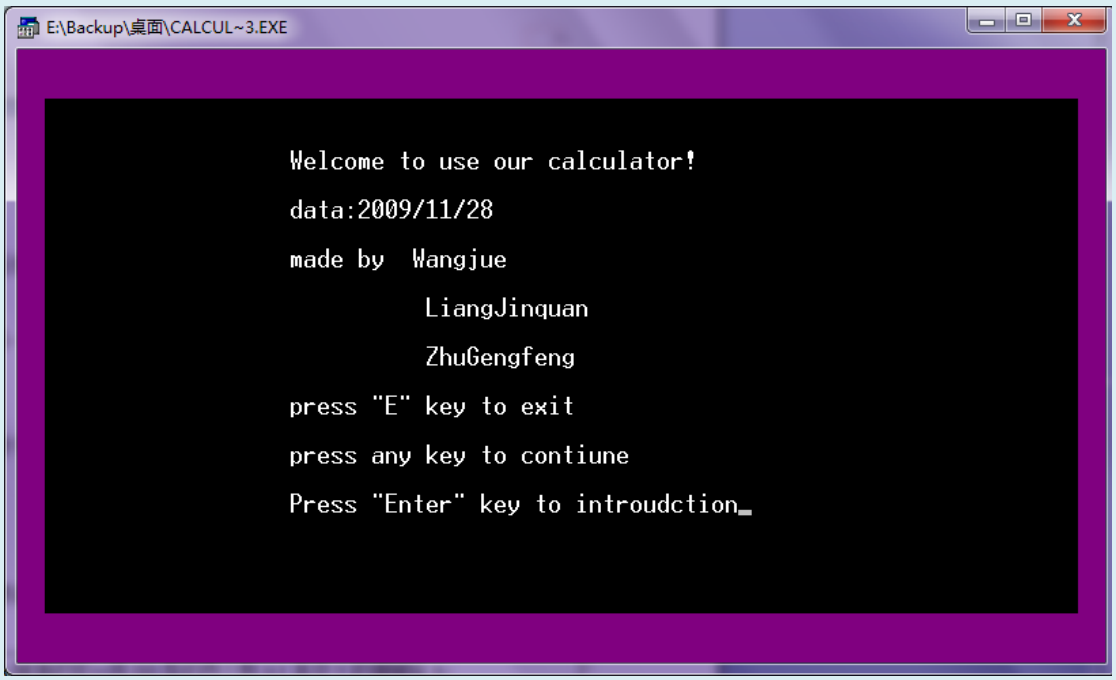
```
endintdiv:
    newadd number3,number4
    dec cx
    push cx
endm
;**除法的逐减宏**
```

9. 其它宏和函数

其它宏和函数附件源代码中都有注释，由于不是本程序核心内容，这里就不在详细说明。

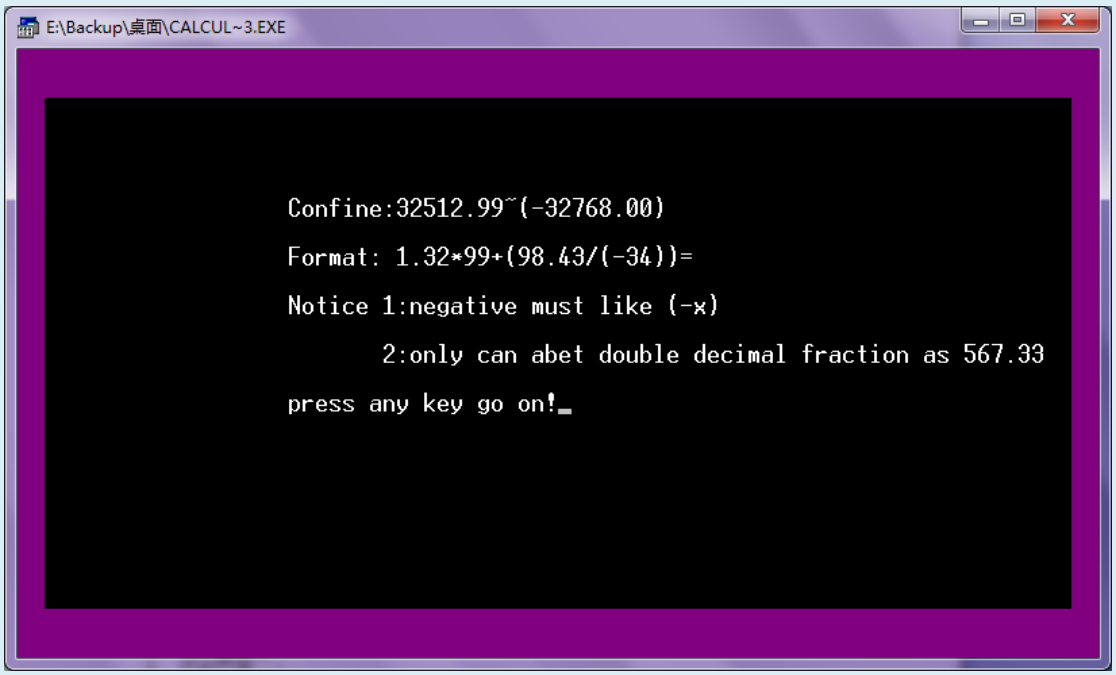
五、 运行结果

1. 欢迎界面一



2. 欢迎界面二

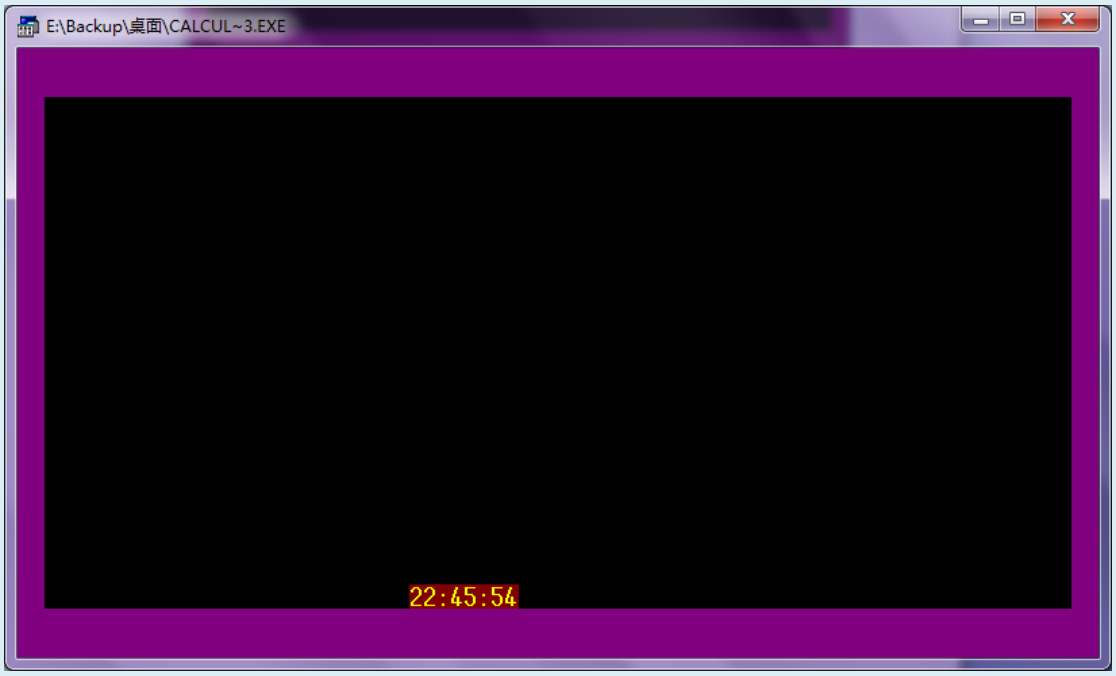
（通过欢迎界面一按“Enter”进入）



3. 运算界面

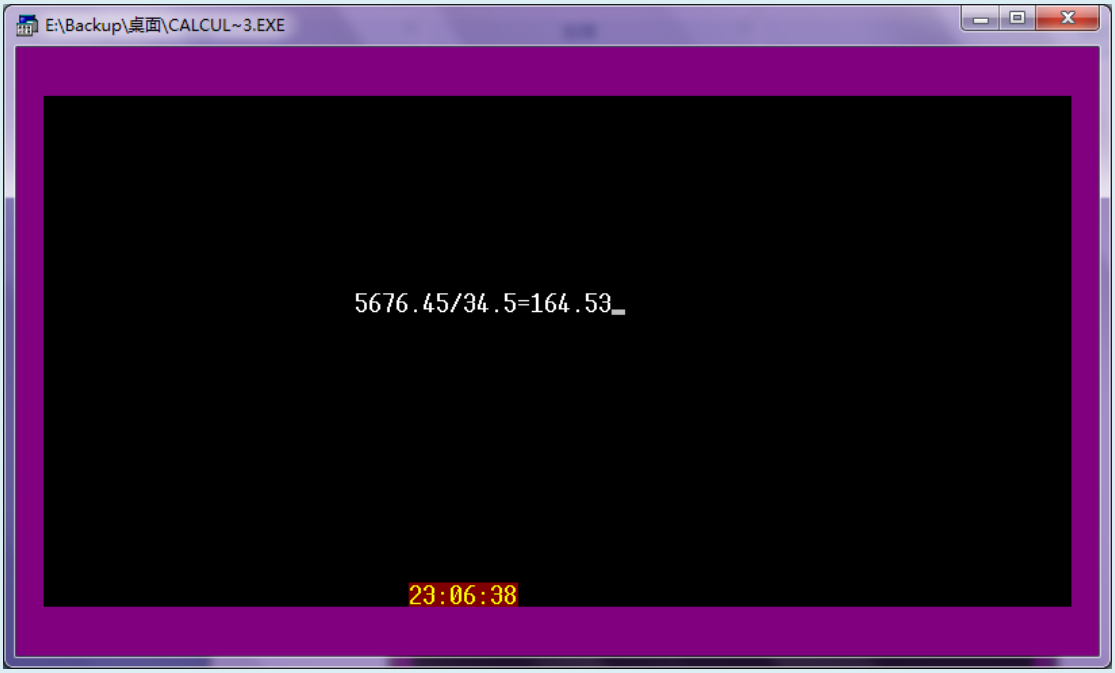
（通过欢迎界面二按任意键进入）

注意：界面的正下方拥有一个及时时钟，可以不断显示现在时刻

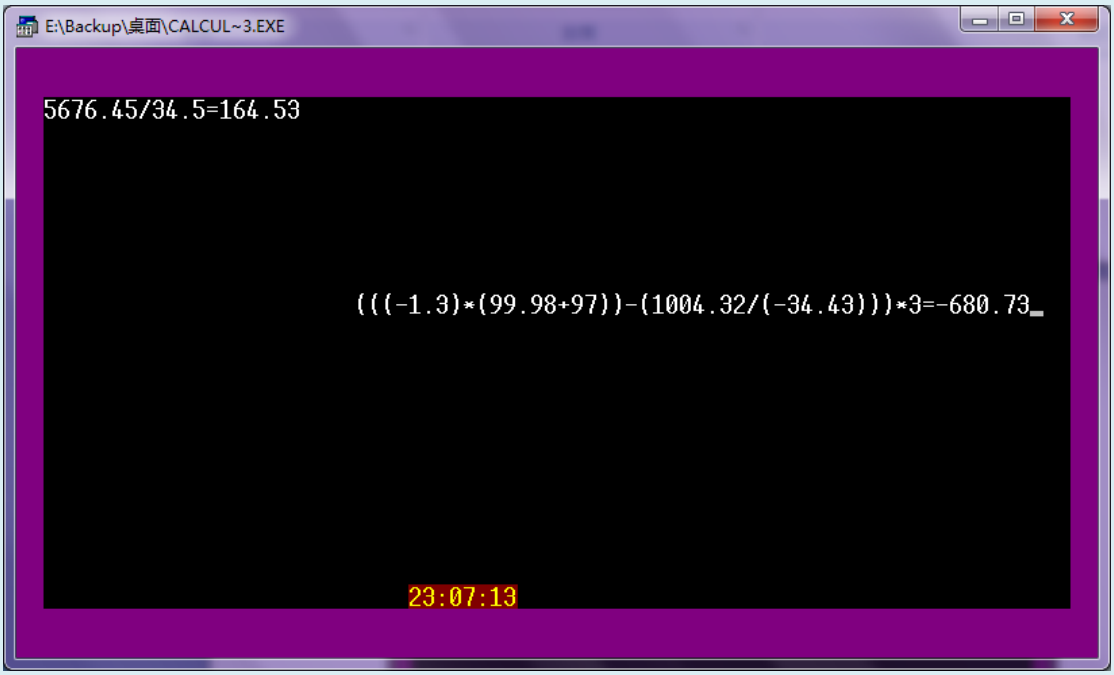


4. 测试程序

i. 简单运算

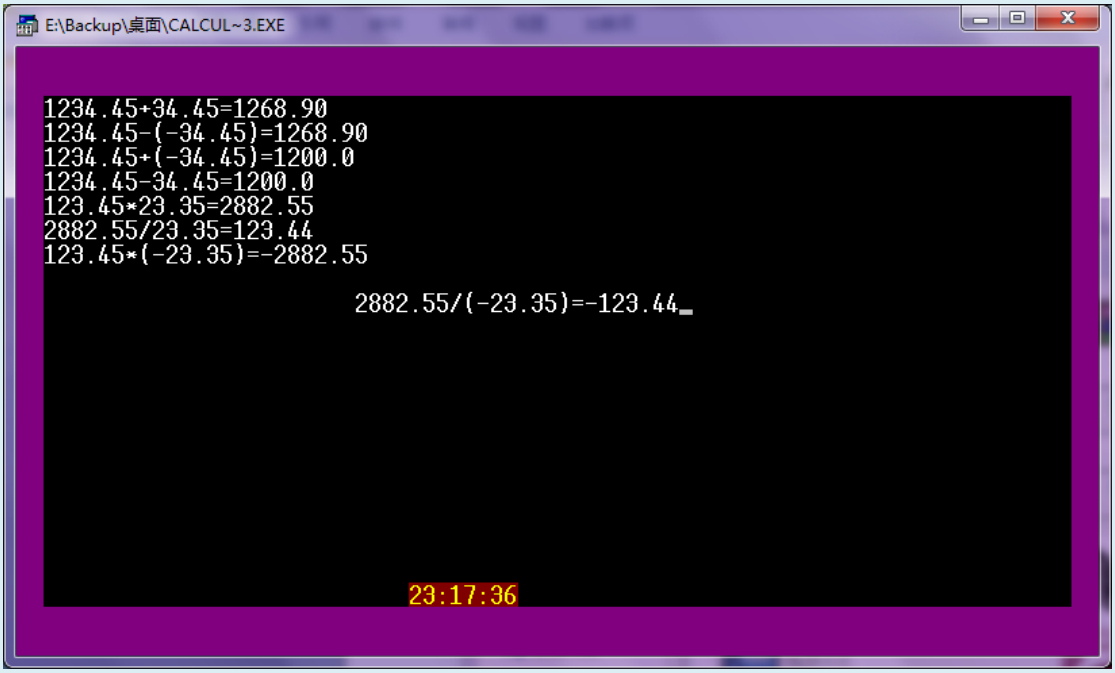


ii. 复杂运算



iii. 多次运算后

特点：正在运算的会在屏幕中央显示，运算过的结果会在屏幕左上角显示



六、 设计与思考

1. 为什么用大量宏和函数

因为，这样可以节省很多代码空间，并且方便分工与单独查错。

2. 实验的难点

最难的地方要选择合适的数据格式在可以完成支持整数部分 2^{15}~-2^{15} 及带两位定点小数，同时能配备好支持它的新的加减乘除法。你可以说利用 dx 与 ax 结合，那么你这样的就无法支持到混合运算当中了。

其次，在支持这个数据格式的四则运算中，除法是一个难点。因为它不像乘法可以通过分配率(a1+b1)*(a2+b2)=a1*a2+a1*b2+a2*1+b1*b2 来实现。

再次，输入部分看似简单，实际要考虑到很多细节，所以也花了大量时间。比如一开始没想到存入的数据可能会符号存进内存后的相同；如何判断存数算法应该结束，对应的存入数和符号；-0.34 在内存中如何表示等等

还有，去括号部分的方法也是修改过两次才得到的。

最后，整个界面的美化及及时时钟也花费了一段时间。

3. 收获与总结（王珏）

整个代码及报告的完成花费了可能我上 50 个小时（只记得自己不知道熬了好几次夜），但是其中可能大部分的时间花在调试上，越到后面代码越长调试越辛苦，8086 解析一个上千条的代码太慢了。

又一次将代码从 0 写到几百，再到上千，再改回 800 条。一条条注释写了满是辛苦。

不过，收获蛮大。从一个月前开始学习汇编到现在，通过自己的书写，调试。开始的，自己很讨厌写注释，到现在习惯的加上注释，从开始不习惯清零，到现在很习惯的清零并用压栈来保证函数及宏不会影响主程序及各自之间。。。

对于注释，我想说：很多时候，注释也是写给自己看的。因为虽然自己写的时候知道那部分用来干什么，可以程序一次，你就可能写了后面，忘了前面。

对于编程，我觉得：不能一开始觉得，想那么高干什么，想到什么就编什么，一步一步来嘛。但是，这样你会发现，新的问题出现，可能是你之前的算法根本无法兼容的。所以，一开始就得想好所有的方面，想好你将来的要用什么方法测试它（最好全面，越全面，你完成后，bug 就越少）。。。。

O(∩_∩)O~就此结束

七、 附件源代码

```
;==给 DD 数 x 赋值的宏==将 DD 型的两个数 X,Y 其中 Y 的值赋给 X
give macro x,y
    mov ax,y
    mov x,ax
    mov ax,y[2]
    mov x[2],ax
endm
;**给 DD 数 x 赋值的宏**将 DD 型的两个数 X,Y 其中 Y 的值赋给 X

;==判断正负宏==     判断 DD 型 x,y 的正负利用 fhx、fhy 两个常数记录下并将
;                   他们都化为正数方便运算
judge macro x,y
local judge1,judge2,judge3,judge4;;标号注释，用于保证宏的重复调用
```

```
                    ;;定位的标号不错乱
        mov ax,x
        cmp ax,0
        jge judge1
        mov fhx,1;;记录 x 的正负
        neg x;;转换为正数
judge1:
        mov ax,x[2]
        cmp ax,0
        jge judge2
        mov fhx,1
        neg x[2]
judge2:
        mov ax,y
        cmp ax,0
        jge judge3
        mov fhy,1
        neg y
judge3:
        mov ax,y[2]
        cmp ax,0
        jge judge4
        mov fhy,1
        neg y[2]
judge4:
endm
;**判断正负宏结束**

;==根据介入 y 值的 0，1 将 x 结果转为正或负==
change macro x,y
local change1
        cmp y,0
        je change1
        neg x
```

```
    neg x[2]
change1:
endm
;**根据介入 y 值的 0，1 将 x 结果转为正或负**

;==进位宏开始== 对小数部分除以 100，将商进位，余数补回小数
carry macro x
    push cx;;免除对宏外面的 cx,dx 值造成干扰
    push dx
    mov ax,x[2]
    mov cx,100
    mov dx,0
    div cx
    mov x[2],dx
    add x,ax
    pop dx
    pop cx
endm
;**进位宏结束**

;==新的加法宏开始== 算法，同号相加，异号相减
newadd macro x,y
    local
subsub,endnewadd,returnadd1,xbig,endadd1,endadd2,endadd3,endadda,endaddc
    judge number3,number4;;不能直接代用 x,y.
                ;;x 其实既 number3,y 既 number4。原因，下面一条注释
    mov ax,fhx
    cmp fhy,ax
    jne  subsub
;;------两个数符号相等则，直接两部分相加
    mov ax,y
    add x,ax
    mov ax,y[2]
    add x[2],ax
```

```
    carry number3
    change number3,fhx
    jmp endnewadd
;;---如果符号相反则，大数减去小数
subsub:
    mov ax,y
    cmp x,ax
    ja   xbig
    jne   endadda
    mov ax,y[2]
    cmp x[2],ax
    ja   xbig
endadda:
    mov ax,x
    sub y,ax
    add y[2],100
    mov ax,x[2]
    sub y[2],ax
    cmp y[2],100
    jnb   endadd1
    sub y,1
    jmp endaddc
endadd1:
    sub y[2],100
endaddc:
    give number3,number4
    change number3,fhy
    jmp endnewadd
xbig:
    mov ax,y
    sub x,ax
    add x[2],100
    mov ax,y[2]
    sub x[2],ax
```

```
    cmp x[2],100
    jnb   endadd2
    sub x,1
    jmp endadd3
endadd2:
    sub x[2],100
endadd3:
    change number3,fhx
    jmp endnewadd
endnewadd:
    mov fhx,0
    mov fhy,0
endm
;**新的加法宏结束**

;==新的减法宏==
newsub macro x,y
    mov fhx,1
    change number4,fhx
    mov fhx,0
    newadd number3,number4
endm
;**新的减法宏**

;==新的乘法宏==     算法  (a1+b1)*(a2+b2)=a1*a2+a1*b2+a2*1+b1*b2
newmul macro x,y
    judge number3,number4
    push bx
    push dx
    mov bx,y
    mov ax,x
    mul bx
    push ax;;压栈用于后面的加法
    mov ax,x[2]
```

```
    mul bx
    push ax;;正数部分乘小数部分的结果可以直接加到小数部分
    mov bx,y[2]
    mov ax,x
    mul bx
    push ax
    mov ax,x[2]
    mul bx
    mov dx,0
    mov bx,100;;小数部分乘小数部分的结果必需再缩小 100 倍，才
              ;;能再加回小数位
    div bx
    mov x[2],ax
    pop ax
    add x[2],ax
    pop ax
    add x[2],ax
    pop ax
    mov x,ax
    carry number3
    mov ax,fhy
    xor fhx,ax
    change number3,fhx
    mov fhx,0
    mov fhy,0
    pop dx
    pop bx
endm
;**新的乘法宏**

;==新的除法宏== 利用减法完成除法运送，但是为了减少逐减次数，所以
                ;先用 a1/(a2+1)得要一个商，这个商一定不会大于逐减
                ;次数，所以就可以从（a1+b1）-商*（a2+b2)开始逐减，
                ;直到减出负数后，回加一个（a2+b2)得（a3+b3)。这时
```

```
                ;候的次数，就是结果的整数部分。最后将(a3+b3)*100
                ;按前面的方法,就可以得到，结果的两位小数部分。
newdiv macro x,y
local endnewdiv
    push bx
    push dx
    push cx
    judge number3,number4
    give fhdx,fhx
    mov fhx,0
    mov fhy,0
    give divn4,number4
    intdiv number3,number4;;求结果的整数部分
    mov bx,100;;将减完的剩余数扩大 100 倍
    mov ax,number3
    mul bx
    mov number3,ax
    mov ax,number3[2];;小数位扩大 100 倍，就等于直接进入整数位
    add number3,ax
    mov number3[2],0
    give number4,divn4
    intdiv number3,number4;;求结果的小数部分
    pop number3[2]
    pop number3
endnewdiv:
    mov ax,fhdy
    xor fhdx,ax
    change number3,fhdx
    mov fhdx,0
    mov fhdy,0
    pop cx
    pop dx
    pop bx
endm
```

```
;**新的除法宏**

;==除法的逐减宏==
intdiv macro x,y
local intdiv1,intdiv2,endintdiv;;内嵌宏，用于逐减
      mov ax,x
      mov bx,y
      add bx,1
      mov dx,0
      div bx
      push ax
      mov bx,y
      mul bx
      mov y,ax
      pop ax
      push ax
      mov bx,y[2]
      mul bx
      mov y[2],ax
      carry number4
      pop cx
      newsub number3,number4
      give number4,divn4;防止减法运算后改变的了 number4 的值
intdiv1:
      newsub number3,number4
      give number4,divn4
      inc cx
      cmp number3,0
      jl endintdiv
      je intdiv2
      jmp intdiv1
intdiv2:
      cmp number3[2],0
      jl endintdiv
```

```
    jmp intdiv1
endintdiv:
    newadd number3,number4
    dec cx
    push cx
endm
;**除法的逐减宏**

;==设置光标宏==
curse macro cury,curx
    mov ah,2
    mov dh,cury
    mov dl,curx
    mov bh,0
    int 10h
endm
;**设置光标宏**

;==定位字符串显示宏==
menu   macro op1,op2,op3 ;菜单显示宏定义 将 op3 的内容显示到 op1,op2 的位置
    mov ah,02h
    mov bh,00h
    mov dh,op1
    mov dl,op2
    int 10h
    mov ah,09h
    lea dx,op3
    int 21h
endm
;**定位字符串显示宏**

;==清屏加色宏==
scroll macro n,ulr,ulc,lrr,lrc,att
```

```
    mov ah,06
    mov al,n;n=上卷行数;n=0 时，整个窗口空白
    mov ch,ulr;左上角行号
    mov cl,ulc;左上角列号
    mov dh,lrr;右下角行号
    mov dl,lrc;右下角列号
    mov bh,att;卷入行属性
    int 10h
endm
;**清屏加色宏**

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;程序开始
data segment
    fhx dw 0;记录 x 值的正负
    fhy dw 0;记录 y 值的正负
    fhdx dw 0;记录除法运算中的 x 值正负
    fhdy dw 0;记录除法运算中的 y 值正负
    divn4 dd 0;除法中帮助暂存 number4
    number0 db 100;计入键盘输入字符串
            db 0
            db 100 dup(0);
    number dw 200 dup(0);存入输入公式处
    number2 dw 200 dup(0);读取去括号后的公式暂存处
    number3 dd 0;存储用于计算的 x 数
    number4 dd 0;存储用于计算的 y 数
    crx    db 20;记录光标列
    cry    db 10;记录光标行
    crx2 db 2;记录光标列 2
    cry2 db 2;记录光标行 2
    memb dw 0;记录 bx 中的值是否储存过
    memx db 0;记录是否应该进入小数存储部分
    memcl db 0;记录已经存储到小数的第几位
    rsi dw 0
    begain db 'Welcome to use our calculator!','$'
```

```
    begain1 db 'data:2009/11/28','$'
    begain2 db 'made by ： Wangjue','$'
    begain3 db '                LiangJinquan','$'
    begain4 db '                ZhuGengfeng','$'
    begain5 db 'press "E" key to exit','$'
    begain6 db 'press any key to contiune','$'
    begain7 db 'Press "Enter" key to introudction','$'
    help    db 'Confine:32512.99~(-32768.00)','$'
    help1   db 'Format: 1.32*99+(98.43/(-34))= ','$'
    help2   db 'Notice 1:negative must like (-x)','$'
    help3   db '            2:only can abet double decimal fraction as 567.33','$'
    help4   db 'press any key go on!','$'
    error1 db ' Error!',13,10,'$'
    DBUFFER DB 8 DUP (':'),12 DUP (' ');时间的底
data ends
code segment
assume cs:code,ds:data,es:data
start:
    mov ax,data
    mov ds,ax
    mov ax,data
    mov es,ax
;----欢迎界面-----
    scroll 0,0,0,24,79,0;清屏
    scroll 25,0,0,24,79,50h;开外窗口，品红底
    scroll 21,2,2,22,77,0fh;开内窗口，黑底白字
    menu 4,20,begain
    menu 6,20,begain1
    menu 8,20,begain2
    menu 10,20,begain3
    menu 12,20,begain4
    menu 14,20,begain5
    menu 16,20,begain6
    menu 18,20,begain7
```

```
    mov ah,01
    int 21h
    cmp al,0dh;'Enter'
    jne helpo
    scroll 21,2,2,22,77,0fh;清屏，内窗
    menu 6,20,help
    menu 8,20,help1
    menu 10,20,help2
    menu 12,20,help3
    menu 14,20,help4
    mov ah,01
    int 21h
    scroll 21,2,2,22,77,0fh;清屏，内窗
    jmp start1
helpo:
    cmp al,45h;'E'
    je exit
    scroll 21,2,2,22,77,0fh;清屏，内窗
;----清零处理-----
start1:
    call time
  start3:
    curse   10,25;光标定位中间
    mov cx,200
    mov si,0
  sclear:
    mov number[si],0
    add si,2
    loop sclear
    mov cx,0
    mov si,0
    mov bx,0
    mov crx,25
    scroll 1,10,25,10,77,0fh;清除原来等式
```

```
;---开始----
      call write
      call loopcount
      curse cry,crx
      call output
      mov ah,01
      int 21h
      scroll 1,cry2,2,cry2,77,0fh;清除原来等式
      menu cry2,crx2,number0[2]
      inc cry2
      cmp cry2,10
      jl    start2
      mov cry2,2
start2:
      jmp start1
exit:
      mov ah,4ch
      int 21h
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;程序结束
;====写入函数======
;数据格式：定义一个 DD 双字数组，双字的第一个字存数的整数部分，第二个字
;                存数的小数部分。但是为了最后的输出方便，以及乘法运算的小数
;                部分运算后向整数部分进位，小数部分采用 100 进一制

;存数范围：十进制 32512.99~(-32767.00),十六进制 7F00.63~8001.00(补码)

;存数算法：整数部分    正数 ai=ai*10+a(i+1),负数对正数求补码
;                小数部分    正数 bi=b1*10+b2,负数对正数求补码

;存数说明：采用单个字符一个一个输入的方式,字符为 0~9 时进入存数算法,当输
;                入是运算符号时结束存数算法,并且开始存数及存运算符.但是这里
要
;                考虑一个问题不能每次有运算符号进入时都存数,比方  A+((X)),如果
每
```

```
;              次都存,A 就会被存 3 次.所以这里我们定义了一个常数 remb 记录每个数存
;              入情况,方便存数时判断

;负数处理：负数的介入要求格式为(-X),因为在读取部分,会依次去括号,这样就
;              变为了-X,所以只需要判断一串公式开头位'-'号,就知道介入的为负
;              数,再直接对 X 求补。(注：这步处理放在支持混合运算的 si_ze 宏部分)

;小数处理: 通过介入'.'来判断是否开始存入小数部分

;特别说明： 1：由于运算符号的 ASCLL 较小,会与介入的数字相等,所以把运算符的
;                  ASCLL 码加上 7F00H 在存入,这样也就要把 7F00H~7FFFH 预留给运算
;                  符号,同时这也会导致存数范围的缩小,不过个人觉得是值的.

;输入格式: 按照平时正常等式的输入方式输入,以等号结尾.
;              例如：11*(1-3)/(-3)=
write proc near
;--写入函数主部分----
startw:
    lea dx,number0
    mov ah,0ah
    int 21h
    xor cx,cx
    mov cl,number0[1]
    mov si,cx
    add si,2
    mov rsi,si
    mov si,0
    mov di,2
    mov bx,0
startw2:
```

```
mov al,number0[di]
inc di
inc crx
mov ah,0
cmp al,45h;'E'  用于程序退出
jz    exit
cmp al,2ah;'*'
jz    memory;转入存数及存运算符
cmp al,2fh;'/'
jz    memory
cmp al,2bh;'/'
jz    memory
cmp al,2dh;'-'
jz    memory
cmp al,29h;')'
jz    memory
cmp al,28h;'('
jz    memory
cmp al,2eh;'.'
jz    memoryx;转入存入小数程序
cmp al,3dh;'='
jz    memory
cmp al,0dh;'CR'
jz     endwrite
sub al,30h;为存数算法做准备,让'1'真正变成二进制码的 1
cmp al,0
jl    error;报错程序
cmp al,9
ja    error
cmp memx,2
je    error
cmp memx,1
je    arithmeticx;小数存数算法程序
jmp arithmetic;整数存数算法程序
```

```
;---memory 存入部分---
memory:
      cmp memb,0;remb 位 0 时表示数已经数存储过；注：开始为 remb 赋值时
                  ;也一定要为 0,因为这是为了防止'(1+3)*4'这样直接以运
                 ;算符开始的等式
      je    memory1
      mov number[si],bx;存入数
      cmp memx,1;
      je    memory2;存小数部分程序
      add si,4
     gomemory1:
      add ax,7f00h;将运算符转换到 7F00~7FFF 之间
      mov number[si],ax;存入运算符
      mov memb,0;说明数已经存储过
      mov memx,0
      mov memcl,0
      mov bx,0
      add si,4
     loop startw2
     jmp endwrite
  memory1:;只存运算符号
      add ax,7f00h
      mov number[si],ax
      add si,4
      loop startw2
      jmp endwrite
  memory2:
       add si,2
      jmp gomemory1
;-memoryx 小数存入部分-
memoryx:
      mov memx,1;当 remx 为 1 时表示开始准备小数存入
      mov number[si],bx
      mov bx,0
```

```
        add si,2;没有加四,应为这下是要转入存小数的下个字节
        loop startw2
        jmp endwrite
;-arithmetic 存数算法--
arithmetic:
        ;算法 bx=bx*10+ax
        push ax
        mov ax,bx
        push cx
        mov cx,10
        mul cx
        pop cx
        pop bx
        add ax,bx
        mov bx,ax
        mov memb,1;说明开始存数
        cmp bx,7f00h;保证数在合适范围内
        jae error
        loop startw2
        jmp endwrite
;-arithmeticx 存数算法--
arithmeticx:
        cmp memcl,1
        je    arithmetix1
        push cx
        mov cx,10
        mul cx
        add bx,ax
        pop cx
        mov memb,1
        inc memcl
        loop startw2
        jmp endwrite
    arithmetix1:
```

```
    add bx,ax
    inc memcl
    loop startw2
    jmp endwrite
;---error 报错部分----
error:
    curse cry,crx
    lea dx,error1
    mov ah,9
    int 21h
    jmp start1;重新运行程序
endwrite:
    mov di,0
    ret
write endp
;***写入函数结束***

;===去括号函数=====
;程序说明：比方'11+(3+99)='这个等式,先从 number 的首位开始每隔两个字取出数
;              与')'的现 ASCLL 码 7f29h 比较从而找到第一个')'的位置,再到回去找到
;              最近的'('的位置,从而将中间没括号的等式读出到 number3 中,接着利
;              用 si_ze 函数算出这个等式的值,并且赋值回 number 中,以替代原来的括
;              号及括号能的等式
loopcount proc near
  startlp:
        mov bp,0
        mov di,0
        mov bx,0
        mov si,0
  startl:
        mov ax,number[si]
        cmp ax,7f29h
```

```
        je    rsee
        add si,4
        cmp ax,7f3dh
        je    lastl
        jmp startl
rsee:
        sub si,4
        mov ax,number[si]
        cmp ax,7f28h
        je rwrite
        jmp rsee
 rwrite:
      push si
        ; mov lct2,si
        mov di,0
      add si,2
    rwrite1:
        add si,2
        mov ax,number[si]
        mov number2[di],ax
        cmp ax,7f29h
        je    rcount
        add di,2
        jmp rwrite1
 rcount:
      push si
        ; mov lct1,si
        call si_ze
      pop ax
      pop di
      push ax
        ; mov di,lct2
        mov ax,number3
        mov number[di],ax
```

```
            mov number3,0
              mov ax,number3[2]
              add di,2
              mov number[di],ax
            mov number3[2],0
            pop si
              ; mov si,lct1
              add si,2
            rcount1:
                    add si,2
                    add di,2
                    mov ax,number[si]
                    mov number[di],ax
                    cmp ax,7f3dh
                    je    startlp
                    jmp rcount1

 lastl:
                mov si,0
       lastll:
                mov ax,number[si]
                mov number2[si],ax
                add si,2
                cmp ax,7f3dh
                je    endl
                jmp lastll

 endl:
          call si_ze
                ret
loopcount endp
;**去括号函数结束**

;==混合四则运算函数==
```

```
;说明：  通过先不处理加减，先把乘除运算进行，将算出的结果带回原来
;        的地方取代。最终得到一个只有加减的公式。来方便运算
;特别说明：  本运算还承贷着将（-1）转为 ffffh 存入 number 中,既负数输入
si_ze proc near
      mov si,4
      mov ax,number2
      cmp ax,7f2dh;'-'
      jne sstart
      give number3,number2[4]
      mov fhx,1
      change number3,fhx
      mov fhx,0
      jmp end4ze
sstart:
      mov ax,number2[si]
       cmp ax,7f2ah;'*'
       je mull;计算乘法
       cmp ax,7f2fh;'/'
       je divv
       cmp ax,7f3dh;'='
       je sznext;去完乘除后跳转
       cmp ax,7f29h;')'
       je sznext
      add si,4
      jmp sstart
mull:
      sub si,4
      give number3,number2[si]
      add si,8
      give number4,number2[si]
      newmul number3,number4
      sub si,8
      give number2[si],number3
      call sloop
```

```
    jmp sstart
divv:
    sub si,4
    give number3,number2[si]
    add si,8
    give number4,number2[si]
    newdiv number3,number4
    sub si,8
    give number2[si],number3
    call sloop
    jmp sstart
sznext:
    give number3,number2
    mov si,0
sznext1:
    add si,4
    mov ax,number2[si]
    cmp ax,7f2bh
    je   addd
    cmp ax,7f2dh
    je   subb
    cmp ax,7f3dh
    je end4ze
    cmp ax,7f29h
    je end4ze
    jmp sznext1
addd:
    add si,4
    give number4,number2[si]
    newadd number3,number4
    jmp sznext1
subb:
    add si,4
    give number4,number2[si]
```

```
      newsub number3,number4
      jmp sznext1
end4ze:
        RET
si_ze endp

sloop proc near;用于循环赋值
      mov di,si
      add si,8
   sloop1:
      add di,4
      add si,4
      mov ax,number2[si]
      push ax
      give number2[di],number2[si]
      pop ax
      cmp ax,7f3dh;等号
      je    endsloop
      cmp ax,7f29h;括号
      je    endsloop
      jmp sloop1
endsloop:
      ret
sloop endp
;**混合四则运算结束**

;==输出函数==;通过除十取余的方法，将数从低位到高位逐一取出再加 30h 转为
;                    ascll 输出。注：先判断数的正负来考虑是否输出符号，接着输出
;                    整数部分再输出一个“.”小数点最后输出小数部分
output proc near
      mov si,rsi
      judge number3,number4
      cmp fhx,1
      jne output1
```

```
        mov number0[si],2dh
        inc si
        mov dx,2dh
        mov ah,02h
        int 21h
output1:
        mov ax,number3
        mov bx,10
        mov cx,0
        call intoutput
        mov number0[si],2eh
        inc si
        mov dx,2eh;输出小数点
        mov ah,02h
        int 21h
        mov ax,number3[2]
        mov bx,10
        mov cx,0
        call intoutput
        mov number0[si],'$'
        ret
output endp

intoutput proc near
output2:
        mov dx,0
        div bx
        push dx
        inc cx
        cmp ax,0
        je    output3
        jmp output2
output3:
        pop ax
```

```
    add ax,30h
    mov number0[si],al
    inc si
    mov dx,ax
    mov ah,02
    int 21h
    loop output3
    ret
intoutput endp
;**输出函数**

CRLF PROC NEAR              ;回车、显示功能过程定义，属性为 NEAR
      MOV DL,0DH              ;把回车的 ASCII 码 0DH 传给 DL
      MOV AH,02H            ;送 DOS  的中断调用功能号
      INT 21H                ; DOS  的中断调用

      MOV DL,0AH            ; 把换行的 ASCII 码 0AH 传给 DL
      MOV AH,02H            ; 送 DOS  的中断调用功能号
      INT 21H                ; DOS  的中断调用
      RET                    ; 返回
CRLF ENDP                   ;完成过程定义

TIME    PROC NEAR             ;显示时间子程序
DISPLAY1:MOV SI,0
         MOV BX,100
         DIV BL
         MOV AH,2CH           ;取时间
         INT 21H
         MOV AL,CH
         CALL BCDASC          ;将时间数值转换成 ASCII 码字符
         INC SI
         MOV AL,CL
         CALL BCDASC
         INC SI
```

```
        MOV AL,DH
        CALL BCDASC
        MOV BP,OFFSET DBUFFER
        MOV DX,161dH
        MOV CX,8
        MOV BX,004EH
        MOV AX,1301H
        INT 10H
        MOV AH,02H
        MOV DX,0300H
        MOV BH,0
        INT 10H
        MOV BX,0018H
RE:     MOV CX,0FFFFH
REA:    LOOP REA
        DEC BX
        JNZ RE
        MOV AH,01H
        INT 16H
        JE   DISPLAY1
        JMP start3
        RET
TIME   ENDP

BCDASC PROC NEAR                    ;时间数值转换成 ASCII 码字符子程序
        PUSH BX
        CBW
        MOV BL,10
        DIV BL
        ADD AL,'0'
        MOV DBUFFER[SI],AL
        INC SI
        ADD AH,'0'
        MOV DBUFFER[SI],AH
```

```
        INC SI
        POP BX
        RET
BCDASC ENDP

code ends
end start
```