## 基于nasm的计算器代码实现

[基于nasm的计算器代码实现 1](#_Toc163940664)

[第一章 项目介绍 2](#_Toc163940665)

[1.1 功能介绍 2](#_Toc163940666)

[1.2 设计思路 2](#_Toc163940667)

[第二章 项目代码分析 3](#_Toc163940668)

[2.1 程序流程分析： 3](#_Toc163940669)

[2.2 程序代码分析 3](#_Toc163940670)

[第三章 项目运行方式 11](#_Toc163940671)

[3.1 汇编，链接，运行，调试说明 11](#_Toc163940672)

[3.2 运行过程： 11](#_Toc163940673)

### 第一章 项目介绍

#### 1.1 功能介绍

利用汇编语言实现从键盘中输入两个10进制数之间的加、减、乘、除计算表达式，然后输出计算结果，再让用户选择根据计算结果再次计算或者结束程序。

#### 1.2 设计思路

通过从键盘中循环读取字符，存入既定的内存空间中，以最后一位输入的非十进制数字符作为输入结束标志。两个10进制数字符串通过加、减、乘、除运算符作为分割。两个10进制数字符串边输入边转化为10进制数并通过判断运算符进行加、减、乘、除计算，然后输出结果。

### 第二章 项目代码分析

#### 2.1 程序流程分析：

简易计算器程序的任务主要有处理输入数据，计算，显示算式结果。需要计算机先对输入数据进行转化分割然后进行计算，最后输出输入的算式及计算结果。

##### **2.1.1 程序流程图解：**

系统输出：Please input x [op] y:

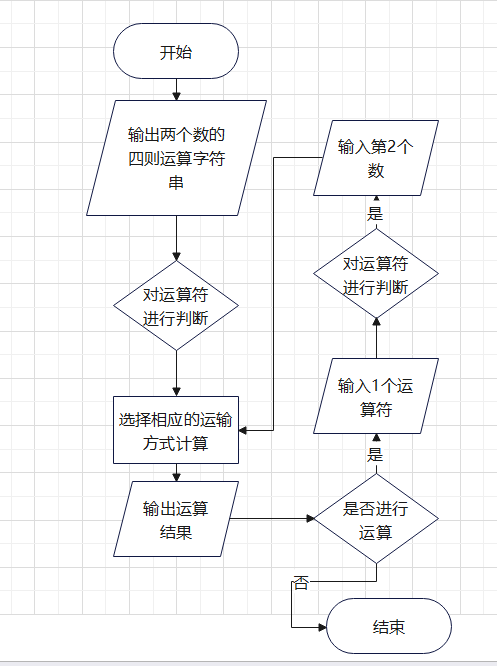
用户输入四则运算式

系统输出：x/y=z （进行除法运算）

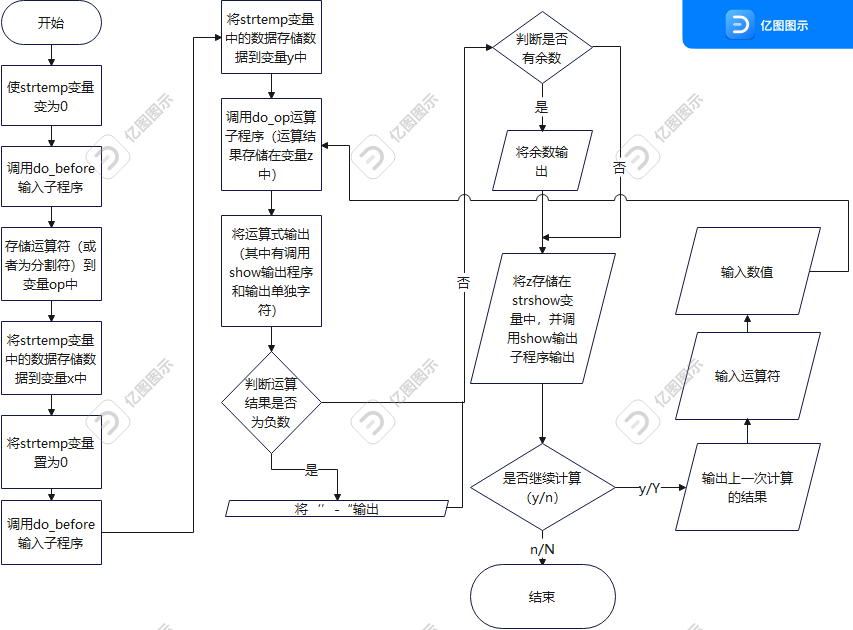
remainder=余数（其他运算没有这部分）

系统输出：Continue typing or no(y/n)

程序流程图如下：



##### **2.1.2 主程序流程图解：**



#### 2.2 程序代码分析

##### 2.2.1 初始变量设置子程序

声明段名，利用伪指令创建需要使用的变量及创建字符串提示信息

##### 2.2.2 初始界面子程序（begin）：

1. 依次调用对应的字符串提示信息来在控制台显示初始界面

2. 调用do\_before子程序来读入用户输入的x值和y值和运算符

         call do\_before ; 调用do\_before函数来读入第一位数字x

         mov [op],al    ; 把运算符存入[op]中,al中存的为运算符

         mov ax,[strtemp]         ; 把读到的x值移入[x]中,strtemp存的为数字的值

         mov [x],ax

         mov dword [strtemp],0         ; 恢复strtemp

         call do\_before         ; 读入第二位数字y

         mov ax,[strtemp]

         mov [y],ax

##### 2.2.3 运算并输出结果子程序（whiledo）：

1. 首先进入运算函数do\_op

2. 以字符串拼接的方式将x，运算符，y，“=”这四个值或符号拼接在一起输出到界面上

3. 判断输出结果是否带负号，如果是的话就把符号标志变量symbol改为负号

         mov dl,[symbol]    ; symbol初值为'+',如果减法出现负数那么改为'-'

         cmp dl,'-'

         jne lan     ; 不为'-'的话进入lan函数

         mov ah,02h  ; 是'-'号的话就输出,注意这个是负号不是减号

         int 21H

4. 把运算结果输出到“=”号的后面

5. 判断运算符是否为除法，如果是除法的话在运算结果后面加上余数的输出，不是除法的话则跳入二次运算do\_move函数

       mov bl,[op]

       cmp bl,'/'

       jne do\_move

       call crlf     ; 换行

       mov dx,message5      ; 输出'remainder='

       mov ah,09h;显示字符串

       int 21h

       mov bx,[remainder]

       mov [strshow],bx

       call show

##### 2.2.4 让用户继续选择输入或结束子程序（do\_move）

1. 输出字符串: Continue typing or no(y/n)

2. 对用户输入的y或n进行判断，然后转入不同的函数

##### 2.2.5 二次运算子程序（lan2）和（done6）

1. 把上一次的运算的结果输出到界面上

2. 读入下一次运算的运算符

3. 判断运算符是什么符号，如果是非运算符则转入报错函数do\_error

4. 如果是正确的运算符，则记录运算符，把上次运算结果z记为x，然后再次读入y值，最后转入运算总函数whiledo

  mov [op],al; 读入运算符

       mov  ax,[z]

       mov [x],ax       ; 把上次运算结果z记为x

       mov dword [strtemp],0       ; 准备再次读入新的y值

       call do\_before

       mov ax,[strtemp]

       mov [y],ax

       jmp wihledo       ; 跳到运算总函数whiledo

##### 2.2.6终止主程序函数（over）

1. 结束程序，返回操作系统

       MOV    AH, 4CH

       INT    21H

##### 2.2.7 判断读入字符类型子程序（do\_before）：

1. 子程序设计思路：

在计算器中，输入的是10进制数字，而在汇编语言中读入字符只能一位一位的读取，故需要设置一个循环算法，将输入的数字以16进制形式放入寄存器中，而输出则再设置一个循环算法，将16进制转化为10进制后处理为ASCII码进行输出。

其次还有判断输入分割和结束的标志，我们这个程序的分割两个10进制数的标志为加减乘除运算符，结束标志为非10进制数字符（加减乘除运算符也是），对输入还未结束的非10进制数字符和非加减乘除运算符会视为错误输入，结束程序。

2. 实现方式：

先判断输入的字符是否为加减乘除符号或回车键，是的话返回主程序

不是运算符或回车键的话再判断是否为数字，对寄存器al的值减去符号“0”的值，这样即可将数字转为ASCII码所代表的数字，是数字的话保存到[temp]中，不是的话转入报错函数

          mov dword [temp],0

          sub al,'0'       ;将数字转化为ASCII码代表的数字

          cmp al,0       ;判断输入的是否是数字(小于0)

          jl do\_error       ;不是数字的话结束输入

          cmp al,9      ;判断输入的是否是数字(大于9)

          jg do\_error      ;不是数字的话结束输入

将[temp]内保存的单个数字存入保存整个数字的字符串[strtemp]中，然后给bx赋值10，再让ax乘bx以实现十进制进位。

重复上面的步骤，直到所有数字都存入[strtemp]中

          xor ah,ah     ;高位清零

          mov [temp],ax     ; 保留ax原来的值,temp存储一个数字,比如9

          mov ax,[strtemp]  ; strtemp相当于整个数字,比如999

          mov bx,10  ; 给bx赋10相当于十进制进位

          ; mul bx相当于ax \* bx,然后就让ax中的值完成了进位

          mul  bx    ; ten 中存储的是10

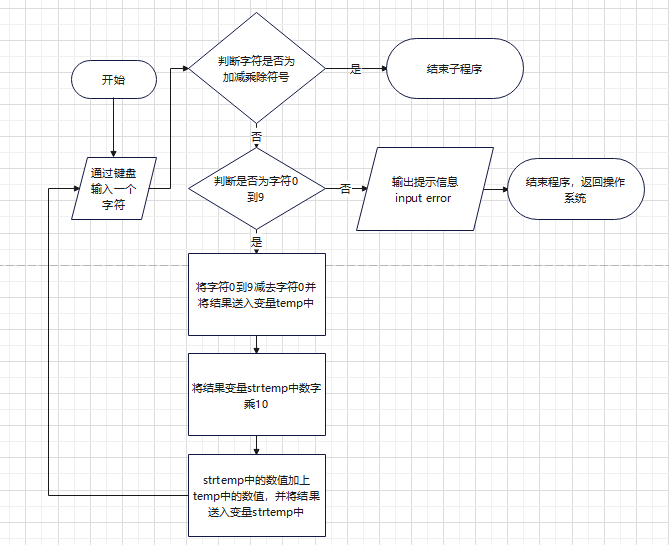
          add ax,[temp]     ; 进位后在加上原来的数

          mov [strtemp],ax  ; 把得到的多个数字存到strtemp中

          ; 数字没存完的话就再循环来存,直到遇到运算符或结尾符(回车符)

          jmp do\_before  ; 输入未结束，循环调用

3. 输入子程序流程图如下：



##### 2.2.8 报错函数（do\_error）

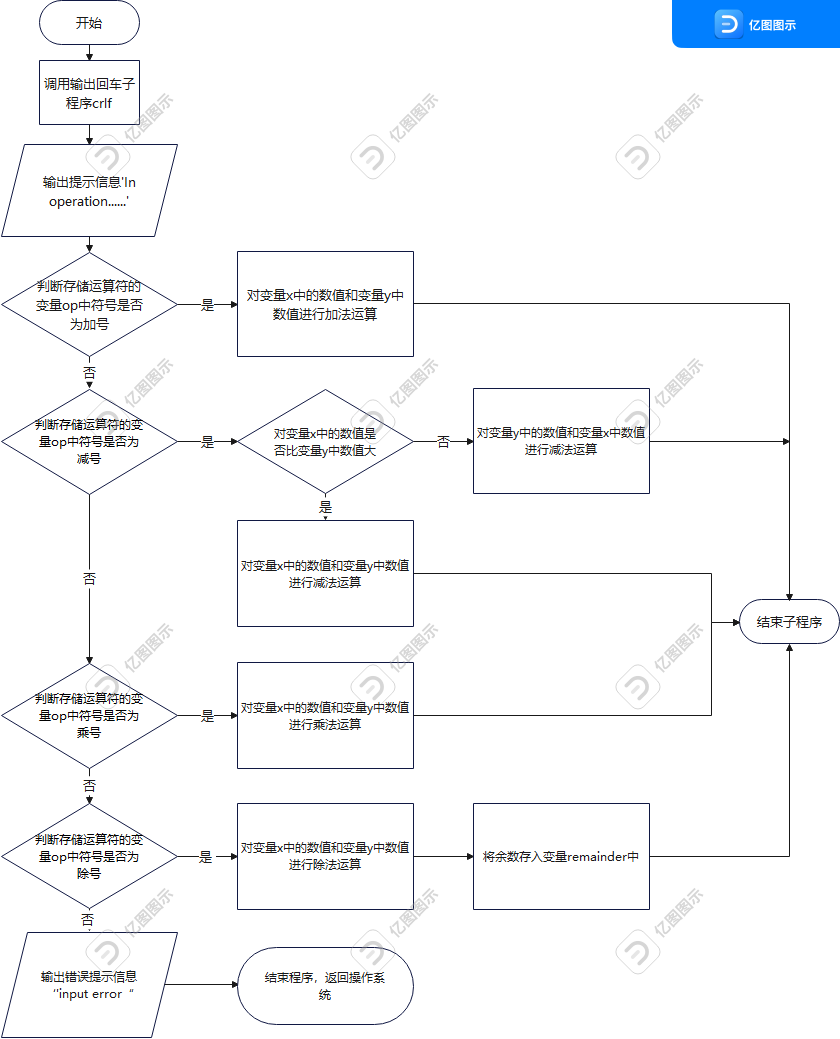
1. 调用报错信息字符串,在界面上输出input error

2. 清零al内的值，返回begin2函数重新读取数字

##### 2.2.9 运算起始函数，也可以充当加法函数（do\_op）

1. 对存储运算符的变量（op）进行判断，选择正确的四则运算方式对输入的两个运算数据变量（x和y）进行运算并存储在运算结果结果变量（z）中。

运算子程序流程图如下：



##### 2.2.10 减法函数（done1）

1. 先判断运算符是否为减号，是的话继续程序，不是的话再转入乘法函数done2

2. 判断减法后的结果是否会产生负数，不产生负数的话转入无符号减法函数do\_sub，产生负数的话就进行负号减法运算

              cmp ax,[y] ;[x]和[y]进行比较

              jge do\_sub  ; 有符号大于等于则跳转到"无负号减法do\_sub函数"

              mov byte [symbol],'-'  ; x小于y的话就进行有负号减法运算

              mov bx,[y]

              sub bx,ax

              mov ax,bx

              jmp done4     ; 输出结果

       do\_sub:       ; 无负号减法函数

              sub ax,[y]

              JMP done4

##### 2.2.11乘法函数（done2）

①先判断运算符是否为乘号，是的话继续程序，不是的话再转入除法函数done3

②进行乘法运算并输出结果

              mul dword [y]

              JMP done4     ; 输出结果

##### 2.2.12 除法函数（done3）

1. 先判断运算符是否为除号，是的话继续程序，不是的话即用户输入了非法运算符，转入报错函数终止程序

2. 进行除法运算

              mov bx,[y]

              mov bh,0

              div  bl

3. 把余数存入[remainder],然后输出结果

              mov [remainder],ah

              mov ah,0

              jmp done4     ; 输出结果

##### 2.2.13 保存结果函数（done4）

将结果存入[z]中，返回主程序

              mov [z],ax

              ret

##### 2.2.14 非法运算符报错函数（done5）

1. 当存入用户输入的运算符时，先进行判断是否为加减乘除符号，如果都不是的话调用报错信息“input error”

2. 恢复断点，返回主程序

##### 2.2.15 显示结果并结束子程序（show）

1. 给bx赋值10，为下一步的div指令做铺垫

2. 用div指令将结果每次除以10，然后把余数压入栈

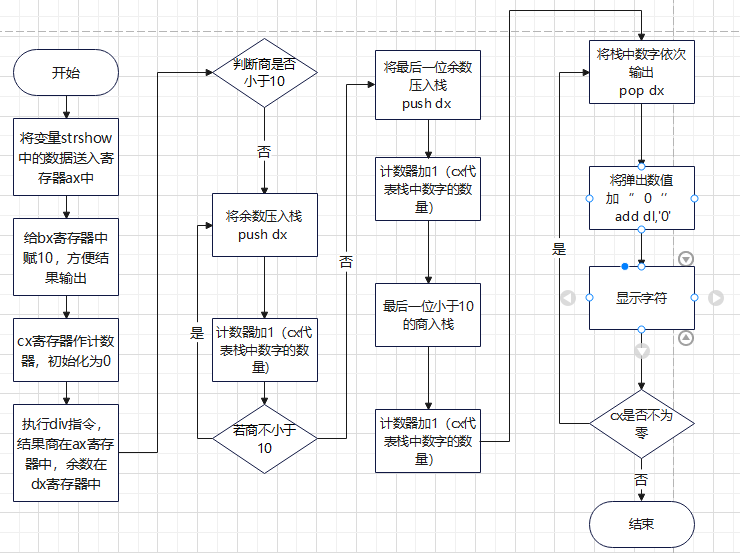
3. 判断商是否小于0，这是作为判断是否除到了最后一位数的依据

4. 如果商小于0，则把余数和商都压入栈

如果商大于0，则重复②③的操作以保证整串数字都压入栈

5. 将数字依次出栈并打印至界面

6. 输出子程序流程图如下：



##### 2.2.16 回车换行子程序（crlf）

对输出内容进行回车换行处理，使程序输出内容更加美观，更有层次性，具体源码如下：

crlf:      ;回车换行

            mov ah,2

            mov dl,0DH

            int 21H

            mov dl,0AH

            int 21H

            ret

### 第三章 项目运行方式

#### 3.1 汇编，链接，运行，调试说明

##### 3.1.1 汇编、链接使用nasm的过程

在windows下打开cmd，切换到文件所在位置，输入编译命令nasm kt3.asm -f bin -o kt3.com，编译成功后可以得到一个以.com结尾的二进制文件

##### 3.1.2 运行、调试使用DoxBox

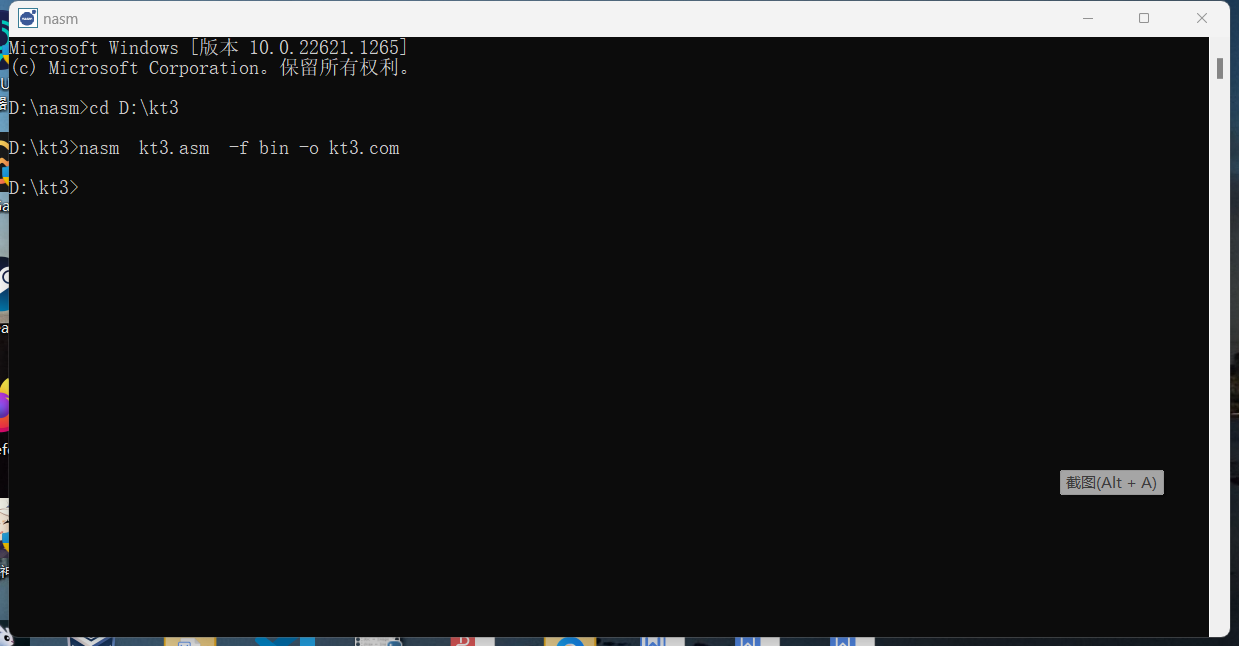
1. 挂载本地文件夹：mount c c:\Users\10799\Desktop\huibian

2. 切换到挂载的文件夹上：c:

3. 运行以.com为结尾的文件：例如: kt3.com

#### 3.2 运行过程：

1. 打开nasm，切换到文件目录，输入nasm kt3.asm -f bin -o kt3.com进行编译



图：nasm汇编

2. 打开DOSbox，输入mount c c:\Users\10799\Desktop\huibian进行挂载本地文件夹，输入c;切换到挂载的文件夹上，打开文件kt3.com

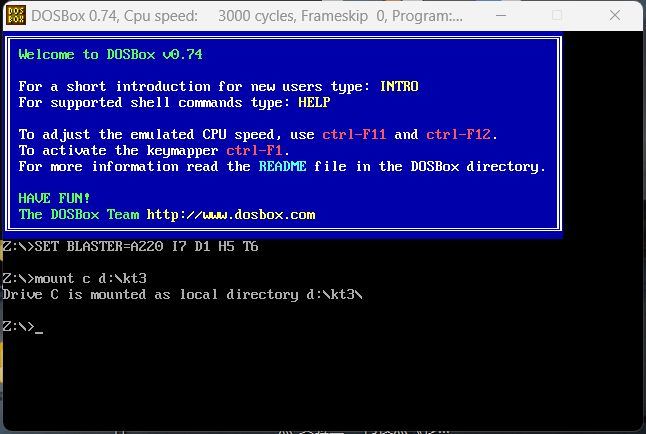


图:程序运行初始界面运行过程

3. 运行程序进行加减乘除的运算以及循环运算功能的展示

（1） 加法运算，如图进行111+45和222+766的运算

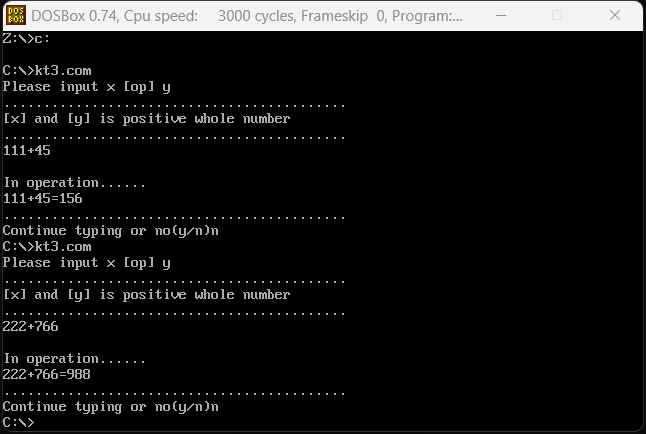


图:加法运行过程

（2） 减法运算，如图进行100-34和87-345的运算

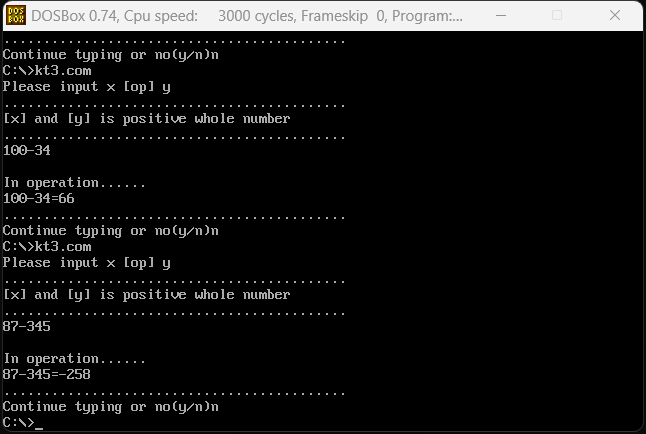


图:减法运行过程

（3） 乘法运算，如图进行345\*765和23+76的运算

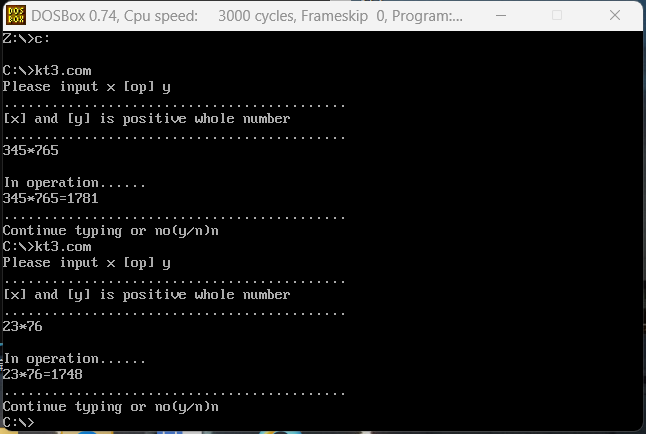
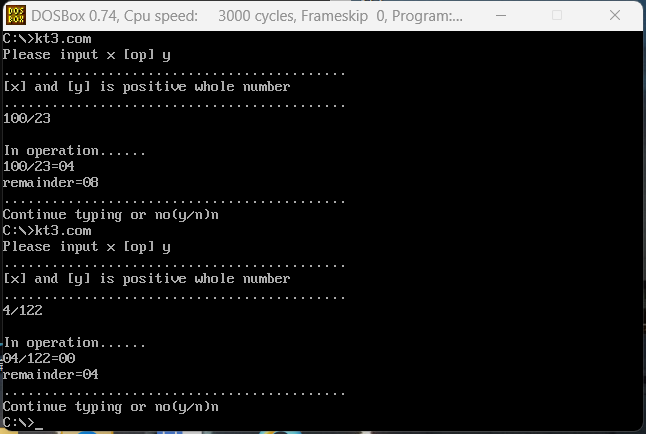


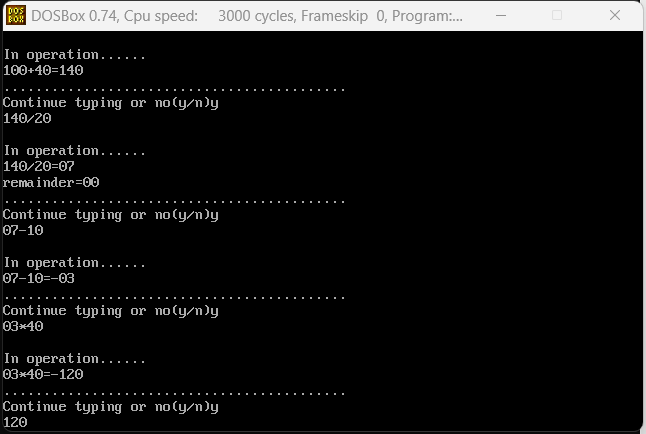
图:乘法运行过程

（4） 除法运算，如图进行100/23和4/122的运算



图：除法运行过程

（5） 循环运算功能，如图进行100+40、140/20、07-10、-03\*40的运算



图：循环运算过程