Descripción de los macros más importantes

Sumar: Tiene como parámetros 2 números a los que desea sumar estos números ya tienen que estar convertidos de ASCII a decimal para poder operarlos, devuelve también en una variable el resultado de la operación junto con el signo correcto que esta tiene que tener.

Restar: Tiene como parámetros 2 números a los que desea restar estos números ya tienen que estar convertidos de ASCII a decimal para poder operarlos, devuelve también en una variable el resultado de la operación junto con el signo correcto que esta tiene que tener.

Multiplicar: Tiene como parámetros 2 números a los que desea multiplicar estos números ya tienen que estar convertidos de ASCII a decimal para poder operarlos, devuelve también en una variable el resultado de la operación junto con el signo correcto que esta tiene que tener.

Dividir: Tiene como parámetros 2 números a los que desea dividir estos números ya tienen que estar convertidos de ASCII a decimal para poder operarlos, devuelve también en una variable el resultado de la operación junto con el signo correcto que esta tiene que tener.

Print: Imprime cadena que se le paso como parametros.

Abrir: Abre un archivo en un ruta especifica y se queda cargado en un handle creado anteriormente.

Cerrar: Tiene como parámetro el handle de abrir y cerrar un archivo abierto anteriormente.

Escribir: Escribe lo que tenga una cadena al archivo, este ya tiene que existir para poder escribir en ellos.

Factorial: Este método tiene como parámetro el número de factorial que desea, este se va operando paso a paso para mostrar el resultado en un ciclo, primero obtiene el factorial deseado, para luego convertirlo a ASCII y poder ser visualizado en la pantalla.

Conversor decimal: Este método tiene como parámetro un número que se encuentra en decimal, le suma 3030h para convertir este decimal a un código ASCII reconocible por un humano.

```
;Macros que se pueden llamar
include macros.asm
.model Large
; -------------- SEGMENTO DE PILA -----------------
.stack
; -------------- Variables a utilizar -----------------
.data
arreglo db 15 dup('$'), '$'
;Donde se guarda el reporte
reporte db 30000 dup(' '), '$'
;Donde se guarda las operaciones
operaciones db 800 dup('$'), '$'
;Donde se guarda la entrada
entrada db 800 dup('$'), '$'
;Donde se almacena el numero 1
;numero1 db 5 dup('$'), '$'
;Donde se almacena el numero 2
;numero2 db 5 dup('$'), '$'
;Donde se almacena el simbolo
simbolo db 5 dup('$'), '$'
;Donde se guarda los simbolos encontrados de la entrada
operarReporte db 800 dup('$'), '$'
imprimirtemporal db 2 dup('$'), '$'
indiceOperarReporte db 0
;Identificador dos caracteres
dosCaracarcteres db 10 dup('$'), '$'
;Identificador
identificadorOP db 30 dup('$'), '$'
;Boleano
bool db 5 dup('$'), '$'
bool2 db 5 dup('$'), '$'
bool3 db 5 dup('$'), '$'
bool4 db 5 dup('$'), '$'
bool5 db 5 dup('$'), '$'
bool6 db 5 dup('$'), '$'
bool7 db 5 dup('$'), '$'
bool8 db 5 dup('$'), '$'

boolPaseSimbolo db 5 dup('$'), '$'

tipoCaso db 5 dup('$'), '$'
contadoNumeros db 0
```

```
contadoPila db 0

boolOperacion db 5 dup('$'), '$'
caracterTemporal db 5 dup('$'), '$'
caracterTemporalResolver db 5 dup('$'), '$'
caracterTemporalOP db 5 dup('$'), '$'
contadorNumero db 0
;es igual
igual db 0ah,0dh, 'Igual' , '$'
;es distinto
distinto db 0ah,0dh, 'Distinto' , '$'
;handle
handlerentrada dw ?

;limpiar reporte
;reporteLimpiar db 256


caracterACaracter db 5 dup('$'), '$'

ascii db 5 dup('$'), '$'

numero1 db 100 dup('$') , '$'
number1 db 100 dup('$') , '$'
number2 db 100 dup('$') , '$'
correcto db 100 dup('$') , '$'
movimientos db 100 dup('$') , '$'

verificarEsNumero db 100 dup('$') , '$'


;indice reportes
indiceReporte db 0
indiceOP db 100 dup('$') , '$'
contadorIndiceOP db 0

numero2 db 100 dup(' ') , '$'
num1 db 100 dup('$'), '$'
num2 db 100 dup('$'), '$'
resul db 0
num3 db 100 dup('$'), '$'
num4 db 100 dup('$'), '$'
```

```
registroAH db 100 dup('$'), '$'


prueba_limpieza db 100 dup('9'), '$'
indice_reporte db 100


test1  db 0
test2  db 0


resultadoFacto db 100 dup('$') , '$'
resultadoFac db 1
resul2 db 0
resultado db 100 dup('$') , '$'
resultado2 db 100 dup('$') , '$'
resultadoFactorial db 100 dup('$') , '$'
conver db 100 dup('$') , '$'

;Concatenar factorial
concatenarFactorial db 100 dup('$') , '$'

;Datos del archivo
file db 'c:\entrrep.html','00h' ;ojo con el 00h es importante
handler dw ?
buffer db 20000 dup(' '), '$'
rute db 100 ,'00h'

;Nombre del archivo
nombreArchivo db 'reporte.html',0
;Nombre del archivo entrada
nombreArchivoEntrada db 'entrrep.html',0
;test salida
sali db '1.txt',0
;indice de operaciones
indiceOperaciones db 0
;operaciones
cadenaOperacion db 0ah,0dh, 'Operacion_' , '$'
;Indice en caso de reinicio
indiceEscribir db 0
;Indice escritura actual
```

```
indiceEscribirActual db 0
;Espacio
espacioR db " " , '$'




;DEBUG
debug db 0ah,0dh, 'PASO' , '$'

;Simbolos
mas db 0ah,0dh, '+' , '$'
menos db 0ah,0dh, '-' , '$'
multi db 0ah,0dh, '*' , '$'
divi db 0ah,0dh, '/' , '$'
signo db 5 dup('$'), '$'
signo2 db 5 dup('$'), '$'
signo3 db 5 dup('$'), '$'
contador db 0

err1 db 0ah,0dh, 'Error al abrir el archivo puede que no exista' , '$'
err2 db 0ah,0dh, 'Error al cerrar el archivo' , '$'
err3 db 0ah,0dh, 'Error al escribir en el archivo' , '$'
err4 db 0ah,0dh, 'Error al crear en el archivo' , '$'
err5 db 0ah,0dh, 'Error al leer en el archivo' , '$'



;Descripcion al iniciar
;Universidad
deseaGuardar db 0ah,0dh, 'Desea guardar la operacion Y/N: ' , '$'
;reporte creado
reporteCreado db 0ah,0dh, 'Reporte creado con exito' , '$'
;Universidad
universidad db 0ah,0dh, 'Universidad de San Carlos de Guatemala' , '$'
;Facultad
facultad db 0ah,0dh, 'Facultad de Ingenieria' , '$'
;Escuela
escuela db 0ah,0dh, 'Escuela de Ciencias y Sistemas' , '$'
;Curso
```

```
curso db 0ah,0dh, 'Arquitectura de ensambladores y computadores 1' , '$'
;Seccion
seccion db 0ah,0dh, 'Seccion B' , '$'
;Semestre
semestre db 0ah,0dh, 'Primer Semestre 2021' , '$'
;Nombre completo
nombre db 0ah,0dh, 'Carlos Antonio Velasquez Castellanos' , '$'
;Carne
carne db 0ah,0dh, '201403654' , '$'
;Descripcion
descripcion db 0ah,0dh, 'Primera practica de Assembler' , '$'
;Salto de linea
skip db 0ah,0dh, ' ' , '$'
;Mensaje de ingreso de archivo
msjAbrir db 0ah,0dh, 'Ingrese la ruta del archivo: ' , '$'

;Generacion del reporte
linea1 db  '<html>$' , 0
linea2 db  '<head>$' , 0
linea3 db  '<title>Reporte Operaciones</title> ' , '$'
linea4 db 0ah,0dh, '<style>  ' , '$'
linea5 db 0ah,0dh, 'body{background-color: #e6e6ff;} ' , '$'
linea6 db 0ah,0dh, '</style> ' , '$'
linea7 db 0ah,0dh, '</head> ' , '$'
linea8 db 0ah,0dh, '<body>' , '$'

linea9 db 0ah,0dh, '<h2 style="text-align:center;">Practica 3 Arqui 1 Seccion
B</h2> ' , '$'
linea10 db 0ah,0dh, '<h4 style="text-align:center;">Estudiante: Carlos Antonio
Velasquez Castellanos</h4>' , '$'
linea11 db 0ah,0dh, '<h4 style="text-align:center;">Carnet: 201403654</h4>' ,
'$'
linea12 db 0ah,0dh, '<script type="text/javascript">' , '$'
linea13 db 0ah,0dh, 'var d = new Date();' , '$'
linea14 db 0ah,0dh, ' var fecha = d.getDate() + "/" + (d.getMonth() + 1) + "/"
+ d.getFullYear();' , '$'
linea15 db 0ah,0dh, 'var tituloFecha = "<h4
style=\"text-align:center;\">Fecha: "+fecha+"</h4>";' , '$'
linea16 db 0ah,0dh, 'var hora = "<h4 style=\"text-align:center;\">Hora: "+
d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds()+"</h4>";' , '$'
linea17 db 0ah,0dh, ' document.write(tituloFecha);' , '$'
linea18 db 0ah,0dh, 'document.writeln(hora);' , '$'
```

```
linea19 db 0ah,0dh, '</script>' , '$'
linea20 db 0ah,0dh, '<div style="text-align:center;"> ' , '$'
linea21 db 0ah,0dh, '<table border="1" style="margin: 0 auto;"> ' , '$'
;linea22 db 0ah,0dh, '<tr>' , '$'
linea23 db 0ah,0dh, ' <td style="text-align:center;"  bgcolor=#006699>ID de
Operacion</td> ' , '$'
linea22 db 0ah,0dh, '<tr>' , '$'
linea222 db 0ah,0dh, ' ' , '$'
linea24 db 0ah,0dh, ' <td style="text-align:center;"
bgcolor=#006699>Operacion</td> ' , '$'
linea25 db 0ah,0dh, ' <td style="text-align:center;"
bgcolor=#006699>Resultado</td> ' , '$'
linea26 db 0ah,0dh, '</tr> ' , '$'

lineaTD db 0ah,0dh, ' <td style="text-align:center;"  bgcolor=#006699> ','$'
lineaTDFin db 0ah,0dh, '</td>','$'

linea27 db 0ah,0dh, '</table> ' , '$'
linea28 db 0ah,0dh, '</div> ' , '$'
linea29 db 0ah,0dh, '</body>' , '$'
linea30 db 0ah,0dh, '<html>' , '$'


;Cadenas del menu principal
;Seleccion de opciones
rotulo  db 0ah,0dh, "             MENU PRINCIPAL" , '$'
marco   db 0ah,0dh, "        *_   _   _   _   _   _*" , '$'
marco1  db 0ah,0dh, " ^      | `_' `-' `_' `-' `_' `|      ^" , '$'
marco2  db 0ah,0dh, " |      | 1. Cargar Archivo    |      |" , '$'
marco3  db 0ah,0dh, " |      | 2. Modo Calculadora  |      |" , '$'
marco4  db 0ah,0dh, " |  (*)  | 3. Factorial        |  \^/  |" , '$'
marco5  db 0ah,0dh, " |_<''>_ | 4. Crear Reporte     | _(#)_ |" , '$'
marco6  db 0ah,0dh, "o+o \ / \0 5. Salir            0/ \ / (=)" , '$'
marco7  db 0ah,0dh, " 0'\ ^ /\/_   _   _   _   _   _ \/\ ^ /`0*" , '$'
marco8  db 0ah,0dh, "   /_^_\ | `_' `-' `_' `-' `_' `| /_^_\" , '$'
marco9  db 0ah,0dh, "   || || |                      | || ||" , '$'
marco10 db 0ah,0dh, "   d|_|b_T______________________T_d|_|b" , '$'

;Mensaje numero 1
ingrese1 db 0ah,0dh, 'Ingrese el primer numero: ' , '$'
;Mensaje numero 2
ingrese2 db 0ah,0dh, 'Ingrese el segundo numero: ' , '$'
```

```
;Mensaje operando
operando db 0ah,0dh, 'Ingrese el operando: ' , '$'
;Mensaje resultado
msjResultado db 0ah,0dh, 'El resultado es: ' , '$'
;Mensaje numero 1
ingreseFactorial db 0ah,0dh, 'Ingrese numero: ' , '$'




mensaje db 0ah,0dh, 'En ciclo' , '$'
saltolinea db 10,'$'
mensaje2 db 10,'El texto es: ','$'


;----------------SEGMENTO DE CODIGO---------------------

.code


main proc far
mov ax,@data
mov ds,ax
mov es,ax

    principio:
         mov si,0
         push si

        ;Carga inicial
        concatenarCadena linea1,reporte,indiceReporte
        concatenarCadena linea2,reporte,indiceReporte
        concatenarCadena linea3,reporte,indiceReporte
        concatenarCadena linea4,reporte,indiceReporte
        concatenarCadena linea5,reporte,indiceReporte
        concatenarCadena linea6,reporte,indiceReporte
        concatenarCadena linea7,reporte,indiceReporte
        concatenarCadena linea8,reporte,indiceReporte
        concatenarCadena linea9,reporte,indiceReporte
        concatenarCadena linea10,reporte,indiceReporte
        concatenarCadena linea11,reporte,indiceReporte
        concatenarCadena linea12,reporte,indiceReporte
```

```
    concatenarCadena linea13,reporte,indiceReporte
    concatenarCadena linea14,reporte,indiceReporte
    concatenarCadena linea15,reporte,indiceReporte
    concatenarCadena linea16,reporte,indiceReporte
    concatenarCadena linea17,reporte,indiceReporte
    concatenarCadena linea18,reporte,indiceReporte
    concatenarCadena linea19,reporte,indiceReporte
    concatenarCadena linea20,reporte,indiceReporte
    concatenarCadena linea21,reporte,indiceReporte
    concatenarCadena linea22,reporte,indiceReporte
    concatenarCadena linea23,reporte,indiceReporte
    concatenarCadena linea24,reporte,indiceReporte
    concatenarCadena linea25,reporte,indiceReporte
    concatenarCadena linea26,reporte,indiceReporte
    mov bool[0], 48
    mov bool2[0], 48
    mov bool3[0], 48
    mov bool4[0], 48
    mov bool5[0], 48
    mov bool5[1], 48
    mov bool5[2], 48
    mov bool6[0], 48
    mov bool7[0], 48
    mov bool8[0], 48
    mov bool8[1], 48
    mov boolOperacion[0], 48
    mov boolOperacion[1], 48
    mov boolPaseSimbolo[0], 48



    ;mov reporte[1599],36
    jmp inicio

inicio:
     clear
     print universidad
     print facultad
     print escuela
     print curso
     print seccion
     print semestre
```

```
        print nombre
        print carne
        print descripcion
        print skip
       ; mov si,0
        ;push si

        mov di,0
        jmp menu

menu:
        mov movimientos[0],48
        mov movimientos[1],33
        mov movimientos[2],61
        mov movimientos[3],49
        mov movimientos[4],44
        mov movimientos[5],32

       ;print operarReporte

       print rotulo
       print marco;
       print marco1;
       print marco2;
       print marco3;
       print marco4;
       print marco5;
       print marco6;
       print marco7;
       print marco8;
       print marco9;
       print marco10;
       print skip
       getChar
       ;cmp para comparar
       cmp al,49 ;mnemonio 31h = 1 en hexadecimal, ascii 49
       je opcion1
       cmp al,50 ;mnemonio 32h = 2 en hexadecimal, ascii 50
       je opcion2
       cmp al,51 ;mnemonio 33h = 3 en hexadecimal, ascii 51
       je opcion3
       cmp al,52 ;mnemonio 34h = 4 en hexadecimal, ascii 52
```

```
        je opcion4
        cmp al,53 ;mnemonio 33h = 3 en hexadecimal, ascii 53
        je salir
        jmp menu

    opcion1:
        ;Mostramos el mensaje de abrir
        print msjAbrir
        ;Obtenemos la ruta del archivo
        ObtenerTexto rute
        abrir rute,handlerentrada  ;le mandamos la ruta y el handler,que será
la referencia al fichero
        limpiar buffer, SIZEOF buffer,24h  ;limpiamos la variable donde
guardaremos los datos del archivo
        leer handlerentrada, buffer, SIZEOF buffer ;leemos el archivo
        cerrar handlerentrada

        ;Empieza la lectura de los archivos

        jmp inicioLeer
    inicioLeer:
         ;Empieza la lectura de los archivos
         mov di,0
         mov al,buffer[di]
         cmp al,60
         je menor


         inc di
         jmp inicioLeer

    menor:
         ;Encontre la etiqueta mayor por ahora no importa OPERACIONES
         inc di
         mov al,buffer[di]
         cmp al,62
         je mayor

         ;print igual

         jmp menor
    mayor:
```

```
      ;Encontrar el menor de la etiqueta del nombre
       inc di
       mov al,buffer[di]
       cmp al,60
       je Oper1

      jmp mayor

Oper1:
       pop si
       mov reporte[si],60
       inc si
       mov reporte[si],116
       inc si
       mov reporte[si],114
       inc si
       mov reporte[si],62
       inc si
       mov reporte[si],60
       inc si
       mov reporte[si],116
       inc si
       mov reporte[si],100
       inc si
       mov reporte[si],62
       inc si
       push si

Oper:
       ;Concatenamos el nombre al reporte
       inc di
       mov al,buffer[di]
       cmp al,62
       je simboloMenor1

       pop si

       mov reporte[si],al
       inc si
       push si
      jmp Oper
```

```
simboloMenor1:

    pop si
    mov reporte[si],60
    inc si
    mov reporte[si],47
    inc si
    mov reporte[si],116
    inc si
    mov reporte[si],100
    inc si
    mov reporte[si],62
    inc si
    push si

simboloMenor:
    ;Encontramos sum resta multi division
    inc di
    mov al,buffer[di]
    cmp al,60
    je guardarOP

    cmp al, 45
    je esNegativo
    cmp al, 48
    je esNumber
    cmp al, 49
    je esNumber
    cmp al, 50
    je esNumber
    cmp al, 51
    je esNumber
    cmp al, 52
    je esNumber
    cmp al, 53
    je esNumber
    cmp al, 54
    je esNumber
    cmp al, 55
    je esNumber
    cmp al, 56
    je esNumber
```

```
        cmp al, 57
        je esNumber



        jmp simboloMenor




    guardarOP:
        ;Mientras no sea mayor concatenar la etiqueta
        pop si
        mov reporte[si],32
        inc si
        push si


        xor si,si
        actualizarContador indiceOperarReporte
        mov operarReporte[si], 32
        xor si,si

        inc di
        mov al,buffer[di]
        ;si es suma
        cmp al,83
        je  EtiquetaSum1
        cmp al,115
        je  EtiquetaSum1
        ;si es resta
        cmp al,82
        je  EtiquetaRes
        cmp al,114
        je  EtiquetaRes
        ;si es multi
        cmp al,77
        je  EtiquetaMul
        cmp al,109
        je  EtiquetaMul
```

```
        ;Si es division
        cmp al,68
        je  EtiquetaDiv
        cmp al,100
        je  EtiquetaDiv

        cmp al, 45
        je esNegativo
        cmp al, 48
        je esNumber
        cmp al, 49
        je esNumber
        cmp al, 50
        je esNumber
        cmp al, 51
        je esNumber
        cmp al, 52
        je esNumber
        cmp al, 53
        je esNumber
        cmp al, 54
        je esNumber
        cmp al, 55
        je esNumber
        cmp al, 56
        je esNumber
        cmp al, 57
        je esNumber

        ;en caso es simbolo de cerrar en etiqueta es porque termino
        ;temporalmente lo mando al menu pero me tendria que ir a detectar
otra etiqueta
        cmp al,47
        je  finEtiqueta


        jmp guardarOP

    esNegativo:
        mov bool3[0],49

        pop si
```

```
        mov reporte[si],al
        inc si
        push si

        xor si,si
        actualizarContador indiceOperarReporte
        mov operarReporte[si], 95
        xor si,si

        jmp Numeral1



    finEtiqueta:


        ;Cierra la etiqueta de la operacion VOY AL METODO QUE VA A OPERAR LA
CADENA
        ;TENGO QUE ABRIR RESULTADO


        ;CUANDO TERMINE EL METODO CIERRA EL RESULTADO

        ;TENGO QUE EVALUAR DONDE RECOLOCO EL PUNTERO PARA QUE SIGA LEYENDO EL
PROXIMO OPERADOR
        jmp retroceso


        ;regresamos al menu
        jmp menu


    retroceso:
        inc di
        mov al,buffer[di]
        cmp al,60
        je retroceso2

        cmp di, 19999
        je ultimos  ;si ocurre algun error que se vaya a menu

       jmp retroceso
```

```
retroceso2:
    inc di
    mov al,buffer[di]
    cmp al,47   ; Si es / no a salido de la etiqueta aun y que vuelva
    je retroceso

    cmp di, 19999
    je ultimos

    ;verifico que sea un valor y comenzar el retroceso

    ;solo en caso que encuentre un valor no retroceso a la etiqueta
    cmp al,118
    je temporal
    cmp al,86
    je temporal


    ;En caso sea una letra
    mov caracterTemporal[0],al
    esLetra caracterTemporal,bool4

    mov al,bool4[0]
    cmp al,49
    je encontre

    je retroceso2

ultimos:

     pop si
     mov reporte[si],60
     inc si
     mov reporte[si],47
     inc si
     mov reporte[si],116
     inc si
     mov reporte[si],100
     inc si
     mov reporte[si],62
     inc si
```

```
        push si


        ;resolvemos la operacion no hay nada mas y operamos para obtener el
resultado



        ;regresamos al menu

        ;print operarReporte

        jmp menu

    temporal:
        ;no tengo que regresar al nombre si no al un valor
        inc di
        mov al,buffer[di]
        cmp al,62   ; Si es / no a salido de la etiqueta aun y que vuelva
        je simboloMenor
        ;print debug

        jmp temporal




    encontre:
        ;Guardamos las operaciones
        pop si
         mov reporte[si],60
         inc si
         mov reporte[si],47
         inc si
         mov reporte[si],116
         inc si
         mov reporte[si],100
         inc si
         mov reporte[si],62
         inc si
         push si
```

```
    ;Operamos para obtener el resultado
    ;xor si,si  ;limpiamos para empezar a utilizarlo
    ;mov si,-1
    ;mov bl,0
    ;mov cl,0
    ;xor ax,ax
    ;jmp resolver

    ;cuando terminamos una operacion regresamos validar otra
    ;bajamos di y nos vamos al inicio para detectar etiqueta
    dec di
    ;Limpiamos variables
     mov bool2[0],48
    jmp Oper1

;*****************************Operacion resolver************
resolver:
    ;print universidad
    inc si
    mov al, operarReporte[si]

    print debug

    cmp si,600
    je menu


    ;verificar simbolo
    cmp al,43 ;sumar
    je concatenarSimbolo
    cmp al,45 ;restar
    je concatenarSimbolo
    cmp al,42 ;multiplicar
    je concatenarSimbolo
    cmp al,47 ;dividir
    je concatenarSimbolo

    ;verifico que venga espacio es el que delimita el numero
    cmp al, 32
    je guardarNumero

    ;ok no viene un simbolo y viene un numero con signo
```

```
        mov caracterTemporalOP[0],al
        esNumero caracterTemporalOP,boolOperacion
        ;miramos que salio si es numero
        mov al , boolOperacion[0]
        cmp al , 49
        ;salio un true nos vamos a guardar un numero
        je concatenarNumero

        cmp al,36
        je menu

        jmp resolver

    concatenarSimbolo:
        ;ingreso al a la pila que es un simbolo
        print nombre
        push ax
        ;aumento la pila
        mov al,contadoPila
        add al,1
        mov contadoPila,al
        xor ax,ax ;limpiamos al

        ;verificamos que tengamos llenos los numeros
        mov al,contadoNumeros
        cmp al,0
        je resolver ;no tenemos ninguno
        cmp al,1
        je resolverLimpiar ;tenemos solo uno y limpiamos
        cmp al,2
        je desApilar ;tenemos solo uno y limpiamos

        jmp resolver
    resolverLimpiar:
        ;si desapile no reseteo

        mov contadoNumeros,0
        jmp resolver

    desApilarPush:
        push ax
        jmp resolver
```

```
desApilar:
    xor ax,ax
    xor bx,bx
    xor cx,cx
    xor dx,dx

    print nombre




    ;le quitamos 3 a la pila
    mov cl,contadoPila
    sub cl,3
    mov contadoPila,cl
    ;pila esta vacia?
    cmp cl,0
    jl mostrarResultados

    ;numero 2 guardamos en bx
    pop bx
    ;numero 1 guardamos en ax
    pop ax
    ;Operacion guadamos en dx
    pop dx

    ;miramos cuando parar
    cmp ax,43 ;sumar
    je desApilarPush
    cmp ax,45 ;restar
    je desApilarPush
    cmp ax,42 ;multiplicar
    je desApilarPush
    cmp ax,47 ;dividir
    je desApilarPush

    ;miramos que simbolo sea
    cmp dl,43 ;sumar
    je sumando
    cmp dl,45 ;restar
    je restando
```

```
    cmp dl,42 ;multiplicar
    je multiplicando
    cmp dl,47 ;dividir
    je dividiendo

encontrarSimbolo:
    ;miramos que simbolo sea
    cmp dl,43 ;sumar
    je sumando
    cmp dl,45 ;restar
    je restando
    cmp dl,42 ;multiplicar
    je multiplicando
    cmp dl,47 ;dividir
    je dividiendo

sumando:
    ;mov cl, boolPaseSimbolo[0]
    ;cmp cl, 49
    add ax,bx
    mov imprimirtemporal[0],al
    print universidad
    print imprimirtemporal
    push ax   ;guardamos ax
    jmp desApilar

restando:
    sub ax,bx
    mov imprimirtemporal[0],al
    print universidad
    print imprimirtemporal
    push ax   ;guardamos ax
    jmp desApilar

multiplicando:
    imul bx
    mov imprimirtemporal[0],al
    print universidad
    print imprimirtemporal
    push ax   ;guardamos ax
    jmp desApilar
```

```
dividiendo:
    idiv bx
    mov imprimirtemporal[0],al
    print universidad
    print imprimirtemporal
    push ax   ;guardamos ax
    jmp desApilar




concatenarNumero:
    ;concateno el numero en numero1
    ;aumento cl

    mov al, boolOperacion[1]
    cmp al,48
    je guardarCaso1 ;guardar el numero1[0]
    cmp al,49
    je guardarCaso2 ;guardar el numero1[0]
    cmp al,50
    je guardarCaso3 ;guardar el numero1[0]
    cmp al,51
    je resolver ;guardar el numero1[0]
guardarCaso1:
     mov boolOperacion[1], 49 ;actualizamos boolOperacion
     mov al, operarReporte[si] ;restauro el numero
     mov numero1[0],al; guardo el numero
     jmp resolver ;regreso a resolver
guardarCaso2:
     mov boolOperacion[1], 50 ;actualizamos boolOperacion
     mov al, operarReporte[si] ;restauro el numero
     mov numero1[1],al; guardo el numero
     jmp resolver ;regreso a resolver
guardarCaso3:
     mov boolOperacion[1], 51 ;actualizamos boolOperacion
     mov al, operarReporte[si] ;restauro el numero
     mov numero1[2],al; guardo el numero
     jmp resolver ;regreso a resolver

guardarNumero:
```

```
        ;convierto el numero de ascii a decimal con signo
        mov al, boolOperacion[1]
        cmp al, 48
        je resolver ;Aun no tiene ningun numero

        ;primero aplicamos un corrector
        corrector numero1
        ;Tenemos en numero correcto y lo convertimos decimal
        ;aumento la pila
        mov al,contadoPila
        add al,1
        mov contadoPila,al
        ;aumento el numero encontrado
        mov al,contadoNumeros
        add al,1
        mov contadoNumeros,al
        ;metemos a pila el numero
        print numero1
        convertirPushar numero1
        limpiar numero1,SIZEOF numero1,36
        print numero1



        ;jmp menu
        jmp resolver


    mostrarResultados:
         ;Agregamos TD
         pop si
         mov reporte[si],60
         inc si
         mov reporte[si],116
         inc si
         mov reporte[si],100
         inc si
         mov reporte[si],62
         inc si
         push si

         cmp al,0
         jl signoResultante
```

```
        mov resultado2,al
        NoImprimirDecimal resultado2,conver

        pop si
        mov al,43
        mov reporte[si],al
        inc si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        pop si
       mov reporte[si],60
       inc si
       mov reporte[si],47
       inc si
       mov reporte[si],116
       inc si
       mov reporte[si],100
       inc si
       mov reporte[si],62
       inc si
       push si


        jmp menu

signoResultante:
        pop si
        mov al,45
        mov reporte[si],al
        inc si
        push si

        mov resultado2,al
        NoImprimirDecimal resultado2,conver
```

```
        pop si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        pop si
      mov reporte[si],60
      inc si
      mov reporte[si],47
      inc si
      mov reporte[si],116
      inc si
      mov reporte[si],100
      inc si
      mov reporte[si],62
      inc si
      push si

       jmp menu
;*****************************FIN Operacion resolver************

EtiquetaSum1:
      ;Comparar si no agrego td
      mov al, bool2[0]
      cmp al, 49
      je antesSum

      pop si
      mov reporte[si],60
      inc si
      mov reporte[si],116
      inc si
      mov reporte[si],100
      inc si
      mov reporte[si],62
      inc si
      push si
      mov bool2[0],49
```

```
antesSum:
     pop si
     mov reporte[si],43
     inc si
     push si

     xor si,si
     actualizarContador indiceOperarReporte
     mov operarReporte[si], 43
     xor si,si



EtiquetaSum:
     ;Tengo que hacer que termine hasta mayor
     inc di
     mov al,buffer[di]
     cmp al,62
     je EtiquetaSum2

     jmp EtiquetaSum

EtiquetaSum2:
     ;Tengo que hacer que termine hasta menor VALOR
     inc di
     mov al,buffer[di]
     cmp al,60
     je EtiquetaSum3

     jmp EtiquetaSum2

EtiquetaSum3:
     inc di
     mov al,buffer[di]  ;Termine valor
     cmp al,62
     je Numeral1

     jmp EtiquetaSum3

Numeral1:
    ;AQUI YA VERIFICO SI ES NUMERO o Etiqueta
```

```
        inc di
        mov al,buffer[di]  ;Termine numero1

        ;Verificamos si es un valor numerico o es un anidado
        cmp al, 45
        je esNegativo
        cmp al, 48
        je esNumber
        cmp al, 49
        je esNumber
        cmp al, 50
        je esNumber
        cmp al, 51
        je esNumber
        cmp al, 52
        je esNumber
        cmp al, 53
        je esNumber
        cmp al, 54
        je esNumber
        cmp al, 55
        je esNumber
        cmp al, 56
        je esNumber
        cmp al, 57
        je esNumber
        cmp al, 60   ; termino o es una multi,divi,resta
        je EvaluarCaso

      jmp Numeral1



  EvaluarCaso:
      ;En caso haya pasado por un retroceso
      ;mov al, bool8[1]
      ;cmp al, 49
      ;en caso que vaya OP
      ;je guardarOP1
```

```
    ;tengo que ver si vengo de guardar un numero o si no hay nada
    mov al, bool3[0]
    ;En caso venga de una numero limpio hasta valor
    cmp al, 49
    je caso1
    cmp al,48
    je caso2

caso1:
    ;en cualquiera de los dos caso limpio la variable
    mov bool3[0],48
    ;en caso hay que llevarlo hasta analizar el proximo valor

    ;Tengo que hacer que termine hasta menor VALOR
     inc di
     mov al,buffer[di]
     cmp al,62
     je caso11

     jmp caso1

caso11:
     inc di
     mov al,buffer[di]  ;Termine valor
     cmp al,60
     je caso111

     jmp caso11

caso111:
     inc di
     mov al,buffer[di]  ;Termine valor
     cmp al,62
     je guardarOP

     jmp caso111


caso2:
    ;en cualquiera de los dos caso limpio la variable
    mov bool3[0],48
```

```
    ;en caso venga limpio nos vamos al operador de nuevo
    jmp guardarOP

esNumber:
     mov bool3[0],49

     pop si
     mov reporte[si],al
     inc si
     push si

     xor si,si
     actualizarContador indiceOperarReporte
     mov operarReporte[si], al
     xor si,si

     jmp Numeral1

EtiquetaRes:
    ;Comparar que no haya agregado ya el <TD>
    mov al, bool2[0]
    cmp al, 49
    je antesRes

     pop si
     mov reporte[si],60
     inc si
     mov reporte[si],116
     inc si
     mov reporte[si],100
     inc si
     mov reporte[si],62
     inc si
     push si

    mov bool2[0],49

antesRes:
     pop si
     mov reporte[si],45
     inc si
     push si
```

```
     xor si,si
     actualizarContador indiceOperarReporte
     mov operarReporte[si], 45
     xor si,si

EtiquetatRes1:
     ;Tengo que hacer que termine hasta mayor
     inc di
     mov al,buffer[di]
     cmp al,62
     je EtiquetatRes2

     jmp EtiquetatRes1

EtiquetatRes2:
     ;Tengo que hacer que termine hasta menor VALOR
     inc di
     mov al,buffer[di]
     cmp al,60
     je EtiquetatRes3

     jmp EtiquetatRes2

EtiquetatRes3:
     inc di
     mov al,buffer[di]  ;Termine valor
     mov si,0
     cmp al,62
     je Numeral1

     jmp EtiquetatRes3

EtiquetaMul:
    ;Comparar que no haya agregado ya el <TD>
    mov al, bool2[0]
    cmp al, 49
    je antesMul

    pop si
    mov reporte[si],60
```

```
    inc si
    mov reporte[si],116
    inc si
    mov reporte[si],100
    inc si
    mov reporte[si],62
    inc si
    push si

    mov bool2[0],49

antesMul:
     pop si
     mov reporte[si],42
     inc si
     push si

     xor si,si
     actualizarContador indiceOperarReporte
     mov operarReporte[si], 42
     xor si,si


EtiquetaMul1:
     ;jmp menu
     inc di
     mov al,buffer[di]
     cmp al,62
     je EtiquetaMul2

     jmp EtiquetaMul1

EtiquetaMul2:
     ;Tengo que hacer que termine hasta menor VALOR
     inc di
     mov al,buffer[di]
     cmp al,60
     je EtiquetaMul3

     jmp EtiquetaMul2

EtiquetaMul3:
```

```
        inc di
        mov al,buffer[di]  ;Termine valor
        mov si,0
        cmp al,62
        je Numeral1

        jmp EtiquetaMul3

    EtiquetaDiv:

        ;Comparar que no haya agregado ya el <TD>
        mov al, bool2[0]
        cmp al, 49
        je antesDiv

        pop si
        mov reporte[si],60
        inc si
        mov reporte[si],116
        inc si
        mov reporte[si],100
        inc si
        mov reporte[si],62
        inc si
        push si

        mov bool2[0],49

    antesDiv:
        pop si
        mov reporte[si],47
        inc si
        push si

        xor si,si
        actualizarContador indiceOperarReporte
        mov operarReporte[si], 47
        xor si,si


    EtiquetaDiv1:
        ;Tengo que hacer que termine hasta mayor
```

```
        inc di
        mov al,buffer[di]
        cmp al,62
        je EtiquetaDiv2

        jmp EtiquetaDiv1

    EtiquetaDiv2:
        ;Tengo que hacer que termine hasta menor VALOR
        inc di
        mov al,buffer[di]
        cmp al,60
        je EtiquetaDiv3

        jmp EtiquetaDiv2

    EtiquetaDiv3:
        inc di
        mov al,buffer[di]  ;Termine valor
        mov si,0
        cmp al,62
        je Numeral1

        jmp EtiquetaDiv3

    opcion2:
        ;tr de inicio
        concatenarCadena linea22, reporte
        concatenarCadena lineaTD, reporte
        concatenarCadena cadenaOperacion, reporte

        mov al,contadorIndiceOP
        add al,48
        mov indiceOP[0],al

        mov al,contadorIndiceOP
        add al,1
        mov contadorIndiceOP,al

        pop si
        mov al,indiceOP[0]
        mov reporte[si],al
```

```
        inc si
        push si

        concatenarCadena lineaTDFin, reporte

        print saltolinea
        print ingrese1
        ObtenerTexto numero1
        print operando
        ObtenerTexto simbolo
        print ingrese2
        ObtenerTexto numero2

        concatenarCadena lineaTD, reporte, indiceReporte
        concatenarCadena numero1, reporte, indiceReporte
        concatenarCadena simbolo, reporte, indiceReporte
        concatenarCadena numero2, reporte, indiceReporte

        mov al,simbolo
        cmp al,43 ;sumar
        je sum
        cmp al,45 ;restar
        je resta
        cmp al,42 ;multiplicar
        je multiplica
        cmp al,47 ;dividir
        je divide

        jmp menu

    opcion3:
        ;limpiamos factorial
        mov resultadoFactorial[0],36
        mov resultadoFactorial[1],36
        mov resultadoFactorial[2],36
        mov resultadoFactorial[3],36
        mov resultadoFactorial[4],36
        mov resultadoFactorial[5],36
```

```
        factorialNormal
msjResultado,movimientos,ingreseFactorial,saltolinea,resultadoFactorial,result
adoFac,resultadoFacto



        print saltolinea
        jmp menu

    opcion4:

        ;Segundo concatenamos el finalizador del documento
        concatenarCadena linea27,reporte,indiceReporte
        concatenarCadena linea28,reporte,indiceReporte
        concatenarCadena linea29,reporte,indiceReporte
        concatenarCadena linea30,reporte,indiceReporte

        limpiar rute, SIZEOF rute,24h ;limpiamos el arreglo bufferentrada con
$
        limpiar buffer, SIZEOF buffer,24h ;limpiamos el arreglo bufferentrada
con $

        limpiar nombreArchivo, SIZEOF nombreArchivo,24h ;limpiamos el arreglo
bufferentrada con $

        ;ObtenerTexto nombreArchivo
        mov nombreArchivo[0],114
        mov nombreArchivo[1],101
        mov nombreArchivo[2],112
        mov nombreArchivo[3],111
        mov nombreArchivo[4],114
        mov nombreArchivo[5],116
        mov nombreArchivo[6],101
        mov nombreArchivo[7],49



        crear nombreArchivo, handler
        escribir  handler, reporte, SIZEOF reporte
        cerrar handler

        print saltolinea
        print reporteCreado
```

```
    print saltolinea

    jmp menu

sum:
    extractorCompleto numero1,num1,num2,test1,signo
    conversor num1,resul,num2
    extractorCompleto numero2,num3,num4,test2,signo2
    conversor num3,resul2,num4

    sumar resul,resul2,resultado2,test1,test2,signo3

    jmp limbo

sum2:
    print ingrese2
    ObtenerTexto numero2

    pop si
    mov al,numero2[0]
    mov reporte[si],al
    inc si
    mov al,numero2[1]
    mov reporte[si],al
    inc si
    push si

    extractorCompleto numero2,num3,num4,test2,signo2

    conversor num3,resul2,num4

    sumar resultado2,resul2,resultado2,test1,test2,signo3

    jmp limbo

resta:

        extractorCompleto numero1,num1,num2,test1,signo
        conversor num1,resul,num2
        extractorCompleto numero2,num3,num4,test2,signo2
        conversor num3,resul2,num4
```

```
        xor al,al
        mov al,signo[0]
        cmp al, 45
        je restaAlternativa

        ;Rumbo normal
        xor al,al
        restar resul,resul2,resultado2,test1,test2,signo3

        jmp limbo

restaAlternativa:
        xor al,al
        mov al,signo2[0]
        cmp al, 45
        je restaComoSuma

        ;Rumbo normal
        xor al,al
        restar resul,resul2,resultado2,test1,test2,signo3

        jmp limbo


restaComoSuma:
        xor al,al
        mov al,1
        mov test1,al
        xor al,al
        sumar resul,resul2,resultado2,test1,test2,signo3

        jmp limboRes

resta2:

        print ingrese2
        ObtenerTexto numero2

        pop si
        mov al,numero2[0]
        mov reporte[si],al
        inc si
```

```
        mov al,numero2[1]
        mov reporte[si],al
        inc si
        push si

        extractorCompleto numero2,num3,num4,test2,signo2
        conversor num3,resul2,num4

        ;multiplicar resul,resul2,resultado2

        ;Tenemos que hacer la ley de signos
        xor al,al
        mov al,signo3[0]
        cmp al, 45
        je restaAlternativa2

        ;Rumbo normal
        xor al,al
        restar resultado2,resul2,resultado2,test1,test2,signo3
        jmp limbo

restaAlternativa2:
    xor al,al
    mov al,signo2[0]
    cmp al, 45
    je restaComoSuma2

    ;Rumbo normal
    xor al,al
    restar resultado2,resul2,resultado2,test1,test2,signo3

    jmp limbo


restaComoSuma2:
    xor al,al
    mov al,1
    mov test1,al
    xor al,al
    sumar resultado2,resul2,resultado2,test1,test2,signo3

    jmp limboRes
```

```
multiplica:

    extractorCompleto numero1,num1,num2,test1,signo
    conversor num1,resul,num2
    extractorCompleto numero2,num3,num4,test2,signo2
    conversor num3,resul2,num4

    ;Tenemos que hacer la ley de signos

    multiplicar resul,resul2,resultado2,test1,test2,registroAH
    leySignos signo,signo2,signo3


    jmp limboMulti

divide:
    extractorCompleto numero1,num1,num2,test1,signo
    conversor num1,resul,num2
    extractorCompleto numero2,num3,num4,test2,signo2
    conversor num3,resul2,num4

    dividir resul,resul2,resultado2,test1,test2
    leySignos signo,signo2,signo3

    jmp limboMulti

multiplica2:

    print ingrese2
    ObtenerTexto numero2

    pop si
    mov al,numero2[0]
    mov reporte[si],al
    inc si
    mov al,numero2[1]
    mov reporte[si],al
    inc si
    push si
```

```
    extractorCompleto numero2,num3,num4,test2,signo2
    conversor num3,resul2,num4

    ;Tenemos que hacer la ley de signos

    multiplicar resultado2,resul2,resultado2,test1,test2,registroAH
    leySignos signo3,signo2,signo3


    jmp limboMulti

divide2:
   print ingrese2
   ObtenerTexto numero2

    pop si
    mov al,numero2[0]
    mov reporte[si],al
    inc si
    mov al,numero2[1]
    mov reporte[si],al
    inc si
    push si

    extractorCompleto numero2,num3,num4,test2,signo2
    conversor num3,resul2,num4

    dividir resultado2,resul2,resultado2,test1,test2

    jmp limboMulti


limbo:
    cuentaIntento contador

    mov al, contador
    cmp al, 10
    je mostrarLimbo

    ;obtenemos la direccion del programa
    print saltolinea
    print operando
```

```
        getChar

        mov simbolo,al

        pop si
        mov al,simbolo[0]
        mov reporte[si],al
        inc si
        push si

        mov al,simbolo
        ;mov al,simbolo
        cmp al,43 ;sumar
        je sum2
        cmp al,45 ;restar
        je resta2
        cmp al,42 ;multiplicar
        je multiplica2
        cmp al,47 ;dividir
        je divide2

        ;Si no es cualquiera de estas opciones muestra el resultado y se le
pide si desea guardar la operacion
        print saltolinea
        print msjResultado
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea



        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
        inc si
        mov al,conver[0]
```

```
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;cerramos tr
        concatenarCadenaMas linea26, reporte, indiceReporte

        ;reset contador
        mov al,0
        mov contador,al


        ;lo guarda?
        print saltolinea
        print deseaGuardar
        print saltolinea
        getchar


         jmp menu

    limboRes:
        ;obtenemos la direccion del programa
        cuentaIntento contador

        mov al, contador
        cmp al, 10
        je mostrarLimboRes

        print saltolinea
        print operando
        getChar

        mov simbolo,al

        pop si
        mov al,simbolo[0]
```

```
        mov reporte[si],al
        inc si
        push si

        ;concatenarCadenaMas numero2, operaciones, indiceOperaciones

        mov al,simbolo

        cmp al,43 ;sumar
        je sum2
        cmp al,45 ;restar
        je resta2
        cmp al,42 ;multiplicar
        je multiplica2
        cmp al,47 ;dividir
        je divide2

        print saltolinea
        print msjResultado
        xor al,al
        mov al,36
        mov signo3[0],al
        xor al,al
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;reset contador
        mov al,0
        mov contador,al

        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
        inc si
        mov al,conver[0]
```

```
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;cerramos tr
        concatenarCadenaMas linea26, reporte, indiceReporte



        ;lo guarda?
        print saltolinea
        print deseaGuardar
        print saltolinea
        getchar



        jmp menu

    limboMulti:
        cuentaIntento contador

        mov al, contador
        cmp al, 10
        je mostrarLimboMulti


        ;obtenemos la direccion del programa
        print saltolinea
        print operando
        getChar

        mov simbolo,al

        pop si
        mov al,simbolo[0]
        mov reporte[si],al
        inc si
        push si
```

```
        mov al,simbolo

        ;mov al,simbolo
        cmp al,43 ;sumar
        je sum2
        cmp al,45 ;restar
        je resta2
        cmp al,42 ;multiplicar
        je multiplica2
        cmp al,47 ;dividir
        je divide2

        mov al, contador
        cmp al, 1
        je limboMulti2

        ;Si no es cualquiera de estas opciones muestra el resultado y se le
pide si desea guardar la operacion
        print saltolinea
        print msjResultado
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;reset contador
        mov al,0
        mov contador,al


        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte
        ;concatenarCadenaMas cadenaOperacion, reporte, indiceReporte
        ;concatenarCadenaMas resultado2, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
```

```
        inc si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;cerramos tr
        concatenarCadenaMas linea26, reporte, indiceReporte

        ;lo guarda?
        print saltolinea
        print deseaGuardar
        print saltolinea
        getchar


        jmp menu

    limboMulti2:
        ;muestra el resultado
        leySignos signo,signo2,signo3
        print saltolinea
        print msjResultado
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;reset contador
        mov al,0
        mov contador,al


        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte
```

```
        ;concatenarCadenaMas cadenaOperacion, reporte, indiceReporte
        ;concatenarCadenaMas resultado2, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
        inc si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;cerramos tr
        concatenarCadenaMas linea26, reporte, indiceReporte

        ;lo guarda?
        print saltolinea
        print deseaGuardar
        print saltolinea
        getchar


        ;lo guarda?



        jmp menu

    mostrarLimbo:
        ;MAXIMO OPERACIONES realizadas
        print saltolinea
        print msjResultado
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;concate cadena final
```

```
    concatenarCadenaMas lineaTDFin, reporte, indiceReporte

    ;resultado
    concatenarCadenaMas lineaTD, reporte, indiceReporte

    pop si
    mov al,signo3[0]
    mov reporte[si],al
    inc si
    mov al,conver[0]
    mov reporte[si],al
    inc si
    mov al,conver[1]
    mov reporte[si],al
    inc si
    push si

    concatenarCadenaMas lineaTDFin, reporte, indiceReporte

    ;cerramos tr
    concatenarCadenaMas linea26, reporte, indiceReporte

    ;reset contador
    mov al,0
    mov contador,al

    ;lo guarda?
    print saltolinea
    print deseaGuardar
    print saltolinea
    getchar









    jmp menu

mostrarLimboRes:
    ;MAXIMO OPERACIONES realizadas
```

```
        ;resultado
        print saltolinea
        print msjResultado
        xor al,al
        mov al,36
        mov signo3[0],al
        xor al,al
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;reset contador
        mov al,0
        mov contador,al

        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
        inc si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;cerramos tr
        concatenarCadenaMas linea26, reporte, indiceReporte



        ;lo guarda?
        print saltolinea
        print deseaGuardar
```

```
        print saltolinea
        getchar

        jmp menu

    mostrarLimboMulti:
        ;MAXIMO OPERACIONES realizadas
        ;resultado
        leySignos signo3,signo2,signo3
        print saltolinea
        print msjResultado
        print signo3
        imprimirDecimal resultado2,conver
        print saltolinea

        ;reset contador
        mov al,0
        mov contador,al


        ;concate cadena final
        concatenarCadenaMas lineaTDFin, reporte, indiceReporte

        ;resultado
        concatenarCadenaMas lineaTD, reporte, indiceReporte
        ;concatenarCadenaMas cadenaOperacion, reporte, indiceReporte
        ;concatenarCadenaMas resultado2, reporte, indiceReporte

        pop si
        mov al,signo3[0]
        mov reporte[si],al
        inc si
        mov al,conver[0]
        mov reporte[si],al
        inc si
        mov al,conver[1]
        mov reporte[si],al
        inc si
        push si

        concatenarCadenaMas lineaTDFin, reporte, indiceReporte
```

```
    ;cerramos tr
    concatenarCadenaMas linea26, reporte, indiceReporte

    ;lo guarda?
    print saltolinea
    print deseaGuardar
    print saltolinea
    getchar

    jmp menu

FuncionLoop:
    Mientras:
        print mensaje ;imprime el mensaje
        Loop Mientras ;lo lleva a la etiqueta mientras pero decrementa cx
        jmp menu ; y cuando cx ya es 0 , avanza y ejecuta este jmp

salir:
    close

Error1:
    print saltolinea
    print err1
    getChar
    jmp menu

Error2:
    print saltolinea
    print err2
    getChar
    jmp menu

Error3:
    print saltolinea
    print err3
    getChar
    jmp menu

Error4:
    print saltolinea
    print err4
    getChar
```

```
        jmp menu

    Error5:
        print saltolinea
        print err5
        getChar
        jmp menu



main endp
end main




;**************************    MACROS    ******************************

print macro cadena ;imprimir cadenas
    mov ah,09h ;Numero de funcion para imprimir cadena en pantalla
    mov dx, @data ;con esto le decimos que nuestrfo dato se encuentra en la
sección data
    mov ds,dx ;el ds debe apuntar al segmento donde se encuentra la cadena
(osea el dx, que apunta  a data)
    mov dx,offset cadena ;especificamos el largo de la cadena, con la
instrucción offset
    int 21h  ;ejecutamos la interrupción
endm

close macro  ;cerrar el programa
    mov ah, 4ch ;Numero de función que finaliza el programa
    xor al,al ;limpiar al
    int 21h
endm

getChar macro ;obtener caracter
    mov ah,01h ;se guarda en al en código hexadecimal del caracter leído
    int 21h
endm

ObtenerTexto macro cadena ;macro para recibir una cadena, varios caracteres
```

```
LOCAL ObtenerChar, endTexto
;si, cx, di  registros que usualmente se usan como contadores
    xor di,di  ; => mov si, 0  reinica el contador

    ObtenerChar:
        getChar  ;llamamos al método de obtener caracter
        cmp al, 0dh ; como se guarda en al, comparo si al es igual a salto de
línea, ascii de salto de linea en hexadecimal o 10en ascii
        je endTexto ;si es igual que el salto de línea, nos vamos a la
etiqueta endTexto, donde agregamos el $ de dolar a la entrada
        mov cadena[di],al ; mov destino, fuente.  Vamos copiando el ascii del
caracter que se guardó en al, al vector cadena en la posicion del contador si
        inc di ; => si = si+1
        jmp ObtenerChar

    endTexto:
        mov al, 36 ;ascii del signo $ o en hexadecimal 24h
        mov cadena[di],al  ;copiamos el $ a la cadena
endm

clear macro ;limpia pantalla
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
         print skip
endm

concatenarCadena macro origen,destino,indiceEscritura
;xor si,si  ; => mov si, 0  reinica el contador
```

```
LOCAL ObtenerCaracter,  termino

    mov si,0
    pop si


    ObtenerCaracter:
        cmp origen[di], 36
        je termino
        mov al, origen[di]
        mov destino[si], al
        inc si
        inc di
        jmp ObtenerCaracter
    termino:
        push si
        mov di,0

endm

concatenarCadena2 macro origen,destino




endm

concatenarCadenaMas macro origen,destino,indiceEscritura
;xor si,si  ; => mov si, 0  reinica el contador
LOCAL ObtenerCaracter,  termino

    ;mov cl,indiceEscritura
    ;mov bl,indiceEscritura

    mov di,0
    pop si


    ObtenerCaracter:
        cmp origen[di], 36
        je termino
```

```
        mov al, origen[di]
        mov destino[si], al
        inc si
        inc di
        ;inc bl
        jmp ObtenerCaracter
    termino:
        ;actualizamos donde empezamos
        push si
        ;print indiceEscritura
        ;mov di,0

endm



crearArchivo macro nombreArchivo
     mov ax,@data  ;Cargamos el segmento de datos para sacar el nombre del
archivo.
     mov ds,ax
     mov ah,3ch ;instrucción para crear el archivo.
     mov cx,0
     mov dx,offset nombreArchivo ;crea el archivo con el nombre archivo2.txt
indicado indicado en la parte .data
     int 21h
     mov bx,ax
     mov ah,3eh ;cierra el archivo
     int 21h
endm



escribir macro handler, buffer, numbytes
    mov ah, 40h
    mov bx, handler
    mov cx, numbytes
    lea dx, buffer
    int 21h


endm
```

```
escribirArchivo macro cadena,nombreArchivo
     pop si
     mov ah,3dh
     mov al,1h
     mov dx,offset nombreArchivo
     int 21h
     ;Escritura de archivo
     mov bx,ax ; mover hadfile
     mov cx,si ;num de caracteres a grabar
     mov dx,offset cadena
     mov ah,40h
     int 21h
     mov ah,3eh ;Cierre de archivo
     int 21h
endm

abrirArchivo macro nombreArchivo,evento
     xor ax,ax
     xor bx,bx
     xor cx,cx
     xor dx,dx



     lea dx, nombreArchivo
     mov ah, 3dh ;Metodo para llamar la interrupcion abrir archivo
     mov al, 00h ;Permiso lectura 000b-lectura,01b Escritura,10
lectura/escritura
     int 21h     ;Interrupcion
     mov evento, ax ;Desplazamiento de cadena nombre archivo
     .if carry?  ;Compara que exista el archivo
     mov ax,-1
     .endif
endm

escribirArchivoReporte macro bytes, evento, datosBuffer
       xor ax,ax
    xor bx,bx
    xor cx,cx
    xor dx,dx
```

```
     mov ah, 40h     ;Sirve para activar el evento
     mov bx, evento  ;Movemos el evento bx
     mov cx, bytes   ;Mover los bytes que se quiere escribir a cx
     lea dx, datosBuffer
     int 21h         ;Interrupcion 21 llamada
endm

leerArchivo macro bytes, evento, datosBuffer
     mov ah, 3fh     ;Sirve para activar el evento
     mov bx, evento  ;Movemos el evento bx
     mov cx, bytes   ;Mover los bytes que se quiere leer a cx
     ;mov si,cx
     lea dx, datosBuffer
     int 21h         ;Interrupcion 21 llamada
endm

cerrarArchivo macro evento
     mov ah,3eh      ;Activador interrupcion
     mov bx,evento   ;llamar al evento
     int 21h         ;interrupcion 21
endm

;Operaciones aritmeticas

sumar macro  numero1,numero2,resultado,test1,test2,signo3
  LOCAL salto,noSalto,fin

     mov al,numero1 ;Mueve a al el numero1
     imul test1
     mov bl,al
     xor al,al
     mov al,numero2 ;Mueve a al el numero1
     imul test2
     add al,bl ;Le suma a al el numero2

     ;resuelta
        cmp al,1
        jg salto
        cmp al,1
        jmp noSalto
```

```
        salto:
            ;positivo
            mov resultado,al ;al -> resultado
            xor al,al
            mov al,43
            mov signo3[0],al
            xor al,al
            mov al,36
            mov signo3[1],al
            mov test1,1
            jmp fin
        noSalto:
            ;negativo
            neg al
            mov resultado,al ;al -> resultado
            xor al,al
            mov al,45
            mov signo3[0],al
            xor al,al
            mov al,36
            mov signo3[1],al
            mov test1,-1

        fin:

endm

restar macro numero1,numero2,resultado,test1,test2,signo3
    LOCAL salto,noSalto,fin

     mov al,numero2 ;Mueve a al el numero1
     imul test2
     mov bl,al
     xor al,al
     mov al,numero1 ;Mueve a al el numero1
     imul test1
     sub al,bl ;Le suma a al el numero2



     ;resuelta
        cmp al,1
```

```
        jg salto
        cmp al,1
        jmp noSalto


        salto:
            ;positivo
            mov resultado,al ;al -> resultado
            xor al,al
            mov al,43
            mov signo3[0],al
            xor al,al
            mov al,36
            mov signo3[1],al
            mov test1,1
            jmp fin
        noSalto:
            ;negativo
            neg al
            mov resultado,al ;al -> resultado
            xor al,al
            mov al,45
            mov signo3[0],al
            xor al,al
            mov al,36
            mov test1,-1
            mov signo3[1],al

        fin:

endm

multiplicar macro numero1,numero2,resultado,test1,test2,registroAH

   LOCAL salto,noSalto,fin

    mov ax,0000



    mov al,numero1 ;Numero1 hacia al
    imul test1 ;signo 1
```

```
    imul numero2    ;Numero2 por numero1
    imul test2  ;Signo 2




    cmp al,1
       jg salto
    cmp al,1
       jmp noSalto


    salto:
        ;print numero1
        mov resultado,al ;al -> resultado
        mov test1,1
        jmp fin
    noSalto:
        ;print numero1
        mov test1,-1
        neg al ;Conversor a positivo
        mov resultado,al ;al -> resultado
    fin:
            mov ah,00h
            mov al,bl
            mov cl,0ah
            div cl
            add ah,30h
            mov registroAH[2],ah
            mov ah,00h
            div cl
            add ah,30h
            mov registroAH[1],ah
            mov ah,09h
            add al,30h
            mov registroAH[0],al
            ;print registroAH
endm

multiplicarSuma macro numero1,numero2,resultado,test1,test2
LOCAL salto,noSalto,fin
    mov al,numero1 ;Numero1 hacia al
```

```
    imul test1
    imul numero2    ;Numero2 por numero1
    imul test2
    cmp al,1
       jg salto
    cmp al,1
       jmp noSalto


salto:
      ;print numero1
      mov resultado,al ;al -> resultado
      jmp fin
noSalto:
      ;print numero1
      neg al
      mov resultado,al ;al -> resultado
fin:

endm

dividir macro numero1,numero2,resultado,test1,test2
LOCAL salto,noSalto,fin
    mov al,numero2 ;Numero1 hacia al
    imul test2
    mov bl,al
    mov al,numero1 ;Numero1 hacia al
    imul test1
    idiv bl

    ;resultado

    cmp al,1
       jg salto
    cmp al,1
       jmp noSalto


    salto:
        ;print numero1
        mov resultado,al ;al -> resultado
        jmp fin
```

```
    noSalto:
        ;print numero1
        neg al
        mov resultado,al ;al -> resultado
    fin:

endm

factorialNormal macro
cartel,movimientos,ingreseFactorial,saltolinea,concatenar,resultado,concatenaR
esultado
LOCAL Mientras,fin
      ;Limpieza de variables
      xor dx,dx
      xor bl,bl
      xor al,al
      xor si,si
      ;Variables iniciales
      mov bl,0
      mov dl,1
      mov si,0
      mov dl,0

        ;Muestra los mensajes de inicio
        limpiarCadena concatenar
        print saltolinea
        print ingreseFactorial
        getChar

        ;Extraemos al y lo convertimos a decimal
        cmp al,48
        je casoEspecial


        mov cl,al
        sub cl,48
        ;Limpiamos
        and ax,0

        ;al se ingresa con 1
        mov al,resultado  ;
```

```
        Mientras:

             mov bl,cl  ;movemos el cl al bl para comparar
             add bl,48  ;Le sumamos 48 para convertirlo a ASCII
             mov concatenar[si],bl  ;Movemos a la cadena
             inc si   ;incrementa si

             mov concatenar[si], 32  ;agregamos espacio a la cadena
             inc si   ;incrementa si

             ;Compara que sea la ultima para no concatenar el resultado
             cmp cl,1
             je imprimirFactorial

             ;Concatenar *
             mov bl,42
             mov concatenar[si],bl
             inc si
             ;realiza el factorial
             mul cl

            Loop Mientras ;lo lleva a la etiqueta mientras pero decrementa cx


        imprimirFactorial:
              mov concatenar[si], 32  ;concatenamos espacio
              inc si   ;Incrementa si

              aam               ; Conversor a ASCII
              add ax, 3030h     ; agrega 3030h
              push ax           ; agregamos a pila ax
              mov dl, ah        ; movemos ah a dl
              mov concatenaResultado[0], dl  ;Concatenamos el primer resultado
decenas
              pop dx            ; Extraemos dx de la pila
              mov al,dl         ; Copiamos dl a al
              mov concatenaResultado[1], al  ;Agremos las unidades
              ;Mensajes para imprimir el resultado
              print saltolinea
              print concatenar
```

```
            print saltolinea
            print cartel
            print concatenaResultado

            jmp fin

        casoEspecial:
            ;volvemos a hacer uno por uno

            add bl,49
            mov concatenar[si],bl
            inc si

            ;Operaciones
            mov al , 1 ; Se coloca en 1
            mov bl, 1
            mul bl
            add al ,48 ; Se agrega 48 para convertir en ASCII

            ;resultado
            mov concatenaResultado[0],al
            mov concatenaResultado[1],36

            ;Imprimir mensajes
            print saltolinea
            print concatenar
            print saltolinea
            print cartel
            print concatenaResultado

        fin:



endm

factorialMacro macro cartel,movimientos,ingreseFactorial,saltolinea,resultado
LOCAL
Conversor,factorial,limpiar,recopilar,mostrar,Especial1,Especial2,fin,Especial
11,Especial22
        mov bh,0        ;limpiamos bh
        mov si,0        ;el contador para mover la matriz
```

```
        and ax,0
        and bx,0
        and cx,0
        and dx,0
        mov di,6

        jmp Especial1
        ;jmp Conversor



        Conversor:
           sub al,48 ;- 48 para convertir a decimal
           mov cl,al ;movemos temporalmente a C low
           and ax,0  ;limpiamos ax = ah + al
           add al,cl ;le volvemos a sumar cl
           mov bh,al ;movemos a low para b high
           and cx,0  ;limpiamos cx
           and ax,0  ;limpiamos ax
           mov cl,bh ;movemos bh para cl
           dec cx    ;restamos uno a cx
           mov al,bh ;restauramos al
           mov di,0

           limpiarCadena movimientos
            mov bl, 32
            mov movimientos[di],bl
            inc di
            mov bl, cl
            add bl, 49
            mov movimientos[di],bl
            inc di
            mov bl, 32
            mov movimientos[di],bl
            inc di




            mov bl, 33
            mov movimientos[di],bl
```

```
            inc di
            mov bl, 61
            mov movimientos[di],bl
            inc di



        factorial:
            mul cx
            push cx
            push ax

            mov bl, 32
            mov movimientos[di],bl
            inc di
            mov bl, cl
            add bl, 49
            mov movimientos[di],bl
            inc di
            mov bl, 32
            mov movimientos[di],bl
            inc di




            mov bl, 33
            mov movimientos[di],bl
            inc di
            mov bl, 61
            mov movimientos[di],bl
            inc di


            and dx,0
            mov bx,0000h
            mov cx,10
            recopilarCadena
            mostrarCadena cartel,movimientos,saltolinea,resultado
            ;limpiarCadena movimientos
            ;limpiarCadena resultado
```

```
        mov di,0
        mov si,0

      pop ax
      pop cx
      loop factorial
      jmp fin

limpiar:
        and dx,0
        mov bx,0000h
        mov cx,10

recopilar:
        div cx    ;dx almacena el resudio que nos interesa
        push dx   ;almacenamos dx en pila
        add bx,1h ;sumamos 1 a bx
        and dx,0  ;limpiamos dx
        cmp ax,0  ;comparamos que no resta nada del numero
        jne recopilar

MOV AH,2
;print saltolinea
;print cartel

mostrar:
        sub bx,1h                   ;restamos a bx 1
        pop dx                      ;restauramos dx de la pila
        add dl,30h                  ;Agregamos 48 a dl para mostrarlo
        mov resultado[si],dl ;Agregamos resultado
        inc si
        ;int 21H                    ;interrupcion 21
        cmp bx,0                    ;comparamos si ya terminamos
        jne mostrar
        print movimientos
        print saltolinea
        print resultado
        jmp fin

Especial1:
        print saltolinea
```

```
    cmp al,48  ;caso especial 11
    je Especial11
    jmp Especial2


Especial11:
    print movimientos
    print cartel
    add al,1
    mov resultado[0],al
    print resultado
    jmp fin


Especial2:
    mov bl, 49
    mov movimientos[di],bl
    inc di
    mov bl, 33
    mov movimientos[di],bl
    inc di
    mov bl, 61
    mov movimientos[di],bl
    inc di
    mov bl, 49
    mov movimientos[di],bl
    inc di
    print movimientos

    cmp al,49  ;caso especial 1
    je Especial22

    jmp Conversor

Especial22:
    print saltolinea
    print cartel
    mov resultado[0],al
    print resultado
    jmp fin
```

```
        fin:
            limpiarCadena movimientos
            limpiarCadena resultado

endm

recopilarCadena macro
LOCAL recopilar

        recopilar:
            div cx    ;dx almacena el resudio que nos interesa
            push dx   ;almacenamos dx en pila
            add bx,1h ;sumamos 1 a bx
            and dx,0  ;limpiamos dx
            cmp ax,0  ;comparamos que no resta nada del numero
            jne recopilar

            MOV AH,2
endm

mostrarCadena macro cartel,movimientos,saltolinea,resultado
LOCAL  mostrar
        mostrar:
            sub bx,1h                   ;restamos a bx 1
            pop dx                      ;restauramos dx de la pila
            add dl,30h                  ;Agregamos 48 a dl para mostrarlo
            mov resultado[si],dl ;Agregamos resultado
            inc si
            ;int 21H                     ;interrupcion 21
            cmp bx,0                    ;comparamos si ya terminamos
            jne mostrar
            print movimientos
            print resultado



endm

limpiarCadena macro  cadena
     mov cadena[0],36
     mov cadena[1],36
     mov cadena[2],36
```

```
     mov cadena[3],36
     mov cadena[4],36
     mov cadena[5],36
     mov cadena[6],36
     mov cadena[7],36
     mov cadena[8],36
     mov cadena[9],36
     mov cadena[10],36
     mov cadena[11],36
     mov cadena[12],36
     mov cadena[13],36
     mov cadena[14],36
     mov cadena[15],36
     mov cadena[16],36
     mov cadena[17],36




endm

imprimirDecimal macro numero,guardar
    mov al, numero
    aam
    add ax, 3030h
    push ax
    mov dl, ah
    mov guardar[0], dl
    mov ah, 02h
    int 21h
    pop dx
    mov al,dl
    mov guardar[1], al
    mov ah, 02h
    int 21h
endm

NoImprimirDecimal macro numero,guardar
    mov al, numero
    aam
    add ax, 3030h
    push ax
```

```
    mov dl, ah
    mov guardar[0], dl
    mov ah, 02h
    ;int 21h
    pop dx
    mov al,dl
    mov guardar[1], al
    mov ah, 02h
    ;int 21h
endm

imprimirDecimalDirecto macro
    aam               ; -> AH is quotient (1) , AL is remainder (2)
    add ax, 3030h     ; -> AH is "1", AL is "2"
    push ax           ; (1)
    mov dl, ah        ; First we print the tens
    ;mov guardar[0], dl
    mov ah, 02h       ; DOS.PrintChar
    int 21h
    pop dx            ; (1) Secondly we print the ones (moved from AL to DL
via those PUSH AX and POP DX instructions
    mov es,dx
    ;mov guardar[1], dl
    mov ah, 02h       ; DOS.PrintChar
    int 21h

endm

multitest macro numeroResultado
     mov ax , 50
     mov cl, 50
     mul cl
     mov numeroResultado,al
     print numeroResultado
endm

convertirPushar macro numero
LOCAL positivo ,negativo,fin
    xor ax,ax
    mov al, numero[0]
    cmp al, 45
    je negativo
```

```
jmp positivo

positivo:
        mov al ,numero[0]
        ;mov resultado[0], al
        sub al,48
        mov cl,10
        mul cl
        mov bl,al
        mov al, numero[1]
        sub al,48
        add bl,al
        mov cl,-1
        mov al,bl
        mul cl
        ;al tiene el resultado y lo metemos a la pila
        push ax

     jmp fin
negativo:
        mov al ,numero[1]
        ;mov resultado[0], al
        sub al,48
        mov cl,10
        mul cl
        mov bl,al
        mov al, numero[2]
        sub al,48
        add bl,al
        mov cl,-1
        mov al,bl
        mul cl
        ;al tiene el resultado y lo metemos a la pila
        push ax


        jmp fin

fin:

        ;print signo
```

```
endm

conversor macro numero1,resultado,numero2
    mov al ,numero1[0]
    ;mov resultado[0], al
    sub al,48
    mov cl,10
    mul cl
    mov bl,al
    mov al, numero2[0]
    sub al,48
    add bl,al
    ;add bl, numero2
    mov resultado,bl

endm

extractor macro arreglo,numero1,numero2
    mov ax,0000
    mov al ,arreglo[0]
    mov numero1[0],al
    mov al, 36 ;ascii del signo $ o en hexadecimal 24h
    mov numero1[1],al  ;copiamos el $ a la cadena
    mov al,0
    mov al ,arreglo[1]
    mov numero2[0],al
    mov al, 36 ;ascii del signo $ o en hexadecimal 24h
    mov numero2[1],al  ;copiamos el $ a la cadena
endm

corrector macro numero
LOCAL positivo,positivo2 ,negativo,fin ,negativo2
    mov al, numero[0]
    cmp al, 95
    je negativo

    jmp positivo
```

```
;Pero en caso venga un negativo o guion bajo



positivo:
     ;verificar que sea 1 solo numero o dos
     mov al, numero[1]
     cmp al,36   ;es dolar entonces solo hay un digito
     je positivo2


     ;En caso que si hay 2 digitos
     mov numero[2], 36 ;dolar al espacio vacio
     jmp fin

positivo2:
     ;En caso que no hay 2 digitos
     mov al, numero[0]
     mov numero[1], al
     mov al, 48
     mov numero[0], al

     mov numero[2], 36 ;dolar al espacio vacio
     jmp fin

negativo:

        mov numero[0] , 45

        mov al ,numero[2]
        ;verificamos si es negativo unico o decimal
        cmp al,36
        je negativo2
        ;en caso que no seguir normal.
        ;no se modifica solo el signo
        jmp fin

negativo2:
        mov al ,numero[1]
        mov numero[2] , al
        mov numero[1], 48
        jmp fin
```

```
    fin:

            ;print signo



endm

extractorCompleto macro arreglo,numero1,numero2,test1,signo
Local ok,malo,fin
            mov ax,0000

            mov al ,arreglo[0]
            cmp al,47 ;0
                ;print arreglo
                ja ok

            cmp al,45
                ;print arreglo
                je malo
            ;abria que agregar en caso es positivo
            ;sobre un resultado anterior.



      ok:
            mov al ,arreglo[0]
            mov numero1[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero1[1],al  ;copiamos el $ a la cadena
            mov al,0
            mov al ,arreglo[1]
            mov numero2[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero2[1],al  ;copiamos el $ a la cadena
            mov al, 1
            mov test1,al
            mov al,43
            mov signo[0],al
            mov al,36
            mov signo[1],al
            ;print signo
```

```
            jmp fin

      malo:
            mov al ,arreglo[1]
            mov numero1[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero1[1],al  ;copiamos el $ a la cadena
            mov al,0
            mov al ,arreglo[2]
            mov numero2[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero2[1],al  ;copiamos el $ a la cadena
            mov al, -1
            mov test1,al
            mov al,45
            mov signo[0],al
            mov al,36
            mov signo[1],al
            ;print signo
        fin:


endm

cuentaIntento macro numero
      xor ax,0
      mov al,numero
      add al,1
      mov numero,al
endm

extractorCompletoSigno macro arreglo,numero1,numero2,test1,signo
Local ok,malo,fin,positivo
            mov ax,0000

            mov al ,signo
            cmp al,43
                ;print arreglo
                je positivo


            cmp al,45
```

```
            ;print arreglo
            je malo


       ;abria que agregar en caso es positivo
       ;sobre un resultado anterior.
positivo:
       mov al ,arreglo[0]
       mov numero1[0],al
       mov al, 36 ;ascii del signo $ o en hexadecimal 24h
       mov numero1[1],al  ;copiamos el $ a la cadena
       mov al,0
       mov al ,arreglo[1]
       mov numero2[0],al
       mov al, 36 ;ascii del signo $ o en hexadecimal 24h
       mov numero2[1],al  ;copiamos el $ a la cadena
       mov al, 1
       mov test1,al
       mov al,43
       mov signo[0],al
       mov al,36
       mov signo[1],al



     jmp fin
ok:
       mov al ,arreglo[0]
       mov numero1[0],al
       mov al, 36 ;ascii del signo $ o en hexadecimal 24h
       mov numero1[1],al  ;copiamos el $ a la cadena
       mov al,0
       mov al ,arreglo[1]
       mov numero2[0],al
       mov al, 36 ;ascii del signo $ o en hexadecimal 24h
       mov numero2[1],al  ;copiamos el $ a la cadena
       mov al, 1
       mov test1,al
       mov al,43
       mov signo[0],al
       mov al,36
       mov signo[1],al
```

```
            ;print signo
            jmp fin

      malo:
            mov al ,arreglo[1]
            mov numero1[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero1[1],al  ;copiamos el $ a la cadena
            mov al,0
            mov al ,arreglo[2]
            mov numero2[0],al
            mov al, 36 ;ascii del signo $ o en hexadecimal 24h
            mov numero2[1],al  ;copiamos el $ a la cadena
            mov al, -1
            mov test1,al
            mov al,45
            mov signo[0],al
            mov al,36
            mov signo[1],al
            ;print signo
        fin:


endm

leySignos macro signo,signo2,signo3
 LOCAL  multi,divi,flag1,flag11,flag111,flag2,flag22,flag222,fin

          xor al,al
          mov al,signo[0]
          cmp al,45
            je flag1
          cmp al,43
            je flag2


    flag1:
          xor bl,bl
          mov bl,signo2[0]
          cmp bl,45
            je flag11
          cmp bl,43
```

```
        je flag111

flag11:
      xor al,al
      mov al,43
      mov signo3[0],al
      xor al,al
      mov al,36
      mov signo3[1],al

      jmp fin

flag111:
      xor al,al
      mov al,45
      mov signo3[0],al
      xor al,al
      mov al,36
      mov signo3[1],al


      jmp fin

flag2:
     xor bl,bl
     mov bl,signo2[0]
     cmp bl,45
       je flag22
     cmp bl,43
       je flag222

flag22:
      xor al,al
      mov al,45
      mov signo3[0],al
      xor al,al
      mov al,36
      mov signo3[1],al

      jmp fin
flag222:
      xor al,al
```

```
          mov al,43
          mov signo3[0],al
          xor al,al
          mov al,36
          mov signo3[1],al

          jmp fin


    fin:


endm

limpiarCadenaCiclo macro cadena,indice
LOCAL limpieza,conversor
    mov cl,indice
    mov si,0

    conversor:
        inc si
        loop conversor

    mov cl,indice

    limpieza:
        mov cadena[si], 48
        dec si
        loop limpieza




endm

esNumero macro numero,verificador
     mov al,numero
     cmp al, 43
     je esNumber
     cmp al, 45
     je esNumber
     cmp al, 48
```

```
    je esNumber
    cmp al, 49
    je esNumber
    cmp al, 50
    je esNumber
    cmp al, 51
    je esNumber
    cmp al, 52
    je esNumber
    cmp al, 53
    je esNumber
    cmp al, 54
    je esNumber
    cmp al, 55
    je esNumber
    cmp al, 56
    je esNumber
    cmp al, 57
    je esNumber



   jmp fin
      esNumber:
      mov verificador[0] ,1



    fin:
      mov verificador[0],0



endm


compararCadena macro cadena1,cadena2,bool
Local iniciar,validacion1,validacion2,fin
       iniciar:
           mov al,cadena1[0]
           mov bl,cadena2[0]
```

```
            cmp al,bl
            je validacion1


            mov bool[0],48   ;es valida
            jmp fin

        validacion1:
            mov al,cadena1[1]
            mov bl,cadena2[1]

            cmp al,bl
            je validacion2

            mov bool[0],48   ;es valida
            jmp fin
        validacion2:
            mov bool[0],49   ;es valida
            jmp fin
        fin:

endm


if_imprimir macro bool,cartel1,cartel2
LOCAL fin ,no
     mov al, bool[0]

     cmp al,48
     je no

     print cartel2

     jmp fin

     no:
       print cartel1

    fin:
```

```
endm


abrir macro buffer,handler

    mov ah,3dh
    mov al,02h
    lea dx,buffer
    int 21h
    jc Error1
    mov handler,ax

endm

limpiar macro buffer, numbytes, caracter
LOCAL Repetir
    xor si,si
    xor cx,cx
    mov cx,numbytes

    Repetir:
        mov buffer[si], caracter
        inc si
        Loop Repetir
endm

cerrar macro handler

    mov ah,3eh
    mov bx, handler
    int 21h
    jc Error2
    mov handler,ax

endm

leer macro handler,buffer, numbytes
    xor ax,ax
    xor bx,bx
    xor cx,cx
    xor dx,dx
```

```
    mov ah,3fh
    mov bx,handler
    mov cx,numbytes
    lea dx,buffer ; mov dx,offset buffer
    int 21h
    jc  Error5

endm

crear macro buffer, handler
    xor ax,ax
    xor bx,bx
    xor cx,cx
    xor dx,dx

    mov ah,3ch
    mov cx,00h
    lea dx,buffer
    int 21h
    jc Error4
    mov handler, ax

endm

escribir macro handler, buffer, numbytes
    xor ax,ax
    xor bx,bx
    xor cx,cx
    xor dx,dx


    mov ah, 40h
    mov bx, handler
    mov cx, numbytes
    lea dx, buffer
    int 21h
    jc Error3

endm
```

```
esLetra macro caracter,bool
LOCAL falso,verdadero,fin
    mov al,caracter[0]
    cmp al, 65
    je verdadero
    cmp al, 66
    je verdadero
    cmp al, 67
    je verdadero
    cmp al, 68
    je verdadero
    cmp al, 69
    je verdadero
    cmp al, 70
    je verdadero
    cmp al, 71
    je verdadero
    cmp al, 72
    je verdadero
    cmp al, 73
    je verdadero
    cmp al, 74
    je verdadero
    cmp al, 75
    je verdadero
    cmp al, 76
    je verdadero
    cmp al, 77
    je verdadero
    cmp al, 78
    je verdadero
    cmp al, 79
    je verdadero
    cmp al, 80
    je verdadero
    cmp al, 81
    je verdadero
    cmp al, 82
    je verdadero
    cmp al, 83
    je verdadero
    cmp al, 84
```

```
    je verdadero
    cmp al, 85
    je verdadero
    cmp al, 86
    je verdadero
    cmp al, 87
    je verdadero
    cmp al, 88
    je verdadero
    cmp al, 89
    je verdadero
    cmp al, 90
    je verdadero
    cmp al, 97
    je verdadero
    cmp al, 98
    je verdadero
    cmp al, 99
    je verdadero
    cmp al, 100
    je verdadero
    cmp al, 101
    je verdadero
    cmp al, 102
    je verdadero
    cmp al, 103
    je verdadero
    cmp al, 104
    je verdadero
    cmp al, 105
    je verdadero
    cmp al, 106
    je verdadero
    cmp al, 107
    je verdadero
    cmp al, 108
    je verdadero
    cmp al, 109
    je verdadero
    cmp al, 110
    je verdadero
    cmp al, 111
```

```
    je verdadero
    cmp al, 112
    je verdadero
    cmp al, 113
    je verdadero
    cmp al, 114
    je verdadero
    cmp al, 115
    je verdadero
    cmp al, 116
    je verdadero
    cmp al, 117
    je verdadero
    cmp al, 118
    je verdadero
    cmp al, 119
    je verdadero
    cmp al, 120
    je verdadero
    cmp al, 121
    je verdadero
    cmp al, 122
    je verdadero



    falso:
         mov bool[0],48
         jmp fin

    verdadero:
         mov bool[0],49
         jmp fin
    fin:




endm

esNumero macro caracter,bool
LOCAL falso,verdadero,fin
```

```
    mov al,caracter[0]
    cmp al, 48
    je verdadero
    cmp al, 49
    je verdadero
    cmp al, 50
    je verdadero
    cmp al, 51
    je verdadero
    cmp al, 52
    je verdadero
    cmp al, 53
    je verdadero
    cmp al, 54
    je verdadero
    cmp al, 55
    je verdadero
    cmp al, 56
    je verdadero
    cmp al, 57
    je verdadero
    cmp al, 95
    je verdadero


    falso:
         mov bool[0],48
         jmp fin

    verdadero:
         mov bool[0],49
         jmp fin
    fin:




endm


actualizarContador macro numero
LOCAL mientras1
```

```
      mov cl,numero
      xor si,si

      mientras1:
           inc si
           loop mientras1
      mov cl,numero
      add cl,1
      mov numero,cl

endm
```