El compilador usado fue FASM y una de sus caracteristicas principles:
-Es un ensamblador multipaso con en el estilo de la sintaxis de Intel.

```
; multi-segment executable file template.

DATA SEGMENT

   msj_inicio:   db "Universidad de San Carlos de Guatemala", 0Dh,0Ah   ;encabezado
                 db "Facultad de Ingenieria", 0Dh,0Ah
                 db "Arquitectura de Computadores y Ensambladores 1", 0Dh,0Ah
                 db "Primer Semestre 2016", 0Dh,0Ah
                 db "Seccion B", 0Dh,0Ah
                 db "Kevin Ariel Cruz Ortiz", 0Dh,0Ah
                 db "201213059", 0Dh,0Ah
                 db "Proyecto 2", 0Dh,0Ah,24h

   mensaje db "----------Bienvenidos----------", 0dh,0ah ;

           db "(1) Ingresar los coeficientes de la funcion.",0dh,0ah
           db "(2) Imprimir funcion almacenada", 0dh,0ah
           db "(3) Imprimir derivada de la funcion almacenada", 0dh,0ah
           db "(4) Graficar la funcion ", 0dh,0ah
           db "(5) Metodo de Newton ", 0dh,0ah ; SUPER METODO 1
           db "(6) Integrar ", 0dh,0ah ; SUPER INTEGRACION
           db "(7) Salir de la aplicacion ", 0dh,0ah; SALIR DE ESTA ONDAAA
           db "Ingrese una opcion valida $", 0dh,0ah ; INGRESAR NUMERO

   msExito db "Si existe funcion en memoria$",0dh,0ah
  msFracaso db "No existe funcion NO se puede graficar$",0dh,0ah
```

```
msGrafi db "1. Normal ", 0dh,0ah ; GRAFICAR NORMAL
     db "2. Derivada ", 0dh,0ah ; GRAFICAR DERIVADA
     db "3. Integral ", 0dh,0ah ; GRAFICAR INTEGRAL
     db "Ingrese el numero de la opcion que desea $ ", 0dh,0ah

newnew db "---------------------- METODO DE NEWTON ------------------ $", 0dh,0ah ; NEWTON



; Variables a mostrar
uno db " OPCION 1 $", 0dh,0ah ; OPCION 1
coef db "Ingresar valor de constante $", 0dh,0ah; COEFICIENTE
coVALUARX1 db "Ingresar valor de X^1 $", 0dh,0ah ; INGRESAR
coVALUARX2 db "Ingresar valor de X^2 $", 0dh,0ah; INGRESAR
coVALUARX3 db "Ingresar valor de X^3 $", 0dh,0ah; INGRESAR
coVALUARX4 db "Ingresar valor de X^4 $", 0dh,0ah; INGRESAR
coVALUARX5 db "Ingresar valor de X^5 $", 0dh,0ah; INGRESAR
 Pide_Punto db 13,10,"Ingrese el Punto INICIAL:  $", 0dh,0ah;
Pide_Tolerancia db 13,10,"Ingrese la Tolerancia: ( 1[0.001], 10[0.01] O 100[0.1] ) $", 0dh,0ah ;
INGRESAR TOLERANCIA
 NIteracion db 13,10,"Ingrese # Iteraciones:  $", 0dh,0ah ; INGRESAR  ITERACIONES
 IteracionN db 13,10,"# Iteracion: $", 0dh,0ah ; NUMERO DE IT
  Xnumber  db 13,10," Xn : $", 0dh,0ah ; XN WOW
  Xnumber1 db 13,10," Xn+1 :  $", 0dh,0ah ; D:
   ErrorM  db 13,10,"ERROR ENCONTRADO: 0. $", 0dh,0ah ; :D
   ErrorMe db 13,10,"----- ERROR < TOLERANCIA $", 0dh,0ah ; :)
   ErrorMa db 13,10,"ERROR > TOLERANCIA $", 0dh,0ah ; :(
ResultadoNewton db 13,10,"LA RAIZ ESTA EN (APROX):  $", 0dh,0ah ; YEAH YEAH
```

```
xsix db "x^6 $", 0dh,0ah ; COMENTARIO 6
xfive db "x^5 $", 0dh,0ah; COMENTARIO 5
xfour db "x^4 $", 0dh,0ah; COMENTARIO 4
xthree db "x^3 $", 0dh,0ah; COMENTARIO 3
xtwo db "x^2 $", 0dh,0ah; COMENTARIO 2
xone db "x $", 0dh,0ah; COMENTARIO 1
punto db ".$", 0dh,0ah; COMENTARIO P
Merror db "Error caracter incorrecto $", 0dh,0ah; COMENTARIO ERROR
pedir_p db "Ingrese el polinomio $", 0dh,0ah; COMENTARIO PUNTO
despeje_x db "Despeje X: $", 0dh,0ah; COMENTARIO DESPEJAR

espacio db "$",0dh,0ah

; Numeros
Num2 db 2       ; Decena del numero 0
Num3 db 3       ; Decena del numero 1
Num4 db 4       ; Decena del numero 2
Num5 db 5       ; Decena del numero 3
Num6 db 6       ; Decena del numero 4
finS dw 0 ; FIN DEL SISTEMA
; Variables funciones
Unidad0 dw 0   ; Decena del numero 0
Unidad1 dw 0   ; Decena del numero 1
Unidad2 dw 0   ; Decena del numero 2
Unidad3 dw 0   ; Decena del numero 3
Unidad4 dw 0   ; Decena del numero 4
Unidad5 dw 0   ; Decena del numero 5
```

```
; Variables fx
fx0 dw 0        ; Decena del numero 0
fx1 dw 0        ; Decena del numero 1
fx2 dw 0        ; Decena del numero 2
fx3 dw 0        ; Decena del numero 3
fx4 dw 0        ; Decena del numero 4
fx5 dw 0        ; Decena del numero 5

; Variables fx derivada
fxd0 dw 0       ; Decena del numero 0
fxd1 dw 0       ; Decena del numero 1
fxd2 dw 0       ; Decena del numero 2
fxd3 dw 0       ; Decena del numero 3
fxd4 dw 0       ; Decena del numero 4
DIEZ dw 10
Iteraciones dw 0


; Variables para STEFFENSEN
po  dw 0 ; PUNTO INCIAL 0
p1  dw 0 ; PUNTO 1
p2  dw 0 ; PUNTO 2
p3  dw 0 ; PUNTO 3
P0P0 db "po:", 0 ; MENSAJE PUNTO 0
P1P1 db "p1:", 0 ; MENSAJE PUNTO 1
P2P2 db "p2:", 0 ; MENSAJE PUNTO 2
P3P3 db "p3:", 0  ; MENSAJE PUNTO 3

ANT_CO0 dw 0 ; COEF 0
```

```
ANT_CO1 dw 0; COEF 1
ANT_CO2 dw 0; COEF 2
ANT_CO3 dw 0; COEF 3
ANT_CO4 dw 0; COEF 4
ANT_CO5 dw 0 ; COEF 5
N dw 0 ; NUMERO DE ENGASADES

FRAC_RES dw 0; MENSAJE PUNTO 0
P0_RES dw 0; MENSAJE PUNTO 0
VER222 dw 0 ; MENSAJE PUNTO 0
FRAC_FRAC dw 0 ; MENSAJE PUNTO 0
FUNC_XXX dw 0; MENSAJE PUNTO 0
CENTINELA_P dw 0  ; MENSAJE PUNTO 0
FUN_GXX         dw 0  ; MENSAJE PUNTO 0
VER111 dw 0; MENSAJE PUNTO 0
RESULT_RESTD dw 0 ; MENSAJE PUNTO 0

; Variables Derivada
numeroD dw 0  ; contador i de ciclo
constD1 dw 0    ; contador i de ciclo
constD2 dw 0    ; contador i de ciclo
constD3 dw 0    ; contador i de ciclo
constD4 dw 0    ; contador i de ciclo
constD5 dw 0    ; contador i de ciclo


; Variables Integral
constIntegral db " + C$", 0dh,0ah
constI1 dw 0     ; Decena del X^1
```

```
constI2 dw 0     ; Decena del X^2
constI3 dw 0     ; Decena del X^3
constI4 dw 0     ; Decena del X^4
constI33 dw 0    ; Decena del X^4
constI5 dw 0     ; Decena del X^5
constI6 dw 0     ; Decena del X^6


valI1 dw 0       ; Decena del X^1
valI2 dw 0       ; Unidad del X^1
valI3 dw 0       ; Decena del X^2
valI4 dw 0       ; Unidad del X^2
valI5 dw 0       ; Decena del X^3
valI6 dw 0       ; Unidad del X^3


; Variables Graficacion
posx dw 0        ; contador i de ciclo
posy dw 0        ; contador linea y
i dw 0   ; contador i de ciclo
j dw 0   ; contador j de ciclo
x dw 0   ; posicion x en la matriz
aux dw 0         ; posicion y en la matriz
y dw 0   ; posicion y en la matriz
xr dw 0 ; posicion x inicial real en la pantalla
yr dw 0 ; posicion y real en la pantalla
xcc dw 1; tamaño del cuadro
contx dw 0       ; contador linea x
conty dw 0       ; contador linea y
```

```
        ;Variables para NEWTON
        ToleranciaF dw 0
        Xinicial dw 0
        ResultadoFX  dw 0
        ResultadoFXD dw 0
        number dw 1
        XN dw 0
        XN1 dw 0
        DECIMALESSS dw 0       ; DECIMALESSS decimal
        fxF dw 0           ; Flag del fx
        fxDF dw 0          ; Flag del fxd
        THOUSND dw 1000       ; 1000
        FINAL_NUMBBBER dw 0           ; Numero final
        FRACCION_FFF dw 0     ; Flag de las fracciones
        FRACCION_NNN dw 0   ; Flag de las fracciones
        XNRes dw 0




DATA ENDS


CODE SEGMENT


ASSUME CS:CODE, DS:DATA
        START:


                mov AX, DATA
```

```
        mov DS, AX
        mov AH, 09H
        lea DX, mensaje                 ; Imprime el texto.
        int 21H

        mov ah, 0
        mov al, 3h
        int 10h ;imprimir el menu
        mov dx, offset mensaje ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos
    jmp LEEROPCION

LEEROPCION:
    mov ah, 00h ; limpia la pantalla
    mov ah, 8h      ; leer caracter
    int 21h

    cmp al, 31h ; compara a 1
    je COEFI

    cmp al, 32h ; compara a 2
    je PRINTFUN

    cmp al, 33h ; compara a 3
    je DERFUNP

    cmp al, 34h ; compara a 4
    je VERMEMORY
```

```
        cmp al, 35h ; compara a 5
        je NEWTON

        cmp al, 36h ; compara a 6
        je INTFUNP

        cmp al, 37h ; compara a 8
        je STOP


        jne LEEROPCION



GRAFP:

        CALL DERFUN
        CALL LIMPIAR
        CALL INTFUN
        CALL LIMPIAR
        CALL GRAFICAR
        CALL LIMPIAR
RET

SALIR:
        ret ; salir

        WriteCh    macro   char
```

```
        push    ax
        mov     al, char
        mov     ah, 0eh
        int     10h
        pop     ax
            endm


;---------------------------------------------------------- INICIAN PROCEDIMIENTOS DE FUNCION ----------
-----------------------------------


;----------------------------------------------------------    IMPRIMIR FUNCION    ----------------------------------------
----------------------------


VERMEMORY:
    cmp unidad0,00h
    ja EXITO
    cmp unidad1,00h
    ja EXITO
    cmp unidad2,00h
    ja EXITO
    cmp unidad3,00h
    ja EXITO
    cmp unidad4,00h
    ja EXITO
    cmp unidad5,00h
    ja EXITO

    jmp FRACASO
```

```
EXITO:

    mov ah,09
    lea dx,msExito
    int 21h

    mov ah,1
    int 21h

    jmp GRAFP

FRACASO:

    mov ah,09
    lea dx,msFracaso
    int 21h

    mov ah, 1
    int 21h

    jmp START

PRINTFUND:
        call JUMP
```

```
        mov ax,constD4
        call PrintNumb

        mov ah, 09
        lea dx, xfour
        int 21h

;IMPRESION DEL X^3

        mov ax,constD3
        call PrintNumb
        mov ah, 09
        lea dx, xthree
        int 21h

;IMPRESION DEL X^2

        mov ax,constD2
        call PrintNumb

        mov ah, 09
        lea dx, xtwo
        int 21h

;IMPRESION DEL X^1

        mov ax,constD1
        call PrintNumb
```

```
        mov ah, 09
        lea dx, xone
        int 21h

;IMPRESION DE LA CONSTANTE

        mov ax,numeroD
        call PrintNumb

        jmp escape

PRINTFUN:
        call JUMP
;IMPRESION DEL X^5
        mov ax, Unidad5
        call PrintNumb

        mov ah, 09
        lea dx, xfive
        int 21h

;IMPRESION DEL X^4
        mov ax, Unidad4
        call PrintNumb

        mov ah, 09
        lea dx, xfour
        int 21h
```

```
;IMPRESION DEL X^3
        mov ax, Unidad3
        call PrintNumb

        mov ah, 09
        lea dx, xthree
        int 21h

;IMPRESION DEL X^2
        mov ax, Unidad2
        call PrintNumb

        mov ah, 09
        lea dx, xtwo
        int 21h

;IMPRESION DEL X^1
        mov ax, Unidad1
        call PrintNumb

        mov ah, 09
        lea dx, xone
        int 21h

;IMPRESION DE LA CONSTANTE
        mov ax, Unidad0
        call PrintNumb
```

```
        jmp escape



PRINTFUNI:
        call JUMP

        mov ax,constI6
        call PrintNumb
        WriteCh '.'

        mov ax, valI5
        call PrintNumb

        mov ah, 09
        lea dx, xsix
        int 21h

;IMPRESION DEL X^5
        mov ax,constI5
        call PrintNumb
        WriteCh '.'

        mov ax, valI4
        call PrintNumb

        mov ah, 09
        lea dx, xfive
        int 21h
;IMPRESION DEL X^4
```

```
        mov ax,constI4
        call PrintNumb
        WriteCh '.'

        mov ax, valI3
        call PrintNumb

        mov ah, 09
        lea dx, xfour
        int 21h
;IMPRESION DEL X^3
        mov ax,constI33
        call PrintNumb
        WriteCh '.'

        mov ax, valI2
        call PrintNumb

        mov ah, 09
        lea dx, xthree
        int 21h

;IMPRESION DEL X^2
        mov ax, constI2
        call PrintNumb
        WriteCh '.'

        mov ax, valI1
        call PrintNumb
```

```
        mov ah, 09
        lea dx, xtwo
        int 21h
;IMPRESION DEL X^1
        mov ax , constI1
        call PrintNumb
        WriteCh '.'

        mov ax, 00
        call PrintNumb

        mov ah, 09
        lea dx, xone
        int 21h
;IMPRESION DE LA CONSTANTE
        mov dx, offset constIntegral ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos

        jmp escape

;------------------------------------------------------------  DERIVAR ------------------------------------------------------
DERFUNP:
        CALL DERFUN
        call PRINTFUND
        RET
DERFUN:
```

```
;derivamos el termino x^5
mov ax, Unidad5
mov cx, 5
imul cx
mov constD4, ax

;derivamos el termino x^4
mov ax, Unidad4
mov cx, 4
imul cx
mov constD3, ax

;derivamos el termino x^3
mov ax, Unidad3
mov cx, 3
imul cx
mov constD2, ax

;derivamos el termino x^2
mov ax, Unidad2
mov cx, 2
imul cx
mov constD1, ax

;derivamos el termino x^1
mov ax, Unidad1
mov numeroD, ax

;call PRINTFUND
```

```
                ret

;------------------------------------------------------------  INTEGRAR ------------------------------------------------
--------------------

INTFUNP:
        CALL INTFUN
        CALL PRINTFUNI
        RET
INTFUN:
;---------------------------------------------- CTE ---------------------------------------
                mov ax, Unidad0
                mov constl1,ax

;---------------------------------------------- X ---------------------------------------
                xor ax,ax
                xor bx,bx

                mov ax, 2
                mov bx, ax
                mov ax, Unidad1

                cmp ax, 0
                jns ns1
                neg ax
                mov fxF, 1
                ns1:
                idiv bx
                cmp fxF,1
                jne sig1
```

```
                neg ax
                mov fxF,0
                sig1:
                mov constI2, ax
                ;call PrintNumb

                mov valI1, dx
                call DECIMALES
;------------------------------------------------ X^2 ---------------------------------------
                xor ax,ax
                xor bx,bx

                mov ax, 3
                mov bx, ax
                mov ax, Unidad2

                cmp ax, 0
                jns ns2
                neg ax
                mov fxF, 1
                ns2:
                idiv bx
                cmp fxF,1
                jne sig2
                neg ax
                mov fxF,0
                sig2:

                mov constI33, ax
```

```
                mov valI2, dx
                call DECIMALES1
;------------------------------------------------ X^3 ----------------------------------------
                xor ax,ax
                xor bx,bx

                mov bx, 4
                mov ax, Unidad3

                cmp ax, 0
                jns ns3
                neg ax
                mov fxF, 1
                ns3:
                imul bx
                idiv bx
                idiv bx

                cmp fxF,1
                jne sig3
                neg ax
                mov fxF,0
                sig3:
                mov constI4, ax

                mov valI3, dx
                call DECIMALES2
```

```
;------------------------------------------------ X^4 ----------------------------------------
                xor ax,ax
                xor bx,bx



                mov ax, 5
                mov bx, ax
                mov ax, Unidad4

                cmp ax, 0
                jns ns4
                neg ax
                mov fxF, 1
                ns4:
                div bx
                cmp fxF,1
                jne sig4
                neg ax
                mov fxF,0
                sig4:
                mov constl5, ax
                ;call PrintNumb
                mov vall4, dx
                call DECIMALES3

;------------------------------------------------ X^5 ----------------------------------------

                xor ax,ax
                xor bx,bx
```

```
        mov ax, 6
        mov bx, ax
        mov ax, Unidad5

        cmp ax, 0
        jns ns5
        neg ax
        mov fxF, 1
        ns5:
        imul bx
        idiv bx
        idiv bx
        cmp fxF,1
        jne sig5
        neg ax
        mov fxF,0
        sig5:
        mov constI6, ax
        mov valI5, dx
        call DECIMALES4

        ret


DECIMALES:
        mov ax, valI1
        mov bx, 1000
```

```
        mul bx

        mov bx, 2
        div bx

        mov vall1, ax

        ret

DECIMALES1:
        mov ax, vall2
        mov bx, 1000
        mul bx

        mov bx, 3
        div bx

        mov vall2, ax

        ret

DECIMALES2:
        mov ax, vall3
        mov bx, 1000
        mul bx

        mov bx, 4
        div bx
```

```
        mov vall3, ax

        ret
DECIMALES3:
        mov ax, vall4
        mov bx, 1000
        mul bx

        mov bx, 5
        div bx

        mov vall4, ax

ret

DECIMALES4:
        mov ax, vall5
        mov bx, 1000
        mul bx

        mov bx, 6
        div bx

        mov vall5, ax

ret
```

```
;---------------------------------------------- SOLICITAR COEFICIENTES----------------------------------------------
------------------
COEFI:

   mov si,0

        mov ah, 00h      ;LIMPIA PANTALLA

        call CONSTANTE1 ;METODO QUE PIDE LA PRIMER CONSTANTE

        call CONSTANTE2 ;METODO QUE PIDE LA SEGUNDA CONSTANTE

        call CONSTANTE3 ;METODO QUE PIDE LA TERCER CONSTANTE

        call CONSTANTE4 ;METODO QUE PIDE LA CUARTA CONSTANTE

        call CONSTANTE5 ;METODO QUE PIDE LA QUINTA CONSTANTE

        call CONSTANTE6 ;METODO QUE PIDE LA SEXTA CONSTANTE

        jmp START



CONSTANTE1:
        call JUMP

        mov dx, offset coef ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
```

```
        int 21h ; interrupcion de microsoft ms-dos

        call READNUMB
        mov Unidad0, cx


        ret

CONSTANTE2:
        call JUMP

        mov dx, offset coVALUARX1 ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos

        call READNUMB
        mov Unidad1, cx

        ret


CONSTANTE3:
        call JUMP

        mov dx, offset coVALUARX2 ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos
```

```
        call READNUMB
        mov Unidad2, cx

        ret

CONSTANTE4:
        call JUMP

        mov dx, offset coVALUARX3 ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos

        call READNUMB
        mov Unidad3, cx

        ret

CONSTANTE5:
        call JUMP

        mov dx, offset coVALUARX4 ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos

        call READNUMB
        mov Unidad4, cx

        ret
```

```
CONSTANTE6:
        call JUMP

        mov dx, offset coVALUARX5 ;recoger valor
        mov ah, 9 ; funcion escribir una cadena
        int 21h ; interrupcion de microsoft ms-dos

        call READNUMB
        mov Unidad5, cx

        ret
;---------------------------------------------------- NEWTON ------------

NEWTON:
        mov ah, 0          ;limpiar pantalla
        mov al, 3h  ;modo texto
        int 10h

        ;pintar fondo
        ;mov ah, 6h ; funcion para scrollear la pantalla un numero de lineas
        ;mov al, 0h ; cero para borrar la pantalla
        ;mov bh, 00000000b ; fondo azul, color de texto amarillo
        ;mov ch, 0 ; x inicial
        ;mov cl, 0 ; y inicial
        ;mov dh, 24 ; x final
        ;mov dl, 79 ; y final
        ;int 10h
```

```
mov ah, 09
lea dx, newnew
int 21h

;----------------------------- metodos ---------------------
call ASK_POL ;pide polinomio

mov ah,09
lea dx, Pide_Punto ;pide punto
int 21h

call READNUMB
mov Xinicial,cx ;recibe punto

mov ah,09
lea dx, Pide_Tolerancia ; pide la tolerancia a usar
int 21h

call READNUMB
mov ToleranciaF,cx ; recibe la tolerancia

mov ah,09
lea dx, NIteracion ; pide las iteraciones
int 21h

call READNUMB
mov Iteraciones,cx ; recibe el numero de iteraciones
```

```
call derivar
jmp Iteracion

;---------------------- CALCULO DE ITERACION ----------------------
Iteracion:
                ; RESOLVER
                ; X = XN - F(XN)/F'(XN)
                call OP_FUNCT  ; Opera los datos de la funcion
                call OP_FUNCT2 ; Opera los datos de la funcion derivada

                mov ax, Xinicial ;MOVEMOS EL XINICIAL AL XN
                mov XN, ax

                mov ax, ResultadoFX ; RESULTADO DE VALUAR LA FUNCION FX
                mov cx, ResultadoFXD ; RESULTADO DE VALUAR LA FUNCION FX DERIVADA

                cmp ax,0 ;COMPARAMOS PARA COMPROBAR SIGNO
                jns NSig ;NO SIGNO
                neg ax ; NEGAR A AX SI NO TIENE SIGNO
                inc fxF ; INCREMENTAR EL FXF

                NSig: ; N SIGUIENTE ENGASADO
                cmp cx,0 ; PUT YOUR HANDS
                jns NSig2 ; UP UP UP
                neg cx ; NEGAR CX ENGASADO
                inc fxDF ; FEDEX

                NSig2: ; N SIGUIENTE ENGASADA 2
                cmp cx, 1 ; COMPARACION
```

```
je NoDividir ;NO DIVIDE
cmp cx, 0 ; COMPARACION
jz NoDividir ;NO DIVIDE
cmp ax,bx ; YEAH
jne Dividir ; DIVICION
cmp ax,0 ; COMPARACION
je NoDividir ;NO DIVIDE
mov ax, 1 ; MUEVELO MUEVELO
mov dx, 0 ; MUEVELO MUEVELO
jmp NoDividir ;NO DIVIDE

Dividir: ;DIVIDE

idiv cx ; SUPER IDIV


NoDividir: ; NO DIVIDE
mov DECIMALESSS,dx ; MUEVE A LOS DECIMALESSS

mov cx,fxF ; FEDEX
cmp cx,fxDF ; FEDEXDA
je Nposs ; N POCICION S
neg ax ; NEGACION DE AX
mov FRACCION_FFF,1 ; BANDERA DE FRAC_FRAC
Nposs: ; ; NO POS SACATELAS
mov FRACCION_NNN,ax ; FRAC NEW FRAC FRAC
mov fxF,0 ; FEDEX
mov fxDF,0 ;FEDXDA
```

```
mov ax,ResultadoFX ; RESULTADOS
mov bx,ResultadoFXD ; RESULTADOS 2
cmp ax,bx
je zFound ; SI ENCUENTRO 0
jne zNoFound ; SI NO ENCUENTRO 0
zFound:
        cmp ax,0
        jne zNoFound
        mov DECIMALESSS,0 ; SUPER DECIMALES
        jmp zOut ; SACATELAS
zNoFound:
mov ax,DECIMALESSS ; SUPER DECIMALES
mov bx,ResultadoFXD ; RESULTADO DE ESA ODNA
cmp bx,0
jnz zNob ; ZERO NOB
mov bx,1
zNob: ; ZERO NOB
cmp bx,0
jns zNoNeg ; ZERO NO NEG
neg bx
zNoNeg:

mul THOUSND ; SON COMO MIL O ALGO ASI
div bx
add cx,ax
mov DECIMALESSS , cx ; SUPER DECIMALES
mov ax,DECIMALESSS ; SUPER DECIMALES

zOut:
```

```
mov ax,Xinicial
imul THOUSND
cmp ax,0
jns RESR
neg ax
mov fxF,1

RESR:
add ax,XNRes ; LA RESPUESTA
cmp fxF,1
jne RESRR
neg ax
mov fxF,0
RESRR:
push ax

mov ax, FRACCION_NNN
imul THOUSND
cmp ax,0
jns e
neg ax
mov fxF,1
e:
add ax,DECIMALESSS ; LA RESPUESTA CON DECIMALES
cmp fxF,1
jne ee
neg ax
mov fxF,0
```

```
ee:
mov bx,FRACCION_NNN
cmp bx,0
jnz eee
cmp FRACCION_FFF,1
jne eee
neg ax
eee:
mov cx,ax
mov FRACCION_FFF,0

pop ax

sub ax,cx ; VAMOS A RESTARA WIII
mov FINAL_NUMBBBER,ax ; EL NUMERO FINAL

xor ax,ax
xor bx,bx
xor cx,cx
xor dx,dx

mov ax,FINAL_NUMBBBER
cmp ax,0
jns sins
neg ax
mov fxF,1

sins:
        idiv THOUSND
```

```
                cmp fxF,1
                jne sins2
                neg ax

            sins2:
            mov fxF,0

            mov DECIMALESSS,dx ;DECIMALESSS

            mov XN1,ax

            mov ax,DECIMALESSS


;-------------------- Imprimir Resultado de Iteracion----------;
            mov ah,09
            lea dx,IteracionN
            int 21h

            mov ax,number
            call PrintNumb

            mov ah,09
            lea dx,Xnumber
            int 21h

            mov ax,XN
            call PrintNumb
            WriteCh '.'
```

```
        mov ax,XNRes
        call PrintNumb

        mov ah,09
        lea dx,Xnumber1
        int 21h

        mov ax,Xinicial
        call PrintNumb
        WriteCh '.'
        mov ax,DECIMALESSS
        call PrintNumb

        CALL JUMP


        mov ax,XN1
mul THOUSND
cmp ax,0
jns NesNeg
neg ax
NesNeg:
add ax,DECIMALESSS
push ax

mov ax,XN
mul THOUSND
cmp ax,0
jns NesNegxn
```

```
neg ax
NesNegxn:
add ax,XNRes
mov cx,ax

pop ax
sub ax,cx
cmp ax,0
jns NLlevS
neg ax
NLlevS:

call PrintNumb

        mov ah,09
        lea dx, ErrorMe
        int 21h

CALL JUMP

mov cx, ToleranciaF
cmp ax, cx
jle  NEWTONF


inc number
mov ax,XN1
mov Xinicial,ax
mov ax,DECIMALESSS
```

```
            mov XNRes,ax
            mov ax,number
            cmp ax,Iteraciones
            je NEWTONF

            call READNUMB

            jmp  Iteracion
            ret

;-------------------------------------------------------


;readkey
jmp escape

ret

NEWTONF:
                        mov number,1

                        mov ah,09
                        lea dx, ErrorMa
                        int 21h

                        mov ah,09
                        lea dx, ResultadoNewton
                        int 21h
```

```
                    mov ax, Xinicial
                    call PrintNumb

                    ;readkey
                    jmp escape
        ret

        JUMP:
        mov dl, 10
        mov ah, 2h
        int 21h
        RET

        PRINTT:
        mov ah, 2h
        int 21h
        RET

OP_FUNCT proc near
                    ; x^5
                    mov ax,Xinicial
                    imul Xinicial
                    imul Xinicial
                    imul Xinicial
                    imul Xinicial
                    imul fx5
                    mov cx,ax
                    xor ax,ax
```

```
; x^4
mov ax,Xinicial
imul Xinicial
imul Xinicial
imul Xinicial
imul fx4
add cx,ax
xor ax,ax
; x^3
mov ax,Xinicial
imul Xinicial
imul Xinicial
imul fx3
add cx,ax
xor ax,ax
; x^2
mov ax,Xinicial
imul Xinicial
imul fx2
add cx,ax
xor ax,ax
; x^1
mov ax,Xinicial
imul fx1
add cx,ax
; x^0
mov ax, fx0
add cx, ax
mov ResultadoFX,cx
```

```
                ret
        OP_FUNCT endp

ASK_POL:

        call JUMP

        mov ah,09
        lea dx,pedir_p
        int 21h

    call JUMP

        mov ah,09
        lea dx,coVALUARX5
        int 21h

    call READNUMB
        call JUMP
    mov fx5, cx


    mov ah,09
        lea dx,coVALUARX4
        int 21h

    call READNUMB
        call JUMP
    mov fx4, cx
```

```
mov ah,09
        lea dx,coVALUARX3
        int 21h

call READNUMB
call JUMP
        mov fx3, cx

mov ah,09
        lea dx,coVALUARX2
        int 21h

call READNUMB
call JUMP
        mov fx2, cx

mov ah,09
        lea dx,coVALUARX1
        int 21h

call READNUMB
call JUMP
        mov fx1, cx

mov ah,09
        lea dx,coef
        int 21h
```

```
    call READNUMB
    call JUMP
        mov fx0, cx
ret

        OP_FUNCT2 proc near
                ;x^4
                mov ax,Xinicial
                imul Xinicial
                imul Xinicial
                imul Xinicial
                imul fxd4
                mov cx,ax
                xor ax,ax
                ;x^3
                mov ax,Xinicial
                imul Xinicial
                imul Xinicial
                imul fxd3
                add cx,ax
                xor ax,ax
                ;x^2
                mov ax,Xinicial
                imul Xinicial
                imul fxd2
                add cx,ax
                xor ax,ax
                ;x^1
                mov ax,Xinicial
```

```
                imul fxd1
                add cx,ax
                ;x^0
                mov ax, fxd0
                add cx,ax

                mov ResultadoFXD, cx
        ret
OP_FUNCT2 endp

derivar:
        ;derivamos el termino x^5
        mov ax, fx5
        mov cx, 5
        imul cx
        mov fxd4, ax

        ;derivamos el termino x^4
        mov ax, fx4
        mov cx, 4
        imul cx
        mov fxd3, ax

        ;derivamos el termino x^3
        mov ax, fx3
        mov cx, 3
        imul cx
        mov fxd2, ax
```

```
                ;derivamos el termino x^2
                mov ax, fx2
                mov cx, 2
                imul cx
                mov fxd1, ax

                ;derivamos el termino x^1
                mov ax, fx1
                mov cx, 1
                imul cx

                mov fxd0, ax
        ret


;--------------------------------- METODOS ----------------------------
        CALL JUMP

        call ASK_POL

        call pide_gx

        mov ah,09
        lea dx, Pide_Tolerancia ; pide la tolerancia a usar
        int 21h

        call READNUMB
        mov ToleranciaF,cx ; recibe la tolerancia
        call JUMP
```

```
        mov ah,09
        lea dx, Pide_Punto
        int 21h

    call READNUMB
        mov po, cx      ;VALOR DE P0
    call JUMP

        mov ah,09
        lea dx, NIteracion ; pide las iteraciones
        int 21h

        call READNUMB
        mov Iteraciones,cx ; recibe el numero de iteraciones
        call JUMP

    call ITSTEFF1

jmp escape
ret


ITSTEFF1:
    mov ax, po
    mov XINICIAL, ax

    call OP_FUNCT
    mov ax, ResultadoFX
```

```
mov p1, ax
mov ax, p1
mov XINICIAL, ax
call OP_FUNCT
mov ax, ResultadoFX
mov p2, ax

;pn+2=po-((p1-po)^2/(p2-2p1+po))

;primero realizo p1-po
mov ax, p1
mov cx, po
sub ax, cx  ;se almacena en ax

imul ax     ;GUARDANDO VALOR AX
mov bx, ax  ;GUARDANDO VALOR DX
        call PrintNumb

;P2 - 2P1 + P0
;2P1
mov ax, 2
mov cx, p1
imul cx
;P2 -2P1
mov cx, p2
sub cx, ax
;P2 - 2P1 + P0
mov ax, po
add cx, ax
```

```
        push ax
        mov ax, cx
        call PrintNumb
pop ax
mov ax, bx

        idiv cx

mov cx, po
sub cx, ax

mov p3, cx

;imprimir la iteracion
        mov ah,09
lea dx, IteracionN
int 21h

mov ax, number
call PrintNumb
call JUMP

        mov ah,09
lea dx,P0P0
int 21h

mov ax, po
        call PrintNumb
call JUMP
```

```
mov ah,09
lea dx,P1P1
int 21h

mov ax, p1
call PrintNumb

        call JUMP

mov ah,09
lea dx, P2P2
int 21h

mov ax, p2
call PrintNumb

call JUMP

        mov ah,09
lea dx,P3P3
int 21h

mov ax, p3
        call PrintNumb

call JUMP

mov ah,09
```

```
    lea dx, ErrorM
    int 21h

            mov ax, ToleranciaF
    mov ax, po
    mov cx, p3
    sub ax, cx
            call PrintNumb

    call JUMP

            mov cx, ToleranciaF
            cmp ax, cx
    jle  fin_steffensen1
    mov ah,09
    lea dx,ErrorMe
    int 21h

            mov ax,number
            cmp ax,Iteraciones
    jge  fin_steffensen1

            call READNUMB
    inc number
    jmp  ITSTEFF1
ret

fin_steffensen1:
            mov number, 1
```

```
    mov ah,09
    lea dx,ErrorMa
    int 21h

    mov ah,09
    lea dx, ResultadoNewton
    int 21h

    mov ax, p3
    call PrintNumb

            jmp escape
ret



pide_gx:
    ;antes guardo los coeficientes del polinomio
    mov ax, fx0
    mov ANT_CO0, ax
    mov ax, fx1
    mov ANT_CO1, ax
    mov ax, fx2
    mov ANT_CO2, ax
    mov ax, fx3
    mov ANT_CO3, ax
    mov ax, fx4
    mov ANT_CO4, ax
    mov ax, fx5
    mov ANT_CO5, ax
```

```
mov fx0, 0
mov fx1, 0
mov fx2, 0
mov fx3, 0
mov fx4, 0
mov fx5, 0

mov ah,09
lea dx, despeje_x
int 21h

        call jump

        mov ah,09
lea dx, coVALUARX5
int 21h

;guardo el valor ingresado en la variable correspondiente
call READNUMB
        mov fx5, cx    ;guardo el valor ingresado en coef5

        call jump

        mov ah,09
lea dx,coVALUARX4
int 21h

;guardo el valor ingresado en la variable correspondiente
```

```
call READNUMB
        mov fx4, cx    ;guardo el valor ingresado en coef4
call jump

        mov ah,09
lea dx,coVALUARX3
int 21h

;guardo el valor ingresado en la variable correspondiente
call READNUMB
mov fx3, cx    ;guardo el valor ingresado en coef3
call jump

mov ah,09
lea dx,coVALUARX2
int 21h

;guardo el valor ingresado en la variable correspondiente
call READNUMB
        mov fx2, cx    ;guardo el valor ingresado en coef2
call jump

        mov ah,09
lea dx,coef
int 21h

;guardo el valor ingresado en la variable correspondiente
call READNUMB
mov fx0, cx    ;guardo el valor ingresado en coef0
```

```
        call jump
ret

;-------------------====---------- STEFF ----------------------



        LIMPIAR: ; METODO QUE LIMPIA LA MEMORIA
                xor ax,ax
                xor bx,bx
                xor cx,cx
                xor dx,dx
        RET



;------------------------------------------------- METODOS PARA GRAFICAR LAS LINEAS DE LOS EJES X y Y --
-----------------------------------
GRAFICAR:
        call jump

        mov ah,09
        lea dx, msGrafi
        int 21h

        call READNUMB
        mov al, cl

        cmp al, 1 ; compara a 1
```

```
        je NORMAL
        jl GRAFICAR

        cmp al, 2 ; compara a 2
        je DERIVADA

        cmp al, 3 ; compara a 3
        je INTEGRAL
        jg GRAFICAR


        NORMAL:
                mov fxF, 1
                jmp GRAFICAR1
        DERIVADA:
                mov fxF, 2
                jmp GRAFICAR1
        INTEGRAL:
                mov fxF, 3

GRAFICAR1:
        mov ah, 0
        mov al, 13h ; modo de video
        int 10h

        mov x,0
        call GRAFICARX ; EJE X

        mov y,0
```

```
        call GRAFICARY ; EJE Y

                mov x,160
                mov y,100
                mov posx, 0
                mov posy, 0
                mov aux, 0

                call GRAFICARFUN1

                mov x,160
                mov y,100
                mov posx, 0
                mov posy, 0
                mov aux, 0

                call GR2

        jmp escape



;-------------------------------------------------------------------- EJE X ------------------------------------------------------
-------------

GRAFICARX:
        add x, 1 ; Suma 1 a x para poner la linea de las xs
        mov y, 100 ; poner la posicion y relativa a la que queremos acceder
        call CALCULAR ; generar posiciones absolutas
```

```
        call FONDO ; dibujar un cuadro de 8x8 en las posiciones absolutas

        cmp x,320 ;Compara si es igual a 320
        jbe GRAFICARX ; si es menor o igual regresa

        ret

;------------------------------------------------------------------- EJE Y ----------------------------------------------------
-------------

GRAFICARY:
        mov x, 160 ; poner la posicion x relativa a la que queremos acceder
        add y, 1 ; Suma 1 a y para hacer la linea de las ys
        call CALCULAR ; generar posiciones absolutas
        call FONDO ; dibujar un cuadro de 8x8 en las posiciones absolutas

        cmp y,200 ;Compara si es igual a 200
        jbe GRAFICARY ; si es menor o igual regresa

        ret

;------------------------------------------------------------------- FUNCION ----------------------------------------------------
------------------

GRAFICARFUN1:
        xor ax,ax
        mov ax, posy
        mov aux, ax
```

```
        mov posy,0
        call VALUARCONST
        call VALUARX1
        call VALUARX2
        call VALUARX3
        call VALUARX4
        call VALUARX5

        mov ax, posy
        ;call PrintNumb

        call SUMAY

        add posx,1
        add x, 1

        jmp VERIFICAR

        ret

VERIFICAR PROC
        cmp y,0
        js nada2
        cmp y,200
        ja nada2
        cmp x,0
        js nada2
        cmp x,360
        ja nada2
```

```
        jmp GRAFICARFUN1

        nada2:
        RET

VERIFICAR ENDP

SUMAY PROC
        A1:
                call VERIFICARP
                xor ax,ax
                mov ax, posy
                cmp ax, aux
                jg B
                jl CC
        B:
                add aux,1
                cmp aux, ax
                jl AA1
                jg SA
        AA1:
                sub y, 1
                jmp A1
        CC:
                sub aux,1
                cmp aux, ax
                jg AB
```

```
            jl SA
        AB:
            add y, 1
            jmp A1
        SA:
            ret
SUMAY ENDP

VERIFICARP PROC
        cmp y,0
        js nada1
        cmp y,200
        ja nada1
        cmp x,0
        js nada1
        cmp x,320
        ja nada1
        cmp y,100
        je nada1
        cmp x,160
        je nada1

            call CALCULAR1 ; generar posiciones absolutas
            call FONDO1 ; dibujar un cuadro de 8x8 en las posiciones absolutas

        nada1:
        RET

VERIFICARP ENDP
```

```
VERIFICAR1 PROC
        cmp y,0
        js nada1
        cmp y,200
        ja nada1
        cmp x,0
        js nada1
        cmp x,320
        ja nada1

        jmp GR2

        nada11:
        RET

VERIFICAR1 ENDP

GR2:
        xor ax,ax
        mov ax, posy
        mov aux, ax

        mov posy,0
        call VALUARCONST
        call VALUARX1
        call VALUARX2
        call VALUARX3
        call VALUARX4
```

```
        call VALUARX5

        call SUMAY

        sub x, 1
        sub posx,1

        CALL VERIFICAR1

        ret

VALUARCONST:
            cmp fxF, 1
            je cmUni0
            cmp fxF, 2
            je cmUniD0
            cmp fxF, 3
            je cmUniI0
            cmUni0:
            mov ax,Unidad0
            jmp fin0
            cmUniD0:
            mov ax, numeroD
            jmp fin0
            cmUniI0:
            mov ax, constI1
```

```
                mov bx, posx
                imul bx

                jmp fin0

                fin0:
                add posy, ax

                ret

VALUARX1:
                cmp fxF, 1
                je cmUni1
                cmp fxF, 2
                je cmUniD1
                cmp fxF, 3
                je cmUniI1
                cmUni1:
                mov ax,Unidad1
                jmp fin1
                cmUniD1:
                mov ax, constD1
                jmp fin1
                cmUniI1:
                mov ax, constI2
                mov bx, posx
                imul bx

                jmp fin1
```

```
        fin1:
        mov bx, posx
        imul bx

        add posy, ax

            ret

VALUARX2:
        cmp fxF, 1
        je cmUni2
        cmp fxF, 2
        je cmUniD2
        cmp fxF, 3
        je cmUniI2
        cmUni2:
        mov ax,Unidad2
        jmp fin2
        cmUniD2:
        mov ax, constD2
        jmp fin2
        cmUniI2:
        mov ax, constI3
        mov bx, posx
        imul bx

        jmp fin2
```

```
        fin2:
                mov bx, posx
                imul bx
                imul bx
                add posy,ax
                ret

VALUARX3:
        cmp fxF, 1
        je cmUni3
        cmp fxF, 2
        je cmUniD3
        cmp fxF, 3
        je cmUniI3
        cmUni3:
        mov ax,Unidad3
        jmp fin3
        cmUniD3:
        mov ax, constD3
        jmp fin3
        cmUniI3:
        mov ax, constI4
        mov bx, posx
        imul bx

        jmp fin3

        fin3:
                mov bx, posx
```

```
                        imul bx
                        imul bx
                        imul bx


                        add posy,ax
        ret

READNUMB    proc    near
    push    dx
    push    ax
    push    si

    mov     cx, 0

    mov     cs:DO_SIGN, 0

N_DIG:

    cmp si,1
    ja STOP_E

    mov     ah, 00h
    int     16h

    mov     ah, 0eh
    int     10h
```

```
        cmp     al, '-'
        je      PUT_SIGN

        cmp     al, 0dh
        jne     N_CR
        jmp     STOP_E
N_CR:

        inc si

        cmp     al, 8
        jne     VER_RET
        mov     dx, 0
        mov     ax, cx
        div     DIEZ
        mov     cx, ax
        WriteCh   ' '
        WriteCh   8
        jmp     N_DIG
VER_RET:
        cmp     al, '0'
        jae     OK_00
        jmp     soloDigitos
OK_00:
        cmp     al, '9'
        jbe     OK_DDGI
soloDigitos:
        WriteCh   8
```

```
        WriteCh   ' '
        WriteCh   8
        jmp     N_DIG
OK_DDGI:


        push    ax
        mov     ax, cx
        mul     DIEZ
        mov     cx, ax
        pop     ax

        cmp     dx, 0
        jne     LOOK_SZ

        sub     al, 30h

        mov     ah, 0
        mov     dx, cx
        add     cx, ax
        jc      LOOK_SZ2

        jmp     N_DIG

PUT_SIGN:
        mov     cs:DO_SIGN, 1
        jmp     N_DIG

LOOK_SZ2:
```

```
        mov     cx, dx
        mov     dx, 0
LOOK_SZ:
        mov     ax, cx
        div     DIEZ
        mov     cx, ax
        WriteCh    8
        WriteCh    ' '
        WriteCh    8
        jmp     N_DIG



STOP_E:
        cmp     cs:DO_SIGN, 0
        je      sin_signo
        neg     cx
sin_signo:

        pop     si
        pop     ax
        pop     dx
        ret
DO_SIGN     db     ?
READNUMB       endp

VALUARX4:
                    cmp fxF, 1
                    je cmUni4
                    cmp fxF, 2
```

```
                je cmUniD4
                cmp fxF, 3
                je cmUniI4
                cmUni4:
                mov ax,Unidad4
                jmp fin4
                cmUniD4:
                mov ax, constD4
                jmp fin4
                cmUniI4:
                mov ax, constI5
                mov bx, posx
                imul bx

                jmp fin4

                fin4:
                        mov bx,posx
                        imul bx
                        imul bx
                        imul bx
                        imul bx

                        add posy ,ax

                        ret

VALUARX5:
                cmp fxF, 1
```

```
je cmUni5
cmp fxF, 2
je cmUniD5
cmp fxF, 3
je cmUniI5
cmUni5:
mov ax,Unidad5
jmp fin5
cmUniD5:
mov ax, 0
jmp fin5
cmUniI5:
mov ax, constI6
cmp ax, 0
je decc5
jne nodecc5
decc5:
        cmp valI6, 500
        jb nodecc5
mov ax, 1
nodecc5:

mov bx, posx
imul bx

jmp fin5

fin5:
        mov bx,posx
```

```
                imul bx
                imul bx
                imul bx
                imul bx
                imul bx

                add posy,ax

                ret

PrintNumb   proc  near
    ;push  dx
    push  ax

    cmp   ax, 0
    jnz   SEE_Z

    WriteCh  '0'
    jmp   PRINTTT

SEE_Z:
    cmp   ax, 0
    jns   POSTS
    neg   ax

    WriteCh  '-'
            jmp POSTS1
POSTS:
            WriteCh  '+'
```

```
POSTS1:
    call   PRINTT_NS
PRINTTT:
    pop    ax
    ;pop    dx
    ret
PrintNumb     endp

PRINTT_NS  proc   near
    push   ax
    push   bx
    push   cx
    push   dx

    mov    cx, 1

    mov    bx, 10000

    cmp    ax, 0
    jz     PRINTT_Z

STRT:

    cmp    bx,0
    jz     ENDPRINTT

    cmp    cx, 0
    je     CLCL
```

```
    cmp     ax, bx
    jb      JUMPJUMP
CLCL:
    mov     cx, 0

    mov     dx, 0
    div     bx

    add     al, 30h
    WriteCh    al

    mov     ax, dx

JUMPJUMP:
    push    ax
    mov     dx, 0
    mov     ax, bx
    div     DIEZ
    mov     bx, ax
    pop     ax

    jmp     STRT

PRINTT_Z:
                WriteCh    '0'

ENDPRINTT:

    pop     dx
```

```
    pop     cx
    pop     bx
    pop     ax
    ret
PRINTT_NS   endp
; ------------------------------------------------------        COMIENZAN METODOS DE GRAFICACION ----------
---------------------------------------
CALCULAR:
        mov ax, x
        mul xcc
        mov xr, ax

        mov ax, y
        mul xcc
        mov yr, ax

        mov j,0
        ret

;imprimir un cuadro de tamanio xcc
FONDO:
        mov i,0
        mov si,xcc
        cmp j, si
        jb CALCULAR2
        ret

; segunda parte del ciclo fondo
CALCULAR2:
```

```
        mov ah, 0ch ; Instruccion para cambiar el color a un pixel
        mov al, 12 ; color
        mov bh, 0 ;pagina
        mov si, i ;temporal si a i
        add xr, si ; sumarle i a x
        mov si, j ;tempral si a j
        add yr, si ; sumarle j a y
        mov cx, xr ; poner posicion x
        mov dx, yr ; poner posicion y
        mov si, i ; temporal si a i
        sub xr, si ; restarle i a x
        mov si, j ; tempral si a j
        sub yr, si ; restarle j a j
        int 10h ;interrupcion
        inc i
        mov si, xcc
        cmp i, si
        jb CALCULAR2
        inc j
        jmp FONDO

        ;readkey
        jmp ESCAPE

CALCULAR1:
        mov ax, x
        mul xcc
        mov xr, ax
```

```
        mov ax, y
        mul xcc
        mov yr, ax

        mov j,0
        ret

;imprimir un cuadro de tamanio xcc
FONDO1:
        mov i,0
        mov si,xcc
        cmp j, si
        jb CALCULAR3
        ret

; segunda parte del ciclo fondo
CALCULAR3:
        mov ah, 0ch ; Instruccion para cambiar el color a un pixel
        mov al, 03 ; color
        mov bh, 0 ;pagina
        mov si, i ;temporal si a i
        add xr, si ; sumarle i a x
        mov si, j ;tempral si a j
        add yr, si ; sumarle j a y
        mov cx, xr ; poner posicion x
        mov dx, yr ; poner posicion y
        mov si, i ; temporal si a i
        sub xr, si ; restarle i a x
        mov si, j ; tempral si a j
```

```
        sub yr, si ; restarle j a j
        int 10h ;interrupcion
        inc i
        mov si, xcc
        cmp i, si
        jb CALCULAR2
        inc j
        jmp FONDO

        ;readkey
        jmp ESCAPE

; ------------------------------------------------------  FINALIZAN METODOS DE GRAFICACION ----------
-----------------------------------

ESCAPE:
        mov ah,01h
        int 21h
        ;cmp al, 1Bh ; hex la tecla ENTER
        jmp START
        ;jmp ESCAPE

STOP:
                mov AX, 4C00H
                int 21H


CODE ENDS
        END START
```