# Manual tecnico

**Struc func:**

Para implementar la funcionalidad de un poliminomio se usaron dos estrucutras.
TERM: tiene cada coeficiente de x. Entonces sus atributos serian:
exp
num
dem
FUNC: tiene 6 terminos

```
;No se si conviene tener todos los coeficientes como byte
;bytes o words para usar el fpu facil, si lo alimiamos a
;fueran todos words
TERM struct 2
    coefNum sword 0
    coefDem sword 1
    exp sword 0
TERM ends

FUNC struct 2
    terms TERM { }, { }, { }, { }, { }, { }
FUNC ends
```

Se agregaron funciones relacionadas a ala estructura. Cada una de ellas recibia un puntero a la estructura en la que se iba a ejecutar el procedimiento. A todas estas funciones se envolvieron en macros que facilitaban su llamada apilando los parametros necesarios en el orden especificado por la funcion y protegiendo los registros haciendo un pusha y un popa antes de llemar a cada funcion.

**funcTryAddTerm:**

Tatra de agregar un termino a la funcion, espera recibir el string ingresado por el usuario, tiene que seguir, si dicho string no sigue la expresion regular: ***-?[0-9]*** no cambia la funcion.

Retorna ax 1 si agrego termino, ax 2 si no

```
mFuncTryAddTerm macro func, index, grado, userInput
    pushaButAx

    lea ax, userInput
    push ax
    mov ax, grado
    push ax
    mov ax, index
    push ax
    lea ax, func
    push ax
    call funcTryAddTerm

    popaButAx
endm
```

**funcToString:**

Escribe en buffer una representacion en string de la funcion

```
mFuncToString macro func, buffer
    pusha

    lea ax, buffer
    push ax
    lea ax, func
    push ax
    call funcToString

    popa
endm
```

**funcDerivate:**

Recibe dos funciones, una contiene la funcion a derivar y en la otra se retorna la funcion resultante

```
mFuncDerivate macro destFunc, sourceFunc
    pusha

    lea ax, sourceFunc
    push ax
    lea ax, destFunc
    push ax
    call funcDerivate

    popa
endm
```

**funcIntegrate**

Recibe dos funciones, una contiene la funcion a integrar y en la otra se retorna la funcion resultante

```
mFuncIntegrate macro destFunc, sourceFunc
    pusha

    lea ax, sourceFunc
    push ax
    lea ax, destFunc
    push ax
    call funcIntegrate

    popa
endm
```

**funcGraph**

Recibe el limete inferior de x y el limite superior de x, dibuja todos los puntos entre esos dos limites de la funcion. Asume que el modo video esta inicializado y que el reg es contiene la memoria de video

```
mFuncGraph macro func, lowerBound, upperBound
    pusha

    mov ax, upperBound
    push ax
    mov ax, lowerBound
    push ax
    lea ax, func
    push ax
    call funcGraph

    popa
endm
```

**File:**

openFile

Abre el archivo con direccion path

```
;Devuelbe en ax el handler, si ocurrio error flag carry esta en 1
openFile macro path
    mov ah, 3ch
    xor cx, cx
    lea dx, path
    int 21h
endm
```

writeContent

Escribe un string con tamano _size en el archivo con el handler handler

```
;returns: cx numero de bytes escritos, carry flag verdadera si ocurrio error
writeContent macro handler, _size, content
    mov bx, handler
    mov ah, 40h
    mov cx, _size
    lea dx, content
    int 21h
endm
```

closeFile

```
;carry flag verdadera si ocurrio error
closeFile macro handler
    mov ah, 3eh
    mov bx, handler ;bx todavia contiene el handler
    int 21h
endm
```

newFile

```
;retorna en ax el handler del nuevo archivo, carry flag si ocu
newFile macro path
LOCAL End
    ;makeFile
    mov ah, 3ch
    xor cx, cx
    lea dx, path
    int 21h
endm
```