

30-11-2022

# Proyecto integrador - Calculadora

Seminario de traductores de lenguajes

I



Integrantes:

Victor Leonardo Valle Guerra

David Isaac de la Cruz Castillo

PROFESOR: PATIÑO RUIZ, ROBERTO – SECCIÓN: D12

CARRERA: LIC. EN INGENIERÍA EN COMPUTACIÓN

## Proyecto integrador - calculadora

### Resumen

Para poder demostrar los conocimientos y habilidades requeridas a lo largo del curso, se ha planteado el desarrollo de una calculadora en ensamblador que pueda realizar operaciones aritméticas, así como trigonométricas, de forma que se haga uso de las interrupciones adecuadas para poder desempeñar dichas operaciones, así como las estrategias adecuadas para poder implementar un menú desde el cual seleccionar las diversas opciones que puede manejar la calculadora.

Para ello se hizo el programa completamente en ensamblador para la arquitectura x86, usando emu8086 y DOSBOX como entornos de prueba para verificar su funcionamiento correcto y esperado por parte del usuario.

A lo largo del diseño y desarrollo del proyecto se plantearon diversas estrategias para poder hacer el código lo más compacto posible, evitando repetir procedimientos, rutinas o segmentos de código que podrían ser reutilizados a lo largo de las diversas funciones, pudiendo de esta forma analizar que comportamiento y funciones compartían en común cada una de las funciones planeadas a integrar en la calculadora.

Este proceso de reducción aplicó para el diseño en general del programa, para cada función de cada una de las operaciones aritméticas, así como las funciones auxiliares para poder capturar los números o entradas del usuario a usar para las operaciones, entre otras funciones auxiliares tales como aquellas para checar signo, limpiar pantalla, limpiar variables, realizar el cálculo de complemento a dos, entre otras tantas.

El programa de igual manera permite poder graficar operaciones trigonométricas básicas tales como el seno, coseno y la tangente, para que puedan ser visualizadas en la pantalla.

### Problemática

Se requiere de tal manera que se desarrolle una calculadora capaz de realizar las operaciones aritméticas básicas con dos operandos, tales como suma, resta, división, multiplicación y potencia, así como operaciones trigonométricas básicas de forma gráfica tales como seno, coseno y tangente.

### Justificación y uso de resultados

Las decisiones de diseño tomadas para el desarrollo de este programa toman como base los mismos principios y buenas prácticas aplicadas a la programación en lenguajes de programación de alto nivel, para poder mantener el código simple, legible, compacto, lo menos redundante posible, y sobre todo abierto a extensión para futuros cambios con el menor número de complicaciones posibles en el caso de querer escalarlo.

De esta forma además se pudo integrar las funciones e implementaciones para los requerimientos señalados inicialmente, de esta forma pudimos observar la viabilidad de nuestros resultados especulados en base al diseño y visión concebidos al principio del desarrollo del proyecto.

### Fundamento teórico

Para poder realizar el desarrollo del programa planteamos de antemano el colocar los mensajes, la entrada y salida del programa y variables para los escenarios relacionados con la evaluación del

signo, así como otras variables auxiliares para el conteo y reinicio de caracteres de las entradas y salidas del programa dentro del segmento de datos.

Para el segmento de código planteamos y diseñamos 7 funciones, entre ellas, funciones para pedir dos variables numéricas al usuario, donde internamente usan la función de captura de valor desde teclado, así como funciones para poder checar signo en caso de ser necesario antes o después de realizar una operación aritmética, así como la función para poder realizar el cálculo del complemento a dos, de igual manera se contemplaron los procedimientos necesarios para poder limpiar pantalla, poder limpiar variables, imprimir mensajes y resultados, así como poder realizar la graficación de las operaciones trigonométricas.

De forma que el programa se componga de un bloque principal de instrucciones donde se imprima el mensaje del menú, capture la opción del usuario desde el teclado, para posteriormente evaluar opción y dirigir a la llamada de la función relacionada, ya sea una operación aritmética o una operación trigonométrica.

De forma que podemos usar las interrupciones 21H para funciones de impresión de mensajes, espera de lectura o de tecla pulsada, finalización de programa, leer un solo carácter, entre otras tantas funciones. De igual manera podemos hacer uso de las interrupciones 13H para poder cambiar el formato del modo video, y la interrupción 10H para poder dibujar en pantalla de igual manera.

### Objetivo

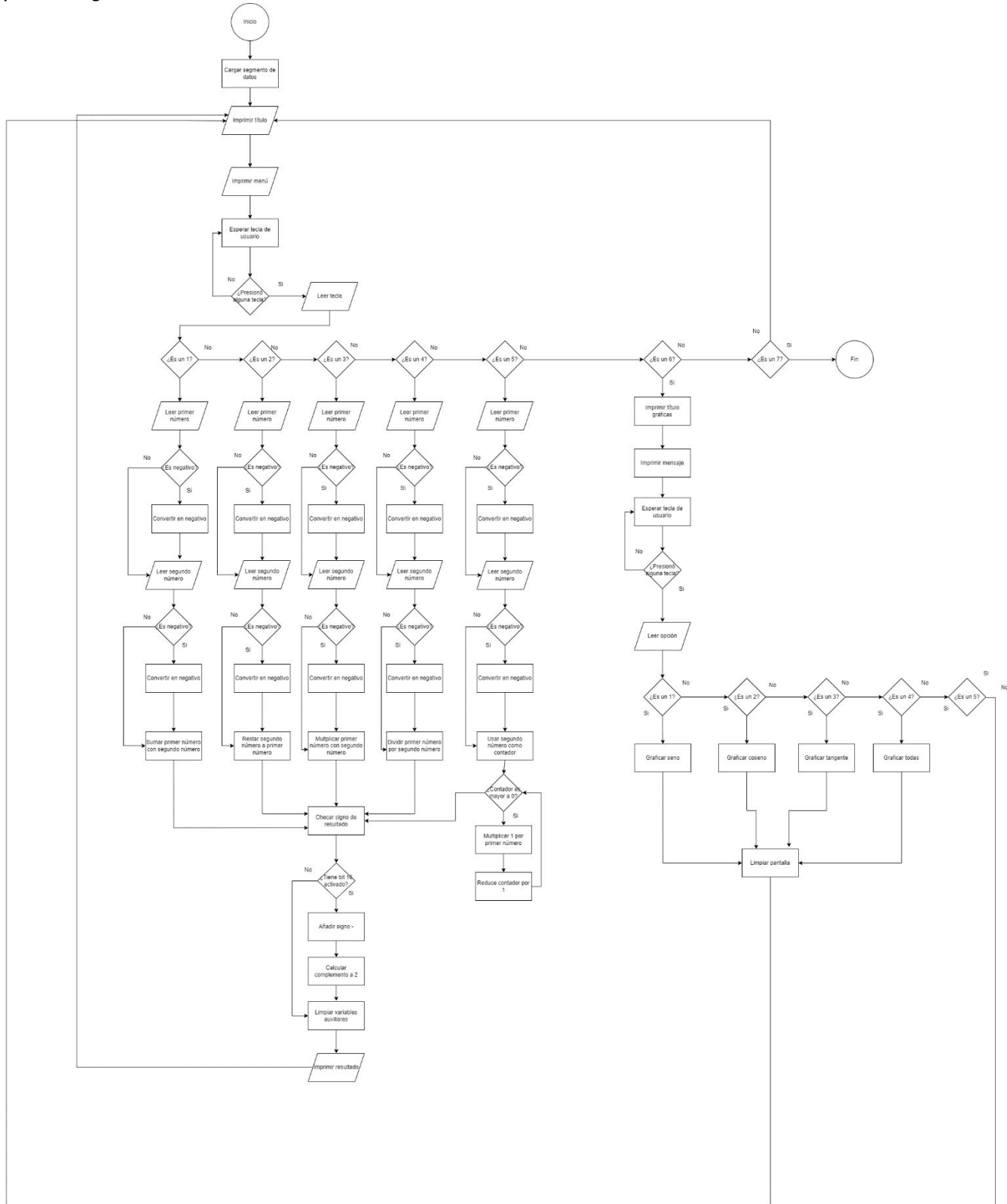
Desarrollar un programa que permita realizar operaciones aritméticas con dos operandos, así como operaciones trigonométricas básicas de forma visual, que además cumpla con los siguientes puntos

- Legibilidad del código
- Escalabilidad
- Capacidad de ser mantenido de forma sencilla
- Compacto o reducido en redundancias
- Simplicidad

### Desarrollo

De igual manera a la forma planteada durante el fundamento teórico, la fase de planeación y posteriores fases de diseño y de desarrollo del programa.

Planteamos un flujograma para poder definir el flujo del programa a través de los diversos escenarios posibles y formas en las que el usuario podría usar la aplicación de forma que también fuera más fácil poder realizar la implementación en código y poder en estar reflejar el flujo del programa de forma deseada.



De manera que realizamos el programa de forma que la sección de datos tiene las variables necesarias para poder desplegar los mensajes al iniciar el programa, el menú, el mensaje de

despedida, así como variables para realizar ciertas operaciones y chequeos respecto las entradas, salidas y su tamaño al comienzo y el final de cada operación seleccionada a realizar.

El programa se compone de un bloque inicial donde se imprime el mensaje del menú, se captura la entrada del usuario para poder comparar y determinar la opción seleccionada por el usuario y de esta manera poder llamar a la función o procedimiento principal que encapsula dicho comportamiento de la operación deseada a realizar.

Por lo tanto, se tiene 7 métodos o procedimientos principales que permiten realizar dichas operaciones, funciones para pedir dos variables numéricas al usuario, donde internamente usan la función de captura de valor desde teclado, así como funciones para poder checar signo en caso de ser necesario antes o después de realizar una operación aritmética, así como la función para poder realizar el cálculo del complemento a dos, de igual manera se contemplaron los procedimientos necesarios para poder limpiar pantalla, poder limpiar variables, imprimir mensajes y resultados, así como poder realizar la graficación de las operaciones trigonométricas.

.MODEL SMALL

.STACK 128

```
;-----  
;----- Segmento de Datos -----  
;-----
```

.DATA

;Mensajes a mostrar

titulo DB 13,10,'--- Calculadora en ensamblador ---',13,10,13,10,'|-- David Isaac De la Cruz Castillo |--',13,10,'|-- Victor Leonardo Valle Guerra --|',13,10,'\$'

mensajeMenu DB '-1. A + B',13,10,'-2. A - B',13,10,'-3. A \* B',13,10,'-4. A / B',13,10,'-5. A ^ B',13,10,'-6. Graficas Trigonometricas',13,10,'-7. Salir',13,10,'\$'

tituloGraf DB 13,10,'--- Selecciona la operacion a graficar ---',13,10,13,10,'\$'

mensajeGraficas DB '-1. Graficar Seno',13,10,'-2. Graficar Coseno',13,10,'-3. Graficar Tangente',13,10,'-4. Todas las anteriores',13,10,'-5. Regresar al menu',13,10,'\$'

msjPedirNumero DB 0ah,0dh,'Escribe un numero: ', '\$'

msjSaltos DB 0ah,0dh,',',13,10,13,10,13,10,'\$'

msjResultado DB 0ah,0dh,',',13,10,'El resultado de la operacion es: ', '\$'

msjFin DB 0ah,0dh,',',13,10,'Gracias por Usar mi programita pongame 100 - :D ',13,10,13,10,'\$'

msjSigno DB '', '\$'

msjSalir DB 'Presione una tecla para continuar... ', '\$'

color DB ?

Aux\_strNum DB 5 dup(' '), '\$' ;Aux\_strNumiliar

reset DB '' ;caracter para reinicio

COUNT DB 0 ;contador de caracteres

COUNT2 DB 0 ;contador de caracteres

COUNT3 DB 0 ;contador de caracteres

posValues DB 1,10,100 ;Variable para posibles valores a ingresar (3 digitos)

posRes DW 1,10,100, 1000, 10000 ;Variable para posibles valores a retornar (5 digitos)

flagRes DB 0 ;Variable para no mostrar 0 a la izquierda

Aux\_intNum DW 0 ;Variable auxiliar para manejar valores numericos

op1\_intNum DW 0 ;Variable para operando 1

op2\_intNum DW 0 ;Variable para operando 2

```

;Seno
EJEY DW
225,227,230,232,234,236,238,240,242,244,246,247,249,251,252,253,254,255,256,257,257,258,258,258,258,258,258,258
,257,257,256,255,254,253,252,251,249,248,246,245,243,241,239,237,235,232,231,229,227,226,224,222,220,218,216,214,21
1,209,207,206,204,202,201,199,198,197,196,195,194,193,193,192,192,192,192,192,192,192,193,193,194,195,196,197,198,1
99,201,202,204,205,207,209,211,213,215,218,219,221,223,224

```

```

;Coseno
EJEY2 DW
258,258,258,258,257,257,256,255,254,253,251,250,248,246,245,243,242,240,237,235,233,231,230,228,226,225,224,222,220
,218,216,214,211,209,207,206,204,202,201,199,198,197,196,195,194,193,193,192,192,192,192,192,192,192,193,193,194,19
5,196,197,198,199,201,202,204,205,207,209,211,213,215,218,219,221,223,224,226,228,230,232,234,236,239,241,243,244,2
46,248,249,251,252,253,254,255,256,257,257,258,258,258,258

```

;Tangente  
EJEY3 DW 225,227,230,232,234,236,239,242,245,247,251,255,260,265,269,276,283,293,308, 315  
,328,359,396,450,5,15,81,109,133,150,162,172,177,184,189,194,198,201,204,207,209,212,215,217,219,221,223,224,225,226,  
228,230,233,235,237,240,243,246,248,252,256,261,266,271,278,285,296,312,334,340,350,360,370,380,400,410,450,15,81,1  
09,133,150,162,172,177,184,189,194,198,201,204,207,209,212,215,217,219,221,223,224,225

COUN DW 0 ;Contador de ejeX  
COUN2 DW 450 ;Contador de ejeY  
OPCGrafs DB 0 ;Variable de opciones para graficas

-----  
----- Segmento deCodigo -----  
-----  
-----

MOV DS,AX

```
CALL limpiarPantalla
LEA DX,titulo ;imprimir el mensajeMenu de titulo de nuestro programa
MOV AH,9h
INT 21h
```

```
LEA DX,mensajeMenu ;imprimir mensajeMenu de menu con opciones
MOV AH,9h
INT 21h
```

Switch de opciones para menu

```
MOV AH,01h      ; pausa y espera a que el usuario precione una tecla
INT 21h         ; interrupcion para capturar
CMP AL,49       ; SI la entrada es 1 en ascci
JE funcion_Suma ; brincar a nuestra funcion para hacer Suma
CMP AL,50       ; SI es un 2
JE funcion_Resta ; brincar a nuestra funcion para hacer Division
CMP AL,51       ; SI es un 3
JE funcion_Mul  ; brincar a nuestra funcion para hacer Multiplicacion
CMP AL,52       ; SI es un 4
JE funcion_Div  ; brincar a nuestra funcion para hacer Division
CMP AL,53       ; SI es un 5
JE funcion_Pot  ; brincar a nuestra funcion para hacer Potencia
```

```
CMP AL,54      ; SI es un 6
JE funcion_Graficas ; brincar a nuestra funcion para hacer Graficas
CMP AL,55      ; SI es un 7
JE finPrograma  ; brincar a fin del programa
CALL hacer3SaltosTexto
JMP menu        ; SI es cualquier otra tecla repetir el despliegue en pantalla
```

```
;-----
;-----  Declaracion de funciones  -----
;-----
```

funcion\_Suma:

```
CALL seleccionarColor
CALL pedir2NumUser

MOV AX, op1_intNum
ADD AX, op2_intNum

CALL checarSigno
CALL mostrarResultado
CALL systemPause
CALL hacer3SaltosTexto
JMP menu
```

funcion\_Resta:

```
CALL seleccionarColor
CALL pedir2NumUser

MOV AX, op1_intNum
SUB AX, op2_intNum

CALL checarSigno
CALL mostrarResultado
CALL systemPause
CALL hacer3SaltosTexto
JMP menu
```

funcion\_Mul:

```
CALL seleccionarColor
CALL pedir2NumUser

MOV AX, op1_intNum
IMUL op2_intNum

CALL checarSigno
CALL mostrarResultado
CALL systemPause
CALL hacer3SaltosTexto
JMP menu
```

funcion\_Div:

```
CALL seleccionarColor
CALL pedir2NumUser

MOV AX, op1_intNum
IDIV op2_intNum
```

*Proyecto integrador – Calculadora – Victor Leonardo Valle Guerra & David Isaac de la Cruz Castillo*

```
CALL checarSigno
CALL mostrarResultado
CALL systemPause
CALL hacer3SaltosTexto
JMP menu
```

funcion\_Pot:

```
CALL seleccionarColor
CALL pedir2NumUser
```

```
MOV AX, 1
MOV CX, op2_intNum
```

potenciar:

```
MUL op1_intNum
loop potenciar
```

```
CALL checarSigno
CALL mostrarResultado
CALL systemPause
CALL systemPause
CALL hacer3SaltosTexto
JMP menu
```

finPrograma:

```
CALL hacer3SaltosTexto
MOV AH,09
MOV DX,offset msjFin      ;Imprime mensajeMenu de fin de programa
INT 21h
MOV AH,04ch               ;Terminamos programa con INT21H
INT 21h
```

funcion\_Graficas:

```
CALL limpiarPantalla
LEA DX,tituloGraf      ;imprimir el titulo de la seccion de graficas
MOV AH,9h
INT 21h
```

```
LEA DX,mensajeGraficas ;imprimir mensajeMenu de menu con opciones para graficas
MOV AH,9h
INT 21h
```

```
;-----
;----- Switch de opciones para Graficas -----
;-----
MOV AH,01H      ;pausa y espera a que el usuario precione una tecla
INT 21h         ;interrupcion para capturar
MOV OPCGrafs, AL
CMP AL, 49      ; SI es un 1
JZ llamarGraficas
CMP AL, 50      ; SI es un 2
JZ llamarGraficas
CMP AL, 51      ; SI es un 3
JZ llamarGraficas
CMP AL, 52      ; SI es un 4
```



```
JZ llamarGraficas
CMP AL,53      ; SI es un 5
JE salir_Graficas ; brincar a nuestra funcion para hacer Potencia
```

```
regresoGraficas:
CALL systemPause
CALL hacer3SaltosTexto
MOV AH, 0
mov aL,03h      ;Funcion para regresar a modo de texto
int 10h ;Llamamos a la INT 10h
JMP funcion_Graficas ; SI es cualquier otra tecla repetir el despliegue en pantalla
```

```
llamarGraficas:
CALL graficarTrigo
JMP regresoGraficas
```

```
salir_Graficas:
JMP menu
```

```
;-----
;----- Funcion para pedir numeros al usuario -----
;
```

pedirNumero PROC NEAR

```
CALL limpiarVariables
```

```
MOV AH,09
MOV DX,offset msjPedirNumero ;Imprimimos el msjPedirNumero
INT 21h
```

```
LEA SI,Aux_strNum ;Cargamos en el registro SI AL primer Aux_strNumiliar
```

getNumbers:

```
INC COUNT ;Incrementamos contador por cada caracter ingresado
MOV AH,01h ;Pedimos un caracter
INT 21h
MOV [SI],AL ;Se guarda en el registro indexado AL Aux_strNumtor
INC SI
CMP AL,0Dh ;Se cicla hasta que se digite un Enter
JZ salirGetNumbersEnter
CMP COUNT,03H ;o se hayan ingresado 3 caracteres
JZ salirGetNumbers
```

```
JMP getNumbers ;SI no, se seguira ciclando
```

salirGetNumbersEnter:

```
DEC COUNT ;Si se presiono enter decrementamos el contador para no leer el enter
```

salirGetNumbers:

```
MOV AX, 0
MOV BX, 0
MOV CX, 0
MOV DX, 0
```

*Proyecto integrador – Calculadora – Víctor Leonardo Valle Guerra & David Isaac de la Cruz Castillo*

```
LEA SI,Aux_strNum    ;cargamos en SI la cadena que contiene vec
MOV AH,0
MOV AL,[SI]+BX       ;Obtenemos caracter de la derecha
CMP AL, 2DH          ;Checamos si es un guion bajo es decir un negativo
JZ esNegativo?       ;Si si es convertimos el numero a complemento a 2
MOV AX, 0             ;Sino regresamos los valores a predeterminado
MOV BX, 0
```

```
getValues:           ;Obtenemos valor de cada caracter
LEA SI,Aux_strNum    ;cargamos en SI la cadena que contiene vec
MOV AH,0
MOV AL,[SI]+BX       ;Obtenemos caracter de la derecha
SUB AL,30H           ;Le restamos 30 para obtener su valor decimal
DEC COUNT
MOV DL, COUNT
MOV SI, DX
MOV CH, 0
MOV CL, posValues[SI] ;Se obtiene su valor segun la posicion (Desenas, centenas, etc)
MUL CX               ;Se multiplica el caracter por su respectivo valor de posicion
ADD Aux_intNum, AX   ;Se va sumando los valores a nuestra variable auxiliar
INC BX
CMP COUNT,0H        ;Ciclamos segun al cantidad de caracteres ingresados
JZ salir
JMP getValues
```

```
salir:
CMP flagRes, 1       ;Checamos si el numero ingresado es negativo
JZ hacerNegativo     ;Si si es se hace negativo
RET                  ;sino, regresamos el procedimiento de donde lo llamaron.
```

```
esNegativo?:
MOV flagRes, 1
DEC COUNT
INC BX
JMP getValues
```

```
hacerNegativo:
MOV AX,Aux_intNum
NEG AX              ;Ponemos el numero en complemento a 2
MOV Aux_intNum, AX
RET
```

pedirNumero ENDP

```
;-----
;----- Reiniciamos nuestras variables -----
;-----
limpiarVariables PROC NEAR
```

```
MOV COUNT, 0
MOV COUNT2, 0
MOV COUNT3, 0
MOV flagRes, 0
MOV Aux_intNum, 0
reinicioAux:      ;Reiniciar el auxiliar de caracteres
MOV CX, 5         ;Caracteres a limpiar
LEA BX, Aux_strNum
MOV DL, reset
```

reinicioChar:

```
MOV [BX], DL    ;Sustituir caracteres por espacios
INC BX
loop reinicioChar
```

```
RET            ;regresamos el procedimiento de donde lo llamaron.
```

limpiarVariables ENDP

```
;-----
;----- Convertimos de Decimal a ASCII -----
;-----
```

mostrarResultado PROC NEAR

CALL limpiarVariables

CMP AX, 0

JZ ponerCero

MOV COUNT2, 8 ;Obtenemos el valor para el caracter 5 de nuestro resultado (10000)

Division:

MOV DH, 0

MOV DL, COUNT2

MOV SI, DX

MOV DX, 0

MOV CX, posRes[SI] ;Obtenemos el valor segun la posicion

SUB COUNT2, 2

MOV Aux\_intNum, AX ;Creamos una copia del valor actual

DIV CX ;Dividimos entre el valor segun la posicion (Descenas, centenas, etc)

CMP AX, 0 ;SI no cabe la division pasar a la siguiente

JZ evaluarBandera

MOV flagRes, 1 ;Si si cabe pasar a ASCII el valor

actualizarElementos:

ADD AL, 30H ;Restamos el correspondiente a ASCII del valor obtenido

MOV CX, 0

MOV CL, COUNT3

MOV SI, CX

MOV Aux\_strNum[SI], AL ;Ponemos el valor ASCII en la posicion correspondiente

MOV AX, DX ;Pasamos el residuo de la division a AX

INC COUNT3

INC COUNT

CMP COUNT, 5 ;Comparamos si se han comparado las 5 posibles posiciones o digitos

JZ salirMostrarRes ;Si es asi salimos

JMP Division ;Sino repetimos el proceso

evaluarBandera:

CMP flagRes, 0 ;Evaluamos si se ha ingresado un caracter anteriormente

JZ siguienteIteracion ;Si no es asi, pasamos a la siguiente iteracion

JMP actualizarElementos ;Si si es asi pasar a ASCII el valor

siguienteIteracion:

MOV AX, Aux\_intNum ;Asignamos a AX el valor que tenia antes de la division

INC COUNT

CMP COUNT, 5 ;Se checa que se hayan comparado los 5 caracteres

JZ salirMostrarRes ;Si si es asi salimos de la funcion

`JMP Division` ;Si no, repetimos el bucle

salirMostrarRes: ;Salimos de la funcion y mostramos el resultado

```
MOV AH,09
MOV DX,offset msjResultado ;Imprime mensaje de Resultado
INT 21h
MOV AH, 03H ; Obtener cursor
MOV BH, 0
INT 10H
LEA SI, color ; Cargar color
MOV BL, color
MOV CX, 1
MOV AL, msjSigno[0] ; Cargar signo
MOV AH, 09H
INT 10H ; Imprimir signo
INC DL
MOV AH, 02H ; Recolocar cursor
INT 10H
MOV SI, 0
```

imprimirResultado:

```
MOV AL, Aux_strNum[SI] ; Cargar resultado
MOV AH, 09H
INT 10h ; Imprimir resultado
INC SI
INC DL
MOV AH, 02H ; Recolocar cursor
INT 10H
CMP SI, 5
JL imprimirResultado ; Repetir por cada caracter
RET
```

ponerCero:

```
MOV Aux_strNum[0], 48
JMP salirMostrarRes
```

mostrarResultado ENDP

```
;-----
;----- Realizamos 3 saltos de linea -----
;-----
```

hacer3SaltosTexto `PROC NEAR`

```
MOV AH,09
MOV DX,offset msjSaltos ;Imprime mensaje con saltos de linea
INT 21h
RET ;regresamos el procedimiento de donde lo llamaron.
```

hacer3SaltosTexto `ENDP`

```
;-----
;----- Limpiar Pantalla -----
;-----
```

limpiarPantalla `PROC NEAR`

```
MOV AH, 0FH      ;Limpiamos pantalla
INT 10H
MOV AH, 0        ;Posicionamos cursor
INT 10H
RET              ;regresamos el procedimiento de donde lo llamaron.
limpiarPantalla ENDP
```

```
;-----
;-----      pausa      -----
;-----
systemPause PROC NEAR
LEA DX, msjSalir  ;mostrar mensaje para continuar
MOV AH, 9h
INT 21h
MOV AH, 01H      ;pausa y espera a que el usuario precione una tecla
INT 21h          ;interrupcion para capturar
RET
systemPause ENDP
```

```
;-----
;----  Llamamos a procedimientos, para pedir 2 numeros  -----
;-----
pedir2NumUser PROC NEAR
;MOV AH, 01h      ;Esperar tecla (Para rellenar buffer de memoria)
;INT 21h
CALL hacer3SaltosTexto
CALL pedirNumero
MOV AX, Aux_intNum
MOV op1_intNum, AX ; Obtenemos el valor del operando 1
CALL pedirNumero
MOV AX, Aux_intNum
MOV op2_intNum, AX ; Obtenemos el valor del operando 2
RET              ;regresamos el procedimiento de donde lo llamaron.
pedir2NumUser ENDP
```

```
;-----
;----      Checar signo de numero      -----
;-----
checarSigno PROC NEAR
MOV BX, AX
SAL AX, 1 ;Esta activado el bit 16 del registro 'AX'?
JC Negativo
MOV AX, BX
MOV msjSigno[0], ' '
RET
Negativo:
MOV AX, BX
MOV msjSigno[0], '-'
CALL compl2ADec
RET          ;regresamos el procedimiento de donde lo llamaron.
checarSigno ENDP
```

```
;-----
;----      Comlemento 2 a Decimal      -----
;-----
compl2ADec PROC NEAR
```

*Proyecto integrador – Calculadora – Víctor Leonardo Valle Guerra & David Isaac de la Cruz Castillo*

```
SUB AX, 1      ;Restamos 1 para hacer proc inverso de Comp a 2
NOT AX        ;Not
RET           ;regresamos el procedimiento de donde lo llamaron.
compl2ADec ENDP
```

```
;-----
;---- seleccionar Color para mostrar Resul -----
;-----
seleccionarColor PROC NEAR
    SUB AX, 48    ;Se asigna un color restando a lo escogido
    LEA SI, color ;48 y ese sera el color a mostrar
    MOV [SI], AL  ;Variando segun la opcion
    RET
seleccionarColor ENDP
```

```
;-----
;---- Graficar funciones trigonometricas -----
;-----
```

graficarTrigo PROC NEAR

```
PIXEL MACRO    ;Macro para pintar graficas por puntos
    MOV AL, 1111B ; Color Blanco
    mov ah, 0ch   ; pone pixel
    int 10h
ENDM
```

```
MOV AX, 0      ;Vaciamos los registros
MOV BX, 0
MOV CX, 0
MOV DX, 0
MOV COUN, 0    ;Inicializamos los contadores
MOV COUN2, 450
mov ah, 00     ; Preparar int 10 para video
mov al, 12h    ; configurar la pantalla a 640*400
int 10h
```

```
MOV CX, 0      ;Inicializamos el loop
```

```
;Realizar Curva Postiva
```

```
NEXTP:
```

```
INC CX        ;Incremento de loop
MOV DX, 225   ; Poner el cero del eje X
PIXEL         ; dibujar el pixel del cero
```

```
CMP FLAG, 1    ;Checar si ya se dibujo el EJE Y 0
JZ continuarNormal ;Si si saltar el dibujarlo
CMP CX, 306    ;Si no checar ahora si estamos a la mitad de la grafica
JZ imprimir0Y  ;Dibujar Eje de Y en 0
```

```
continuarNormal:
```

```
CMP COUN, 202  ;Si el contador recorrio todo el arreglo
JZ resetCOUN   ;Reiniciarlo a 0
```

CMP OPCGrafs, 49 ;si se presiono 1 en el menu grafico

JZ graficarSen ;Mostrar grafica de seno

CMP OPCGrafs, 50 ;Si se preisono 2

JZ graficarCos ;Mostrar grafica coseno

CMP OPCGrafs, 51 ;Si se presiono 3

JZ graficarTan ;Mostrar Tangente

CMP OPCGrafs, 52 ;Si se presiono 4

JZ graficarTodas ;Mostrar las 3 graficas juntas

continuarGraf:

ADD COUN, 2 ;Anadimos 2 al contador ya que manejamos registros de 2 bytes

continuar:

CMP CX, 640 ; Si ya llegamos al ancho total

JZ FIN ;Terminamos de graficar

JMP NEXTP ; Sino seguimos imprimiendo pixeles

resetCOUN: ;Reinicia el contador a 0

MOV COUN, 0

JMP continuar

imprimir0Y: ; Dibujamos eje de Y para 0

MOV FLAG, 1 ; indica que se dibujo el eje de Y

DEC COUN2

CMP COUN2, 0

JZ continuarNormal

MOV DH, 0

MOV DX, COUN2 ; Recorremos contador de 0 a 450

PIXEL

JMP imprimir0Y

FIN:

RET ;Acabamos el procedimiento

graficarSen:

CALL GrafSeno ;Imprimimos seno

JMP continuarGraf

graficarCos:

CALL GrafCos ;Imprimimos Coseno

JMP continuarGraf

graficarTan:

CALL GrafTan ;Imprimimos Tangente

JMP continuarGraf

graficarTodas: ;Imprimimos todas las graficas

CALL GrafSeno

```
CALL GrafCos  
CALL GrafTan  
JMP continuarGrafi
```

```
graficarTrigo ENDP
```

```
GrafSeno PROC NEAR  
MOV SI, COUN ; Pasar el indice desde el Count  
MOV DH, 0  
MOV DX, 450  
SUB DX, EJY2[SI] ; en DX la posicion del arreglo de entre 0 a 101  
PIXEL  
RET  
GrafSeno ENDP
```

```
GrafCos PROC NEAR  
MOV SI, COUN ; Pasar el indice desde el Count  
MOV DH, 0  
MOV DX, 450  
SUB DX, EJY2[SI] ; en DX la posicion del arreglo de entre 0 a 101  
PIXEL  
RET  
GrafCos ENDP
```

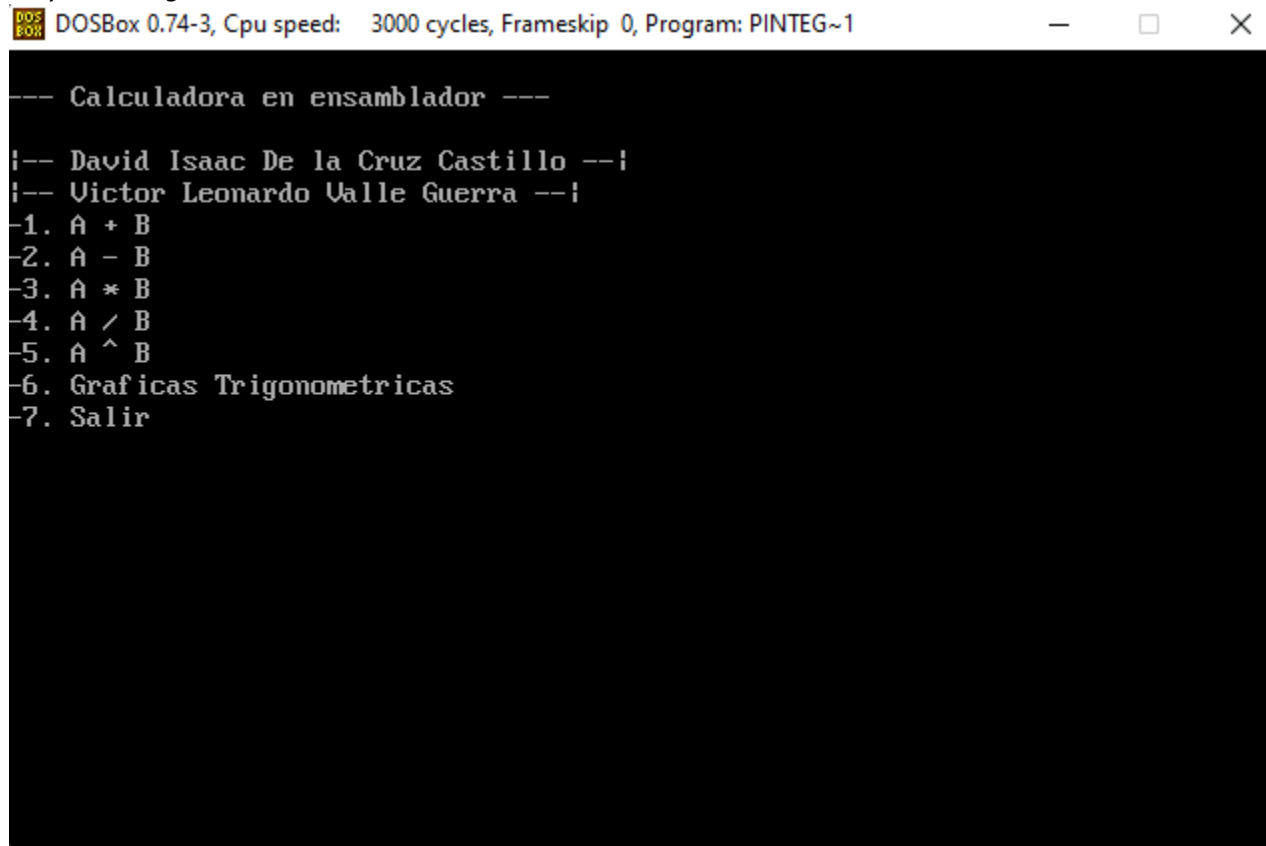
```
GrafTan PROC NEAR  
MOV SI, COUN ; Pasar el indice desde el Count  
MOV DH, 0  
MOV DX, 450  
SUB DX, EJY3[SI] ; en DX la posicion del arreglo de entre 0 a 101  
PIXEL  
RET  
GrafTan ENDP
```

```
END ;Terminamos programa
```

Para realizar las pruebas correspondientes del programa, estas tuvieron que ser realizadas de forma manual en emu8086 para poder de igual manera observar de forma más detallada lo que el estaba sucediendo en memoria y en los registros, así como de igual manera se realizaron pruebas en DOSBOX para poder ver el programa ejecutarse a una velocidad más realista a lo que vería el usuario final.

El menú de la aplicación se puede observar de la siguiente manera.



A screenshot of a DOSBox window. The title bar reads "DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: PINTEG~1". The window contains a text-based menu for a calculator program. The menu is displayed in a monospaced font on a black background. The menu options are: "---- Calculadora en ensamblador ----", "!-- David Isaac De la Cruz Castillo --!", "!-- Victor Leonardo Valle Guerra --!", "1. A + B", "2. A - B", "3. A \* B", "4. A / B", "5. A ^ B", "6. Graficas Trigonometricas", and "7. Salir".

```
---- Calculadora en ensamblador ----
!-- David Isaac De la Cruz Castillo --!
!-- Victor Leonardo Valle Guerra --!
1. A + B
2. A - B
3. A * B
4. A / B
5. A ^ B
6. Graficas Trigonometricas
7. Salir
```

Ingresando una tecla se puede seleccionar una de las opciones del menú.

Una vez que este ingrese dentro de dicha opción, solo será necesario ingresar un número de máximo 2 dígitos con o sin signo, de igual manera este proceso se repite para el segundo número, de forma que posteriormente realizará la operación deseada y mostrará el resultado de otro color.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: PINTEG~1

--- Calculadora en ensamblador ---

!-- David Isaac De la Cruz Castillo --!
!-- Victor Leonardo Valle Guerra --!
-1. A + B
-2. A - B
-3. A * B
-4. A / B
-5. A ^ B
-6. Graficas Trigonometricas
-7. Salir
1

Escribe un numero: 5
Escribe un numero: 4

El resultado de la operacion es: 9 Presione una tecla para continuar..._
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: PINTEG~1

--- Calculadora en ensamblador ---

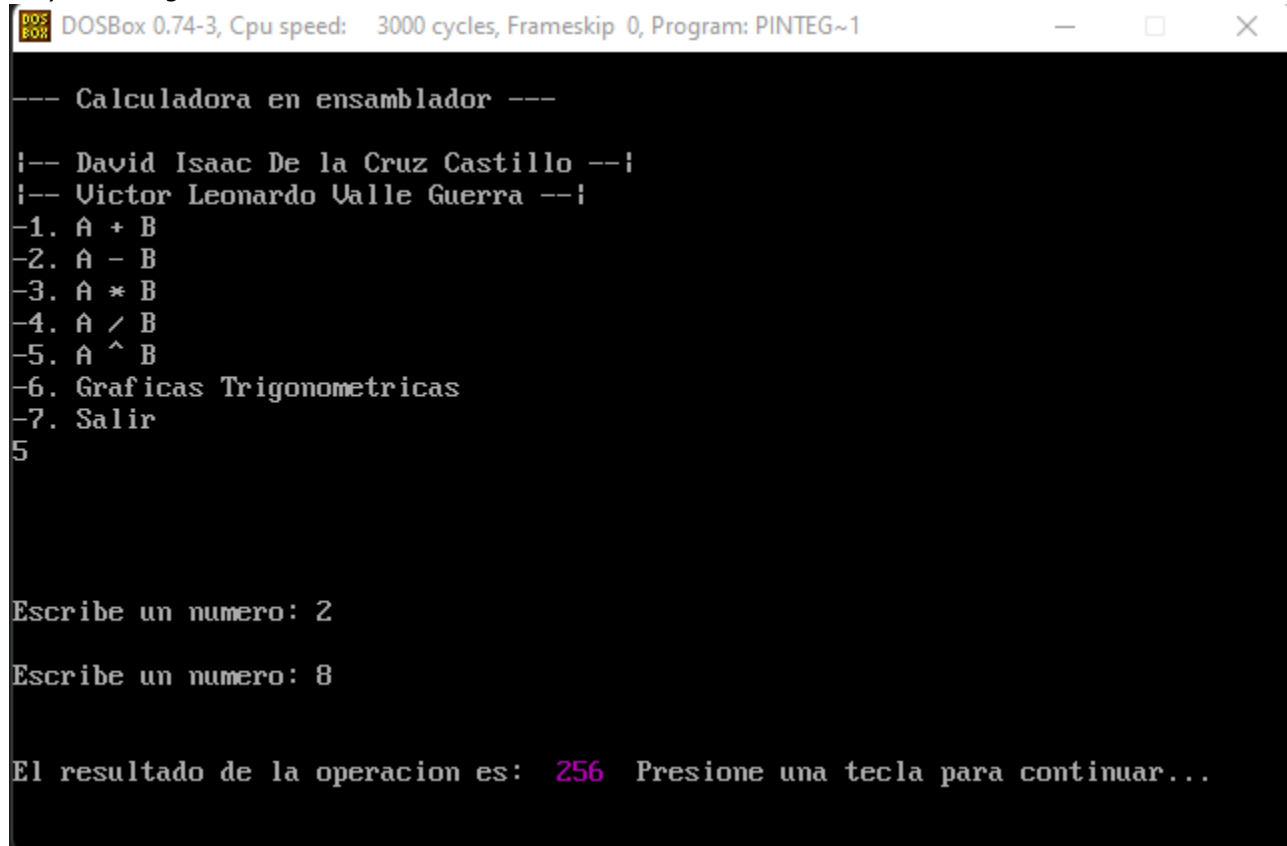
!-- David Isaac De la Cruz Castillo --!
!-- Victor Leonardo Valle Guerra --!
-1. A + B
-2. A - B
-3. A * B
-4. A / B
-5. A ^ B
-6. Graficas Trigonometricas
-7. Salir
2

Escribe un numero: -10
Escribe un numero: -5

El resultado de la operacion es: -5 Presione una tecla para continuar...
```

```
DOS BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: PINTEG~1
--- Calculadora en ensamblador ---
|-- David Isaac De la Cruz Castillo --|
|-- Victor Leonardo Valle Guerra --|
-1. A + B
-2. A - B
-3. A * B
-4. A / B
-5. A ^ B
-6. Graficas Trigonometricas
-7. Salir
3
Escribe un numero: -3
Escribe un numero: -3
El resultado de la operacion es: 9 Presione una tecla para continuar...
```

```
DOS BOX DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: PINTEG~1
--- Calculadora en ensamblador ---
|-- David Isaac De la Cruz Castillo --|
|-- Victor Leonardo Valle Guerra --|
-1. A + B
-2. A - B
-3. A * B
-4. A / B
-5. A ^ B
-6. Graficas Trigonometricas
-7. Salir
4
Escribe un numero: 10
Escribe un numero: -1
El resultado de la operacion es: -10 Presione una tecla para continuar..._
```

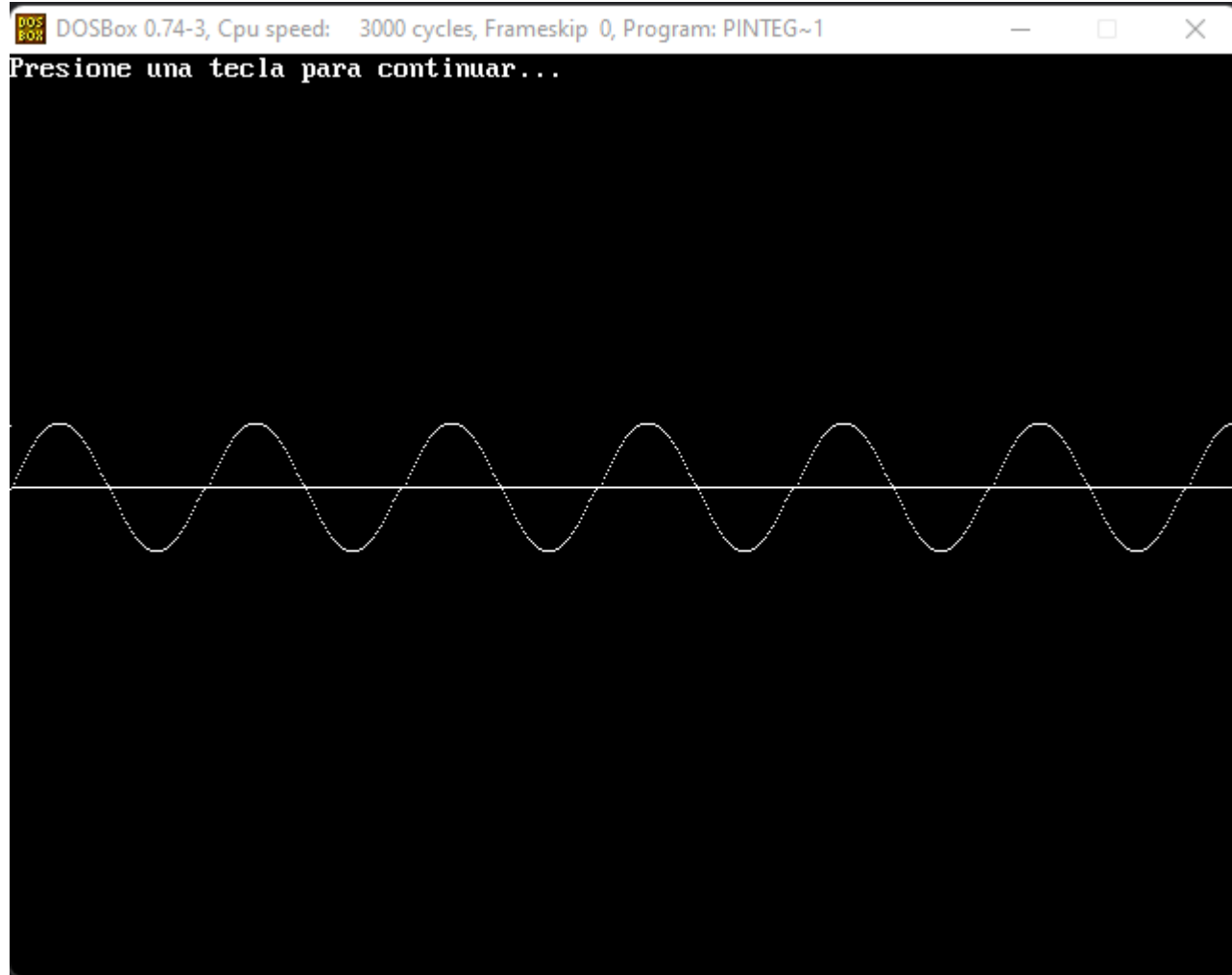


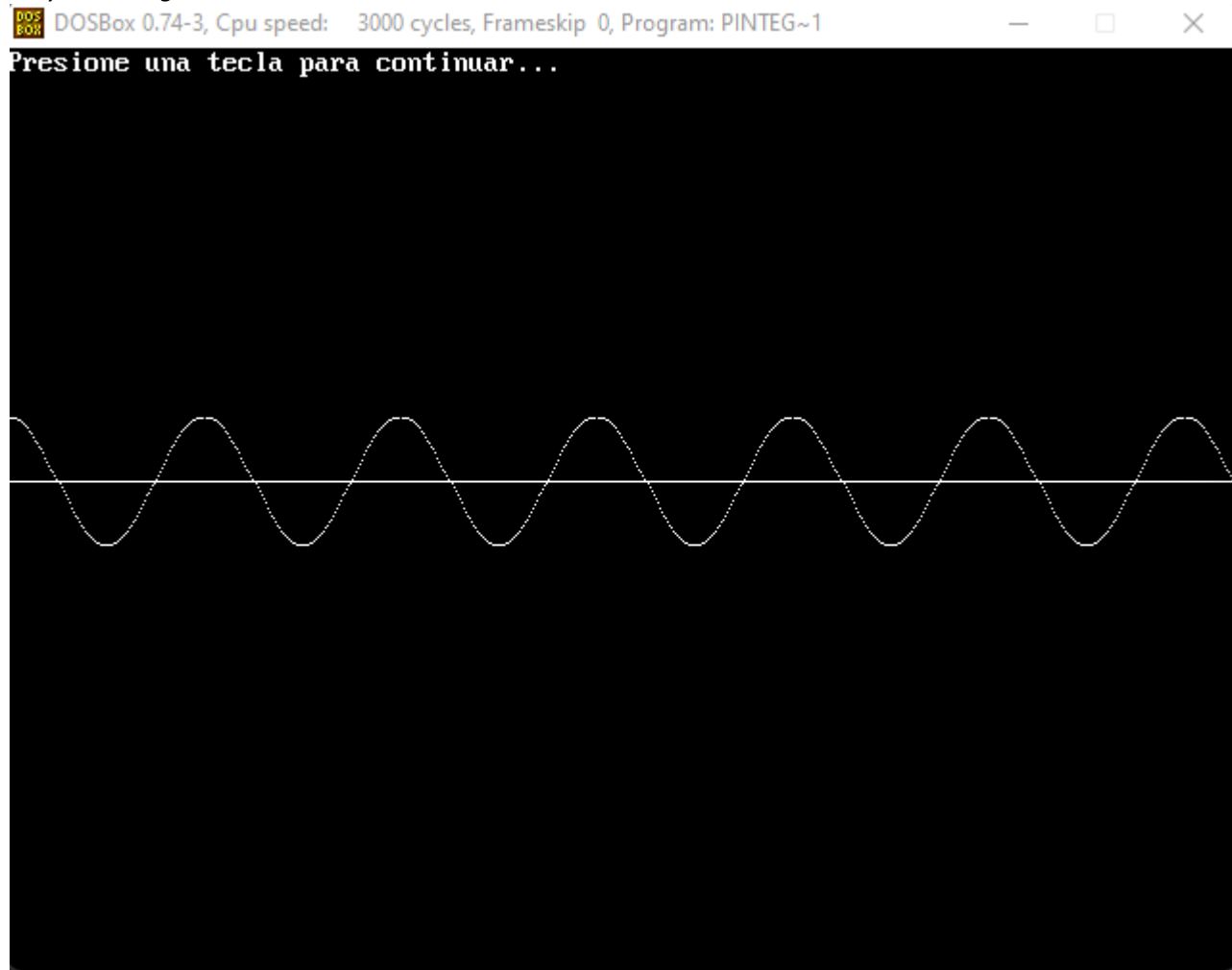
```
--- Calculadora en ensamblador ---  
  
!-- David Isaac De la Cruz Castillo --!  
!-- Victor Leonardo Valle Guerra --!  
-1. A + B  
-2. A - B  
-3. A * B  
-4. A / B  
-5. A ^ B  
-6. Graficas Trigonometricas  
-7. Salir  
5  
  
Escribe un numero: 2  
Escribe un numero: 8  
  
El resultado de la operacion es: 256 Presione una tecla para continuar...
```

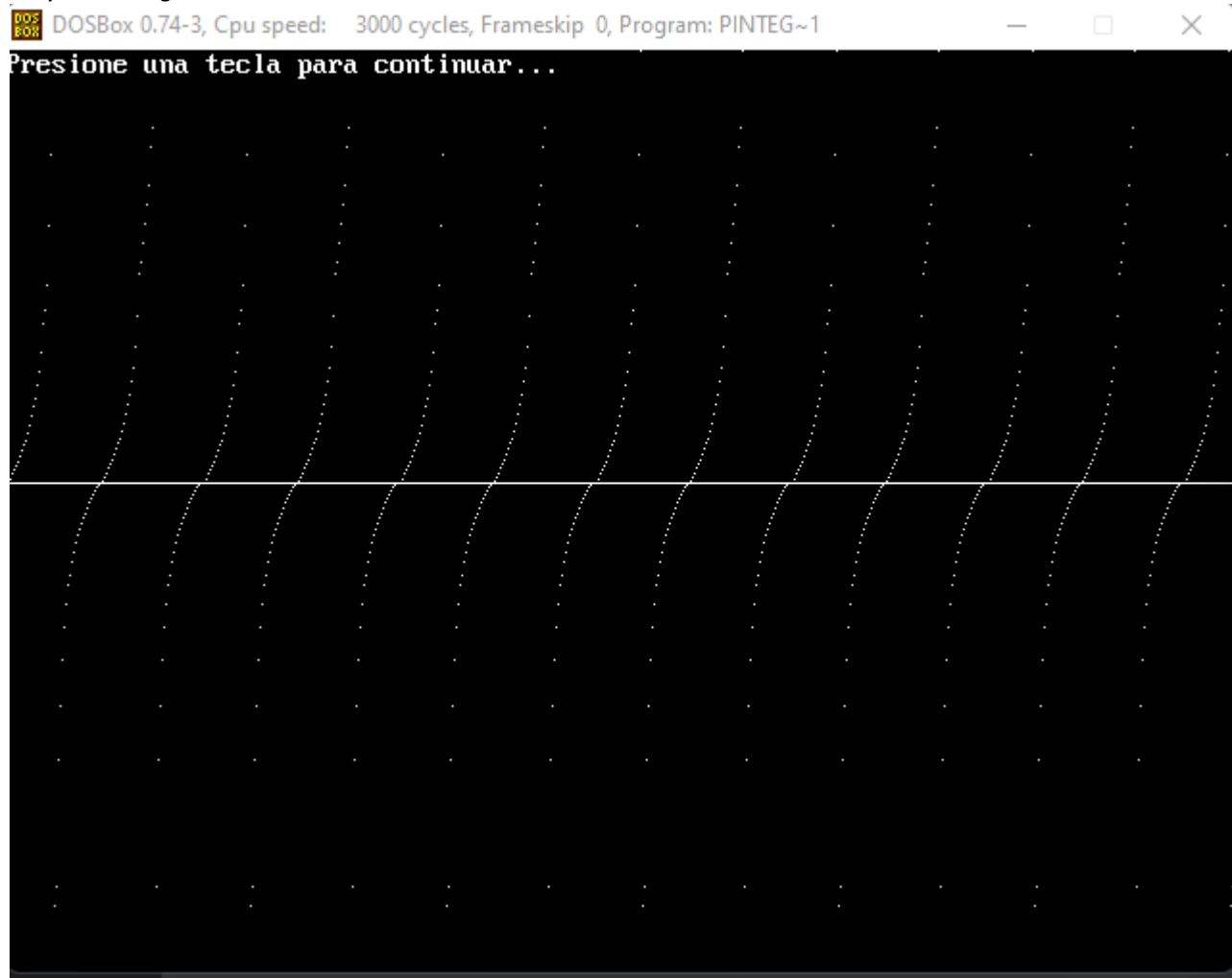
De igual manera la opción de gráficas trigonométricas tiene un submenú para poder graficar las funciones de seno, coseno y tangente o todas juntas.

--- Selecciona la operacion a graficar ---

- 1. Graficar Seno
- 2. Graficar Coseno
- 3. Graficar Tangente
- 4. Todas las anteriores
- 5. Regresar al menu

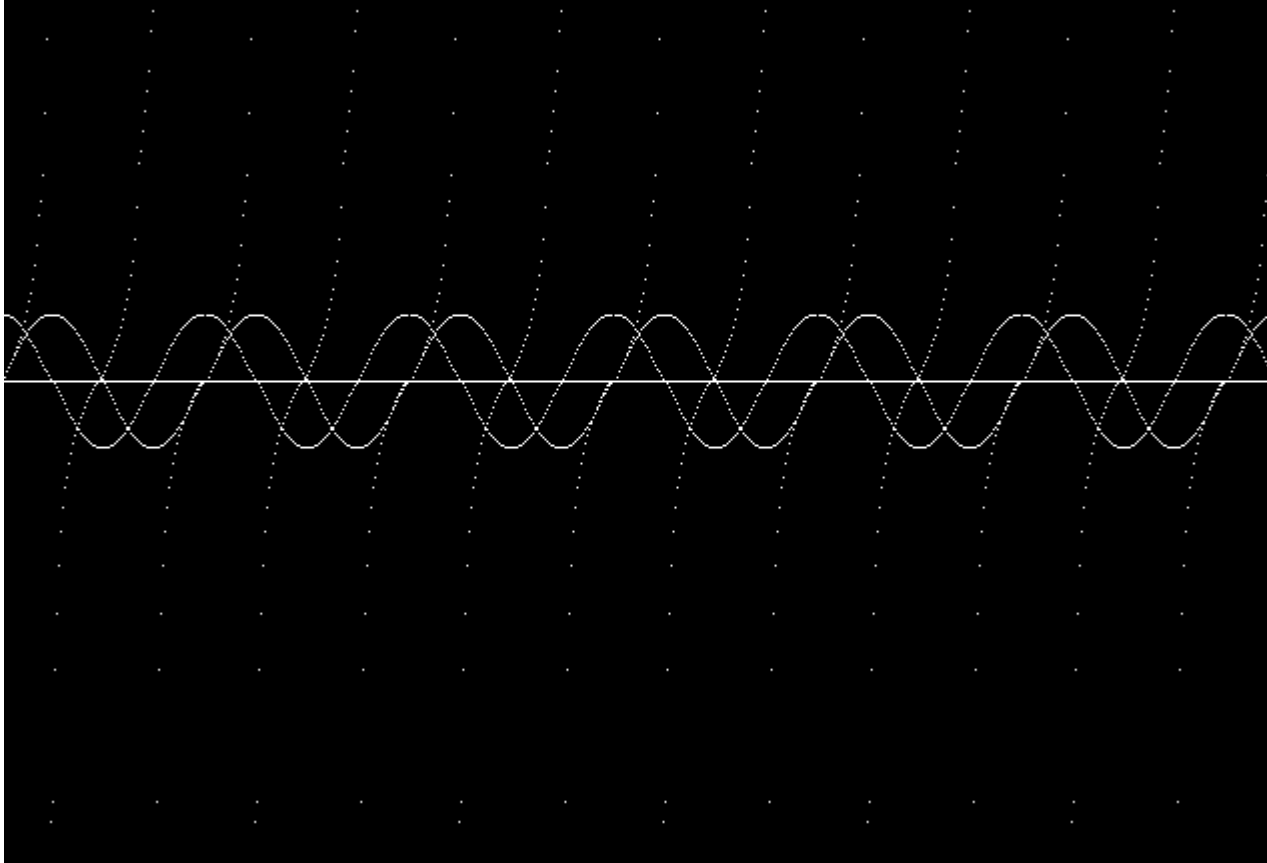








Presione una tecla para continuar...



### Conclusiones:

El lenguaje ensamblador tiene un gran poder expresivo al igual que sus pares en capas de abstracción superiores, pero tiene la ventaja de no tener que pasar por dichas capas de abstracción para poder comunicarse con el hardware del equipo donde se ejecuta.

Posee una curva de aprendizaje pronunciada, especialmente para aquellos estudiantes o profesionistas que no tengan aún bien digerido los conceptos fundamentales de la computación y de la arquitectura de una computadora, pero demuestra como son los pilares y la piedra angular de un gran número de profesiones y campos contemporáneos dentro del área de las tecnologías de la información.

Claramente, no siempre es mejor hacer uso del lenguaje ensamblador, pues requiere una capacidad diferente de abstracción, y una diferente estrategia para poder controlar el flujo del programa y tener noción de la imagen completa de que hace y como funciona el programa y los diversos escenarios que se pueden dar con los diversos factores y parámetros que maneja durante su ejecución, añadiendo a todo esto al tener que conocer la arquitectura del equipo donde se ejecuta el programa, haciendo que se aliente por solo usarlo cuando realmente sea requerido, ya sea por restricciones de memoria bastante austeras, o cuando se requiera tiempos de respuesta o complejidad temporal bastante reducida para la ejecución y uso del mismo, como lo es en el caso del desarrollo de

firmware para sistemas embebidos u otros sistemas con capacidades de hardware limitadas o que poseen una sola responsabilidad o uso para algún proceso industrial específico.

De forma que esto se decidirá a criterio del desarrollador, y con base en los requerimientos y limitaciones, así como alcances delimitados por la naturaleza del proyecto o circunstancia. Ya que en muchos escenarios se prefiere la velocidad y facilidad de poder desarrollarlo software, así como de escalarlos, corregirlo, actualizarlo y mantenerlo que su complejidad temporal o espacial.

Además de que, en muchos escenarios, si no es que, en la mayoría, el compilador realizará un mucho mejor trabajo de optimización al compilar a que si nosotros directamente quisiéramos desarrollar la implementación directamente en ensamblador.

De forma que este proyecto nos permite más que nada de forma didáctica poder comprender la relación y funcionamiento entre el hardware y el lenguaje ensamblador, y como este posteriormente sirvieron para poder llevar a la industria del software a donde estamos, por medio de la añadidura de capas de abstracción que facilitan el trabajo y desarrollo de los programadores para ciertos escenarios donde precisamente el consumo de memoria y el tiempo de ejecución no sean una preocupación, o en aquellos donde no se requiera hablar o tratar directamente con el hardware.

Además, pudiendo darnos a c como lenguaje de programación el cual es idóneo en escenarios donde queremos lo mejor de ambos mundos.

## Bibliografía

Abel, P. (1995). *Lenguaje ensamblador y programación para IBM PC y compatibles*. British Columbia: Pearson.

Intel. (1979). *The 8086 Family User's Manual*. Santa Clara: Intel Corporation.

Intel. (1981). *iAPX 86, 88 User's manual*. Santa Clara: Intel Corporation.

Intel. (1983). *iAPX 286 Programmer's Reference*. Santa Clara: Intel Corporation.

Intel. (1983). *iAPX 286 Programmer's Reference Manual*. Santa Clara: Intel Corporation.

Intel. (2019). *Intel 64 and IA-32 Architectures Optimization Reference Manual*. Santa Clara: Intel Corporation.

Toppr. (2019, Junio 19). *Toppr*. Retrieved from Toppr: <https://www.toppr.com/guides/computer-science/computer-fundamentals/system-software/assembler/>

Virginia, U. o. (2022, Marzo 08). *x86 Assembly Guide*. Retrieved from University of Virginia Computer Science: <https://www.cs.virginia.edu/~evans/cs216/guides/x86.html>

World, C. (2008, Junio 05). *Timeline: A brief history of the x86 microprocessor*. Retrieved from Computer World: <https://www.computerworld.com/article/2535019/timeline--a-brief-history-of-the-x86-microprocessor.html>

