Arellano Granados Angel Mariano

218123444

Seminario de Traductores de Lenguajes I

I7026 D02

Reporte de Proyecto Final

Proyecto Final

Descripción

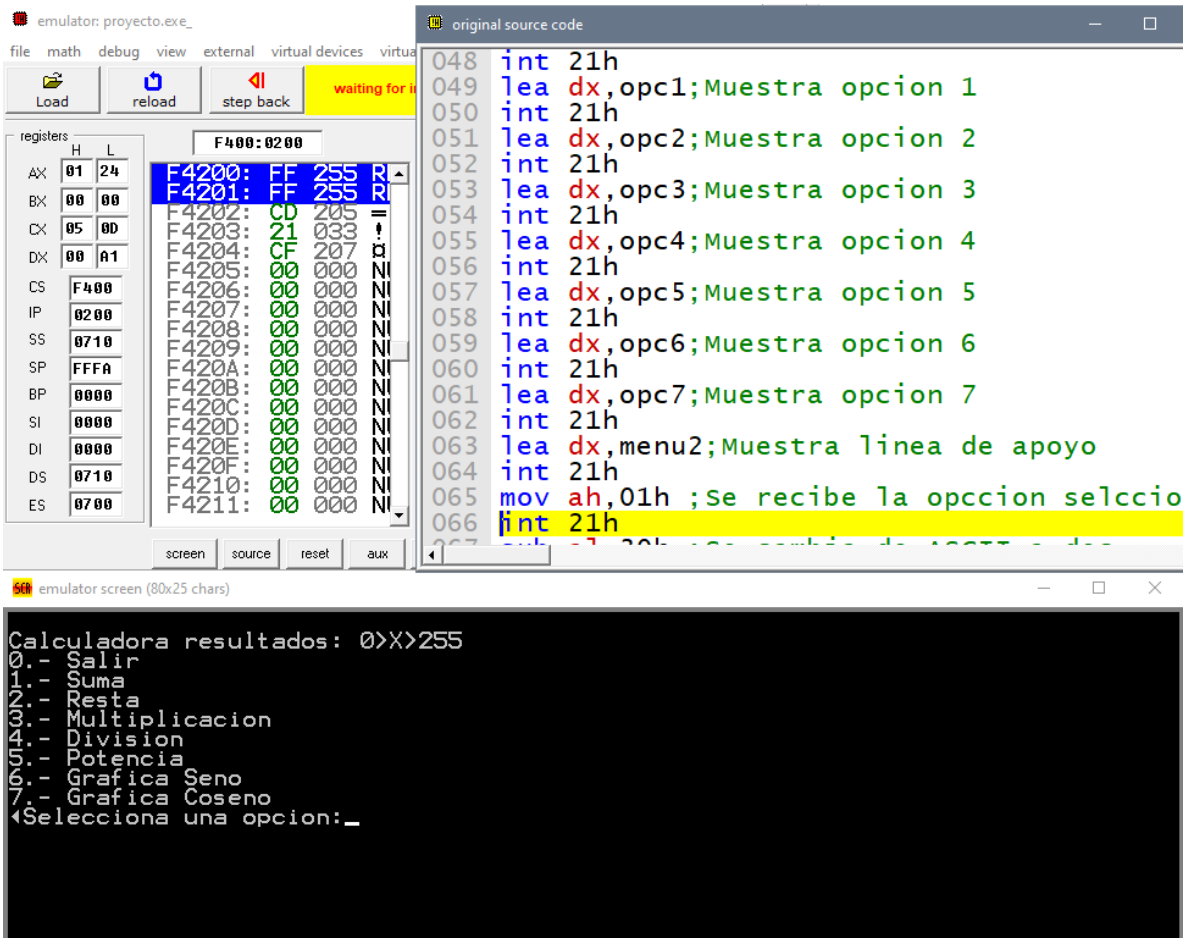
Desarrolla un programa de una calculadora en lenguaje ensamblador, implementando las operaciones aritméticas básicas (sume, resta, multiplicación y división), además de la potencia. Agrega las funciones para graficar las funciones seno y coseno.

Desarrollo y Resultados

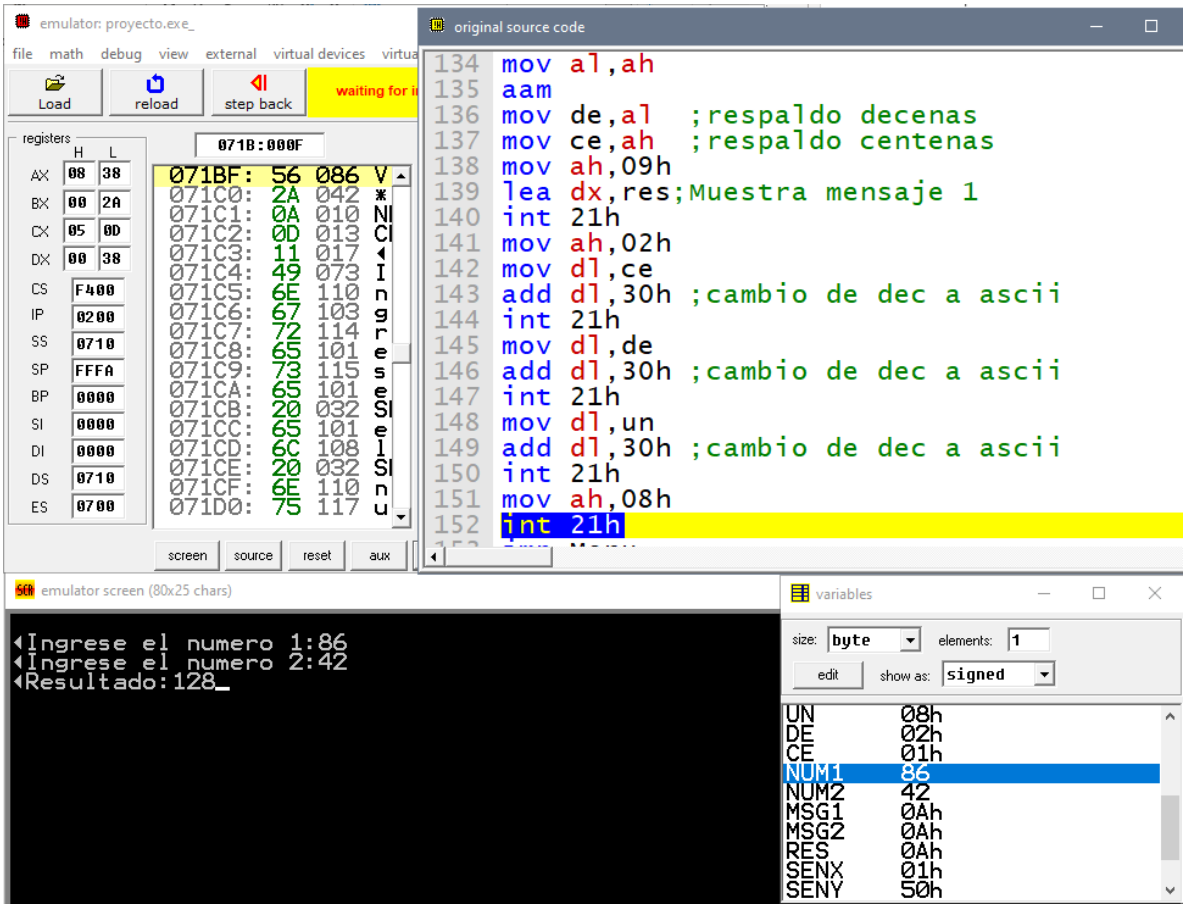
Para este proyecto se aprovechó y escalo el código de la actividad 6 – Parte 2 donde se capturaban tres números, se almacenaban y mostraban, agregando un módulo que efectuase la correspondiente operación antes de mostrar el resultado.

La aplicación de este método tiene una complicación dado a que uso la instrucción AAM para convertir los resultados en hexadecimal a ASCII, esta instrucción dentro del EMU8086 solo funciona con números entre 0 y 255, si no esta dentro de este rango se muestra un resultado erróneo, creando varios problemas con la operación de potencia, pero cualquier operación que el resultado este en ese rango funciona a la perfección.

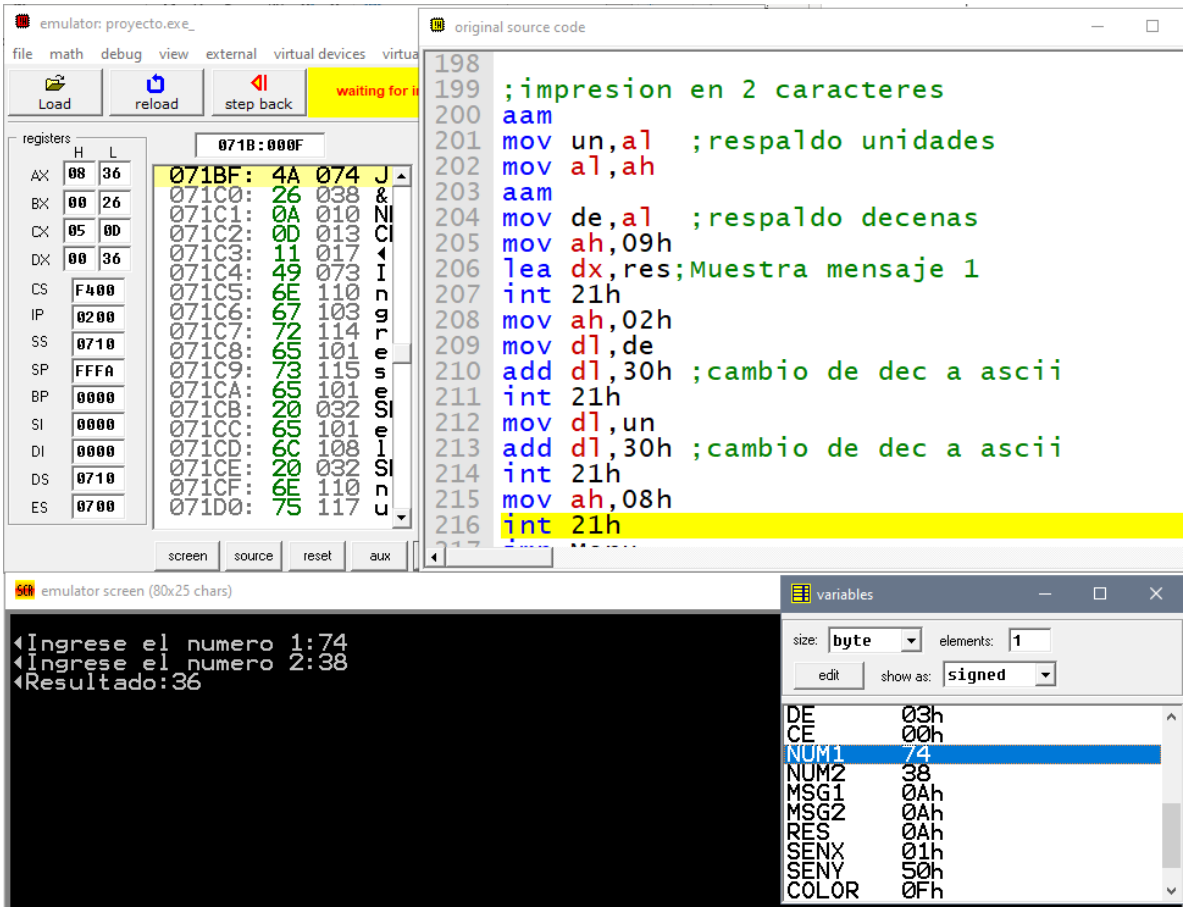
Ya en implementación el programa de divide en 9 etiquetas principales una para el menú, 4 para las operaciones aritméticas, una para la potencia, 2 para las graficas y una ultima para terminar el programa; La etiqueta de Menú imprime varias cadenas para mostrar el menú de opciones y espera recibir el numero de una de las opciones para saltar a la etiqueta de la operación, sino recibe una entrada valida se cicla.



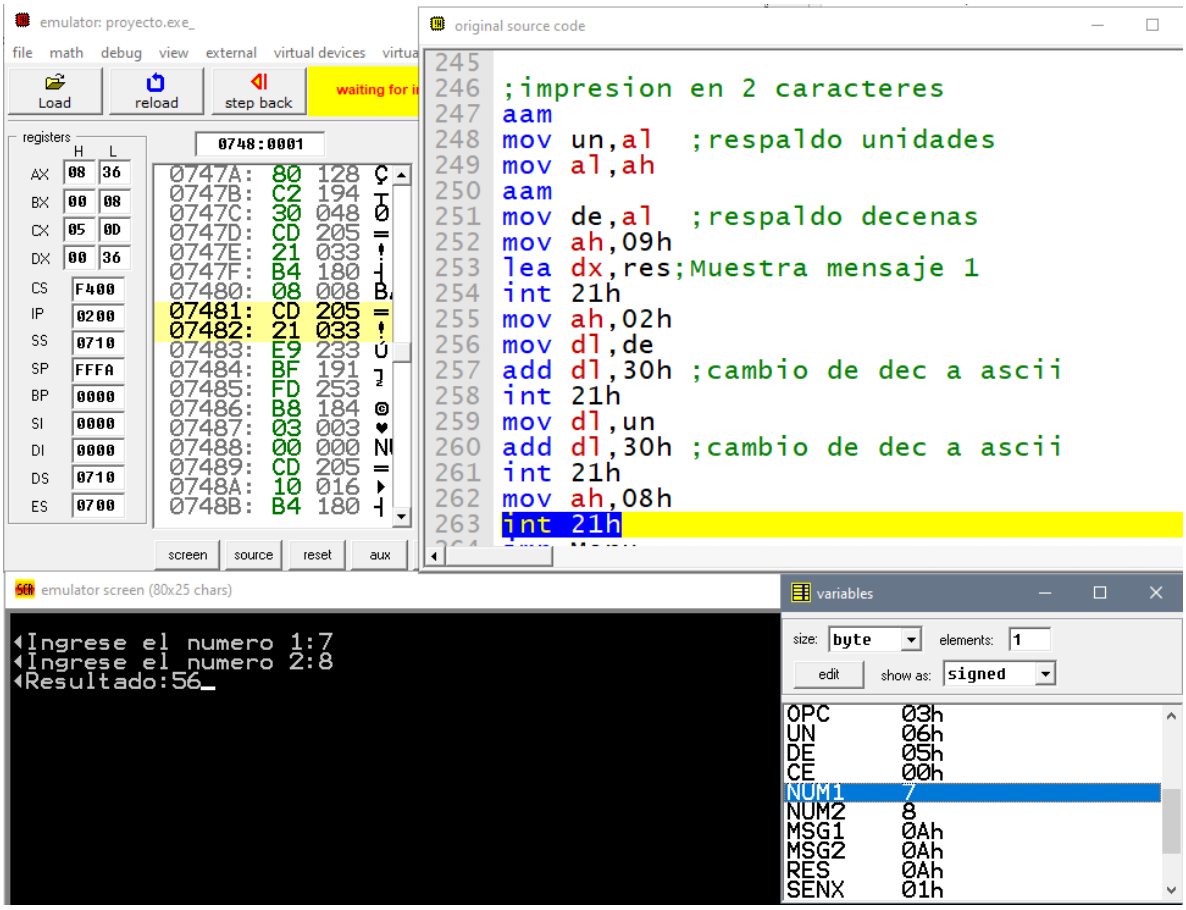
Al recibir un número del 0 al 7 salta a la respectiva etiqueta con ayuda de varias comparaciones y saltos condicionales que comparan el carácter recibido; si recibe el numero 1 salta a la etiqueta Sum, que recibe 2 números de 2 cifras y muestra el resultado en 3 cifras, donde el rango de todas las sumas va de 0+0=0 a 99+99=198 no teniendo problemas con AAM.



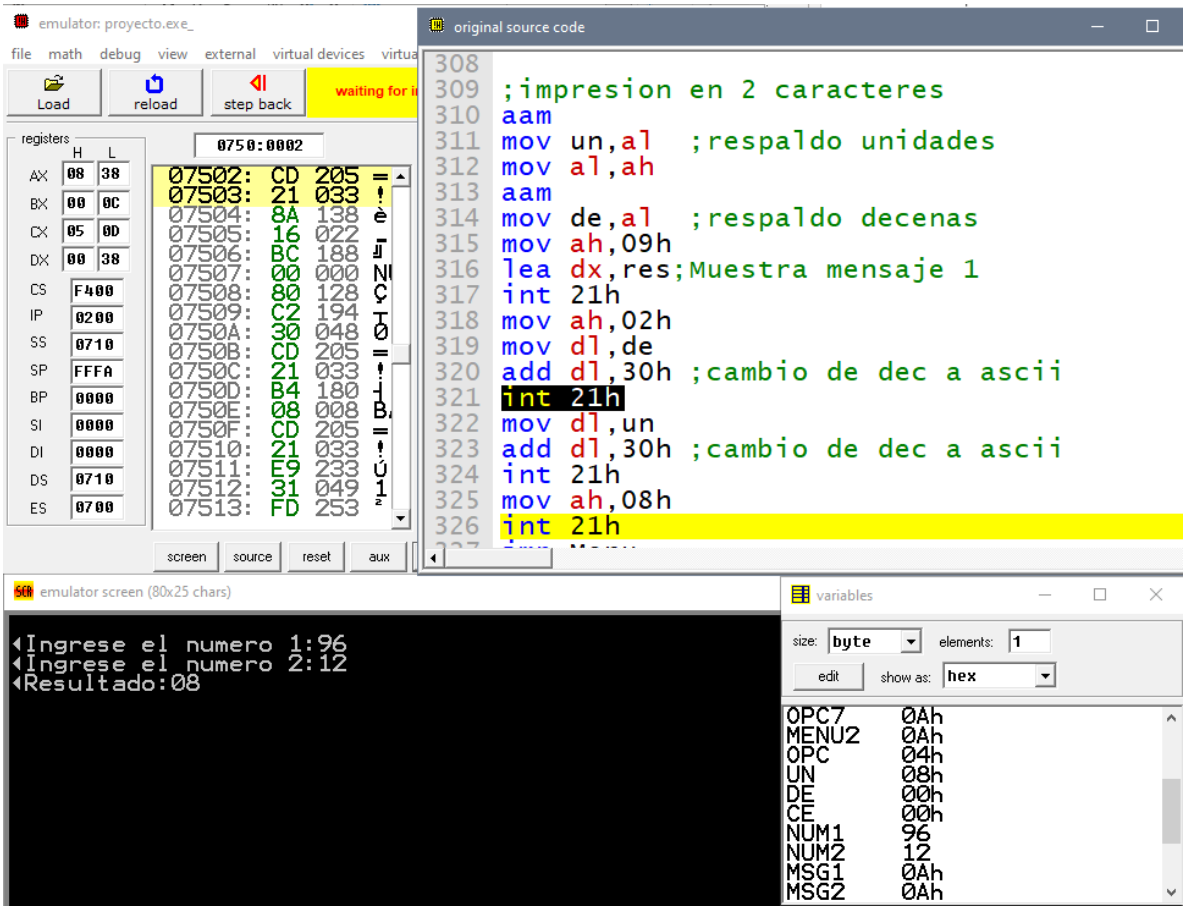
El programa se pausa hasta que se presione cualquier tecla del teclado para volver a regresar al menú principal; en el menú ingresamos el número 2 y salta a la etiqueta Res, que recibe 2 números de 2 cifras y muestra el resultado en 2 cifras, donde el rango de todas las restas va de 0-99=-99 a 99-0=99 donde tenemos varias restas con problemas con AAM, pues esta no detecta números negativos.



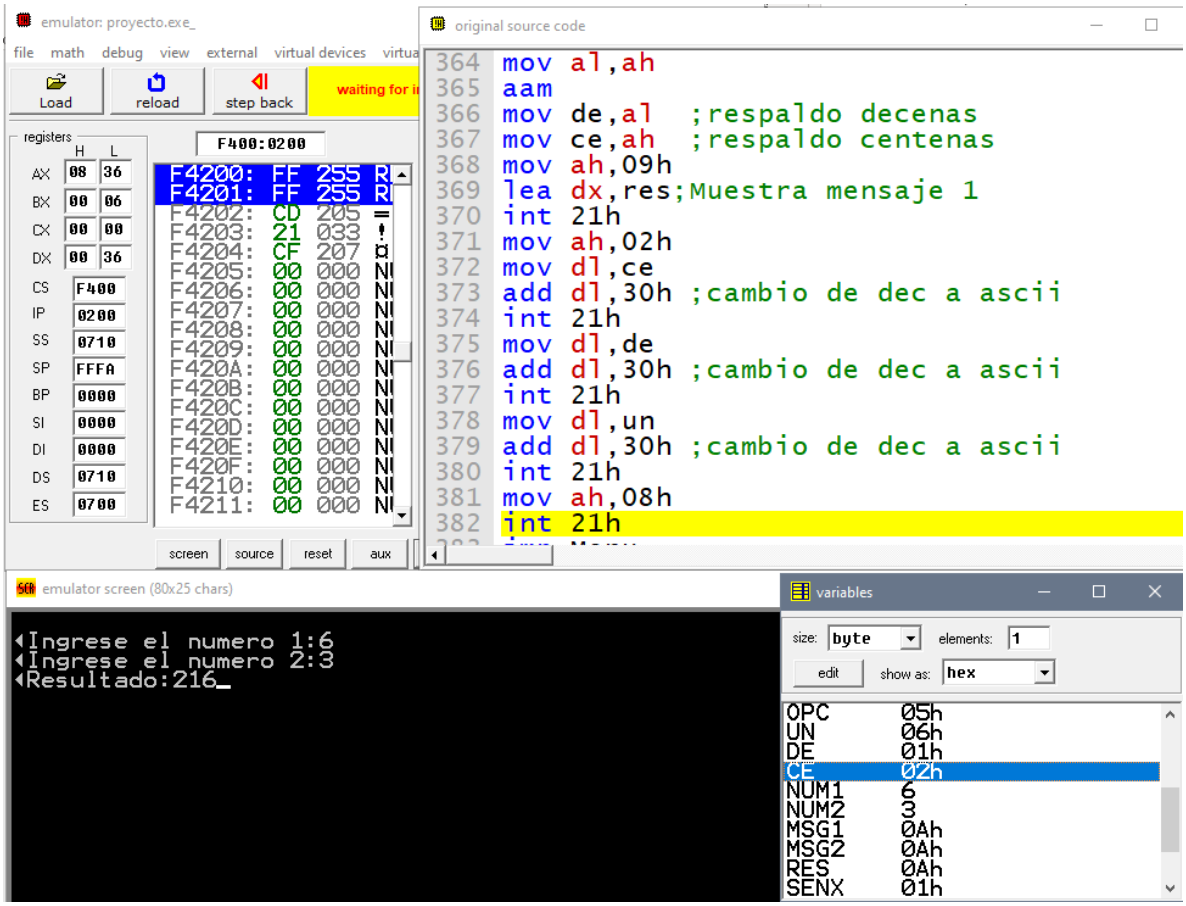
Si el resultado de la resta es un numero negativo el programa muestra un resultado erróneo, también se pausa hasta presionar una tecla; regresando al menú ingresamos el número 3 y salta a la etiqueta Mult, que recibe 2 números de una cifra y muestra el resultado en 2 cifras, donde el rango de todas las multiplicaciones va de 0*0=0 a 9*9=81 no teniendo problemas con AAM.



Ahora ingresamos el número 4 y salta a la etiqueta Divi que solo muestra resultados enteros omitiendo el resto, lo equivalente a la división entera, que recibe 2 números de 2 cifras y muestra el resultado en 2 cifras, donde el rango de todas las divisiones va de 0/1=0 a 99/99=1 no teniendo problemas con AAM, pero si se intenta 0/0 el programa se detiene.

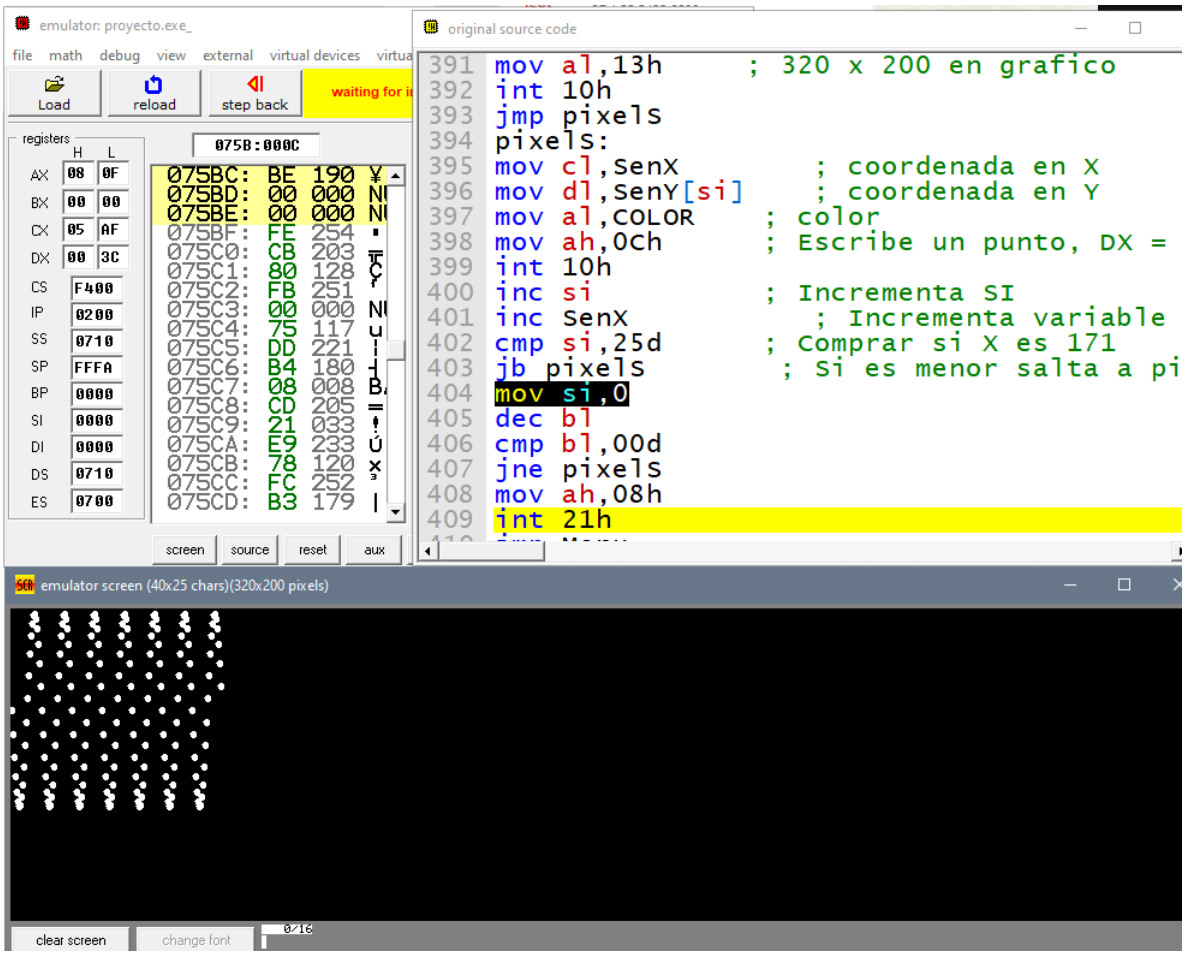


Con la opción número 5 se salta a la etiqueta Pot, que recibe 2 números de una cifra y muestra el resultado en 3 cifras, donde el rango de todas las potencias va de 0^1=0 a 9^9=387,420,489 teniendo muchos problemas con AAM pues la gran mayoría de combinaciones el resultado es mayor a 255, para lograr que se efectuara la potencia se tomaba el primer número y se coloca en los registros AL y BL y el segundo número en CL, a si con un loop multiplicaría AL y BL tantas veces diga CL.

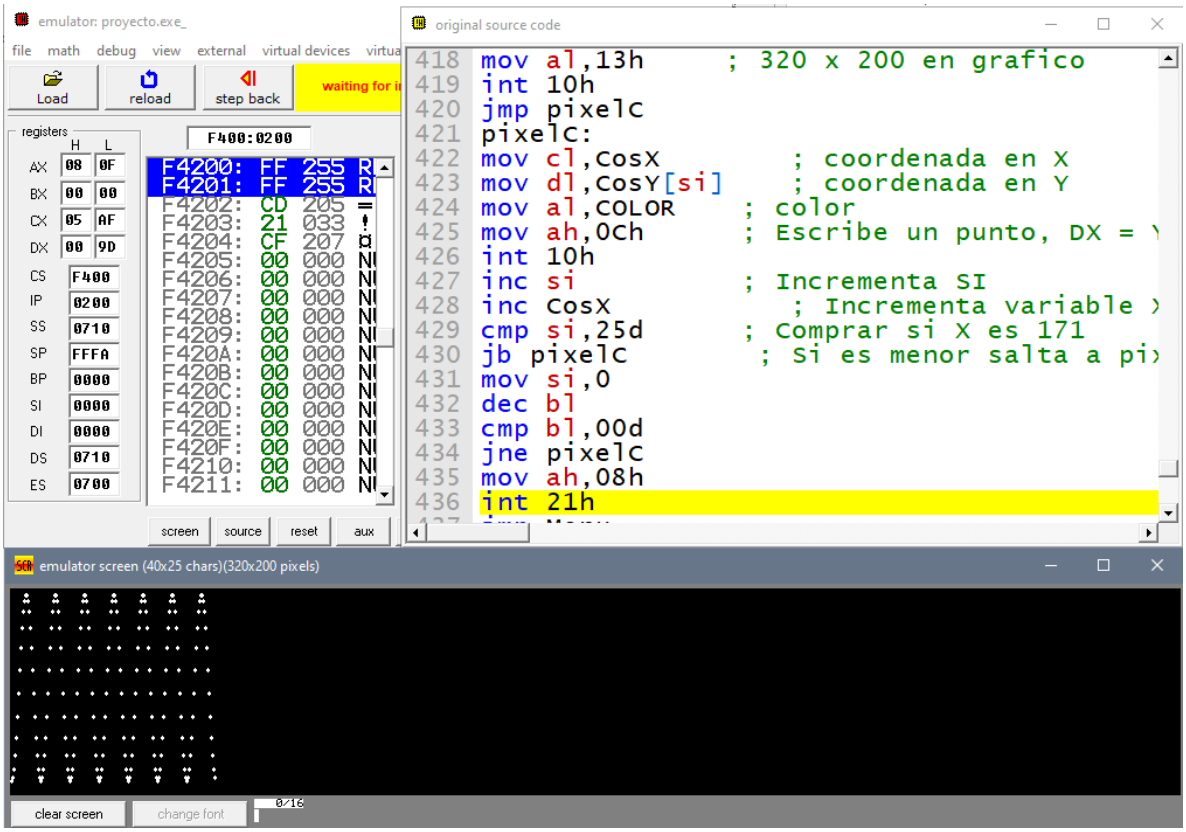


Empezando con las gráficas que corresponden a las opciones 6 y 7 estas saltan a las etiquetas Sen y Cos respectivamente, donde ambas tienen el mismo funcionamiento inician el modo del video y con ayuda de otra etiqueta se crea un ciclo que pinta un pixel en una coordenada dada por X y Y, donde X comienza en 1 y Y recorre los valores de un arreglo, tras cada repetición X aumenta 1 y Y pasa al siguiente valor del arreglo, cuando ya no quedan valores en el arreglo termina el ciclo, se reinicia en puntero para regresar al inicio del arreglo y se vuelve iniciar el ciclo, esto 7 veces para que se vean varias oscilaciones de la gráfica.

Seno:

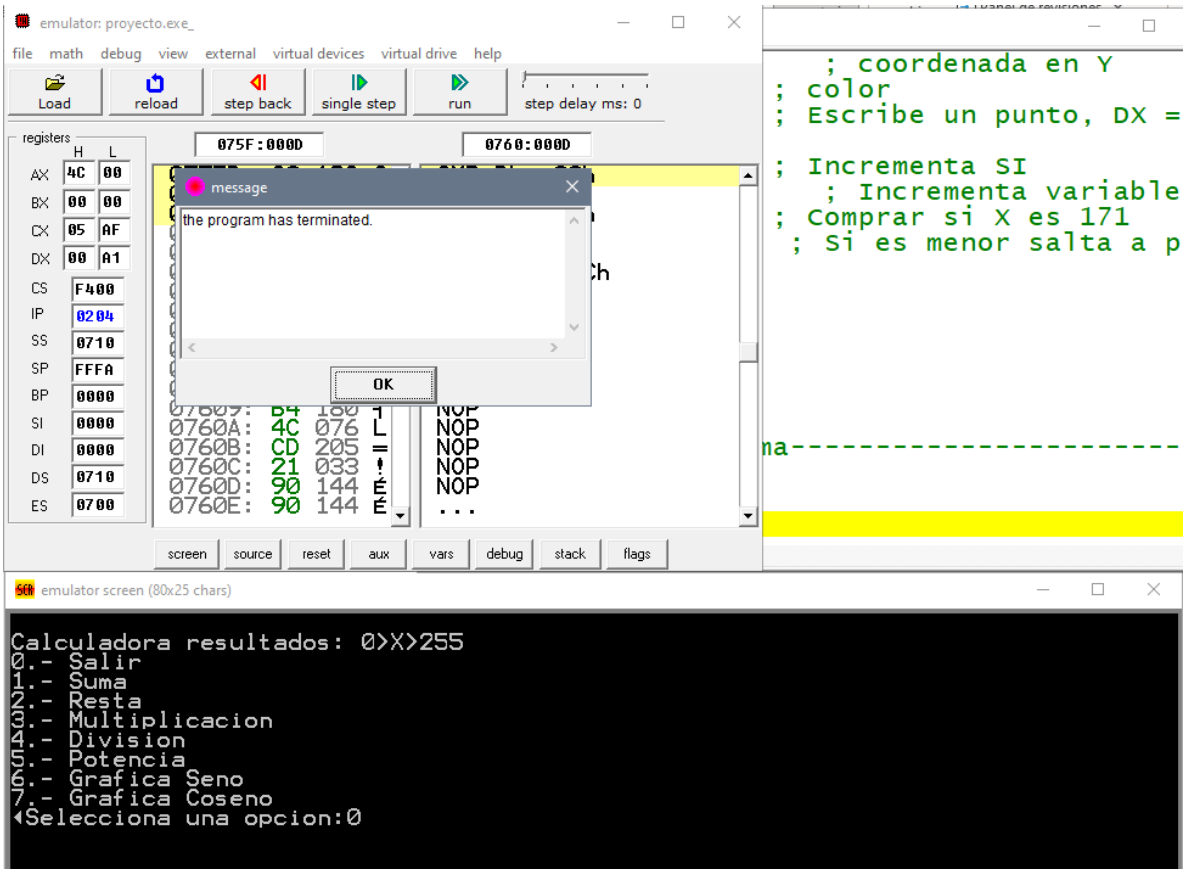

emulator: proyecto.exe_
file math debug view external virtual devices
Load
reload
step back
registers
AX 08 0F
BX 00 00
CX 05 AF
DX 00 3C
CS F400
IP 0200
SS 0710
SP FFFA
BP 0000
SI 0000
DI 0000
DS 0710
ES 0700
075B:000C
075BC: BE 190
075BD: 00 000
075BE: 00 000
075BF: FE 254
075C0: CB 203
075C1: 80 128
075C2: FB 251
075C3: 00 000
075C4: 75 117
075C5: DD 221
075C6: B4 180
075C7: 08 008
075C8: CD 205
075C9: 21 033
075CA: E9 233
075CB: 78 120
075CC: FC 252
075CD: B3 179
screen
source
reset
aux
original source code
391 mov al,13h ; 320 x 200 en grafico
392 int 10h
393 jmp pixels
394 pixels:
395 mov cl,SenX ; coordenada en X
396 mov dl,SenY[si] ; coordenada en Y
397 mov al,COLOR ; color
398 mov ah,0Ch ; Escribe un punto, DX =
399 int 10h
400 inc si ; Incrementa SI
401 inc SenX ; Incrementa variable
402 cmp si,25d ; Comprar si X es 171
403 jb pixels ; Si es menor salta a pi
404 mov si,0
405 dec bl
406 cmp bl,00d
407 jne pixels
408 mov ah,08h
409 int 21h
emulator screen (40x25 chars)(320x200 pixels)
clear screen
change font

Coseno:



Los valores de ambos arreglos fueron obtenidos a través de Excel con una función dada por mi maestro de catedra Juan Meza.

Por último, la opción con el numero 0 que termina el programa.



Reflexión

Durante la elaboración de este proyecto fui capaz de unir y mejorar bastante código de otras tareas de esta materia y la catedra para crear algo más grande y complejo comparado a lo que normalmente hacíamos, sin embargo considero que pude mejorar el código bastante eliminando la limitación de los resultados y modularizar más es código dado a que repito mucho las mismos segmentos de código varias veces dado a no poder implementar de manera correcta los procedimientos, aun así estoy feliz con el resultado.

Código

```
.model .stack

.data

menu1 db 09,10,13,'Calculadora resultados: 0>X>255','$'

opc0  db 10,13,'0.- Salir','$'

opc1  db 10,13,'1.- Suma','$'

opc2  db 10,13,'2.- Resta','$'
```

```
opc3  db 10,13,'3.- Multiplicacion','$'
opc4  db 10,13,'4.- Division','$'
opc5  db 10,13,'5.- Potencia','$'
opc6  db 10,13,'6.- Grafica Seno','$'
opc7  db 10,13,'7.- Grafica Coseno','$'
menu2 db 10,13,17,'Selecciona una opcion:','$'
opc   db 0

un db 0
de db 0
ce db 0
num1 db 0
num2 db 0
msg1 db 10,13,17,'Ingrese el numero 1:','$'
msg2 db 10,13,17,'Ingrese el numero 2:','$'
res db 10,13,17,'Resultado:','$'

SenX DB 1
SenY                                                                    DB
80,100,119,135,148,156,160,159,152,142,127,109,90,70,51,33,18,8,1,0,4,12,25,4
1,60
COLOR DB 0fh

CosX DB 1
CosY                                                                    DB
160,157,150,138,123,105,85,65,46,29,15,6,1,1,6,15,29,46,65,85,105,123,138,150,
157

.code
   mov ax,data
```

```
    mov ds,ax  ;Se inicia en segmento de datos
;Menu----------------------------------------------------
Menu:
    mov un,0
    mov de,0
    mov ce,0
    mov num1,0
    mov num2,0
    mov ax,03h
    int 10h
    mov ah,09h
    lea dx,menu1;Muestra linea de apoyo
    int 21h
    lea dx,opc0;Muestra opcion 0
    int 21h
    lea dx,opc1;Muestra opcion 1
    int 21h
    lea dx,opc2;Muestra opcion 2
    int 21h
    lea dx,opc3;Muestra opcion 3
    int 21h
    lea dx,opc4;Muestra opcion 4
    int 21h
    lea dx,opc5;Muestra opcion 5
    int 21h
    lea dx,opc6;Muestra opcion 6
    int 21h
    lea dx,opc7;Muestra opcion 7
```

```
    int 21h
    lea dx,menu2;Muestra linea de apoyo
    int 21h
    mov ah,01h ;Se recibe la opccion selccionada
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov opc,al
    cmp opc,0d
    je Salir
    cmp opc,1d
    je Sum
    cmp opc,2d
    je Rest
    cmp opc,3d
    je Mult
    cmp opc,4d
    je Divi
    cmp opc,5d
    je Pot
    cmp opc,6d
    je Sen
    cmp opc,7d
    je Cos
    cmp opc,7d
    jg Menu
;Suma----------------------------------------------------
Sum:
    mov ax,03h
```

```
int 10h
mov ah,09h
lea dx,msg1;Muestra mensaje 1
int 21h
mov ah,01h ;Introdusco las decenas num 1
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov de,al
mov ah,01h ;Introdusco las unidades num 1
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov un,al
mov al,de
mov bl,10
mul bl     ; al = de * 10
add al,un   ; al = de * 10 + un
mov num1,al;se obtiene el primer numero
mov ah,09h
lea dx,msg2;Muestra mensaje 2
int 21h
mov ah,01h ;Introdusco las decenas num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov de,al
mov ah,01h ;Introdusco las unidades num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov un,al
```

```
mov al,de
mov bl,10
mul bl     ; al = de * 10
add al,un  ; al = de * 10 + un
mov num2,al;se obtiene el segundo numero

;Sumador
xor ax,ax  ;lipiamos registros
xor bx,bx
mov al,num1
mov bl,num2
add al,bl  ;Se suman los numeros

;impresion en 3 caracteres
aam
mov un,al  ;respaldo unidades
mov al,ah
aam
mov de,al  ;respaldo decenas
mov ce,ah  ;respaldo centenas
mov ah,09h
lea dx,res;Muestra mensaje 1
int 21h
mov ah,02h
mov dl,ce
add dl,30h ;cambio de dec a ascii
int 21h
mov dl,de
```

```
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov dl,un
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov ah,08h
    int 21h
    jmp Menu
`
;Resta-------------------------------------------------------
Rest:
    mov ax,03h
    int 10h
    mov ah,09h
    lea dx,msg1;Muestra mensaje 1
    int 21h
    mov ah,01h ;Introdusco las decenas num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov de,al
    mov ah,01h ;Introdusco las unidades num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov un,al
    mov al,de
    mov bl,10
    mul bl     ; al = de * 10
    add al,un   ; al = de * 10 + un
```

```
mov num1,al;se obtiene el primer numero
mov ah,09h
lea dx,msg2;Muestra mensaje 2
int 21h
mov ah,01h ;Introdusco las decenas num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov de,al
mov ah,01h ;Introdusco las unidades num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov un,al
mov al,de
mov bl,10
mul bl     ; al = de * 10
add al,un  ; al = de * 10 + un
mov num2,al;se obtiene el segundo numero

;Restador
xor ax,ax  ;lipiamos registros
xor bx,bx
mov al,num1
mov bl,num2
sub al,bl  ;Se restan los numeros

;impresion en 2 caracteres
aam
mov un,al  ;respaldo unidades
```

```
    mov al,ah
    aam
    mov de,al  ;respaldo decenas
    mov ah,09h
    lea dx,res;Muestra mensaje 1
    int 21h
    mov ah,02h
    mov dl,de
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov dl,un
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov ah,08h
    int 21h
    jmp Menu
;Multiplicacion----------------------------------------------
mult:
    mov ax,03h
    int 10h
    mov ah,09h
    lea dx,msg1;Muestra mensaje 1
    int 21h
    mov ah,01h ;Introdusco las unidades num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov un,al
    mov num1,al;se obtiene el primer numero
```

```
mov ah,09h
lea dx,msg2;Muestra mensaje 2
int 21h
mov ah,01h ;Introdusco las unidades num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov un,al
mov num2,al;se obtiene el segundo numero

;Multiplicador
xor ax,ax  ;lipiamos registros
xor bx,bx
mov al,num1
mov bl,num2
mul bl      ;Se multiplican los numeros

;impresion en 2 caracteres
aam
mov un,al  ;respaldo unidades
mov al,ah
aam
mov de,al  ;respaldo decenas
mov ah,09h
lea dx,res;Muestra mensaje 1
int 21h
mov ah,02h
mov dl,de
add dl,30h ;cambio de dec a ascii
```

```
    int 21h
    mov dl,un
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov ah,08h
    int 21h
    jmp Menu
;Division-----------------------------------------------------
Divi:
    mov ax,03h
    int 10h
    mov ah,09h
    lea dx,msg1;Muestra mensaje 1
    int 21h
    mov ah,01h ;Introdusco las decenas num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov de,al
    mov ah,01h ;Introdusco las unidades num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov un,al
    mov al,de
    mov bl,10
    mul bl     ; al = de * 10
    add al,un   ; al = de * 10 + un
    mov num1,al;se obtiene el primer numero
    mov ah,09h
```

```
lea dx,msg2;Muestra mensaje 2
int 21h
mov ah,01h ;Introdusco las decenas num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov de,al
mov ah,01h ;Introdusco las unidades num 2
int 21h
sub al,30h ;Se cambia de ASCII a dec
mov un,al
mov al,de
mov bl,10
mul bl     ; al = de * 10
add al,un  ; al = de * 10 + un
mov num2,al;se obtiene el segundo numero

;Divisor
xor ax,ax  ;lipiamos registros
xor bx,bx
mov al,num1
mov bl,num2
div bl      ;Se multiplican los numeros

;impresion en 2 caracteres
aam
mov un,al  ;respaldo unidades
mov al,ah
aam
```

```
    mov de,al  ;respaldo decenas
    mov ah,09h
    lea dx,res;Muestra mensaje 1
    int 21h
    mov ah,02h
    mov dl,de
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov dl,un
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov ah,08h
    int 21h
    jmp Menu
;Potencia-----------------------------------------------------
Pot:
    mov ax,03h
    int 10h
    mov ah,09h
    lea dx,msg1;Muestra mensaje 1
    int 21h
    mov ah,01h ;Introdusco las unidades num 1
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov un,al
    mov num1,al;se obtiene el primer numero
    mov ah,09h
    lea dx,msg2;Muestra mensaje 2
```

```
    int 21h
    mov ah,01h ;Introdusco las unidades num 2
    int 21h
    sub al,30h ;Se cambia de ASCII a dec
    mov un,al
    mov num2,al;se obtiene el segundo numero

    ;Potenciador
    xor ax,ax  ;lipiamos registros
    xor bx,bx
    xor cx,cx
    mov al,num1
    mov bl,num1
    mov cl,num2
    dec cl
bucle:
    mul bl      ;Se multiplican los numeros
    loop bucle

    ;impresion en 3 caracteres
    aam
    mov un,al  ;respaldo unidades
    mov al,ah
    aam
    mov de,al  ;respaldo decenas
    mov ce,ah  ;respaldo centenas
    mov ah,09h
    lea dx,res;Muestra mensaje 1
```

```
    int 21h
    mov ah,02h
    mov dl,ce
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov dl,de
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov dl,un
    add dl,30h ;cambio de dec a ascii
    int 21h
    mov ah,08h
    int 21h
    jmp Menu
;Seno-------------------------------------------------------
Sen:
    mov bl,7
    mov si,0
    mov SenX,1
    ; establece el modo de video
    mov ah,0
    mov al,13h     ; 320 x 200 en grafico
    int 10h
    jmp pixelS
    pixelS:
        mov cl,SenX        ; coordenada en X
        mov dl,SenY[si]    ; coordenada en Y
        mov al,COLOR    ; color
```

```
        mov ah,0Ch      ; Escribe un punto, DX = Y, CX = X,
        int 10h
        inc si          ; Incrementa SI
        inc SenX        ; Incrementa variable X
        cmp si,25d      ; Comprar si X es 171
        jb pixelS       ; Si es menor salta a pixel
    mov si,0
    dec bl
    cmp bl,00d
    jne pixelS
    mov ah,08h
    int 21h
    jmp Menu
;Coseno-------------------------------------------------------
Cos:
    mov bl,7
    mov si,0
    mov CosX,1
    ; establece el modo de video
    mov ah,0
    mov al,13h     ; 320 x 200 en grafico
    int 10h
    jmp pixelC
    pixelC:
        mov cl,CosX        ; coordenada en X
        mov dl,CosY[si]    ; coordenada en Y
        mov al,COLOR    ; color
        mov ah,0Ch      ; Escribe un punto, DX = Y, CX = X,
```

```
        int 10h
        inc si          ; Incrementa SI
        inc CosX          ; Incrementa variable X
        cmp si,25d      ; Comprar si X es 171
        jb pixelC       ; Si es menor salta a pixel
    mov si,0
    dec bl
    cmp bl,00d
    jne pixelC
    mov ah,08h
    int 21h
    jmp Menu
;Termina programa-----------------------------------------------
Salir:
    mov ah,04ch
    int 21h
    end
```