

# Seminario de Solución de problemas de Traductores de Lenguajes I

Centro Universitario de Ciencias Exactas en  
ingenierías

Universidad de Guadalajara



Maestro: Tonatiuh Hernandez Casas

Juan Antonio Pérez Juárez  
Código: 215660996  
Carrera: INCO

# Actividad Integradora

Marco Teórico

Primero debemos definir lo que es el EMU8086.

El emu8086 es un emulador del microprocesador 8086 (Intel o AMD compatible) con assembler integrado. A diferencia del entorno de programación en assembler utilizado anteriormente en la cátedra (MASM), este entorno corre sobre Windows y cuenta con una interfaz gráfica muy amigable e intuitiva que facilita el aprendizaje del lenguaje de programación en assembler.

Dado que en un entorno emulado de microprocesador no es posible implementar una interfaz real de entrada/salida, el emu8086 permite interfacear con dispositivos virtuales y emular una comunicación con el espacio de E/S. Para esto, el emu8086 cuenta con una serie de dispositivos virtuales preexistentes en el software base, listos para ser utilizados, entre los que se encuentran una impresora, un cruce de calles con semáforos, un termómetro, un motor paso a paso, etc. No obstante, la cátedra ha desarrollado dispositivos adicionales con características particulares para la realización del segundo trabajo práctico.

Así que se hará el desarrollo de una calculadora simple en este entorno desarrollo haciendo uso de el lenguaje ensamblador, veamos un poco de lo que es ensamblador.

El lenguaje ensamblador o assembler (en inglés: assembler language y la abreviación asm) es un lenguaje de programación que se usa en los microprocesadores. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Además, el hardware de la computadora posee interrupciones de software propias definidas por un número específico, por ejemplo, la interrupción 10h maneja los servicios de video, la 16h el teclado, la 17h la impresora. Una de las interrupciones más utilizada es la 21h, que es la interrupción a los servicios del sistema operativo. La interrupción 21h es muy importante porque entre todas las funciones que posee incluye funciones de manejo del hardware, teclado, video, archivos, terminación de programa, incluso la interpretación de comandos del sistema operativo sin perjuicio de las interrupciones propias del hardware. Es decir, que hay dispositivos que podemos controlar desde su propia interrupción o a través de las del sistema operativo.

Veamos algunas de sus ventajas:

- El código escrito en lenguaje ensamblador posee una cierta dificultad de ser entendido ya que su estructura se acerca al lenguaje máquina, es decir, es un lenguaje de bajo nivel.
- El lenguaje ensamblador es difícilmente portable, es decir, un código escrito para un microprocesador, puede necesitar ser modificado, para poder ser usado en otra máquina distinta. Al cambiar a una máquina con arquitectura diferente, generalmente es necesario reescribirlo completamente.

Pero ahora pasemos al desarrollo de esta actividad.

Pero para poder programar en ensamblador necesitamos hablar antes de eso de las interrupciones en ensamblador.

Las interrupciones son habilidades de comunicación entre hardware y software con el programa que se está desarrollando que le permiten ampliar su capacidad al manejar hardware a través del sistema operativo, o recibir datos o mensajes del sistema operativo, avisar que el programa principal ha terminado, establecer la comunicación hacia algún dispositivo de entrada o salida como la pantalla, el teclado, el mouse o una impresora.

Es muy importante que un programa de cómputo posea la capacidad de interactuar con el usuario final, por lo menos a través de la pantalla y el teclado en el caso de una computadora personal. En el ensamblador utilizamos el servicio de interrupciones del sistema operativo para comunicarnos con el hardware.

Es así que podemos programar rutinas que muestren en pantallas resultados o, incluso, gráficos que muestren al usuario diversas informaciones útiles; también es posible hacer subrutinas para que se introduzcan datos como cadenas de texto o valores numéricos e, incluso, comandos, a través del teclado y el ratón.

## Desarrollo de la Actividad

Desarrollar una calculadora en lenguaje ensamblador implementando las operaciones aritméticas básicas (Suma, resta, multiplicación, división y potencia).

Así que ahora veamos el código en Ensamblador para hacer esto posible.

```
Unset
;Hecho por Juan Antonio Perez Juarez
;INCO 2024B
;215660996

.model small
.stack 100h
.data
    ; Datos del programa 1 (Calculadora)
    msg1 db 10,13,'Ingrese el primer numero: $'
    msg2 db 10,13,'Ingrese el segundo numero: $'
    msgSuma db 10,13,'Suma: $'
    msgResta db 10,13,'Resta: $'
    msgMult db 10,13,'Multiplicacion: $'
    msgDiv db 10,13,'Division: $'
    msgPot db 10,13,'Potencia: $'
    num1 db ?
    num2 db ?
    result dw ?

    ; Datos del programa 2 (angulo y trigonometria)
    msgAng db 10,13,'Ingrese un angulo en grados (0-90): $'
    msgSeno db 10,13,'Seno: $'
    msgCoseno db 10,13,'Coseno: $'
```

```

    msgGrafica db 10,13,'Grafica de Seno y Coseno: $'
    msgLinea db ' * $'
    msgEspacio db ' $'
    anguloGrados db ?
    resultadoSeno dw 0
    resultadoCoseno dw 0

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Ejecutar el primer programa (Calculadora)
    call calculadora

    ; Ejecutar el segundo programa (angulo y trigonometría)
    call angulo_trigonometria

    ; Terminar el programa
    jmp fin

; --- Calculadora ---
calculadora proc
    ; Mostrar mensaje para primer numero
    lea dx, msg1
    mov ah, 09h
    int 21h

    ; Leer primer numero
    mov ah, 01h
    int 21h
    sub al, 30h
    mov num1, al

    ; Mostrar mensaje para segundo numero
    lea dx, msg2
    mov ah, 09h
    int 21h

    ; Leer segundo numero
    mov ah, 01h
    int 21h
    sub al, 30h
    mov num2, al

    ; Suma
    lea dx, msgSuma

```

```
mov ah, 09h
int 21h
mov al, num1
add al, num2
mov ah, 0
mov result, ax
call mostrar_resultado
```

```
; RESTA
lea dx, msgResta
mov ah, 09h
int 21h
```

```
mov al, num1
sub al, num2
mov ah, 0
mov result, ax
call mostrar_resultado
```

```
; MULTIPLICACION
lea dx, msgMult
mov ah, 09h
int 21h
```

```
mov al, num1
mul num2
mov result, ax
call mostrar_resultado
```

```
; DIVISION
lea dx, msgDiv
mov ah, 09h
int 21h
```

```
mov ax, 0
mov al, num1
div num2
mov ah, 0
mov result, ax
call mostrar_resultado
```

```
; POTENCIA
lea dx, msgPot
mov ah, 09h
int 21h
```

```
mov cx, 0
mov cl, num2
```

```

        mov ax, 1
        mov bl, num1

        cmp cl, 0
        je fin_potencia

ciclo_potencia:
        mul bl
        loop ciclo_potencia

fin_potencia:
        mov result, ax
        call mostrar_resultado

mostrar_resultado proc
        mov ax, result
        mov cx, 0
        mov bx, 10

convertir:
        mov dx, 0
        div bx
        push dx
        inc cx
        cmp ax, 0
        jne convertir

mostrar:
        pop dx
        add dl, 30h
        mov ah, 02h
        int 21h
        loop mostrar

        ret

; --- trigonometria ---
angulo_trigonometria proc
        ; Mostrar mensaje para angulo
        lea dx, msgAng
        mov ah, 09h
        int 21h

        ; Leer angulo
        mov ah, 01h
        int 21h
        sub al, 30h

```

```

    mov anguloGrados, al

    ; Calcular seno y coseno
    call calcular_seno_coseno

    ; Mostrar resultados
    lea dx, msgSeno
    mov ah, 09h
    int 21h
    mov ax, [resultadoSeno]
    call mostrar_resultado

    lea dx, msgCoseno
    mov ah, 09h
    int 21h
    mov ax, [resultadoCoseno]
    call mostrar_resultado

    ; Graficar seno y coseno
    call graficar_trigonometrica

    ret
angulo_trigonometria endp

    ; Procedimiento para calculo de seno y coseno
calcular_seno_coseno proc
    ; Tabla de aproximaciones para 0 a 90 grados
    xor ax, ax
    mov al, anguloGrados

    ; Aproximaciones de seno
    cmp al, 30
    jle seno_menor_30
    cmp al, 60
    jle seno_entre_30_60
    jmp seno_mayor_60

seno_menor_30:
    mov word ptr [resultadoSeno], 50 ; 0.5
    jmp calcular_coseno

seno_entre_30_60:
    mov word ptr [resultadoSeno], 87 ; 0.87
    jmp calcular_coseno

seno_mayor_60:
    mov word ptr [resultadoSeno], 100 ; 1.0
    jmp calcular_coseno

```



```

calcular_coseno:
    ; Aproximaciones de coseno
    cmp al, 30
    jle coseno_menor_30
    cmp al, 60
    jle coseno_entre_30_60
    jmp coseno_mayor_60

coseno_menor_30:
    mov word ptr [resultadoCoseno], 87 ; 0.87
    ret

coseno_entre_30_60:
    mov word ptr [resultadoCoseno], 50 ; 0.5
    ret

coseno_mayor_60:
    mov word ptr [resultadoCoseno], 10 ; 0.1
    ret
calcular_seno_coseno endp

; Procedimiento para graficar funciones trigonometricas
graficar_trigonometrica proc
    lea dx, msgGrafica
    mov ah, 09h
    int 21h

    ; Primera linea - eje horizontal
    mov cx, 10
grafica_eje:
    lea dx, msgEspacio
    mov ah, 09h
    int 21h
    loop grafica_eje

    ; Calcular altura de seno y coseno
    mov ax, [resultadoSeno]
    mov bl, 5
    div bl
    mov cl, al ; Altura de linea de seno

    mov ax, [resultadoCoseno]
    mov bl, 5
    div bl
    mov ch, al ; Altura de linea de coseno

    ; Graficar seno

```

```

        mov cx, 5
grafica_seno:
    push cx
    lea dx, msgLinea
    mov ah, 09h
    int 21h
    pop cx
    loop grafica_seno

    ; Nueva linea
    mov dl, 10
    mov ah, 02h
    int 21h

    ; Graficar coseno
    mov cx, 5
grafica_coseno:
    push cx
    lea dx, msgLinea
    mov ah, 09h
    int 21h
    pop cx
    loop grafica_coseno

    ret
graficar_trigonometrica endp

; Procedimiento para mostrar resultado
mostrar_resultadoAri proc
    mov cx, 0
    mov bx, 10

convertirAri:
    mov dx, 0
    div bx
    push dx
    inc cx
    cmp ax, 0
    jne convertir

mostrarAri:
    pop dx
    add dl, 30h
    mov ah, 02h
    int 21h
    loop mostrar

    ; Salto de linea

```

```

mov dl, 10
mov ah, 02h
int 21h

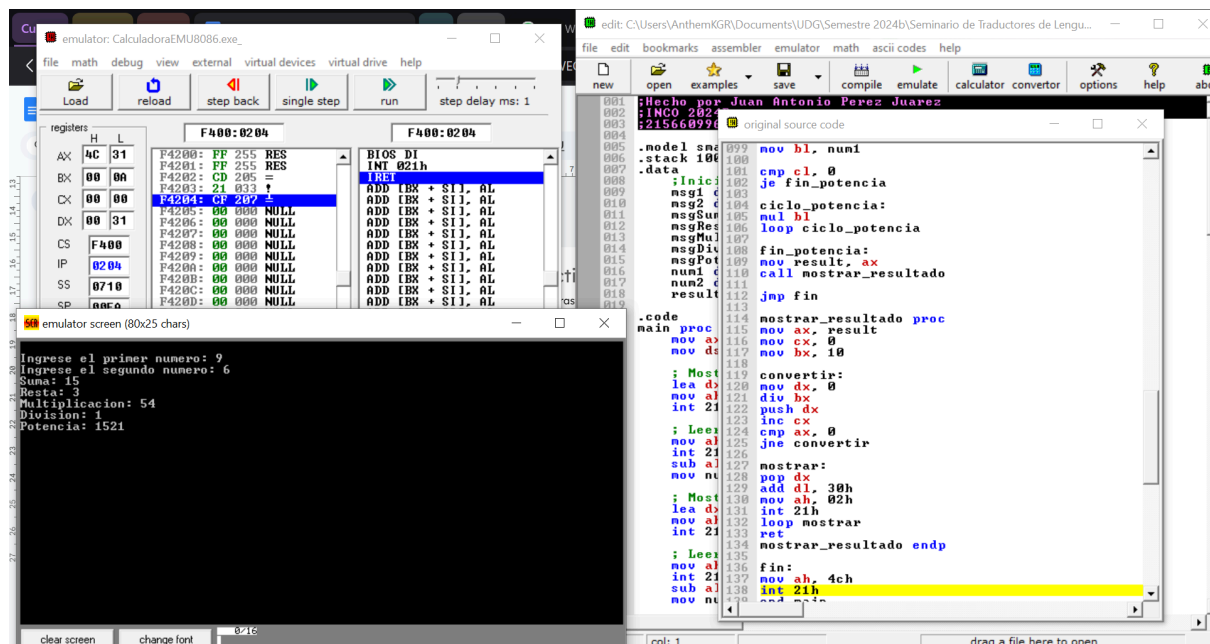
ret
mostrar_resultadoAri endp

fin:
mov ah, 4ch
int 21h
end main

```

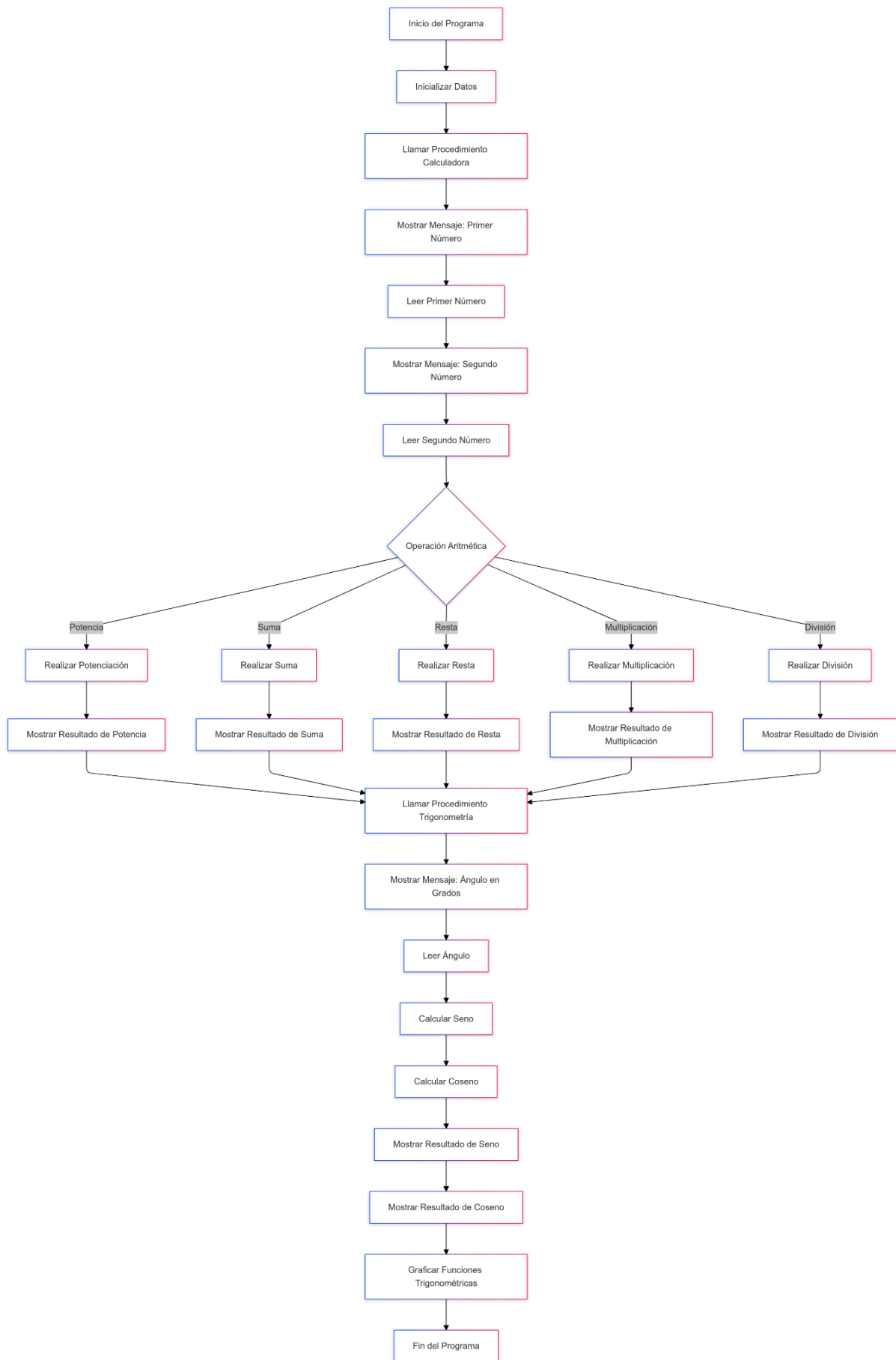
## Resultados de la Actividad

Ahora veamos unas cuantas capturas de pantalla de lo que hace este código.



Así se ve una ejecución del código completa usando 2 números. (Estas capturas son de la versión anterior de mi programa en la que no calculaba el seno y coseno)

El flujo de desarrollo del programa funciona de esta manera.  
(En la siguiente hoja)



Primero Inicia los segmentos, eso es sencillo.

Y pide al usuario que escriba el primer número, por cuestiones de tiempo solo pude hacer que fueran números del 0 al 9. En ambos casos, en los dos números.

Una vez guardado los números en una dirección de memoria comienza a procesarlos para hacerlos de manera automática, sin que el usuario haga nada, cuando termina de procesar los datos. Los despliega en la consola para comprobar su resultado.

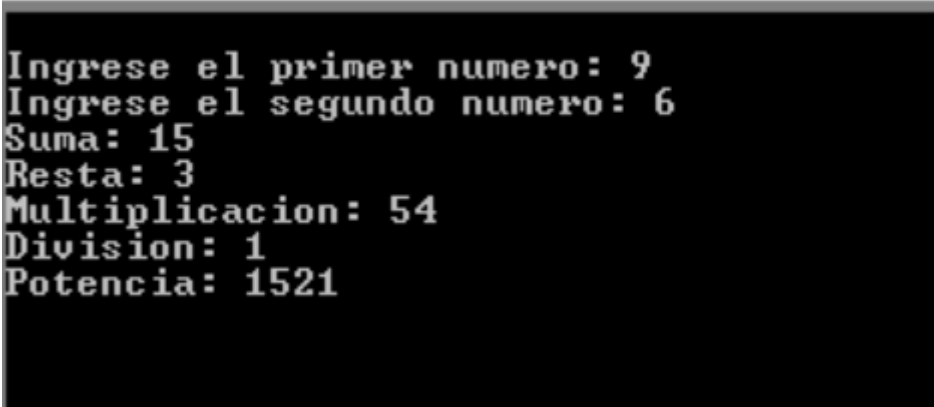
Aunque no está del todo terminado porque no puede manejar la división por cero, en el apartado de división y tampoco maneja los números negativos en la resta.

Cuando este proceso termina, le pide al usuario que ingrese el valor de un ángulo para que pueda graficar de manera super rudimentaria, con el mismo número hace una aproximación al seno y al coseno con una tabla de valores predefinida por mí. (Y por una IA, tengo que ser honesto).

Después de la graficación la ejecución del programa termina y devuelve el control al sistema operativo.

Así que veamos unas capturas de pantalla de como funciona el programa actualizado.

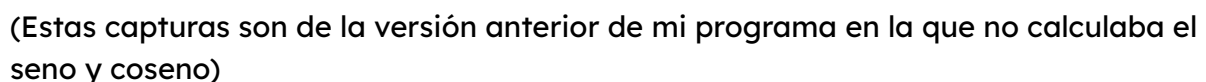
 emulator screen (80x25 chars)



```
Ingrese el primer numero: 9
Ingrese el segundo numero: 6
Suma: 15
Resta: 3
Multiplicacion: 54
Division: 1
Potencia: 1521
```

Es por eso que trato de poner el numero mas grande al inicio por que sigue la lógica de programación de Numero1 menos Numero2.

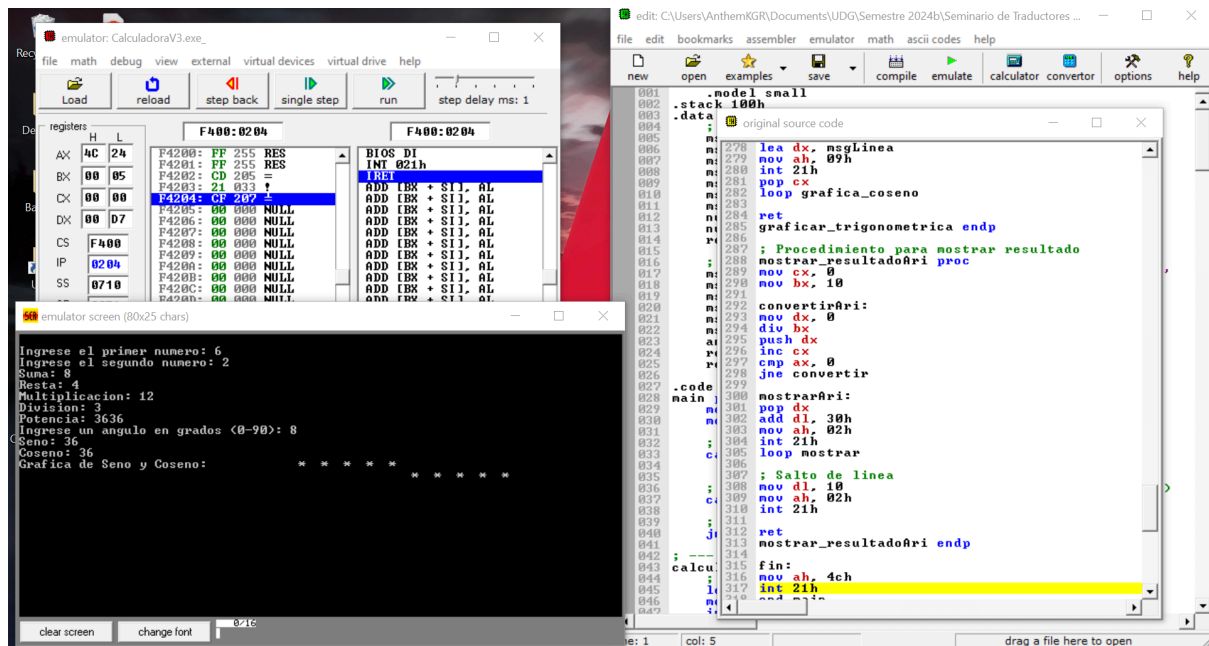
Así me evito quedar en ridículo porque sale un número negativo sin convertir a Ascii. (Que trucazo, ¿No?)



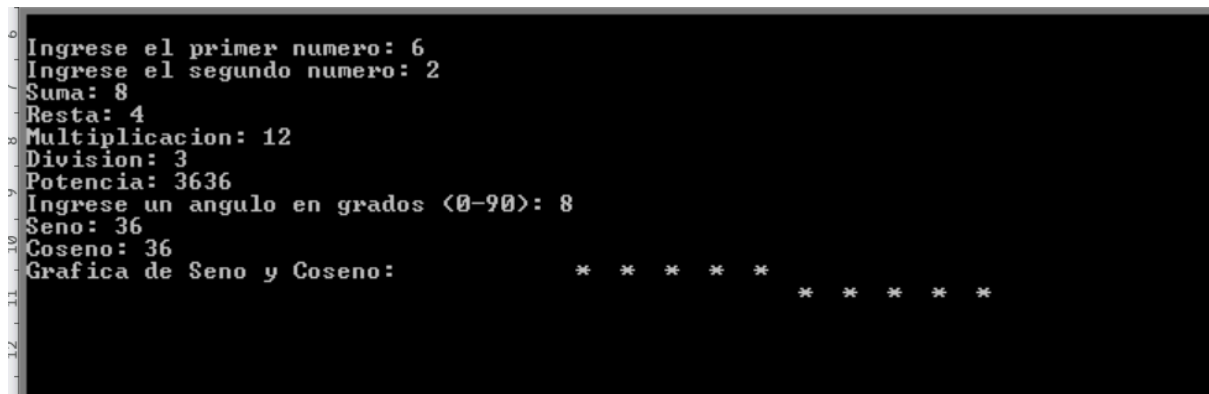
Y pues ya una con un número negativo, ya que, ni modo.

Y ahora veamos las capturas de pantalla del programa actualizado y podría decirse que terminado.

(Siguiendo página)

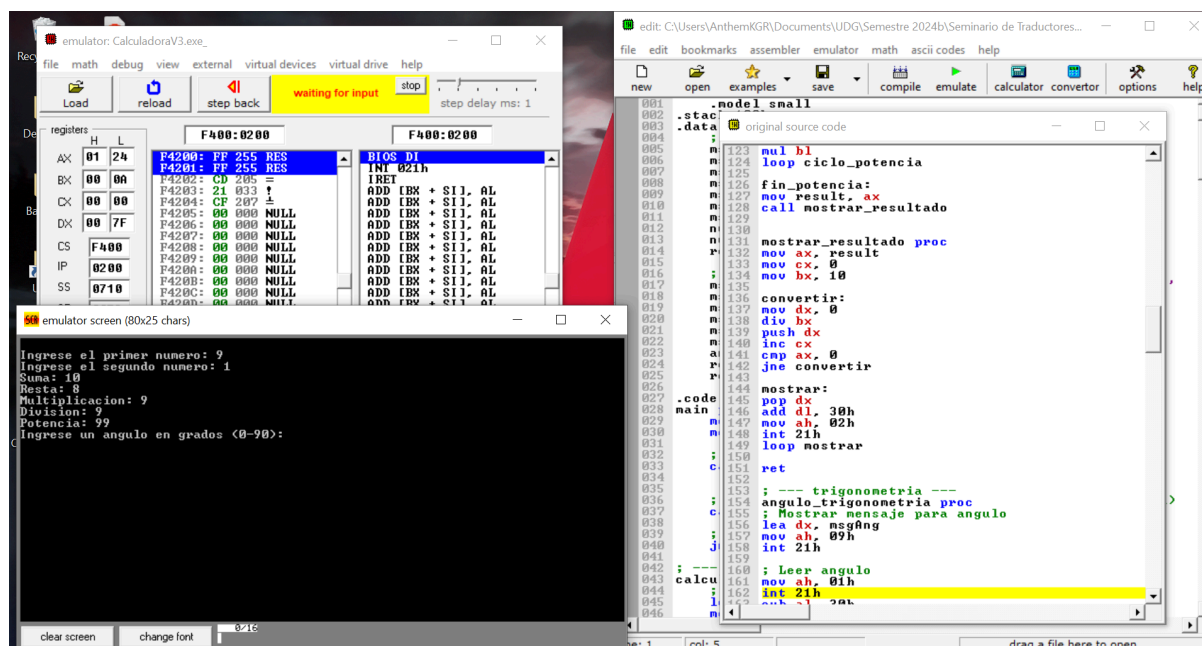


Ahora cuando termina las funciones aritméticas simple comienza preguntando al usuario el número de un ángulo, aquí para no complicarme mucho solo admite un dígito, del 0 al 9, por lo que solo tengo eso valores, después calcula la aproximación muy redondeada y la gráfica según una tabla de valores ya definida.



Ahí es donde se puede ver brevemente la graficación de la tabla, que básicamente es una impresión de caracteres en pantalla, los cuales se hacen con un ciclo cuyo valor está contenida en una variable, en seno está en el registro CL y el coseno en CH. Estos valores representan la altura de la gráfica, bueno, si a esto se le puede llamar gráfica.

Veamos otra captura de pantalla.  
(Siguiente página)



Parece que funciona bien ¿no?

## Reflexión

Realmente me odie mucho en esta actividad, procrastiné demasiado lo que causó que no pudiera hacer más, para que el proyecto y sobre todo el código fuera muy presentable, fue una semana muy pesada en el trabajo, no es justificación pero fue un factor que me mermó, en fin, dejando de lado las excusas, para mí este programa fue realmente desafiante, como le había comentado en reportes anteriores no soy bueno en lógica de programación y menos en un lenguaje de programación tan complicado como lo es ensamblador.

Además de tener todas las tentaciones de hacerlo con IA, como por ejemplo mi buen amigo ChatGPT, pero creo que es algo digno de presentar, no es mucho, pero es mi trabajo honesto, el viernes fue cuando ya no pude avanzar en el código y la verdad use tutoriales y páginas de dudosa procedencia pero todo bien.

No pude enviar el reporte ni el código antes porque tuve un corte de internet en mi zona por vandalismo, así que no pude entregar mi reporte a tiempo, pero eso me dio tiempo para complementar aún más mi reporte y un poco el código, la verdad no fue la gran cosa, de hecho creo que solo me metí el pie porque un módulo no funciona como debe funcionar, dejando de lado la presión y las ganas de pasar, aprovecharé estas líneas para hacer una reflexión del curso en sí, muchas gracias por su tiempo profe y su paciencia hacia mí, me sirvió de mucho por que me había frustrado anteriormente con el lenguaje ensamblador, aun sigo siendo malísimo en ello pero al menos tengo una idea de lo que voy haciendo y sinó tengo herramientas que me pueden orientar.

Gracias por ayudarme a comprender de mejor manera este lenguaje y gracias por su paciencia.

Y una disculpa por no ser el alumno más dedicado, aún estoy trabajando en ello.



## Bibliografía:

colaboradores de Wikipedia. (2024, November 13). Lenguaje ensamblador. Wikipedia, La Enciclopedia Libre.

[https://es.wikipedia.org/wiki/Lenguaje\\_ensamblador](https://es.wikipedia.org/wiki/Lenguaje_ensamblador)

De Arquitectura, V. T. L. E. (2016, November 16). Emulador 8086. Arquitectura De Hardware.

<https://arquitectura9728.wordpress.com/2016/11/16/emulador-8086/>

ASCIIFlow. (n.d.). <https://asciiflow.com/#/>

Ensamblador - CALCULADORA EN ASSEMBLER. (n.d.).

<https://www.lawebdelprogramador.com/foros/Ensamblador/1292366-CALCULADORA-EN-ASSEMBLER.html>

ojoanalogo/simple-asm-calc: 🇺🇸 Calculadora simple hecha en asm 8086. (n.d.). GitHub. <https://github.com/ojoanalogo/simple-asm-calc>

About Calculator in asm 8086 - 2 Questions. (n.d.). Stack Overflow.

<https://stackoverflow.com/questions/36811810/about-calculator-in-asm-8086-2-questions>

Simple-Calculator-Using-Assembly-Language/Simple Calculator.asm at master · IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language. (n.d.). GitHub.

<https://github.com/IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language/blob/master/Simple%20Calculator.asm>

Simple-Calculator-Using-Assembly-Language/Simple Calculator.asm at master · IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language. (n.d.). GitHub.

<https://github.com/IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language/blob/master/Simple%20Calculator.asm>

Haseeeb21/Calculator-Assembly: A Scientific Calculator coded in Assembly Language which takes numbers as input and performs the selected operations and displays the result. Operations include +, -, x, /, %, Square, Cube and some Binary Operations. (n.d.). GitHub.

<https://github.com/Haseeeb21/Calculator-Assembly>

colaboradores de Wikipedia. (2022, January 12). Int 10h. Wikipedia, La Enciclopedia Libre. [https://es.wikipedia.org/wiki/Int\\_10h](https://es.wikipedia.org/wiki/Int_10h)

Funciones del DOS. (s. f.)

[http://arantxa.ii.uam.es/~gdrivera/labetcii/int\\_dos.htm#:~:text=INT%2021H%](http://arantxa.ii.uam.es/~gdrivera/labetcii/int_dos.htm#:~:text=INT%2021H%20)

Grupo de Arquitectura de Computadores y Diseño Lógico. UEX, 1997., Germán Galeano Gil & Juan A. Gómez Puildo. (1996). Tabla de interrupciones (1.a ed., Vol. 1).

[http://ebadillo\\_computacion.tripod.com/ensamblador/8086\\_int.pdf](http://ebadillo_computacion.tripod.com/ensamblador/8086_int.pdf)

Programación en ensamblador. (n.d.).

<https://moisesrbb.tripod.com/unidad2.htm>

Universidad de Sevilla. (s. f.). Ensamblador 8086/88 (Benemérita Universidad Autónoma de Puebla, Ed.; 2.a ed., Vol. 1).

[https://www.cs.buap.mx/~mgonzalez/asm\\_mododir2.pdf](https://www.cs.buap.mx/~mgonzalez/asm_mododir2.pdf)

Newest “assembly+math” Questions. (n.d.). Stack Overflow.

<https://stackoverflow.com/questions/tagged/assembly+math?tab=Newest>

Assembly Programming tutorial. (n.d.).

[https://www.tutorialspoint.com/assembly\\_programming/index.htm](https://www.tutorialspoint.com/assembly_programming/index.htm)

**Que joya de página es esta, se la recomiendo 100% profe.**

Intel® 64 and IA-32 Architectures Software Developer Manuals. (n.d.).

Intel.

<https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html>

Build software better, together. (n.d.). GitHub.

<https://github.com/search?q=calculadora%20ASM&type=repositories>