Portada

Proyecto: Calculadora de Números Racionales en Turbo Assembler
Autor: Braunny Madrigal Barrantes (C24436)
Curso: CI-0118 Lenguaje Ensamblador
Institución: Universidad de Costa Rica
Escuela: Escuela de Ciencias de la Computación e Informática
Profesor: Dr. Carlos Vargas
Ciclo: I Ciclo del 2024

Contenido

1. Descripción del problema

Se programa en lenguaje ensamblador la simulación de “una calculadora” que únicamente opere con números racionales. La calculadora permite únicamente las operaciones de suma, resta, multiplicación y división. Se programan las entradas y salidas de datos del programa con el uso de distintas interrupciones del BIOS. El programa tiene como objetivo entender cómo realizar aritmética en ensamblador, Turbo Assembler si se es específico, y el cómo se deben utilizar las interrupciones más importantes del mismo.

2. Alcances y limitaciones del programa

Alcances:

El programa abarca operaciones aritméticas básicas como suma, resta, multiplicación y división, permitiendo a los usuarios realizar cálculos con números racionales de manera sencilla. La interfaz de usuario está diseñada para mostrar tanto la operación actual como el resultado, proporcionando una experiencia visual clara y directa. La interfaz de usuario también cuenta con una ventana de error que se actualiza a tiempo real, la misma indica si lo que hasta al momento ha escrito el usuario es válido. La interfaz consigue el diseño clásico gracias al uso de

interrupciones del BIOS (INT 10H para las impresiones e INT 21H para generar un retraso a la hora de procesar lo hecho por el usuario final).

El programa maneja entradas del teclado y del mouse mediante el uso de interrupciones del BIOS (INT 16H para la salida en pantalla e INT 33H para la entrada del teclado y mouse), asegurando una interacción eficiente con el usuario.

El programa también es capaz de verificar operaciones inválidas, como dividir por cero, escribir un número en lugar de un operador o escribir un operador en lugar de un número.

Además del aspecto clásico, se busca recrear la sensación de trabajar con una calculadora antigua. Es por ello que se decide usar la interrupción INT 21H para generar una espera o retraso a la calculadora, ese retraso obliga al usuario a mantener presionado el botón deseado antes de que el mismo sea mostrado en pantalla. Se considera que este añadido dota de personalidad a la calculadora.

Por último, se considera importante mencionar el porque se ha descrito en reiteradas ocasiones que la calculadora cuenta con soporte de teclado. Dicho soporte existe ya que el usuario puede salirse de la calculadora en cualquier momento con simplemente presionar la tecla ESC.

Limitaciones:

Hay un formato estricto en el que las operaciones deben ser digitadas. El formato es el siguiente: “ `A/B operador C/D =` ”, donde las letras representan los números como tal y se establece que los números deben de ser escritos como fracciones siempre, eso quiere decir que para escribir cero, se tendrian que, por ejemplo, escribir un numerador cero, la barra de división y luego cualquier número a utilizar como denominador.

Cada número, representados por letras en el párrafo anterior, pueden ser de máximo dos dígitos,. Sobre el operador igual, se tiene que este solo puede ser escrito una vez se cumpla con el formato

establecido, si no, el programa detectara error y lo comunicará. Al programa detectar error este se asegura que no computará la operación invalida que el usuario acabase de digitar.

Además, para seguir con el formato, se establece que el programa no opera con números negativos, ya que estos disturban el formato anteriormente establecido. Además de que las calculadoras básicas no trabajan con negativos. También, es importante mencionar, que obviamente la calculadora solo es capaz de computar una operación a la vez, es decir solo puede haber un operador en ventana (además de los que dividen a las dos fracciones).

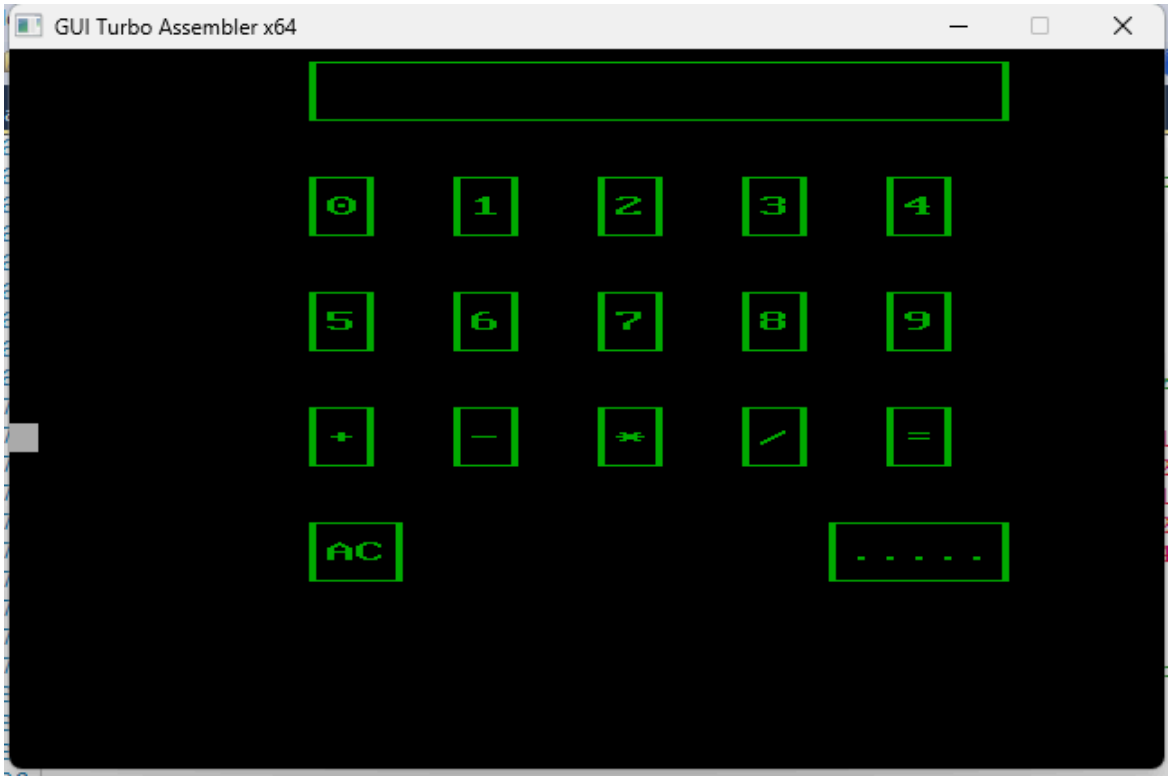
Otra limitación, tiene que ver en la forma en que los resultados son impresos, ya que por sugerencia del profesor encargado del curso se establece que no es necesario simplificar las operaciones. Por tanto, los resultados de cualquier operación que la calculadora realice se encuentran sin simplificar.

Por último, se tiene la limitación más importante del programa. Esta no se origina por culpa de cómo ha sido programada la calculadora, si no que se da por culpa del emulador del ensamblador de Turbo Assembler que se utiliza. La limitación tiene que ver con la forma de usar el mouse, ya que el emulador utilizado presenta un “bug” al manejar las coordenadas del mouse y la única forma conocida de manejarlo es comunicarle al usuario como iniciar de forma correcta del mismo. Dicho uso correcto es simple, basta con que el primer click que el usuario da en la ventana generada por el emulador al correr el programa sea en la posición del puntero mostrado.

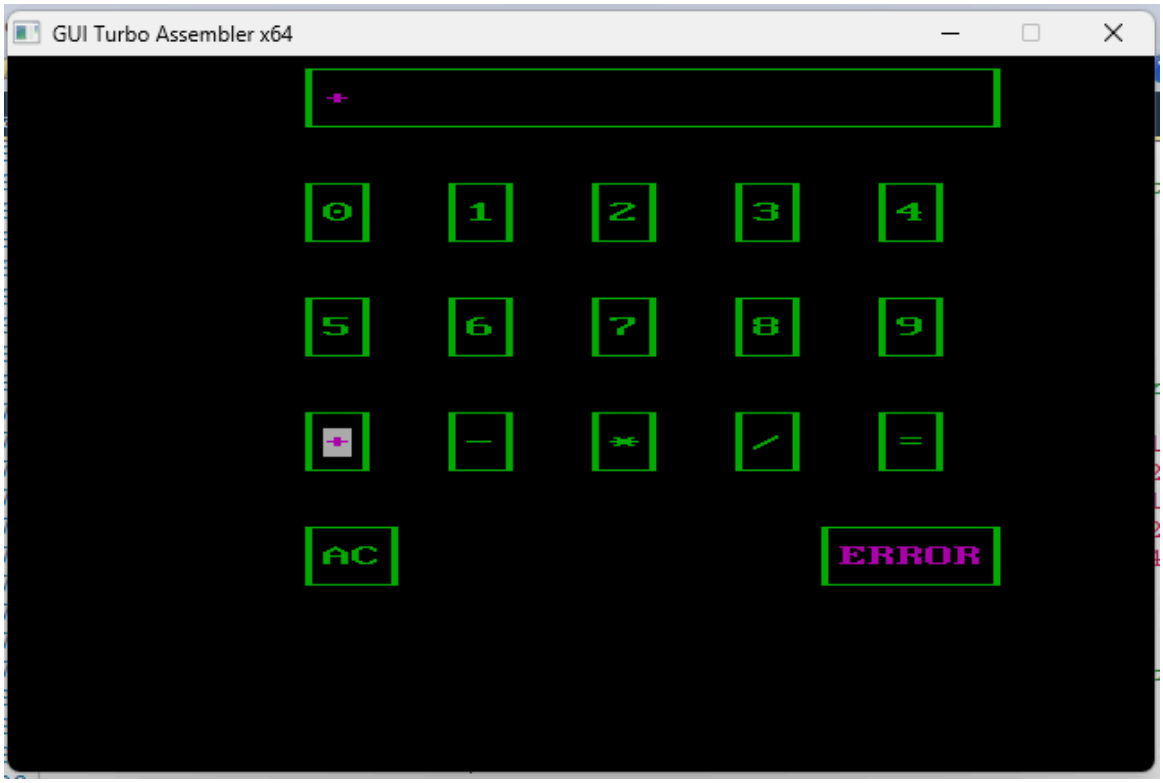
3. Interfaz de usuario

Todas las impresiones en la misma han sido generadas haciendo uso de la interrupción del BIOS INT 10H.

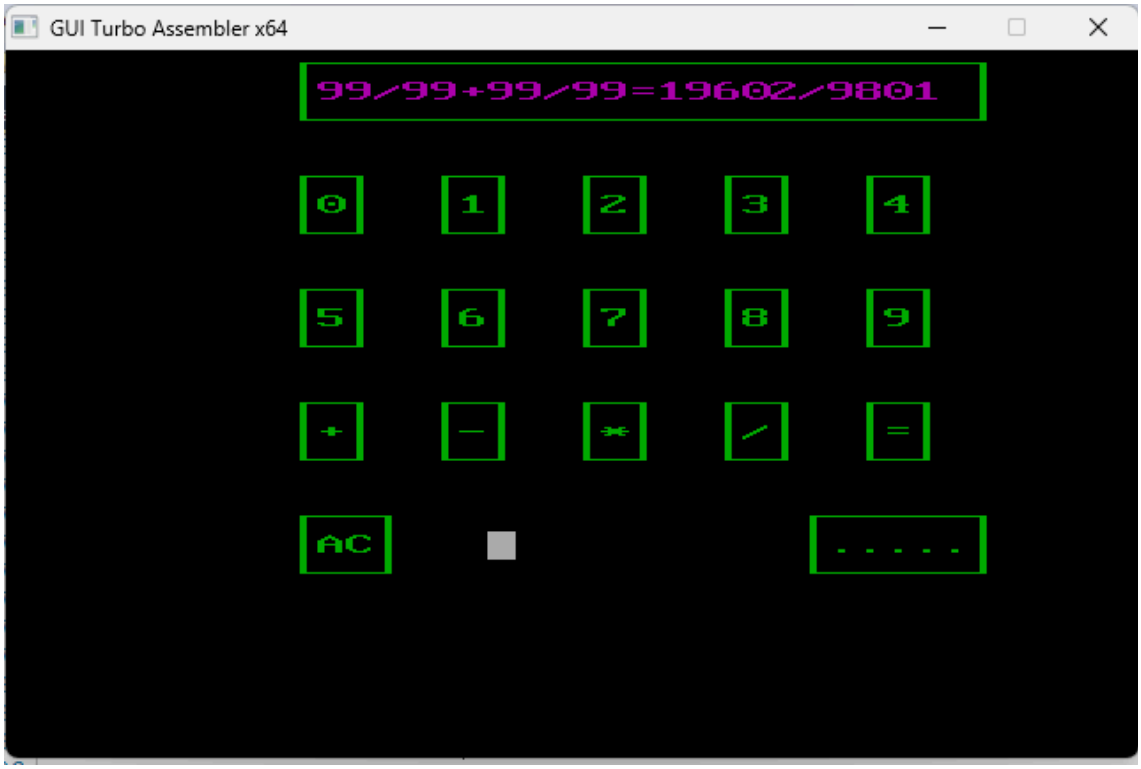
Lo primero que ve el usuario al ejecutar el programa:



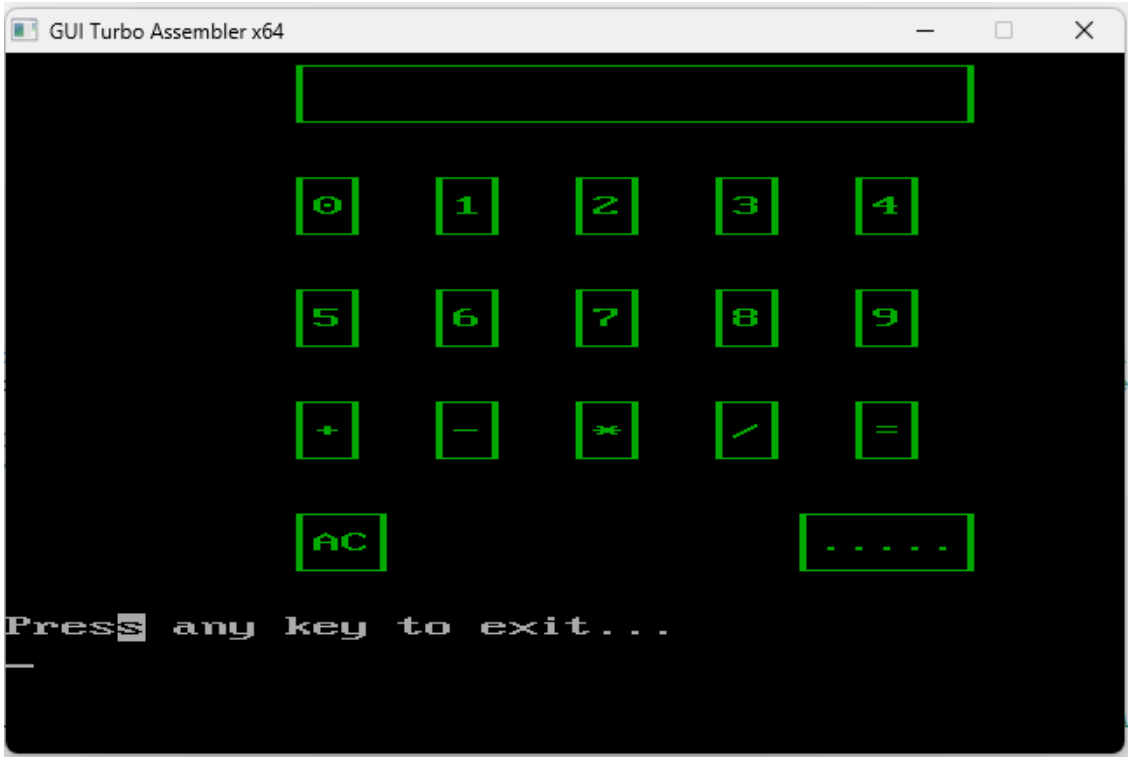
Lo que ve el usuario al proveer algo invalido en la ventana de operaciones:



Lo que ve el usuario al usar el operador igual, si en la ventana de operaciones se encuentra escrita una operación válida:



Lo que ve el usuario si decide presionar la tecla ESC:



4. Algoritmos utilizados

En este programa hay muy pocos procedimientos que tengan un mínimo de complejidad algorítmica a nivel implementación. Pero los que sí lo tuvieron, se decidió hacer un pseudocódigo en estilo python para desarrollarlos antes de convertirlos a lenguaje ensamblador.

Algoritmo para determinar que se debería hacer según el valor almacenado en “auxB01”.

```
def doWhat():  # Algoritmo para decidir que hacer con el valor en auxB01
    if(auxB01 != 0) and (contArray < 12):
        if (valAsciiAct != "d"):
            print(valAsciiAct) en col(contVent)
            ++contVent
            call storeInArray()
        else:
            call cleanVentanita
    return
```

Algoritmo para determinar de qué forma se debería almacenar “auxB01” en el array. Osea que procedimiento de almacenamiento llamar.

```
def storeInArray():
    switch = {
        0: FirstNum,
        1: SeconNum,
        2: Operator,
        3: FirstNum,
        4: SeconNum,
        5: Operator,
        6: FirstNum,
        7: SeconNum,
        8: Operator,
        9: FirstNum,
        10: SeconNum,
        11: Operator
    }
    func = switch.get(contArray, lambda: None)
    func()
    return
```

Las funciones que dictan las tres distintas formas de almacenar el auxB01 en el array.

```
def FirstNum():
    ventanita[contArray] = number(valAsciiAct)
    ++contArray
    return

def SeconNum():
    if(47 < valAsciiAct < 58):  # is a number
        ventanita[contArray] = number(valAsciiAct)
        ++contArray
    else:
        auxaux = ventanita[contArray-1]
        ventanita[contArray-1] = 0
        ventanita[contArray] = auxaux
        ventanita[contArray+1] = valAsciiAct
        contArray = contArray + 2
    return

def Operator():
    ventanita[contArray] = valAsciiAct
    ++contArray
    return
```

Algoritmo que chequea, en base al valor almacenado en las distintas posiciones del array, si hay error o no.

```
def CheckErr():
    error = 0
    if(arrayVenta[0] > 9):
        print("error")
    if(arrayVenta[1] > 9):
        print("error")
    if(arrayVenta[2] != diviOpAscii):
        print("error")
    if(arrayVenta[3] > 9):
        print("error")
    if(arrayVenta[4] > 9 or arrayVenta[4] < 1):
```

```
        print("error")
    if(arrayVenta[5] < multOpAscii or arrayVenta[5] > diviOpAscii):
        print("error")
    if(arrayVenta[6] > 9):
        print("error")
    if(arrayVenta[7] > 9):
        print("error")
    if(arrayVenta[8] != diviOpAscii):
        print("error")
    if(arrayVenta[9] > 9):
        print("error")
    if(arrayVenta[10] > 9 or arrayVenta[10] < 1):
        print("error")
    if(arrayVenta[11] > 0 and arrayVenta[11] != iguaOpAscii):
        print("error")
    return
```

Algoritmo que chequea si es necesario ir a computar un resultado y obtiene los datos que serán utilizados, ello en base a los ascii digitados por el usuario.

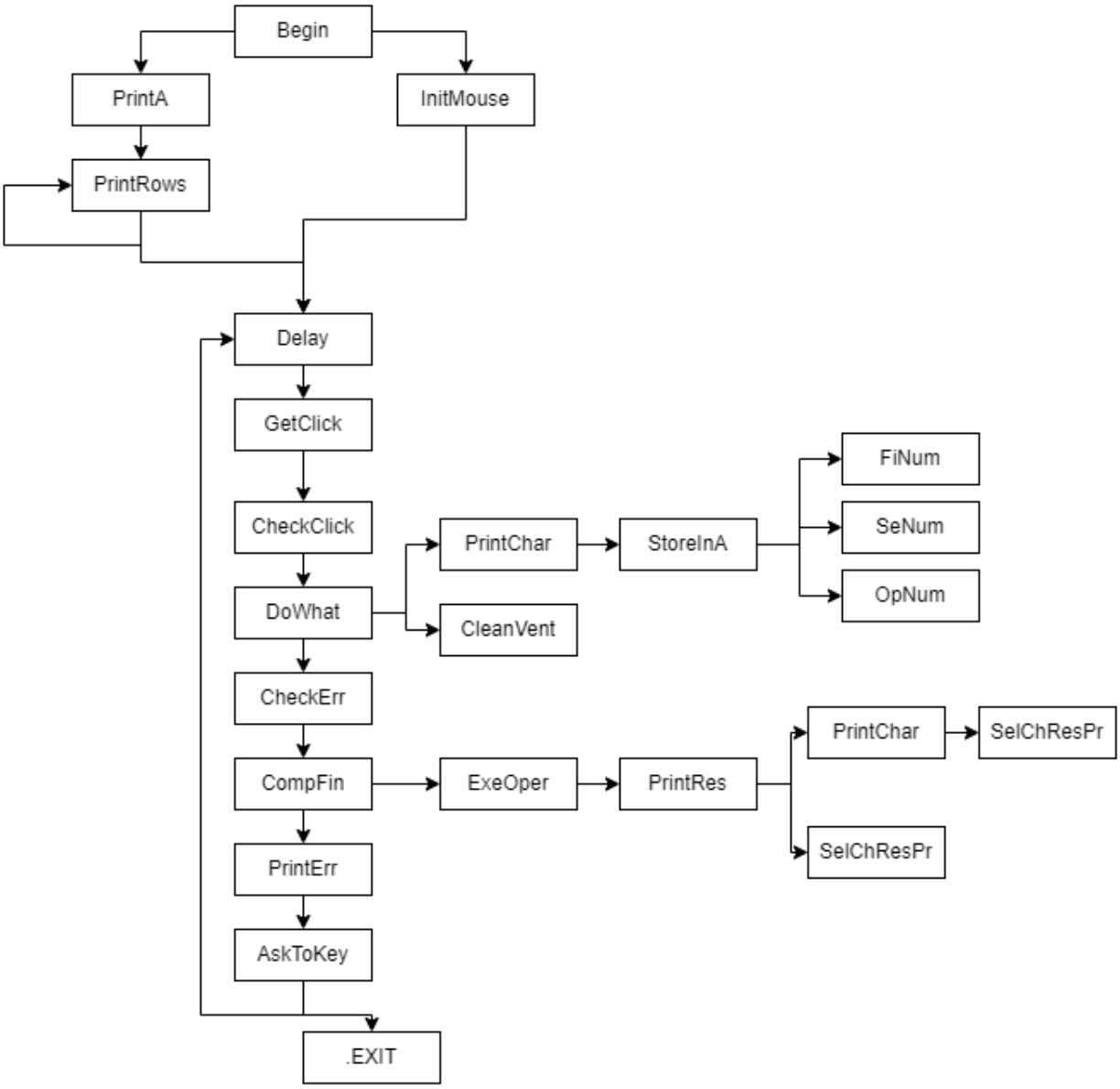
```
def CompFin():
    if (error != 1 and arrayVenta[11] == iguaOpAscii):
        numA = (arrayVenta[0]*10 + arrayVenta[1])
        numB = (arrayVenta[3]*10 + arrayVenta[4])
        opeR = arrayVenta[5]
        numC = (arrayVenta[6]*10 + arrayVenta[7])
        numD = (arrayVenta[9]*10 + arrayVenta[10])
        call ExeOper()
        call PrintFinRes()
    return
```

Algoritmo que determina cómo hacer cada una de las operaciones.

```
def ExeOper():
    if(opeR == sumaOpAscii):
        jmp suma
    if(opeR == restOpAscii):
        jmp resta
    if(opeR == multOpAscii):
        jmp multi
```

```
    if(operR == diviOpAscii)
        jmp divi
    suma:
        E = (numA * numD) + (numC * numB)
        F = numB * numD
        jmp retExeOper
    resta:
        E = (numA * numD) - (numC * numB)
        F = numB * numD
        jmp retExeOper
    multi:
        E = numA * numC
        F = numB * numD
        jmp retExeOper
    divi:
        E = numA * numD
        F = numB * numC
        jmp retExeOper
    retExeOper:
        return
```

5. Estructura del programa



6. Descripción de constantes y de variables esenciales

Constantes:

- interFilaTamanno: Cantidad de filas en el array interf.
- teclaEscape: Valor ASCII de la tecla escape.
- sumaOpAscii: Valor ASCII del operador suma.
- restOpAscii: Valor ASCII del operador resta.
- multOpAscii: Valor ASCII del operador de multiplicar.

- diviOpAscii: Valor ASCII del operador de dividir.
- iguaOpAscii: Valor ASCII del operador de igual.
- miniRecFilG: Fila en la que se escribe en el mini rec de los errores.
- miniRecColI: Primer columna del mini rec de los errores.
- miniRecColF: Última columna del mini rec de los errores.
- sizeArrNumAscii: Tamaño del array rNumAscii.
- mensajeError: Mensaje de error que se escribe en el mini recuadro.

Variables:

- clickIzq: Funciona como booleano para determinar si se ha presionado el click izquierdo del mouse.
- mouseCol: Número de columna en la que se encuentra el puntero del mouse.
- mouseFil: Número de fila en la que se encuentra el puntero del mouse.
- seconds: Segundos para el delay.
- auxB01: Variable auxiliar de 8 bits.
- auxB02: Variable auxiliar de 8 bits.
- auxW01: Variable auxiliar de 16 bits.
- arrayVenta: Array de valores que el usuario ha escrito en la ventanita.
- contaArray: Índice para saber en qué parte del array arrayVenta toca insertar.
- contaVenta: Columna en la que toca imprimir en la ventana.
- FilaSup: Fila superior del botón en tal posición.
- ColuIzq: Columna izquierda del botón en tal posición.
- FilaInf: Fila inferior del botón en tal posición.
- ColuDer: Columna derecha del botón en tal posición.
- Caracte: Valor ASCII del caracter en tal posición.
- varError: Funciona como booleano para saber si el usuario ingresa datos a la calculadora (ventanita) de forma incorrecta.
- varNegativo: Funciona como booleano para saber si toca imprimir un signo de negativo antes del resultado.
- numA: AA, considerando ambos dígitos.
- numB: BB, considerando ambos dígitos.

- opeR: Operador.
- numC: CC, considerando ambos dígitos.
- numD: DD, considerando ambos dígitos.
- numE: EEEE, considerando los 4 dígitos.
- numF: FFFF, considerando los 4 dígitos.
- impY: Funciona como booleano para saber si ya se imprimió el resultado.
- arrNumAscii: Array con valores a imprimir de número E o F en ASCII.
- auxB01: Variable auxiliar de 8 bits.
- auxB02: Variable auxiliar de 8 bits.
- auxW01: Variable auxiliar de 16 bits.
- interf: Valores ASCII que serán impresos para representar la interfaz.

7. Detalles importantes de la implementación

Uso de interrupciones del BIOS:
La calculadora utiliza las interrupciones del BIOS para manejar las entradas y salidas de datos, garantizando una interacción fluida y eficiente con el usuario. Se emplea la interrupción INT 10H para gestionar las impresiones en pantalla, permitiendo así la visualización de la interfaz de usuario y los resultados de las operaciones. Para manejar la entrada de datos desde el teclado y el mouse, se utiliza la interrupción INT 16H y INT 33H, respectivamente. Adicionalmente, se utiliza la interrupción INT 21H para implementar un retraso, emulando el comportamiento de las calculadoras antiguas y mejorando la experiencia del usuario.

Interfaz gráfica:
La interfaz gráfica de la calculadora se construye utilizando arrays predefinidos con valores ASCII. Cada botón y área de la interfaz se define en términos de sus posiciones en la pantalla, permitiendo una visualización clara y coherente de los elementos de la calculadora. La interfaz incluye una ventana principal para la entrada y salida de datos, así como una ventana de error que se actualiza en tiempo real para informar al usuario sobre entradas inválidas.

Procedimientos modulares:

Para mantener el código organizado y facilitar su mantenimiento, se ha adoptado una estructura modular. Los procedimientos están claramente definidos y se encargan de tareas específicas, como el almacenamiento de valores en arrays, la verificación de errores, y la ejecución de operaciones aritméticas. Esto no solo mejora la legibilidad del código, sino que también simplifica la detección y corrección de errores.

Manejo de errores:

El programa incluye robustos mecanismos para la detección y gestión de errores. Antes de realizar cualquier operación, se verifica que la entrada del usuario cumpla con el formato establecido. Se identifican y manejan errores comunes, como la división por cero y el uso de operadores inválidos, asegurando que el programa no se detenga inesperadamente y proporcionando retroalimentación inmediata al usuario.

Algoritmos clave:

Se han implementado varios algoritmos clave para el correcto funcionamiento de la calculadora. Por ejemplo, el algoritmo que decide qué hacer con el valor actual almacenado en “auxB01” y los procedimientos para almacenar estos valores en un array. También se ha desarrollado un algoritmo que verifica la validez de la entrada del usuario y otro que ejecuta las operaciones aritméticas basadas en los valores ingresados. Estos algoritmos están escritos en pseudocódigo estilo Python antes de ser traducidos a lenguaje ensamblador, lo cual facilita su comprensión y verificación.

Simulación de calculadora antigua:

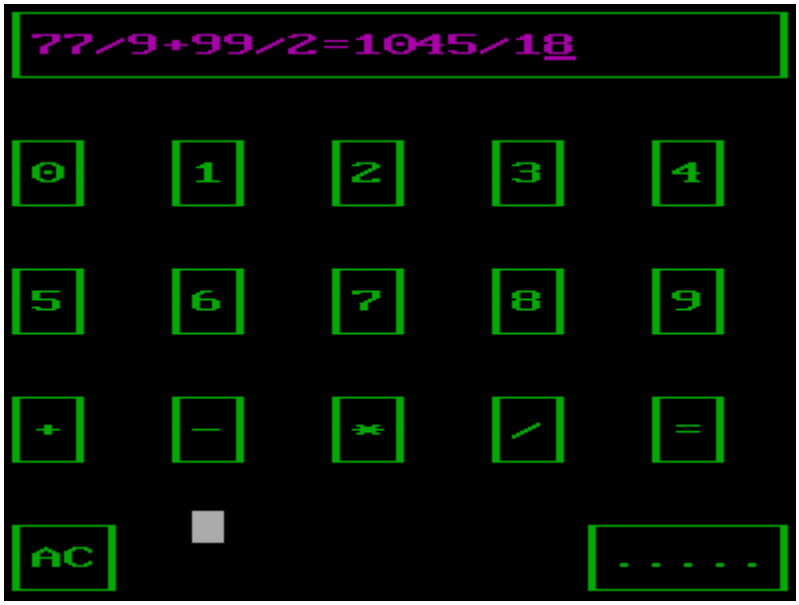
Para recrear la sensación de usar una calculadora antigua, se introduce un retraso en la respuesta del programa utilizando la interrupción INT 21H. Este retraso obliga al usuario a mantener presionado el botón deseado antes de que el mismo sea mostrado en pantalla, añadiendo un toque de autenticidad a la experiencia de uso.

Soporte de teclado y salida rápida:

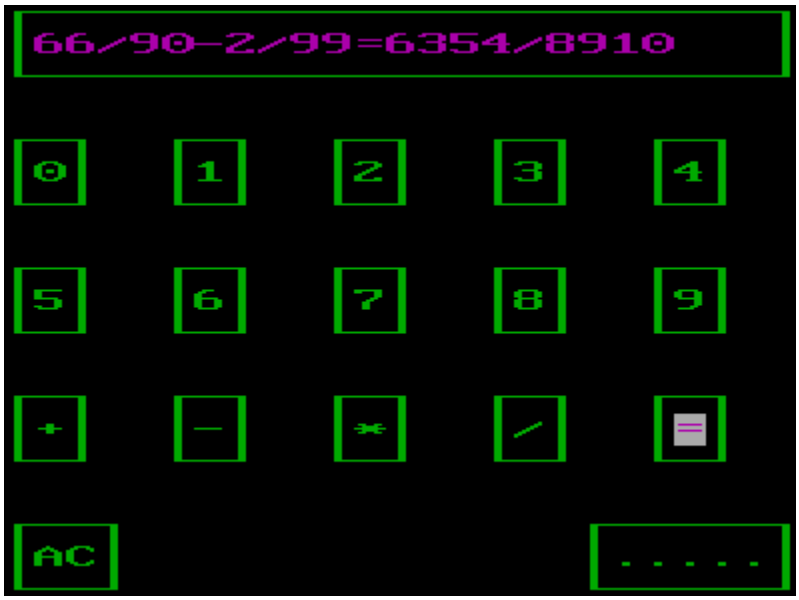
La calculadora permite al usuario salir del programa en cualquier momento simplemente presionando la tecla ESC. Este soporte de teclado facilita la interacción y proporciona una manera rápida y eficiente de finalizar el uso de la calculadora.

8. Datos de prueba utilizados (casos de prueba)

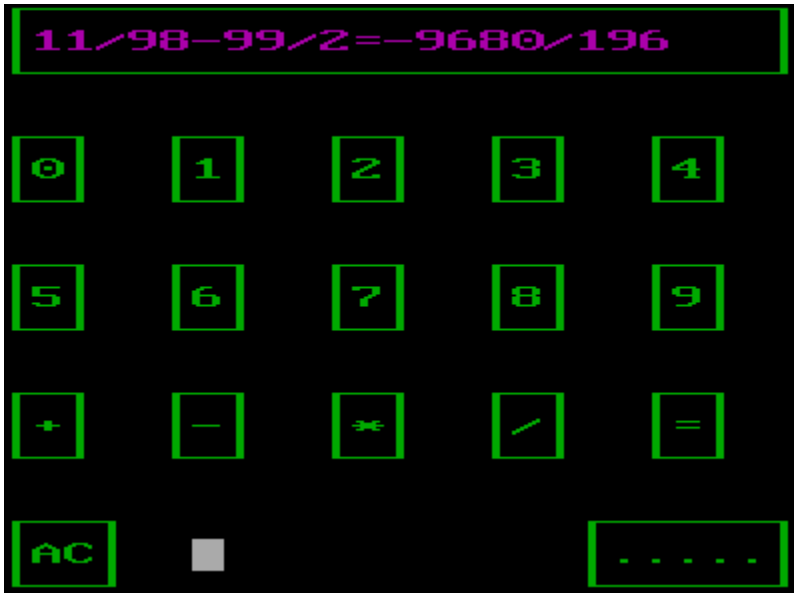
Suma:



Resta con resultado positivo:

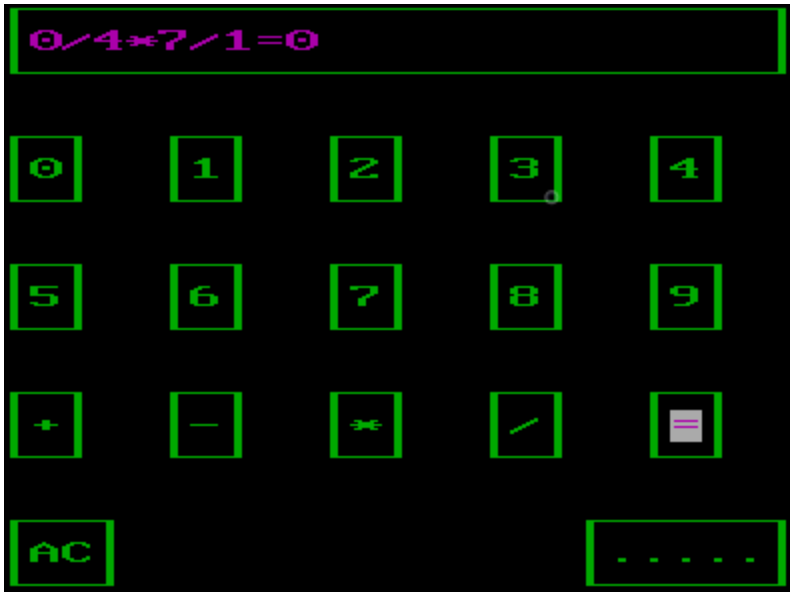


Resta con resultado negativo:



Multiplicación con resultado mayor que cero:

Multiplicación con resultado igual a cero:



División correcta:

División erronea:

9. Análisis de resultados (indicar grado de funcionamiento)

En el desarrollo y prueba de la calculadora de números racionales en Turbo Assembler, se realizaron diversas pruebas para asegurar que las operaciones aritméticas básicas (suma, resta, multiplicación y división) funcionaran correctamente bajo las condiciones establecidas. Los resultados obtenidos confirman que la calculadora realiza las operaciones de manera precisa y fiable, cumpliendo con los requisitos del proyecto.

Aritmética precisa: Todas las operaciones aritméticas básicas fueron ejecutadas correctamente, obteniendo resultados que coinciden con los valores esperados. La calculadora maneja adecuadamente las operaciones válidas y genera mensajes de error cuando se intenta realizar una operación inválida, como dividir por cero.

Manejo de entradas y salidas: La interfaz de usuario, diseñada para mostrar la operación actual y el resultado, funcionó sin problemas. Las interrupciones del BIOS (INT 10H, INT 16H, INT 33H y INT 21H) se utilizaron eficazmente para gestionar las entradas del teclado y mouse, así como las salidas en pantalla.

Detección de errores: El algoritmo de verificación de errores comprobó que todas las entradas cumplieran con el formato especificado, lo que garantiza que la calculadora no procese entradas inválidas. Este mecanismo previene errores comunes y proporciona retroalimentación inmediata al usuario.

Experiencia de usuario: La simulación de una calculadora antigua mediante la introducción de un retraso en la respuesta del programa añade un toque de autenticidad y personalidad a la experiencia de uso. El soporte para teclado, incluyendo la opción de salir del programa con la tecla ESC, mejora la interacción del usuario.

Formato estricto de entrada: La calculadora requiere que las operaciones se ingresen en un formato específico (por ejemplo, A/B operador C/D =). Aunque esto garantiza la consistencia y precisión de los cálculos, puede ser una limitación para algunos usuarios que no estén familiarizados con el formato.

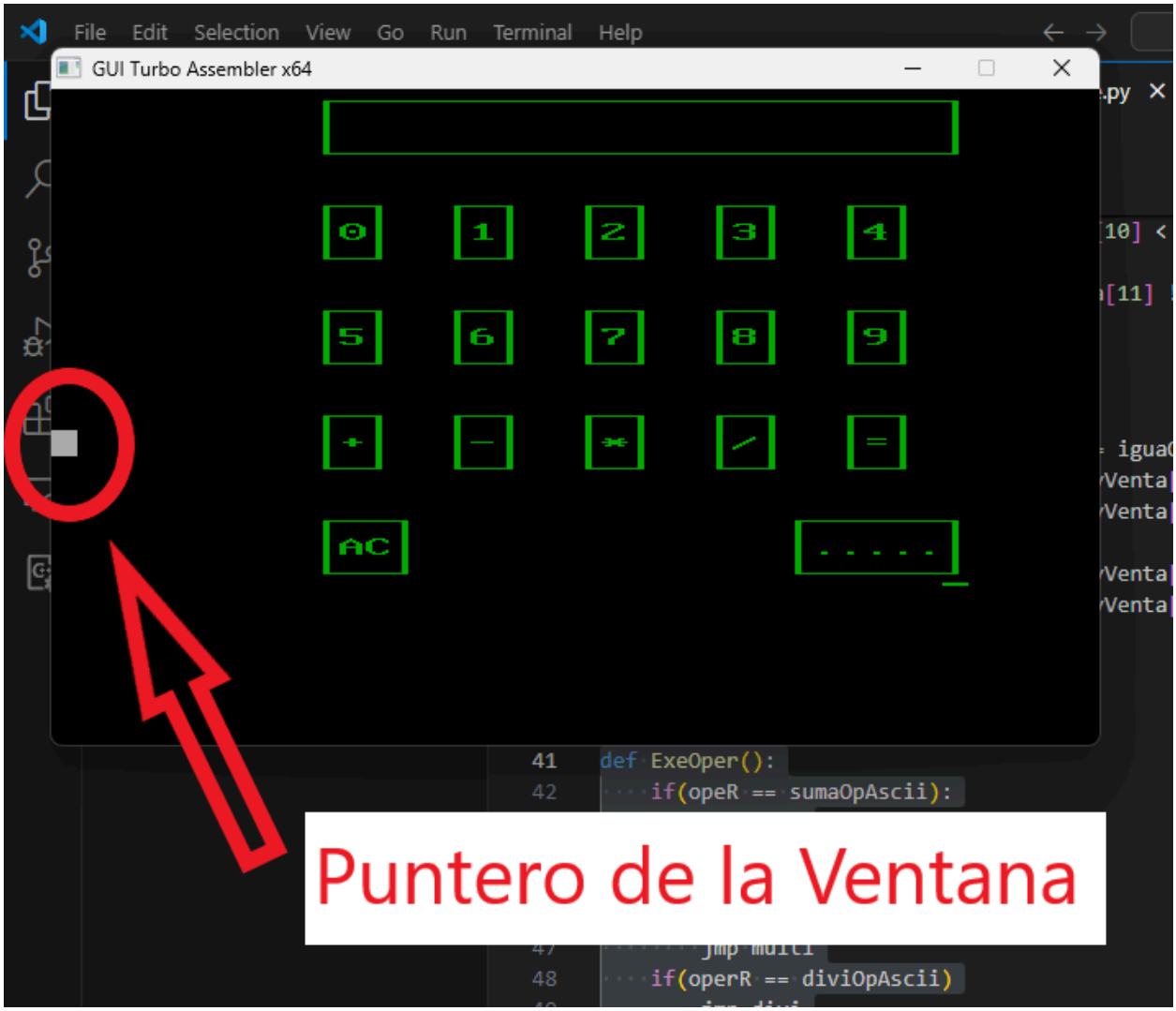
Sin simplificación de fracciones: Los resultados de las operaciones no se simplifican automáticamente, siguiendo la sugerencia del profesor encargado del curso. Aunque esto no afecta la precisión de los cálculos, puede resultar en fracciones no reducidas.

Bug del mouse: Una limitación notable es el bug relacionado con el manejo de las coordenadas del mouse en el emulador de Turbo Assembler. Este problema se resuelve informando al usuario que el primer clic en la ventana generada por el emulador debe ser en la posición mostrada por el puntero del mouse para evitar desajustes en las coordenadas.

10. Breve Guía del Usuario

- Descargar “GUI Turbo Assembler”. El siguiente enlace lo lleva directo al repositorio oficial del programa mencionado. https://github.com/ljnath/GUI-Turbo-Assembler
- Instalar “GUI Turbo Assembler”, para ello seguir las recomendaciones que se dan en la página a la cual redirige el enlace anterior.
- Abrir el archivo “code.asm” con la herramienta previamente descargada.

- Presionar la tecla F9 para hacer los procesos de “assemble”, “build” y “run” de forma sencilla. Con esto inmediatamente se ejecuta el código.
- Importante: Una vez se genera la ventana en la que corre la calculadora, usted deberá direccionar el puntero de su ratón hacia el puntero en formato texto de la ventana generada. Cuando ambos punteros coinciden, es cuando usted puede dar click, en caso contrario es probable que experimente el “bug” de la interrupción INT 33H del programa GUI Turbo Assembler, dicho “bug” ocasiona que el programa malinterprete las coordenadas del mouse.



- Ahora, usted puede hacer uso de todas las herramientas que la calculadora le ofrece. Como detalle, es importante recordar seguir el formato que se ha establecido a lo largo de

esta documentación (A/B op C/D =), de esta forma evitará toparse con el molesto mensaje de error.

- A la hora de dar clic a cualquier botón, recuerde mantenerlo presionado por un tiempo.
- Para computar una operación, debe hacer clic en el botón “=”.
- Para limpiar la ventana y así poder hacer otra operación, basta con presionar el botón “AC”.
- Presione la tecla ESC para salir del programa.