MANUAL TÉCNICO
PROYECTO 2

Elías Abraham Vásquez Soto

201900131

Arquitectura de Computadores y Ensambladores 1 N

OBJETIVOS

General:

Brindar al lector una guía que contenga la información de la lógica utilizada para el desarrollo de aplicación, con el fin de facilitar futuras actualizaciones y modificaciones realizadas por terceros.

Específicos:

- Mostrar al lector una descripción lo más completa y detallada posible del sistema operativo, versión de ensamblador, sintaxis, entre otros utilizados para el desarrollo de la aplicación.
- Proporcionar al lector una concepción y explicación técnica - formal de los procesos y relaciones entre macros, procedimientos y registros que conforman la parte operativa de la aplicación.

INTRODUCCIÓN

Este manual técnico tiene como finalidad dar a conocer al lector que pueda requerir hacer modificaciones futuras al software el desarrollo de la aplicación "Calculadora gráfica" indicando todas las herramientas utilizadas en su desarrollo, sus versiones, requerimientos del sistema, etc.

La aplicación tiene como objetivo ser una calculadora que se maneje en consola, pudiendo almacenar ecuaciones de grado n (donde n es un número entero no mayor a 5), y calcular su derivada e integral, graficar dichas funciones almacenadas en ella, y encontrar los ceros de la función por medio del método de Newton o del método de Steffensen, todo esto utilizando como herramienta un ensamblador de x86, el coprocesador matemático e interrupciones de DOS.

DESCRIPCIÓN DE LA SOLUCIÓN

Para poder desarrollar esta aplicación, se utilizó el ensamblador NASM en un sistema operativo DOS, considerando que en esta fase del programa se implementan gráficas, por lo que el uso de interrupciones de DOS/BIOS es necesaria (estas interrupciones no pueden trabajarse en Linux).

La aplicación cuenta con un buffer de entrada de 16 bits, en donde registra las entradas que el usuario ingresa al programa, y así estas son analizadas y procesadas como instrucciones directas a la calculadora.

Entre las consideraciones más importantes que se tomaron en cuenta para el desarrollo de la aplicación, se encuentran:

- El usuario se comunica con la aplicación utilizando únicamente números, por lo que la correcta e intuitiva navegación entre los distintos menús del programa es vital.

- El usuario puede ingresar los caracteres que desee en la consola, por lo que debe existir una anticipación a los posibles errores de ejecución que puedan generarse. En algunos casos, la recuperación de dichos errores debe ser posible.

- En esta fase se implementan ampliamente los números en punto flotante, por lo que el manejo del concepto de FPU es de vital importancia.

FUNCIONAMIENTO DE LA APLICACIÓN

Manejo de números en punto flotante

En esta última fase de la aplicación, se implementaron nuevas funcionalidades, como gráfica de funciones y búsqueda de ceros por métodos numéricos; funcionalidades que requieren de operar con números en punto flotante, por lo que se recomienda que, si se desea observar o dar mantenimiento al código de la aplicación, se comprenda cómo funciona y cómo se manipula el FPU (Unidad de punto flotante) en lenguaje ensamblador.

El FPU es una parte del computador especialmente diseñada para encargarse de las operaciones con números en punto flotante.

Archivos que componen la aplicación

La aplicación cuenta con los siguientes archivos:

- DOS_CODE.asm: Código del programa en lenguaje ensamblador
- DOS_CODE.obj: Código objeto del programa, ensamblado con NASM.
- DOS_CODE.EXE: Ejecutable de la aplicación, generado con el enlazador VAL (ver Especificaciones técnicas).

Las especificaciones del lenguaje utilizado son:

- Ensamblador NASM (Netwide Assembler): Un ensamblador para la arquitectura de CPU x86 portable a casi todas las plataformas modernas, y con generación de código para muchas plataformas antiguas y nuevas.
- Sintaxis Intel: Se optó por esta sintaxis y no por AT&T, gracias a comodidad del desarrollador, y considerando también que la mayoría de los ensambladores x86 utilizan la sintaxis de Intel.

Estructura del programa

Tomando en cuenta la estructura de un programa NASM, un programa ensamblador puede estar dividido en 3 secciones:

ÁREA DE MACROS

Las macros multilínea se expanden a un nuevo conjunto de líneas de código fuente. El número que aparece tras el nombre de la macro en la línea %macro define el número de parámetros que espera recibir. Con una macro que toma más de un parámetro, los parámetros subsiguientes se referirían como %2, %3 y así sucesivamente.

Para el desarrollo del programa se definieron las siguientes macros:

MACROS I FASE:

- print: Utilizada para imprimir variables en pantalla (no imprime variables numéricas).

- printNumber: Utilizada para imprimir variables numéricas.
- printDecimalNumber: Utilizada para imprimir los coeficientes de la integral, truncados a 1 decimal.
- printFraction: Utilizada para imprimir los coeficientes de la integral, en formato de fracción.
- read: Utilizada para leer datos del teclado.

*** Para más detalles de estas macros, ver Manual Técnico Fase I.**

MACROS II FASE:

- **printWordNumber | printDwordNumber**
  - Macros utilizadas para la impresión en consola de números almacenados en variables de 2 y 4 bytes respectivamente, con capacidad de impresión de hasta 6 dígitos, en el caso de números guardados en variables de 4 bytes.
  - Parámetros que reciben: 1
    - Variable que almacena el número a imprimir
  - Macros auxiliares:
    - Macros de impresión del 6° dígito, 5° dígito, etc.

- **evaluateOriginalFunction | evaluateDerivativeFunction**
  - Macros de pruebas, que evalúan números enteros en la función original y en la derivada de la función original. Macro no utilizada en ninguna funcionalidad del proyecto.
  - Parámetros que recibe: 1
    - Valuar a evaluar en la función

- **evaluateFloatInOriginalFunction**
  - Macro utilizada para evaluar números en punto flotante en la función original. Utilizada para las gráficas y los métodos numéricos.
  - Parámetros que recibe: 1
    - Flotante de 4 bytes a evaluar en la función original.

- **evaluateFloatInDerivativeFunction**
  - Macro utilizada para evaluar números en punto flotante en la derivada función original. Utilizada para las gráficas y el método de Newton.
  - Parámetros que recibe: 1
    - Flotante de 4 bytes a evaluar en la derivada de la función original.

- **castFloatToInt**
  - Macro que trunca el número en punto flotante que se le envíe. Utilizada para graficar, asegurando pixeles enteros.
  - Parámetros que recibe: 1
    - Flotante de 4 bytes a castear a entero.

- **printFloatingNumber**
  - Macro que utiliza operaciones del FPU para imprimir flotantes.
  - Parámetros que recibe: 1
    - Flotante de 4 bytes a imprimir.

SECCIÓN *.data*

La sección .data es usada para declarar datos o constantes inicializadas. En esta sección se podrán encontrar todos los textos usados a lo largo de la aplicación, así como los coeficientes de las funciones calculadas, y variables auxiliares usadas para graficar funciones y evaluación de métodos numéricos. Los coeficientes siguen el siguiente formato:

- Coeficientes de la función original
  - coef_#: # toma los valores de 0 a 5.
- Coeficientes de la derivada de la función original
  - deriv_c#: # toma los valores de 0 a 4.
- Parte entera y residual de los coeficientes de la integral de la función original
  - integ_e#: parte entera del coeficiente. # toma los valores de 1 a 6.
  - integ_d#: parte residual del coeficiente. # toma los valores de 1 a 6.

SECCIÓN *stack*

La sección bss se utiliza para declarar variables sin valor inicial. En esta sección se declararon distintas variables auxiliares, como el buffer de entrada de teclado, etc.

SECCIÓN *.text*

La sección de texto se utiliza para guardar el código real. La declaración **global _start** indica al núcleo dónde comienza la ejecución del programa.

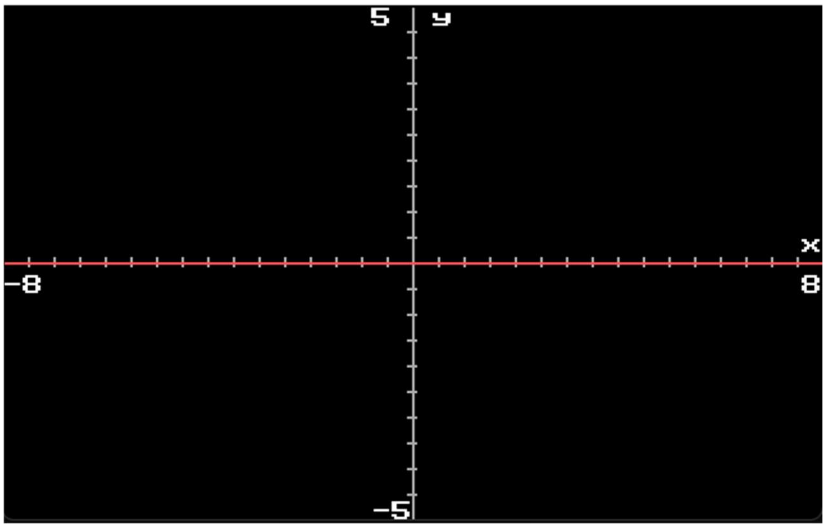
```
;; INICIO DE EJECUCIÓN DE PROGRAMA
global _start

_start:

MENU:

    print text_menu, len_menu

    ; leyendo entrada del teclado
    read buffer_in, 16
```

Los procedimientos declarados en esta sección son:

- **CLEAR_TERMINAL:** Método que se invoca cada vez que se desea limpiar la terminal.
    - Llamada al sistema que realiza (interrupciones): sys_write
        - %eax: 4
        - %ebx: 1
        - %ecx: Código a mandar a consola para limpieza de la misma
        - %edx: longitud del código

- **READ_NUMBER:** Método que se invoca cada vez que se desea leer un coeficiente, guardándolo automáticamente en el registro DL.

- **GRAPH_CARTESIAN_MAP:** Método que se llama cada vez que se desea iniciar a graficar una función. Dibuja en pantalla los ejes 'x' y 'y', cuyo rango de valores es de [-8, 8] en 'x', y [-5, 5] en 'y'.



La escala utilizada es de 20:1 (20 pixeles por cada unidad, ya sea en el eje 'x' o en el eje 'y'). Esto permite una mayor nitidez de las gráficas, ya que los valores de x van aumentando de 0.05 en 0.05 para obtener los respectivos valores en y.

* IMPORTANTE: Este método sólo grafica el plano cartesiano, no las gráficas de funciones. Esto se hace en el código de la opción 5, correspondiente a las gráficas.

ESPECIFICACIONES TÉCNICAS

Herramientas utilizadas

- Lenguaje ensamblador

  NASM versión 2.16.01

- Librerías utilizadas

  No se utilizaron librerías en el desarrollo de la aplicación.

- IDE utilizado

  Sublime Text versión 4

- Sistema operativo

  DOS (DosBOX para simularlo).

Requerimientos mínimos de sistema

- Hardware
  - Procesador: Intel o AMD de doble núcleo a 2 Ghz o superior
  - Memoria RAM: 384 MB
  - Disco duro: 25 GB
  - Tarjeta gráfica VGA
  - Acceso a internet

- Software
    - Ensamblador NASM: Se recomienda instalar este ensamblador para poder generar los ejecutables que desee.
    - DosBOX: Es un emulador que recrea un entorno similar al sistema MS-DOS, para ejecutar el programa.
    - Linker VAL: Es un programa que toma los objetos generados en los primeros pasos del proceso de compilación, produciendo un ejecutable. Necesario si se desea compilar el programa desde el código fuente.