MANUAL TÉCNICO
PRÁCTICA 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, 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 Linux, específicamente Ubuntu, considerando la ventaja que este ofrece al otorgar un amplio soporte a los programas creados con el mencionado ensamblador.

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.

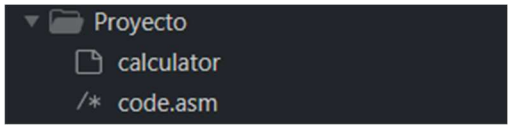
Entre las consideraciones más importantes que se tomaron en cuenta para el desarrollo de la aplicación, se encuentran:

- 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.

FUNCIONAMIENTO DE LA APLICACIÓN

Archivos que componen la aplicación



La aplicación cuenta con un único archivo .asm en donde se encuentra todo el código assembler. 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.

Además, se podrá encontrar con un ejecutable para Linux, aunque si se desea, se puede crear el ejecutable en cualquier máquina Linux por medio de comandos en la terminal (ver detalles en Especificaciones técnicas).

Estructura del programa

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

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:

- **print**

```
%macro print 2
```

  - Utilizada para imprimir variables en pantalla (no imprime variables numéricas).
  - Parámetros que recibe: 2
    1. Variable a imprimir
    2. Longitud de la variable a imprimir
  - Llamada al sistema que realiza (interrupciones): sys_write
    1. %eax: 4
    2. %ebx: 1
    3. %ecx: variable a imprimir
    4. %edx: longitud de variable a imprimir

- **printNumber**

```
%macro printNumber 1
```

  - Utilizada para imprimir variables numéricas.
  - Parámetros que recibe: 1
    1. Variable numérica a imprimir
  - Macros auxiliares:
    1. printTwoDigits

```
%macro printTwoDigits 1
```

    2. printThreeDigits

```
%macro printThreeDigits 1
```

- **printDecimalNumber**

```
%macro printDecimalNumber 6
```

  - Utilizada para imprimir los coeficientes de la integral, truncados a 1 decimal.
  - Parámetros que recibe: 6
    1. Parte entera del coeficiente
    2. Residuo del coeficiente
    3. Tipo de 'x' a imprimir (x^6, x^5, …)
    4. Tamaño de la 'x' a imprimir
    5. Número de coeficiente a imprimir (1-6)
    6. ¿Imprimir el signo '+'? (1: Y, 0: N)

- **printFraction**

```
%macro printFraction 5
```

  - Utilizada para imprimir los coeficientes de la integral, en formato de fracción.
  - Parámetros que recibe: 5
    1. Numerador (coeficiente de la función original)
    2. Denominador (número de coeficiente)
    3. Tipo de 'x' a imprimir (x^6, x^5, …)
    4. Tamaño de la 'x' a imprimir
    5. ¿Imprimir el signo '+'? (1: Y, 0: N)

- **read**

```
%macro read 2
```

  - Utilizada para leer datos del teclado.
  - Parámetros que recibe: 2
    - Variable en donde se guardarán los datos leídos.
    6. Longitud de lo máximo que se leerá.
  - Llamada al sistema que realiza (interrupciones): sys_read
    1. %eax: 3
    2. %ebx: 0
    3. %ecx: variable en donde se guardarán los datos leídos
    4. %edx: longitud de lo máximo que se leerá

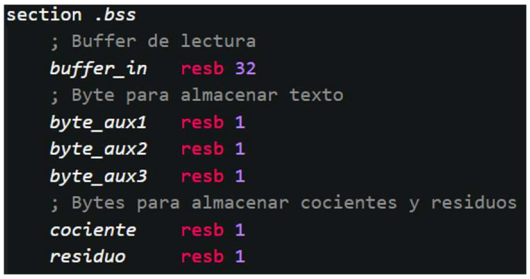
Sección *.data*

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. Estos 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 *.bss*

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.



```
section .bss
    ; Buffer de lectura
    buffer_in    resb 32
    ; Byte para almacenar texto
    byte_aux1    resb 1
    byte_aux2    resb 1
    byte_aux3    resb 1
    ; Bytes para almacenar cocientes y residuos
    cociente     resb 1
    residuo      resb 1
```

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.

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

  Ubuntu 20.04

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.

      El comando de instalación del ensamblador es:

      ```
      sudo apt-get install nasm build-essential
      ```

      El comando para generar código objeto es:

      ```
      nasm -f elf ASM_FILE.asm
      ```

      El comando para generar ejecutables de 32 bits a partir de un archivo en formato elf es:

      ```
      ld -m elf_i386 -o EXEC_NAME OBJECT_FILE.o
      ```