Universidad De San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
LABORATORIO ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
Sección “N”

“MANUAL TECNICO”

Luis Angel Barrera Velásquez

Carné: 202010223

Índice

Objetivos

General:

Brindar al lector información acerca del desarrollo de esta práctica y como se le dio solución para una futura actualización o corrección de errores técnicos.

Específicos:

- Detallar con palabras la solución presentada en el código paso a paso para seguir un orden y lógica del programa.

- Proporcionar los conceptos que fueron empleados para la solución de esta práctica con teoría para que sea más entendible la solución expresada en palabras.

Introducción

Este manual técnico esta elaborado con el propósito de detallar el funcionamiento del código fuente de la practica y en caso de que lo lea alguien con conocimientos sobre programación sea de ayuda para realizar futuras actualizaciones sobre el programa o corrección de algún error que suceda durante la ejecución. Es importante recalcar que su solución de esta practica 1 es bastante sencilla abarcando conceptos bastante básicos de programación en assembler.

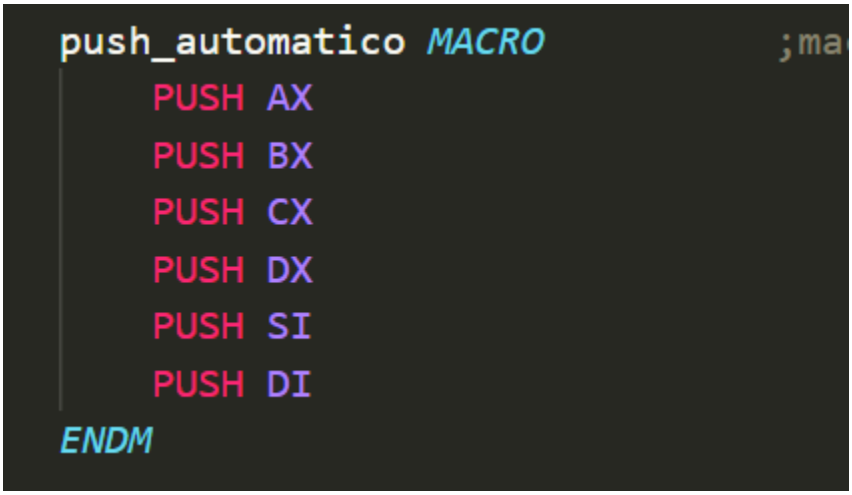
La practica consta de la solución de una calculadora en la cual se ingresa una función de máximo grado 5 y a partir de esto generar la derivada e integral a través de una interfaz elaborada en consola para una agradable interacción con el usuario. A continuación se elabora una guía técnica explicando algunos fragmentos de código para su facil interpretación.

Descripción de la Solución

Para la solución de esta practica fue utilizado el lenguaje assembler utilizando MASM en la versión 6.11 la cual es utilizada para compilar el archivo y generar el ejecutable, para el correcto funcionamiento se vio necesario la utilización de macros los cuales son necesarios para reducir la cantidad de líneas de código y hacer mas entendible y sencilla la solución agregando un toque estético para que no se vea tan complejo interpretar el código por otra persona. A continuación, se agrega una descripción para cada una de las macros utilizadas en esta práctica:

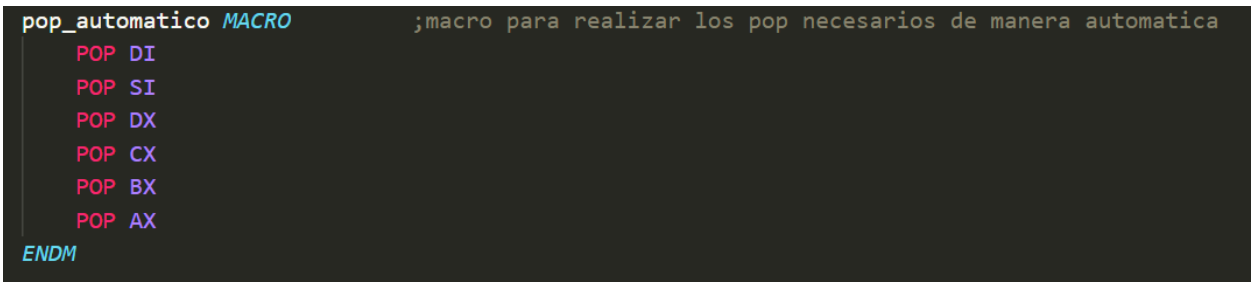
Macro push_automatico

Esta macro es necesaria para realizar los push automáticos durante la ejecución del programa.



Macro pop_automatico

Esta macro es necesaria para realizar los pop automáticos durante la ejecución del programa.



Macro limpiarpantalla

Esta macro fue necesaria para limpiar la pantalla y a la vez configura el color de la consola en fondo cyan y letras de color negro tal como se indica en los comentarios del código luego de limpiar la pantalla es necesario posicionar de nuevo el cursor en la parte superior de la consola para volver a imprimir lo que necesite de nuevo el programa.

```
limpiarpantalla MACRO
    ; PARA CONFIGURAR COLOR DE LA PANTALLA
    MOV ah, 06h
    MOV al, 00H
    mov bh,30h ; 3= COLOR CIAN Y 0 COLOR NEGRO EN LAS LETRAS
    mov cx,0000h
    mov dx,184fh
    int 10h

    ; PARA POSICIONAR EL CURSOR EN ARRIBA DE LA PANTALLA AL INICIO
    mov ah,02h
    mov bh,00h
    mov dx,0000h
    int 10h
ENDM
```

Macro obtenerOpcion

Esta opción lo que realiza es leer la pulsación de una tecla utilizando la interrupción 21h con la instrucción 01H la cual lo que hace es leer cual tecla es pulsada en el teclado, esta macro fue utilizada para elegir las opciones del menú.

```
obteneropcion MACRO             ;macro que sirve para leer la pulsacion de una tecla
    MOV AH, 01H ; leemos el caracter osea la tecla presionada 01H
    INT 21H ;ejecutamos la interrupcion
ENDM
```

Macro imprimir

Esta macro es de gran utilidad es para desplegar en consola el texto que se desee mostrar al usuario solo se llama esta macro y como parámetro se le manda lo que se quiere mostrar pasando la cadena al registro dx y llamando por ultimo la interrupción 21h

```
imprimir MACRO cadena
    push_automatico
    MOV AX, @data ;obtenemos la data
    MOV DS, AX ;pasamos el registro AX al DS
    MOV AH, 09H ;creamos la interrupcion
    MOV DX, offset cadena ;obtenemos la direccion de memoria de cadena
    INT 21H ; ejecutamos la interrupcion
    pop_automatico
ENDM
```

Macro imprimirOriginal

Esta macro es utilizada para imprimir en consola la función original lo que se hace es sumar 30h para convertir el contenido de ese array a código ascii luego mandarlo a imprimir y por ultimo se le vuelve a restar 30h para regresar el contenido a valor decimal: por ejemplo:

1 en ascii es 49 pero en hexadecimal es 31h y al restarle 30h tenemos 01h es decir en decimal.

```
imprimirOriginal MACRO
    LOCAL ERROR
    cmp existeFuncion,00H
    JE ERROR
    add Original5[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original5[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original4[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original4[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original3[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original3[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original2[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original2[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original1[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original1[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original0[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Original0[0002],30H ; regresamos el valor a codigo ascii para poder imprimir

    imprimir iguales
    imprimir salto
    imprimir InicioFuncionOriginal
    imprimir Original5
    imprimir equis5
    imprimir Original4
```

Macro imprimirDerivada

Esta macro es utilizada para imprimir en consola la función derivada lo que se hace es sumar 30h para convertir el contenido de ese array a código ascii luego mandarlo a imprimir y por ultimo se le vuelve a restar 30h para regresar el contenido a valor decimal: por ejemplo:

1 en ascii es 49 pero en hexadecimal es 31h y al restarle 30h tenemos 01h es decir en decimal.

```
imprimirDerivada MACRO
    LOCAL ERROR
    cmp existeFuncion,00H
    JE ERROR
    add Derivada4[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada4[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada3[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada3[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada2[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada2[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada1[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada1[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada0[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Derivada0[0002],30H ; regresamos el valor a codigo ascii para poder imprimir

    imprimir iguales
    imprimir salto

    imprimir InicioFuncionDerivada
    imprimir Derivada4
    imprimir equis4
```

Macro imprimirIntegral

Esta macro es utilizada para imprimir en consola la función integral lo que se hace es sumar 30h para convertir el contenido de ese array a código ascii luego mandarlo a imprimir y por último se le vuelve a restar 30h para regresar el contenido a valor decimal: por ejemplo:

1 en ascii es 49 pero en hexadecimal es 31h y al restarle 30h tenemos 01h es decir en decimal.

```
imprimirIntegral MACRO
    LOCAL ERROR
    cmp existeFuncion,00H
    JE ERROR
    add Integrada6[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada6[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada5[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada5[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada4[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada4[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada3[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada3[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada2[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada2[0002],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada1[0001],30H ; regresamos el valor a codigo ascii para poder imprimir
    add Integrada1[0002],30H ; regresamos el valor a codigo ascii para poder imprimir

    imprimir iguales
    imprimir salto

    imprimir InicioFuncionIntegrada
    imprimir Integrada6
```

Macro reiniciarFuncion

Esta macro se utilizó para reiniciar las variables que guardan los valores de las funciones originales esto con el objetivo que no existan valores basura que puedan afectar en los resultados entonces se hizo esto para regresar las variables a como estaban al inicio de la ejecución de programa, esta macro se usa cuando el usuario vuelve a ingresar una función diferente.

```
reiniciarFuncion MACRO
    MOV Original5[0000], 002BH ;ascci +
    MOV Original5[0001], 00H ;borrar y poner en 0
    MOV Original5[0002], 00H ;borrar y poner en 0
    MOV Original4[0000], 002BH ;ascci +
    MOV Original4[0001], 00H ;borrar y poner en 0
    MOV Original4[0002], 00H ;borrar y poner en 0
    MOV Original3[0000], 002BH ;ascci +
    MOV Original3[0001], 00H ;borrar y poner en 0
    MOV Original3[0002], 00H ;borrar y poner en 0
    MOV Original2[0000], 002BH ;ascci +
    MOV Original2[0001], 00H ;borrar y poner en 0
    MOV Original2[0002], 00H ;borrar y poner en 0
    MOV Original1[0000], 002BH ;ascci +
    MOV Original1[0001], 00H ;borrar y poner en 0
    MOV Original1[0002], 00H ;borrar y poner en 0
    MOV Original0[0000], 002BH ;ascci +
    MOV Original0[0001], 00H ;borrar y poner en 0
    MOV Original0[0002], 00H ;borrar y poner en 0

ENDM
```

Macro reiniciarTemporales

Esta macro fue utilizada para reiniciar las variables temporales las cuales almacenaran los datos que ingrese el usuario para los coeficientes y si el ingreso de coeficientes es exitoso entonces se copia todos los datos a las variables de la función original esto con el objetivo que si existe un error en el ingreso de coeficientes y anteriormente había una función esta no se borre hasta que se ingrese exitosamente otra función.

```
reiniciarTemporales MACRO
    MOV Temporal5[0000], 002BH ;ascci +
    MOV Temporal5[0001], 00H ;borrar y poner en 0
    MOV Temporal5[0002], 00H ;borrar y poner en 0
    MOV Temporal4[0000], 002BH ;ascci +
    MOV Temporal4[0001], 00H ;borrar y poner en 0
    MOV Temporal4[0002], 00H ;borrar y poner en 0
    MOV Temporal3[0000], 002BH ;ascci +
    MOV Temporal3[0001], 00H ;borrar y poner en 0
    MOV Temporal3[0002], 00H ;borrar y poner en 0
    MOV Temporal2[0000], 002BH ;ascci +
    MOV Temporal2[0001], 00H ;borrar y poner en 0
    MOV Temporal2[0002], 00H ;borrar y poner en 0
    MOV Temporal1[0000], 002BH ;ascci +
    MOV Temporal1[0001], 00H ;borrar y poner en 0
    MOV Temporal1[0002], 00H ;borrar y poner en 0
    MOV Temporal0[0000], 002BH ;ascci +
    MOV Temporal0[0001], 00H ;borrar y poner en 0
    MOV Temporal0[0002], 00H ;borrar y poner en 0

ENDM
```

Macro leerValorCoeficiente

Esta macro esta implementada para la lectura del coeficiente que este ingresando el usuario y dentro de esta macro se implementa la validación de errores y este solo acepta números enteros.

```
leerValorCoeficiente MACRO
    LOCAL SIGUIENTE,CICLO,SALIR1,ExisteSigno,SinSigno,SALIR2,ERROR ; DECLARAMOS QUE ESTAS ETIQUETAS
    push_automatico
    XOR SI,SI ; reiniciamos el registro SI origen de memoria

    SIGUIENTE: ; reiniciamos la variable llenandola con simbolos $
        mov TextoIngresado[SI],'$'
        INC SI ; incrementamos SI
        cmp SI, 14D
        jle SIGUIENTE ;realiza el salto solamente si es menor o igual a 15

    ;lectura del string
    mov ah,01h ;mandamos un 1 al registro alto de ax, funcion para leer la entrada desde el teclad
    XOR SI,SI ; reiniciamos el registro SI origen de memoria
    CICLO:
        int 21H ;hacemos la interrupcion para guardar el caracter leido y lo guardamos en AL
        cmp AL, 13D ;verificamos que no sea el enter
        JE SALIR1 ;si es enter salimos
        MOV TextoIngresado[SI], AL
        inc SI
        jmp CICLO

    SALIR1:
        ; si el usuario ingresa     -       4       0
        ;                          0000    0001    0002

        ; posicion 0000 es la posicion si el usuario ingreso signo o no
        CMP TextoIngresado[0000], 002BH;verificamos que el usuario haya ingresado signo +
        JE ExisteSigno ;saltamos para analizar con signo
```

Macro salir

Esta macro su única función es finalizar el programa.

```
salir MACRO
    imprimir mensajeSalida
    MOV AX, 4C00H ; Interrupcion para finalizar el programa
    INT 21H ; Llama a la interrupcion
ENDM
```

Macro derivar

En esta macro se encuentra toda la lógica para derivar y guardar en otras variables el resultado.

```
derivar MACRO
    push_automatico

    ;derivar coeficiente 5 y guardarlo en derivada 4

    xor AX,AX ;reiniciamos el registro ax
    mov al,10D ; valor necesario para multiplicar la decena de la funcion original
    mul Original5[0001] ;multiplicacion de la decena por 10
    xor BX,BX; reiniciamos bx
    mov bl,Original5[0002] ;obtenemos el digito de unidad
    add al,bl; sumamos las decenas con las unidades
    mov bl,5D ;realizamos la multiplicacion de la derivada
    mul bl
    mov bl,10D ;valor para obtener las decenas
    div bl ; realizamos la division y guardamos en ax --- AL el digito decena --- AH digito Unidad
    mov Derivada4[0001], AL ;guardamos la decena
    mov Derivada4[0002], AH ;guardamos la unidad
    mov ah,Original5[0000] ;copiamos el signo
    mov Derivada4[0000],ah

    ;derivar coeficiente 4 y guardarlo en derivada 3
```

Macro integrar

En esta macro se encuentra toda la lógica para integrar y guardar en otras variables el resultado.

```
integrar MACRO
    push_automatico


    ;Integrar coeficiente 5 y guardarlo en Integral 6

    xor AX,AX ;reiniciamos el registro ax
    mov al,10D ; valor necesario para multiplicar la decena de la funcion original
    mul Original5[0001] ;multiplicacion de la decena por 10
    xor BX,BX; reiniciamos bx
    mov bl,Original5[0002] ;obtenemos el digito de unidad
    add al,bl; sumamos las decenas con las unidades
    mov bl,6D ;realizamos la division de la integral
    div bl ; esta division se guarda en AX--- AH parte para decimales----AL parte para enteros
    xor ah,ah ;reiniciamos ah ya que contiene parte decimal que no interesa
    mov bl,10D ;valor para obtener las decenas
    div bl ; realizamos la division y guardamos en ax --- AL el digito decena --- AH digito Unidad
    mov Integrada6[0001], AL ;guardamos la decena
    mov Integrada6[0002], AH ;guardamos la unidad
    mov ah,Original5[0000] ;copiamos el signo
    mov Integrada6[0000],ah
```

Requisitos Mínimos

Dosbox

Este programa nos servirá para compilar nuestro archivo.asm, se puede descargar desde el siguiente link:

https://www.dosbox.com/download.php?main=1

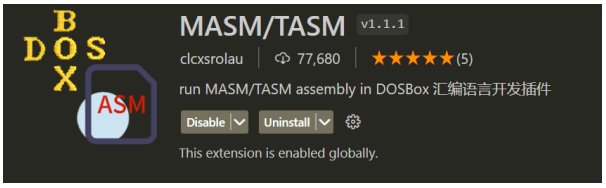
MASM 6.11

En este caso se uso la sintaxis masm para la realización de código de todo el proyecto a continuación se deja un tutorial de instalación y compilación de un programa en assembler usando dosbox y masm 6.11

https://www.youtube.com/watch?v=zvLX0bFpcxM

VisualStudio

Este es opcional pero se puede utilizar únicamente instalación la siguiente extensión:



Y modificando las configuraciones de esta manera: