UNIVERSIDAD SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
INGENIERÍA EN CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORAS Y ENSAMBLADORES 1
SECCIÓN A

MANUAL TÉCNICO

Laboratorio, Proyecto N°2

Leonardo Roney Martínez Maldonado 201780044
César Leonel Chamalé Sicán 201700634

Ciudad de Guatemala, Domingo 10 de Octubre de 2021

INTRODUCCIÓN

La programación en assembler es un lenguaje de bajo nivel muy potente, para introducirse en el entorno de la programación en assembler se debe tener bases firmes de lógica de programación. En este proyecto se llevarán a cabo dos maneras de introducir funciones al proyecto, mediante la carga de una sola o mediante la carga de un archivo de entrada.

En este manual se detalla, cada una de las macros y el flujo de operaciones que se llevan a cabo en el código de assembly de 16 bits. El ejecutable del proyecto sirve para derivar, integrar, resolver y graficar funciones polinómicas de hasta cuarto grado, ya sea que esté en un archivo o por medio del uso de comandos y mostrar todas las funciones en memoria. Además este contará con un analizador de todas las entradas del usuario para verificar que la función esté correcta y cumpla con las condiciones impuestas, tanto en forma de comando como en el archivo.

DISTRIBUCIÓN DE ARCHIVOS

La proyecto posee cuatro archivos los cuales son:

- **Proyecto.asm:** Este posee todo el código de ejecución del programa, es decir, el main, los procedimientos y todas las variables utilizadas a lo largo de los macros, procedimientos y el mismo main.

- **Archivos.asm:** Este posee todas los macros que manipulan los archivos.

- **Macros.asm:** Este posee macros de uso general, que pueden ser útiles a lo largo de macros y procedimientos, por ejemplo, la macro de Imprimir.

ESTRUCTURA GENERAL

Segmento .data

Todas las variables del segmento de data utilizan un prefijo que permite identificar qué acción realizan en la porción de código en la que es utilizada, los prefijos son :

- **msg_** : estos solo almacenan mensajes que despliegan en consola o en la escritura del archivo.

- **var_** : las variables que poseen este prefijo almacenan los datos que varían en tiempo de ejecución.

- **wr_** : las de este prefijo son variables utilizadas para la escritura del archivo

Segmento .code

En este apartado están todos los procedimientos que se utilizan dentro del ciclo principal y en las macros. Todos los procedimientos empiezan con una letra mayúscula al igual que las macros. Los procedimientos realizados y una descripción de lo que realizan:

LimpiarEntradas: limpia los registros ax,bx,cx,dx.

```
;-------------------------------------------------------------------------
;                                          LIMPIAR ENTRADAS
;-------------------------------------------------------------------------
    LimpiarEntradas proc
        mov si,0
        nciclo:
            mov var_input[si],'$'
            inc si
            cmp var_input[si],'$'
            jne nciclo
        mov si,0
        ret
    LimpiarEntradas endp
```

ImprimirFunciones: imprime todas las funciones almacenadas en memoria, va desde la función 1 hasta la 20.

```
;-------------------------------------------------------------------------
;                                          IMPRIMIR FUNCIONES
;-------------------------------------------------------------------------
    ImprimirFunciones proc
        mov msg_id[2],'A'
        Imprimir msg_id
        ImprimirFunc func_1

        mov msg_id[2],'B'
        Imprimir msg_id
        ImprimirFunc func_2

        mov msg_id[2],'C'
        Imprimir msg_id
        ImprimirFunc func_3
```

```
        …
        mov msg_id[2],'T'
        Imprimir msg_id
        ImprimirFunc func20
        ret
    ImprimirFunciones endp
```

LeerLinea: permite la entrada de caracteres hasta que el usuario presione enter, es decir, hasta un salto de línea.

```
;-------------------------------------------------------------------------
;                                                  LEER LA LINEA DE ENTRDA
;-------------------------------------------------------------------------
    LeerLinea proc
        mov si, offset var_input
        loop_input:
            mov ah,1     ;LEER CARACTER
            int 21h
            cmp al,13    ;COMPARAR CON \n
            je concat_input
            mov [si],al ;AGREGAR A MEMORIA SI NO CUMPLE
            inc si       ;PASAR A LA SIGUIENTE CELDA
            jmp loop_input ;FIN DEL LOOP
        concat_input:
            mov di,offset var_input
        ret
    LeerLinea endp
```

Salir: se sale de la aplicación xd.

```
;-------------------------------------------------------------------------
;                                                  COMANDO SALIR
;-------------------------------------------------------------------------
    Salir proc
        mov ax, 4C00H
        int 21h
    Salir endp
```

SubirArchivo: permite que el usuario ingrese la ruta del archivo a subir y llama la macro de cargar archivo.

```
;-------------------------------------------------------------------------
;                                                   INGRESAR ARCHIVO
;-------------------------------------------------------------------------
    SubirArchivo proc
        Imprimir msg_archivo
        call LeerLinea
        CargarArchivo var_input, var_handle, var_texto_archivo, 500
        ret
    SubirArchivo endp
```

IngresarFunción: permite leer la entrada y comprobar la función ingresada por el usuario.

```
;-------------------------------------------------------------------------
;                                                   INGRESAR FUNCION
;-------------------------------------------------------------------------
    IngresarFuncion proc
        mov error_funcion,0
        Imprimir msg_escribe
        call LeerLinea
        SepararTerminos var_input
        cmp error_funcion,1
        je mostrar_error
        GuardarFuncion var_input
        Imprimir msg_func_bien
        jmp if_salir
        mostrar_error:
            Imprimir msg_func_mala
        if_salir:
            LimpiarVariable var_input
        ret
    IngresarFuncion endp
```

DerivarFunción: permite la entrada de una función seleccionada y a parte deriva la función que se asignó en memoria.

```
;-------------------------------------------------------------------------
;                                                DERIVAR FUNCION
;-------------------------------------------------------------------------
    DerivarFuncion proc
        Imprimir msg_ingrese_id
        call LeerLinea
        LimpiarVariable var_funcion
        SeleccionarFuncion var_input
        Imprimir msg_funcion_select
        Imprimir var_funcion

        LimpiarVariable arr_exponentes
        LimpiarVariable arr_coeficientes
        SepararTerminos2 var_funcion
        Imprimir msg_derivida
        Derivar arr_coeficientes, arr_exponentes
        ret
    DerivarFuncion endp
```

IntegrarFunción: permite la entrada de una función seleccionada y a parte integra la función que se asignó en memoria.

```
;-------------------------------------------------------------------------
;                                                INTEGRAR FUNCION
;-------------------------------------------------------------------------
    IntegrarFuncion proc
        Imprimir msg_ingrese_id
        call LeerLinea
        LimpiarVariable var_funcion
        SeleccionarFuncion var_input
        Imprimir msg_funcion_select
        Imprimir var_funcion

        LimpiarVariable arr_exponentes
        LimpiarVariable arr_coeficientes
        SepararTerminos2 var_funcion
        Imprimir msg_integral
        Integrar arr_coeficientes, arr_exponentes
        ret
    IntegrarFuncion endp
```

ResolverFunción: selecciona función y pide que tipo es la función para proceder a resolver.

```
;-------------------------------------------------------------------------
;                                                    RESOLVER FUNCIÓN
    ResolverFuncion proc
        Imprimir msg_ingrese_id
        call LeerLinea
        LimpiarVariable var_funcion
        SeleccionarFuncion var_input
        Imprimir msg_funcion_select
        Imprimir var_funcion
        ;PREGUNTAR QUE TIPO DE FUNCIÓN ES
        LimpiarVariable var_input
        LimpiarVariable arr_exponentes
        LimpiarVariable arr_coeficientes
        SepararTerminos2 var_funcion
        Imprimir msg_submenu0
        Imprimir msg_men11
        call LeerLinea
        cmp var_input[0], '1'    ; Ax = 0 y Ax^2 = 0
        jne resfunc2
        Imprimir msg_resultado
        Imprimir msg_ceroxd
        jmp res_func_salir
        resfunc2:
            cmp var_input[0], '2' ; Ax + B = 0
            jne resfunc3
            ResolverLineal arr_coeficientes
            jmp res_func_salir
        resfunc3:
            cmp var_input[0], '3' ; Ax^2 + B = 0
            jne resfunc4
            ResolverCadratica1 arr_coeficientes
            jmp res_func_salir
        resfunc4:
            cmp var_input[0], '4' ; Ax^2 + Bx + C = 0
            jne res_func_salir
            ResolverCadratica2 arr_coeficientes
            jmp res_func_salir
        LimpiarVariable var_funcion
        res_func_salir:
        ret
    ResolverFuncion endp
```

Main Procedure

Es el procedure principal de la aplicación y cuenta con un ciclo en el que se repite el menú y secciones en donde se compara la entrada del usuario para llamar el procedure correspondiente de esa sección.

```
;-------------------------------------------------------------------------
;                                                        MAIN
;-------------------------------------------------------------------------
main proc
    mov ax,@data
    mov ds,ax

    ciclo_principal:
        Imprimir msg_menu
        call LeerLinea
        call Limpiar

        cmp var_input[0], '1' ; INGRESAR FUNCIÓN
        jne sig2
        call IngresarFuncion
    sig2:
        cmp var_input[0], '2' ; CARGAR ARCHIVO
        jne sig3
        call SubirArchivo
    sig3:
        cmp var_input[0], '3' ; IMPRIMIR FUNCIONES GUARDADAS
        jne sig4
        call ImprimirFunciones
    sig4:
        cmp var_input[0], '4' ; DERIVAR FUNCIÓN
        jne sig5
        call DerivarFuncion
    sig5:
        cmp var_input[0], '5' ; INTEGRAR FUNCIÓN
        jne sig6
        call IntegrarFuncion
    sig6:
        cmp var_input[0], '6' ; RESOLVER FUNCIÓN
        jne sig7
        call ResolverFuncion
    sig7:
        cmp var_input[0], '7' ; GRAFICAR FUNCIÓN
```

```
        jne sig8
        Imprimir msg_bien
    sig8:
        cmp var_input[0], '8' ; ENVIAR ARDUINO
        jne sig9
        Imprimir msg_nose
    sig9:
        cmp var_input[0], '9' ; SALIR XD
        jne fin_ciclo_principal
        call Salir

    fin_ciclo_principal:
        Imprimir msg_seguir
        call LeerLinea
        call LimpiarEntradas
        jmp ciclo_principal
main endp
end main
```

MACROS IMPORTANTES A RESALTAR

Imprimir16ConMasyMenos: esta macro imprime un registro de 16 bits con el signo que corresponde. (Se realizó el mismo procedimiento para registros de 8 bits).

```
;-------------------------------------------------------------------------
;   IMPRIMIR CON MENOS Y MÁS
;-------------------------------------------------------------------------
Imprimir16ConMasyMenos macro valor
    local negat, pos, irse

    cmp valor,32768
    jnc negat
    jmp pos

    negat:
        Imprimir menos
        neg valor
        Imprimir16bits valor
        jmp irse
    pos:
        Imprimir mas
        Imprimir16bits valor
    irse:
endm
```

CargarArchivo: permite la lectura del archivo y esta macro es la encargada de separar, comprobar y guardar en memoria las funciones separadas por ‘;’.

```
;-------------------------------------------------------------------------
;   CARGAR ARCHIVO
;-------------------------------------------------------------------------
CargarArchivo macro ruta, handle, contenedor, cantidad
    LimpiarVariable contenedor
    Agregar00H ruta
    OpenFile ruta,handle
    ReadFile handle,contenedor,cantidad
    CloseFile handle
    SepararFunciones contenedor
    Imprimir msg_archivo_bien
endm
```

SepararFunciones: Analiza la cadena caracter por caracter para separarla por puto y coma.

```
;   SEPARAR FUNCIONES POR ';'
SepararFunciones macro texto
    local ciclo, seguir,seguir2,seguir3,funcion
    push si
    push di
    push ax

    mov si,0
    mov di,0
    ciclo:
        mov error_funcion,0
        ;IGNORAR \n y \r
        cmp texto[si],10 ; '\n'
        je seguir
        cmp texto[si],13 ; '\r'
        je seguir

        cmp texto[si],59 ; ';'
        je funcion

        mov al,texto[si]
        mov var_funcion[di],al
        inc di
    seguir:
        inc si
        cmp texto[si],'$'
        jne ciclo
    funcion:
        SepararTerminos var_funcion
        cmp error_funcion,1
        je seguir2
        GuardarFuncion var_funcion
    seguir2:
        LimpiarVariable var_funcion
        mov di,0
        inc si
        cmp texto[si],'$'
        jne ciclo
    pop ax
    pop di
    pop si  endm
```

SepararFunción: este macro recibe la función de *SepararFunciones* y las separa por signo para obtener los términos de la función.

```
;-------------------------------------------------------------------------
;   SEPARAR FUNCIÓN POR TÉRMINOS
;-------------------------------------------------------------------------
SepararTerminos macro funcion
    local ciclo, seguir,seguir2, negativo, termino, errorxd,finxd
    push si
    push di
    push ax

    mov si,0
    mov di,0
    ciclo:
        cmp funcion[si],'-'
        je negativo
        cmp funcion[si],'+'
        je termino
    seguir:
        mov al,funcion[si]
        mov var_termino[di],al ;GLOBAL
        inc di
        inc si
        cmp funcion[si],'$'
        jne ciclo
        jmp termino
    negativo:
        cmp var_termino[0],'$'
        je seguir
        dec si
    termino:
        ComprobarTermino var_termino
        cmp error_termino,0
        je seguir2
        mov error_funcion,1 ;global
    seguir2:
        LimpiarVariable var_termino
        mov di,0
        inc si
        cmp funcion[si],'$'
        jne ciclo
    finxd:
        pop ax
```

```
        pop di
        pop si
endm
```

ComprobarTermino: este macro es el encargado de analizar el término y verificar las restricciones impuestas en el enunciado.

```
;-------------------------------------------------------------------------
;   COMPROBAR TÉRMINO
;-------------------------------------------------------------------------
ComprobarTermino macro termino
       local  negativo,variable,  numero,  numero2,  seguir2,  exponente,
error,bien, fin_ct, fin_cadena
    push si

    mov error_termino,0
    mov si,0

    cmp termino[si],'-'
    je negativo
    cmp termino[si],'x'
    je variable
    EsNumero termino[si]
    cmp flag_numero,1
    je numero
    jmp error

    negativo:
        inc si
        cmp termino[si],'x'
        je variable
        EsNumero termino[si]
        cmp flag_numero,1
        je numero
        jmp error
    variable:
        inc si
        cmp termino[si],'^'
        je exponente
        jmp fin_ct
    exponente:
        inc si
        EsExponente termino[si]
```

```
        cmp flag_exp,1
        je fin_cadena
        jmp error
    fin_cadena:
        inc si
        cmp termino[si],'$'
        je fin_ct
        jmp error
    numero:
        inc si
        cmp termino[si],'x'
        je variable
        EsNumero termino[si]
        cmp flag_numero,1
        je numero2
        jmp fin_cadena
    numero2:
        inc si
        cmp termino[si],'x'
        je variable
        jmp fin_cadena
    error:
        mov error_termino,1
    fin_ct:
        pop si
endm
```

GuardarFunción: este macro recibe la función a guardar y busca la variable vacía para poder copiarlo.

```
;-------------------------------------------------------------------------
;   GUARDAR FUNCIÓN
;-------------------------------------------------------------------------
GuardarFuncion macro funcion
                                                                    local
s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,noh
ay,exit
    cmp func_1[0],'$'
    jne s2
    Copiar func_1,funcion
    jmp exit
    s2:
```

```
        cmp func_2[0],'$'
        jne s3
        Copiar func_2,funcion
        jmp exit
    s3:
        cmp func_3[0],'$'
        jne s4
        Copiar func_3,funcion
        jmp exit
    s4:
        cmp func_4[0],'$'
        jne s5
        Copiar func_4,funcion
        jmp exit
      . . .
      . . .
      . . .
    s17:
        cmp func17[0],'$'
        jne s18
        Copiar func17,funcion
        jmp exit
    s18:
        cmp func18[0],'$'
        jne s19
        Copiar func18,funcion
        jmp exit
    s19:
        cmp func19[0],'$'
        jne s20
        Copiar func19,funcion
        jmp exit
    s20:
        cmp func20[0],'$'
        jne nohay
        Copiar func20,funcion
        jmp exit
    nohay:
        Imprimir msg_nohay
    exit:
endm
```

DerivarFuncion: macro encargada de derivar la función ya separada en coeficientes y exponentes.

```
;-------------------------------------------------------------------------
;   DERIVAR FUNCIÓN
;-------------------------------------------------------------------------
Derivar macro coeficientes, exponentes
    local ciclo,continuar_ciclo
    push si
    push ax
    push bx
    mov si,0
    ciclo:
        cmp exponentes[si],0 ; IGNORAR CONSTANTES
        je continuar_ciclo
        ;MULTIPLICAR EXP * COEF
        mov al,coeficientes[si]
        mov bl,exponentes[si]
        imul bl
        mov var16_coef, ax
        ;RESTAR EXP-1
        mov al,exponentes[si]
        sub al,1
        mov var_exp,al
        Imprimir16ConMasyMenos var16_coef
        cmp var_exp,0
        je continuar_ciclo
        Imprimir equis
        Imprimir circunflejo
        Imprimir8bits var_exp
    continuar_ciclo:
        inc si
        cmp coeficientes[si],'$'
        jne ciclo
    pop bx
    pop ax
    pop si
endm
```

IntegrarFunción: macro encargada de integrar la función ya separada en coeficientes y exponentes.

```
;-------------------------------------------------------------------------
```

```
;   INTEGRAR FUNCIÓN
;-------------------------------------------------------------------------
Integrar macro coeficientes, exponentes
    local ciclo,continuar_ciclo
    push si
    push ax
    push bx
    mov si,0
    ciclo:
        ;SUMAR 1 AL exp
        mov al,exponentes[si]
        add al,1
        mov var_exp,al
        ;DIVIDIR COEF/(EXP+1)
        mov al,coeficientes[si]
        cbw
        mov bl,var_exp
        idiv bl
        mov var_coef, al
        Imprimir8ConMasyMenos var_coef
        Imprimir equis
        Imprimir circunflejo
        Imprimir8bits var_exp
    continuar_ciclo:
        inc si
        cmp coeficientes[si],'$'
        jne ciclo
        Imprimir mas_c
    pop bx
    pop ax
    pop si
endm
```

ResolverFunción: esta macro resuelve ecuaciones de la forma ax+b.

```
;-------------------------------------------------------------------------
;   RESOLVER FUNCIÓN - LINEAL
;-------------------------------------------------------------------------
ResolverLineal macro coeficientes
    push si
    push ax
    push bx
    mov var_a,0
    mov var_b,0
    mov si,0

    mov al,coeficientes[si]
    mov var_a,al
    inc si
    mov al,coeficientes[si]
    mov var_b,al
    neg var_b
    ;DIVISIÓN -B/A
    mov al,var_b
    cbw
    mov bl,var_a
    idiv bl
    mov var_coef,al
    Imprimir msg_resultado
    Imprimir8ConMasyMenos var_coef
    pop bx
    pop ax
    pop si
endm
```

ResolverCuadrática: este macro resulve ecuaciones de la forma ax^2+b

```
;-------------------------------------------------------------------------
;   RESOLVER FUNCIÓN - CUADRÁTICA 1
;-------------------------------------------------------------------------
ResolverCadratica1 macro coeficientes
    push si
    push ax
    push bx
    mov var_a,0
    mov var_b,0
    mov si,0

    mov al,coeficientes[si]
    mov var_a,al
    inc si
    mov al,coeficientes[si]
    mov var_b,al
    neg var_b
    ;DIVISIÓN -B/A
    mov al,var_b
    cbw
    mov bl,var_a
    idiv bl
    mov var_coef,al
    mov bl,1
    imul bl
    mov var16_coef, ax

    Sqrt var16_coef
    Imprimir msg_resultado
    Imprimir16ConMasyMenos var16_coef

    pop bx
    pop ax
    pop si
endm
```

metodos de video:

```
print macro cadena ;imprimir cadenas
    push ax
    push dx
    mov ax,@data
    mov ds,ax
    mov ah,09h ;Numero de funcion para imprimir cadena en pantalla
     mov dx,offset cadena ;especificamos el largo de la cadena, con la
instrucción offset
    int 21h  ;ejecutamos la interrupción
    pop dx
    pop ax
endm

close macro  ;cerrar el programa
    mov ah, 4ch ;Numero de función que finaliza el programa
    int 21h
endm

getChar macro ;obtener caracter
      mov  ah,01h ;se guarda en al en código hexadecimal del caracter
leído
    int 21h
endm


ModoVideo macro
    ;resolucion de 320x180
    mov ah, 00h
    mov al, 13h
    int 10h
    mov ax, 0A000h
    mov ds, ax  ; DS = A000h (memoria de graficos).
endm


ModoTexto macro
    mov ah, 00h
    mov al, 03h
    int 10h
endm
```

```
PintarMargen macro color
LOCAL primera, segunda, tercera, cuarta, x, y

mov dl, color

; barra horizontal superior
; pixel (x,y) = (20,0) = 20 x 320 + 10 = 6400
 mov di, 6410
primera:
    mov [di],dl
    inc di ; para que pinte a la derecha
    cmp di, 6709 ; 20 x 320 + 319
    jne primera

 ;barra horizontal inferior
 ; (180,0) = 180*320 = 57600
 mov di, 57610
segunda:
    mov [di],dl
    inc di ; para que pinte a la derecha
    cmp di, 57909 ; 180 x 320 + 319
    jne segunda

; barra vertical izquierda
mov di , 6410
tercera:
    mov [di],dl
    add di, 320
    cmp di, 57610
    jne tercera

; barra vertical derecha
mov di , 6709
cuarta:

    mov [di],dl
    add di, 320
    cmp di, 57909
    jne cuarta
```

```
mov di , 32010
x:
    mov [di],dl
    inc di
    cmp di, 32309
    jne x


mov di , 6560
y:
    mov [di],dl
    add di, 320
    cmp di, 57760
    jne y


endm

posicionarCursor macro x,y
    mov ah,02h
    mov dh,x
    mov dl,y
    mov bh,0
    int 10h
endm

imprimirVideo macro caracter, color
    mov ah, 09h
    mov al, caracter ;al guarda el valor que vamos a escribir
    mov bh, 0
    mov bl, color
    mov cx,1
    int 10h
endm
```