UNIVERSIDAD NACIONAL AUTONÓMA DE MÉXICO
FACULTAD DE INGENIERÍA

Estructura y Programación de Computadoras.

Proyecto.

Alumno: Uriarte Ortiz Enrique Yahir

Profesor: M.I. Miguel Israel Barragán Ocampo.

Grupo: 6

Fecha de entrega: Miércoles 11 de Enero 2023.

Semestre: 2023 – 1

INDICE.

OBJETIVO.

Aplicar los conocimientos de programación en ensamblador obtenidos durante el curso de Estructuras y Programación de Computadoras, para desarrollar un menú que muestre la hora del sistema en tiempo real, que ejecute diferentes operaciones con matrices e imprima los datos de entrada y salida, además de poder finalizar con ESC.

INTRODUCCION.

El ensamblador es un programa que "traduce" el programa fuente a un programa objeto (TASM), para posteriormente el Enlazador o Linker genere el ejecutable a partir de este (TLINK). Simplemente con este analizas podemos determinar que el archivo que ejecuta el TASM es el código fundamental para organizar la información en los registros (AX,BX,CX,DX,SI,DI,…,etc) y localidades de memoria.

Aunque antes de desarrollar líneas de código necesitamos conocer algunos conceptos básicos sobre como guardar los datos y como mostrarlos en pantalla como deseamos, como lo son: saber convertir cantidades entre los sistemas numéricos decimal, binario y hexadecimal (ya que esta arquitectura guarda toda la información en sistema hexadecimal); conocer los tamaños de memoria en bits de los registros y de las localidades de memoria, entender las operaciones que se pueden ejecutar sabiendo donde se guardan los resultados dependiendo los bits de esta, saber las partes y los tipos de registros que el CPU de la Arquitectura X86, conocer el código ASCII, comprender la capacidad de los saltos en el código, saber definir variables y cadenas, saber declara interrupciones, entre muchos otros conocimientos dependiendo la cantidad de opciones que se desean realizar.

La extensión para que el ensamblador reconozca los programas fuente es .ASM, una vez traducido el programa fuente, crea un archivo .OBJ, este archivo contiene un "formato intermedio" del programa, después el enlazador con el archivo .OBJ genera un programa ejecutable .EXE.

FILOSOFIA DE DISEÑO.

Antes que nada, debo aclarar que el programa tiene como limite que la matriz resultante o dato de resultado sea menor o igual a 99, sino los datos salen incorrectos por una mala localización, pues el código del programa solo puede trabajar con números de 2 digitos.

Para el desarrollo del código originalmente plante ejecutar todo en un solo procedimiento, sin embargo, cuando llegue a la parte para identificar la opción que el usuario quisiera ejecutar, los jmps serian demasiado largos, siendo imposibles de llevar a la etiqueta donde comenzaba el proceso, considere comer etiquetas y jmps intermedios para poder llegar, pero considere que esto haría mas largo el código. Por lo tanto, opte al final por hacer a cada operación en un procedimiento diferente y ser llamados al procedimiento principal cuando la comparación entre la tecla y la opción en ASCII fueran iguales. Para después de realizar dicho procedimiento se realizará una llamada en el procedimiento principal hacia el mismo procedimiento principal y así mantener la ejecución constante.

En lo que respecta a la parte de código en el procedimiento principal "*inicial*" la ejecución del ciclo para imprimir la hora del sistema, tuvo complicaciones en cuanto a finalizarlo, ya que la idea principal era solo la interrupción para salir int 21h ah = 0Bh, y luego solicitar en una pantalla limpia la opción a ejecutar. Sin embargo, fue más eficiente la int 21h ah = 0Bh, luego cmp al,0FFh y ejecutar un je para salir del ciclo, y luego la interrupción int 21h ah = 08h la cual considera la tecla previamente seleccionada como la que espera para continuar.

Otro procedimiento que se hizo fundamental fue "*confirmación*", este imprime las preguntas "Quiere realizar otra operación? SI(S)/No(ESC)" y dependiendo la respuesta, o se dirige al final de la función, o manda a llamar al procedimiento "*otraOption*" que imprime la pregunta para repetir la operación original con nuevos datos o salir al menú principal con "*ESC*", donde al final lo guarda en bl para realizar esta última comparación en la opción original.

El procedimiento "*retra*" es una función que convierte los valores de hexadecimal obtenidos de las operaciones en decimal con repeticiones de estos loop dependiendo la operación, si la operación es columna o renglón se

hace 4 veces, pero sí de la matriz M3. Lo que realiza es tomar el valor, dividirlo entre 10 e incrementarle a la parte alta y baja del registro ax 30h para colocarlos en el orden correcto y registrarlo en la matriz.

Los procedimientos matrix1 y matrix2 son casi iguales, más que por la diferencia de que cada una es para una matriz de ingreso inicial diferente, primero toma la cadena ingresada y la decrementa en 30h a cada a carácter, para después multiplicar cada numero por 10 y reacomodar los datos en la matriz M1r o M2r, siendo estas más fáciles para ejecutar las operaciones de cada opción.

Los procedimientos "*matrix1imp*" y "*matrix2imp*" así como matrix1 y matrix2 son casi iguales, salvo que cada una es para una matriz respectivamente y que matrix1imp se divide en otro proceso ya que como M1 es mas usada para todas las operaciones del programa a veces nos necesario hacer el cambio de valores que realiza el procedimiento por completo. Ambos procesos leen en si la matrix M1 o M2 y la posicionan para tomar los datos, y recorren toda la matriz adicionándole a cada uno 30h para pasar los valores de nuevo a decimal como se ingresaron, para después ser leídos de nuevo por si y llamar al procedimiento "*matrixposicionar*", e imprimir los valores de una forma mas parecida a la de una matriz de 4x4.

Pero, así como "*matrix1imp*" y "*matrix2imp*" son para M1 y M2, para la matriz M3 esta matrix3imp, en su caso solo se decrementa la dirección de M3 en si y manda llamar a "*matrixposicionar*". Para los 3 procedimientos se manda un valor a la variable "*recuerdaimp*" que representa la posición de columna en la se imprimirá la matriz

"*matrixposicionar*" es el procedimiento diseñado para para imprimir las matrices de una forma mas similar a la que tienen las matrices, y la ultima en ser llamada en los procedimientos "*matrix1imp*", "*matrix2imp*" y "*matrix3imp*", antes de llamar a estas se posiciona en los procedimientos de operaciones principales se posiciona el cursor. Se declara "*MTRx*" a di, siendo esta una cadena para imprimir elementos de las matrices de 4 en 4, luego inserta el contenido de la matriz en "*MTRx*" en un loop 4 veces e imprime está en la posición correspondiente. Ejecuta de nuevo el mismo procedimiento donde se quedo en matriz y lo imprime en el siguiente renglón, ejecutando esto 4 veces. Reconozco que este método se podía simplificar con otro loop 4 veces afuera del primer loop pero, cuando lo estaba desarrollando por alguna razón no lo ejecutaba correctamente e introducía valores incorrectos en "*MTRx*", por lo que opte al final por realizarlo de este modo, por el tiempo de desarrollo del Código.

"*matrixposicionar*" no fue el único caso donde tuve realizar una ejecución diferente al de un loop, para la opción de multiplicación determiné que debían ser 3 loops, uno dentro de otro para obtener el proceso, sin embargo, cuando realizaban este incrementaba 30h a los valores, y al tratar de arreglarlo solo se producían mas errores en la impresión, por lo que decidí probar que en lugar de hacer loop, se realizaran llamadas a procedimientos. Por eso "*multmmmrt*" se manda a llamar 4 veces en "*multmmmr*", y a la vez esta se llama 4 veces en "*multmmm*" que es llamada 4 veces en el procedimiento de la opción para ejecutar la multiplicación de matrices en el menú principal.

En "*multmmmrt*" se pasa el valor de M1r para multilpicarlo con M2r y añadirlo en la dirección de M3, después se posiciona M1r en el siguiente digito del renglón y se posiciona M2r en el siguiente digito de la columna.

En "*multmmmr*" se incrementa la posición de M3 y limpiamos esta, se llama 4 veces a "*multmmmrt*", y se posiciona M3 en el siguiente digito; regresamos la posición en M2r al siguiente digito del que iniciamos y regresamos la posición de M1r de donde comenzamos.

En "*multmmm*" se manda a llamar 4 veces el procedimiento "*multmmmr*" y luego se mueve M1 al siguiente renglón, también se le recuerda a M2r su posición original.

DOCUMENTACION CODIGO FUENTE.

```
page 60,132
title proyecto
.model small
diez equ 10
mueve equ mov
.stack 64
.data ; declaramos variables -------------------------------------------------------------

    ubica db 'AQUI TOY';cadena para ubicar memoria

    M1mnsj db 'M1:','$'
    M2mnsj db 'M2:','$'
    M3mnsj db 'M-Resultado:','$'

    M1 db '00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00','$';Primera matriz de datos
    M2 db '00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00','$';Segunda matriz de datos
    M3 db '00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00   ','$';Matriz resultante
    Columnas db '00,00,00,00  ','$';definimos columna
    Renglones db '00,00,00,00  ','$';definimos renglon
    MTRx db '00,00,00,00','$';definimos matriz para posicion

    ubicando db 'AQUI TOY' ;cadena para ubicar memoria, division entre datos acomodada

    M1r db '0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0','$' ;Primera matriz de datos en hexadecimal
    M2r db '0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0','$' ;Segunda matriz de datos en hexadecimal

    SumDiag db 00 ;definimos resultado de suma de diagonal principal
    SumDiagcad db '00','$' ;cadena para resultado de suma de diagonal principal
    ctaEXT dw 00 ;Cx es de 16 bits
    recuerda dw 00 ;guardar posicion en matriz acomodada
    recuerdb dw 00 ;guardar posicion de M2 matriz para multiplicacion
    recuerdaimp db 00 ;posicion de columna para matriz acomodada

    tiempo db 'HH:MM:SS:CS','$'
    titulo db 'PROGRAMA: OPERACIONES CON MATRICES.','$'
    Opdis db 'Teclee la opcion de dese realizar.','$'
    Opa db 'A) Suma de 2 Matrices.','$'
    Opb db 'B) Obtener Transpuesta.', '$'
    Opc db 'C) Multiplicar Matrices.', '$'
    Opd db 'D) Diagonal principal y suma.','$'
    Ope db 'E) Suma de columnas de una matriz.','$'
    Opf db 'F) Suma de renglones de una matriz.','$'
    Opg db 'Terminar programa - ESC.','$'
    Opax db 'Suma de 2 Matrices de 4X4.','$'
    Opbx db 'Obtener Transpuesta de 4X4.', '$'
    Opcx db 'Multiplicar Matrices.', '$'
    Opdx db 'Diagonal principal y suma.','$'
    Opex db 'Suma de columnas de una matriz.','$'
    Opfx db 'Suma de renglones de una matriz.','$'
    mimpes db 'Imprimir todas las matrices de la operacion? I(si)/N(no)','$'
    Opamp db '-> Introdusca los valores de la primera matriz:','$'
    Opams db '-> Introdusca los valores de la segunda matriz:','$'
    Opminsrt db '- Matrices insertadas -','$'
    Opfin db 'Programa terminado gracias por su preferencia :).','$'
    strOtraOp db 'Quiere realizar otra operacion? SI(S)/No(ESC)','$'
```

Fig.1. Código de proyecto (Variables)

La primera parte del Código se trata de declaración de variables, podríamos decir que existen 3 tipo de estas declaradas, las que son mensajes solo para imprimir como: Opa, Opams, M3mnsj o titulo; las que son para guardar valores en cadenas como las de matrices: M1r, M2, Columnas, MTRx o tiempo; y aquellas que son para guardar valores numéricos o direcciones como: SumDiag, ctaEXT, recuerdb.

También declaramos antes de las variables la página, el titulo del documento, el tamaño del modelo de memoria y el tamaño del stack.

```
.code ; iniciamos codigo ----------------------------------------------------------------
inicio proc far
    assume
    mov ax,@data
    mov ds,ax
    mov es,ax

    call limpia

; imprime cadena ---------------------------------------------------------------------
    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,21           ;columna
    int 10h             ;posiciona el cursor
    lea dx,titulo
    mov ah,09
    int 21h             ;imprime la esquina

    ; imprime menu de operaciones ------------------------------------------------------

    ; Imprime enuncionado de inicio ----------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,8        ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opdis
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion A -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,10       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opa
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion B -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,12       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opb
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion C -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,14       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opc
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion D -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,16       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opd
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion E -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,18       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Ope
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime opcion F -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,20       ;renglón
        mov dl,26       ;columna
        int 10H         ;ubicamos
        lea dx,Opf
        mov ah,09
        int 21h         ;imprime la esquina

    ; Imprime terminar -----------------------------------------------------------------
        mov ah,02
        mov bh,0        ;número de página
        mov dh,23       ;renglón
        mov dl,40       ;columna
        int 10H         ;ubicamos
        lea dx,Opg
        mov ah,09
        int 21h         ;imprime la esquina
```

Fig.2. y ***Fig.3.*** Código de proyecto (segmento de Código)

Después se declara el segmento de Código, y se mandan a imprimir las cadenas de texto que conforman la presentación del menú, con las interrupciones para posicionar la ubicación del cursor, los textos a imprimir en pantalla son: el título, los incisos de cada opción a ejecutar en el programa y un pequeño mensaje en la parte inferior derecha de la pantalla negra para recordarle al usuario que con la tecla ESC se puede salir del programa

```
anima:
; obteniendo tiempo ----------------------------------------------------------------
    mov ah,02ch
    int 21h

; cast de número a ascii -----------------------------------------------------------
    xor ax,ax
    mov al,ch
    mov bl,diez
    div bl          ;separando dígitos
    add al,30h      ;obteniendo su ascii
    add ah,30h;
    mueve si,offset tiempo
    mov [si],al     ;creando cadena de tiempo
    mov [si+1],ah   ;hora

; minutos --------------------------------------------------------------------------
    xor ax,ax
    mov al,cl
    mov bl,diez
    div bl          ;separando dígitos
    add al,30h      ;obteniendo su ascii
    add ah,30h
    mov [si+3],al   ;creando cadena de tiempo
    mov [si+4],ah   ;minutos
```

Fig.4. Código de proyecto (segmento de extracción y análisis del tiempo)

Se coloca posterior una etiqueta para repetir este ciclo, primero extraemos el tiempo del equipo, luego se caste para transformar los valores de hexadecimal a decimal en orden progresivo de izquierda a derecha, dividiendo el dato entre 10 y añadiéndole a la parte alta y baja del resultado 30h, primero con los datos de la hora y luego con los datos para los minutos.

```
; segundos ------------------------------------------------------------------------
    xor ax,ax
    mov al,dh
    mov bl,diez
    div bl          ;separando dígitos
    add al,30h      ;obteniendo su ascii
    add ah,30h
    mov [si+6],al   ;creando cadena de tiempo
    mov [si+7],ah   ;segundos

; centésimas de segundo ------------------------------------------------------------
    xor ax,ax
    mov al,dl
    mov bl,diez
    div bl          ;separando dígitos
    add al,30h      ;obteniendo su ascii
    add ah,30h
    mov [si+9],al   ;creando cadena de tiempo
    mov [si+10],ah  ;segundos

; ubicando cursor en centro de pantalla modo texto ---------------------------------
    mov ah,02h
    mov dh,5        ;renglón
    mov dl,32       ;columna
    mov bh,00       ;núm página
    int 10H         ;ubicamos

; imprimir tiempo ------------------------------------------------------------------
    mov ah,09h
    mov dx,si
    int 21h

; ciclo con salida esc -------------------------------------------------------------
    in al,60h
    dec al
    cmp al,00

; interrupcion  --------------------------------------------------------------------

    mov ah,02
    mov bh,0        ;número de página
    mov dh,21       ;renglón
    mov dl,26       ;columna
    int 10H         ;ubicamos

    mov ah,0Bh
    int 21h
    cmp al,0FFh
    je allright

jnz anima
```

Fig.5. Código de proyecto (segmento de extracción y análisis del tiempo)

Continuamos con el procedimiento de casteo con los segundo y centésimas de segundo. Todos los valores se fueron guardando en la cadena de texto “tiempo”, después ubicamos el cursor en el espacio que se dejo entre el título del menú y las opciones para mandar a imprimir la cadena “tiempo”, así mismo se declara el ciclo de salida para comparar que si el valor de al es diferente de 00 se mantiene el ciclo, luego posicionamos e cursor debajo de la opción f (esto último es solo para estética) y por ultimo se implementa una interrupción para que cuando se tecle cualquier tecla con código ASCII, se salga del ciclo de ”*anima*” y salte a la ejecución de lectura de opción.

```
allright:
call limpia

; imprime cadena -------------------------------------------------------------------
mov ah,08h
int 21h
mov bl,al
xor ax,ax

cmp bl,41h; Comparar con A
je suma
cmp bl,42h; Comparar con B
je transp
cmp bl,43h; Comparar con C
je multi
cmp bl,44h; Comparar con D
je dprin
cmp bl,45h; Comparar con E
je sumcol
cmp bl,46h; Comparar con F
je sumren
cmp bl,1Bh; Comparar con ESC
je exit

jmp inicio
```

Fig.6. Código de proyecto (direccionamiento a opción)

Una vez fuera del ciclo anima, se limpia la pantalla y se implementa una interrupción para introducir un valor en al sin que aparezca en pantalla, pero esta interrupción interpreta la entrada de la interrupción anterior para salir de anima para compara y dependiendo si el valor es igual se salta a la etiqueta correspondiente pero si no se encuentra una igualdad, vuelve a llamar al procedimiento principal "*inicio*", se llama a si misma.

```
    suma:
    call sumarm ; llama a operacion suma de matrices
    jmp inicio  ; llama a ejecutar el menu de nuevo
    transp:
    call tranptr; llama a operacion traspuesta de matriz
    jmp inicio
    multi:
    call multirm; llama a operacion multiplicacion de matrices
    jmp inicio
    dprin:
    call dprinrm; llama a operacion suma de la diagonal principal de la matriz
    jmp inicio
    sumcol:
    call sumcolrm; llama a operacion suma de columnas de la matriz
    jmp inicio
    sumren:
    call sumrenrm; llama a operacion suma de columnas de la matriz
    jmp inicio
    ; finaliza programa ------------------------------------------------------------
    exit:
    call limpia
        mov ah,02
        mov bh,0        ;número de página
        mov dh,12       ;renglón
        mov dl,15       ;columna
        int 10H         ;ubicamos
        lea dx,Opfin    ;mensaje de agradecimiento
        mov ah,09
        int 21h         ;imprime la esquina

    mov ah,08h
    int 21h             ;espera cualquier tecla

    call limpia

    mov ah,04ch         ;termina ejecucion
    mov al,0
    int 21h

inicio endp
```

Fig.7. Código de proyecto (direccionamiento a opción)

Dependiendo la etiqueta es la llamada al procedimiento que se va a realizar, pero después de las llamadas se manda un jmp a "*inicio*", para volver al menú y si el usuario lo desea ejecutar otra operación. En caso de saltar a la etiqueta exit: limpia la pantalla e imprime un mensaje para agradecer el uso del programa, limpia nuevamente y termina la ejecución del programa. Y con eso se finaliza el procedimiento principal del Código, el menú.

```
limpia proc
    MOV AH,0FH          ;limpia pantalla
    INT 10H
    MOV AH,0
    INT 10H
    ret
limpia endp
```

Fig.8. Código de proyecto (función para limpiar la pantalla)

Siendo así, uno de los principales procedimientos que se realizaron fue "*limpia*", para poder limpiar la pantalla cada vez que se fuera a imprimir un texto diferente o se hiciera una entrada a opción a través de interrupciones.

```
otraOption proc
    call limpia
    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,20           ;columna
    int 10h             ;posiciona el cursor
    lea dx,strOtraOp    ;Pregunta para continuar
    mov ah,09
    int 21h             ;imprime la esquina
otraOption endp

confirmacion proc
    mov ah,02
    mov bh,0            ;número de página
    mov dh,12           ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor
    lea dx,mimpes       ;Pregunta para imprimir
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,13           ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor
    mov ah,08h          ;Respuesta para imprimir
    int 21h
    mov bl,al
    xor ax,ax

    cmp bl,49h; Comparar con I
    je impma
    cmp bl,1Bh; Comparar con ESC
    je net

    net:
    call otraOption
    mov ah,08h
    int 21h
    mov bl,al
    xor ax,ax

    impma:

    ret
confirmacion endp
```

Fig.9. Código de proyecto (funciones para imprimir mensajes de impresión y ejecución)

Las funciones "*otraOpcion*" y "*confirmacion*" son impresiones de mensajes parar que el usuario decida si imprimir imprimir las matrices, esto con "*confirmacion*", si fueses el caso que no deseara imprimir las matrices, se envía otro mensaje para preguntar si desea repartir esta operación o salir directamente al menú y seleccionar otra opción.

```
matrix1 proc

    lea si,M1           ;lee la matriz 1
    mov cx,30h          ;48 veces
    inc si
    inc si              ;se coloca en la posicion correcta

    repetir:            ;Convertir a hexadecimal
        mov ah,[si]
        sub ah,30h      ;cambiamos a valor hexa
        mov [si],ah
        inc si
    loop repetir
    lea si,M1           ;Obtener valores hexadecimales reales M1
    lea di,M1r
    mov bl,diez
    mov cx,10h          ;16 veces
    mov dx,0

    realvalm1:          ;guarda en M1r matrix para operar
        inc si
        inc si          ;se coloca en la posicion correcta
        mov al,[si]
        mul bl
        mov ah,[si+1]
        add al,ah       ;AX = valor hexadecimal real
        inc si
        mov [di],al
        inc di
        inc di          ;se coloca en la posicion correcta
    loop realvalm1
    ret
matrix1 endp
```

Fig.10. Código de proyecto (funciones para convertir datos de la matriz uno en decimal a hexadecimal)

Para que el usuario tenga la facilidad de ingresar los datos de su matriz en su sistema numérico, decimal, la computadora trabaja más fácil con datos en hexadecimales, por lo tanto, este procedimiento convierte los datos restándole 30h a los números y sumado los dos dígitos en una nueva matriz para operar más fácil, esto con la función "*matrix1*".

```
matrix2 proc            ;misma ejecucion que matrix1 pero para M2

    lea si,M2
    mov cx,30h
    inc si
    inc si

    repetirb:
        mov ah,[si]
        sub ah,30h
        mov [si],ah
        inc si
    loop repetirb

    lea si,M2
    lea di,M2r
    mov bl,diez
    mov cx,10h
    mov dx,0

    realvalm2:
        inc si
        inc si
        mov al,[si]
        mul bl
        mov ah,[si+1]
        add al,ah
        inc si
        mov [di],al
        inc di
        inc di
        loop realvalm2
    ret
matrix2 endp
```

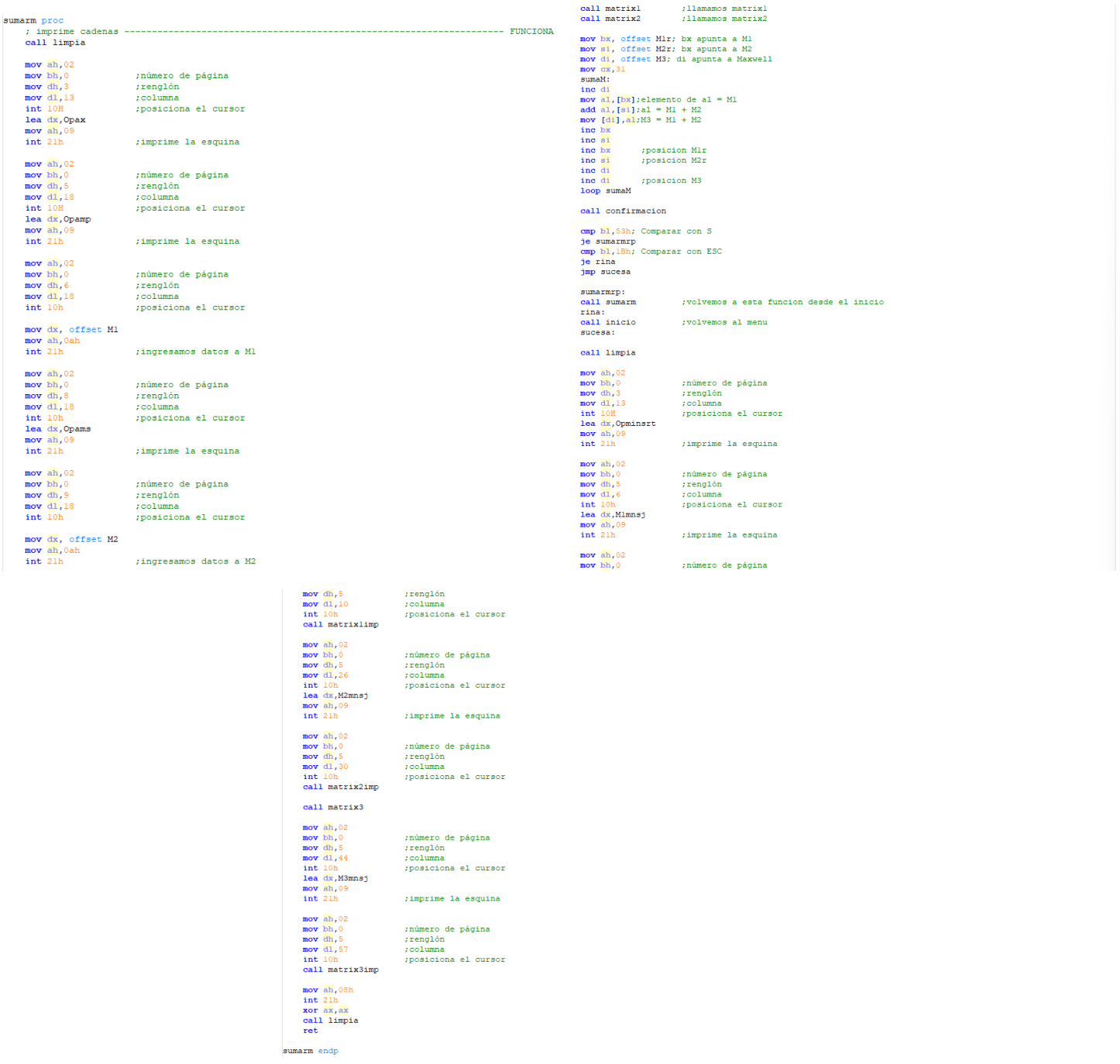
Fig.10. Código de proyecto (funciones para convertir datos de la matriz dos en decimal a hexadecimal)

Así mismo este procedimiento de se realiza para las dos matrices, para el otro caso con "*matrix2*", aunque en algunas operaciones principales del programa solo es necesario ejecutar lo q se encuentra en la matriz uno.

```
matrix3 proc
    lea si,M3           ;Obtener valores hexadecimales reales M3
    mov cx,10h
    mov dx,2Ch

    call retra          ; llama a operacion para Convertir a decimal - hexadecimal
    ret
matrix3 endp

matrixlimp proc
    lea si,M1           ;si = M1
    mov cx,30h
    inc si
    inc si              ;colocamos en posicion adecuada

    repetirc:           ;Convertir a hexadecimal Convertir a decimal - hexadecimal
        mov ah,[si]
        add ah,30h      ;cambiamos a valor hexa
        mov [si],ah
        inc si
    loop repetirc

    call matrixlimpcc   ; llama a operacion para imprimir m1
    ret
matrixlimp endp

matrixlimpcc proc
    mov si, offset M1;      ;si = M1
    mov recuerdaimp,10      ;posicion columna
    call matrixposicionar   ;llama a operacion para ordenar impresion
    ret
matrixlimpcc endp

matrix2imp proc         ;mismo procedimiento que matrixlimp pero para M2
    lea si,M2
    mov cx,30h
    inc si
    inc si

    repetird:
        mov ah,[si]
        add ah,30h
        mov [si],ah
        inc si
    loop repetird

    mov si, offset M2;      ;si = M2
    mov recuerdaimp,30      ;posicion columna
    call matrixposicionar   ;llama a operacion para ordenar impresion
    ret
matrix2imp endp

matrix3imp proc
    mov si, offset M3;      ;si = M3
    dec si
    dec si                  ;colocamos en posicion adecuada
   mov recuerdaimp,57      ;posicion columna
   call matrixposicionar   ;llama a operacion para ordenar impresion
   ret
matrix3imp endp
```

Fig.11. Código de proyecto (funciones para mandar a imprimir matrices)

Así como existe una función para los valores de las matrices 1 y 2, también desarrolle una para la matriz 3, M3, que en su caso el procedimiento solo manda a llama a la opción para "*retra*" ya que los datos obtenidos en M3 estarán en hexadecimal y se cambian a decimal. Y los procedimientos "*matrix1imp*", "*matrix2imp*" y "*matrix3imp*", como ya lo mencionamos en la filosofía de código, son para imprimir los datos de hexadecimal a decimal en la pantalla con un orden en estos que asemeje mas a una matriz de 4x4.



```
sumarm proc
    ; imprime cadenas ------------------------------------------------------------------------ FUNCIONA
    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opax
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1
    mov ah,0ah
    int 21h             ;ingresamos datos a M1

    mov ah,02
    mov bh,0            ;número de página
    mov dh,8            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor
    lea dx,Opams
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,9            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M2
    mov ah,0ah
    int 21h             ;ingresamos datos a M2

    call matrix1        ;llamamos matrix1
    call matrix2        ;llamamos matrix2

    mov bx, offset M1r; bx apunta a M1
    mov si, offset M2r; bx apunta a M2
    mov di, offset M3; di apunta a Maxwell
    mov cx,31
    sumaM:
    inc di
    mov al,[bx];elemento de al = M1
    add al,[si];al = M1 + M2
    mov [di],al;M3 = M1 + M2
    inc bx
    inc si
    inc bx      ;posicion M1r
    inc si      ;posicion M2r
    inc di
    inc di      ;posicion M3
    loop sumaM

    call confirmacion

    cmp bl,53h; Comparar con S
    je sumarmrp
    cmp bl,1Bh; Comparar con ESC
    je rina
    jmp sucesa

    sumarmrp:
    call sumarm         ;volvemos a esta funcion desde el inicio
    rina:
    call inicio         ;volvemos al menu
    sucesa:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,M1mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrix1imp

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,26           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M2mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,30           ;columna
    int 10h             ;posiciona el cursor
    call matrix2imp

    call matrix3

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    call matrix3imp

    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret

sumarm endp
```

Fig.12., ***Fig.13.*** y ***Fig.14.*** Código de proyecto (procedimiento para ejecutar la suma)

Para operación de suma todo el contenido esta en el procedimiento "*sumarm*", está compuesto por tres partes: la impresión de mensajes, la ejecución de operaciones y la impresión de las matrices.
La ejecución para obtener la matriz resultante es: primero ingresa los datos en las cadenas "*M1*" y "*M2*", luego con "*matrix1*" y "*matrix2*" se castea todo en "*M1r*" y "*M2r*" cadena a hexadecimal, luego suma cada posición de las matrices y las coloca en "*M3*", ya por ultimo se manda a llamar al procedimiento "*confirmacion*" donde se

imprimen los mensajes para continuar. Si es que se desea imprimir las matrices, se llama a “*matrix1imp*”, “*matrix2imp*”, “*matrix3*” y “*matrix3imp*” para regresar los datos a decimal e imprimirlos.

```
tranptr proc
    ; imprime cadena ------------------------------------------------------------------------ FUNCIONA
    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opbx
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1   ;dx = M1
    mov ah,0ah
    int 21h

    mov bx, offset M1   ;bx = M1
    mov si, offset M3   ;si = M3
    mov cx,04           ;4 veces
    inc bx
    inc bx              ;posicion correcta de M1
    proceExt:
        mov ctaEXT,cx   ;ctaEXT = cx, guardamos cx
        mov cx,04
        renACol:
            mov al,[bx]
            mov [si],al ;pasamos [M1] -> [M3]
            inc bx      ;cambiamos posicion
            inc si      ;cambiamos posicion
            mov al,[bx]
            mov [si],al ;pasamos [M1] -> [M3]
            add si,11   ;Cambio de renglon
            inc bx
            inc bx      ;cambiamos de numero
        loop renACol
        sub si,45       ;se mueve a la siguente columna
        mov cx,ctaEXT   ;recuperamos cx
    loop proceExt

    call confirmacion
```

```
    cmp bl,53h; Comparar con S
    je tranptrrp
    cmp bl,1Bh; Comparar con ESC
    je rinb
    jmp sucesb

    tranptrrp:
    call tranptr
    rinb:
    call inicio
    sucesb:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,Mlmnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrixlimpcc

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    call matrix3imp

    mov ah,08h


    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret

tranptr endp
```

Fig.15. y ***Fig.16.*** Código de proyecto (procedimiento para ejecutar la transpuesta)

El procedimiento para transpuesta es similar al de “*sumarm*” solo que en este caso solo se ocupa “*M1*” y no se realiza casteo a hexadecimal, solo un reacomodo de datos en “*M3*” de comunas y renglones, guardando con un loop 4 veces por los dígitos del renglón de “*M1*” y moviéndose 11 posiciones en “*M3*”, y otro loop por fuera para recorrer al siguiente renglón en “*M1*”. Se imprime los mismos mensajes de “*confirmacion*” y las matrices se imprimen sin recatearlas a decimal, en M1 solo se llama a “*matrix1impcc*” para organizarla y a “*matrix3impcc*” para “*M3*”.

```
multirm proc
    ; imprime cadena ------------------------------------------------------------------------ FUNACIONA
    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opcx
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1
    mov ah,0ah
    int 21h

    mov ah,02
    mov bh,0            ;número de página
    mov dh,8            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor
    lea dx,Opams
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,9            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M2
    mov ah,0ah
    int 21h

    call matrix1
    call matrix2

    mov bx, offset M1r; bx apunta a M1r
    mov si, offset M2r; si apunta a M2r

    mov di, offset M3; di apunta a M3
    mov recuerdb,si     ;recordamos ubicacion de M2r
    xor ax,ax

    call multmmm
    call multmmm
    call multmmm
    call multmmm        ;llamamos a la multiplicacion 4 veces por convencion de la operacion

    call confirmacion

    cmp bl,53h; Comparar con S
    je multirmrp
    cmp bl,1Bh; Comparar con ESC
    je rinc
    jmp sucesc

    multirmrp:
    call multirm
    rinc:
    call inicio

    sucesc:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,M1mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrixlimp

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,26           ;columna
    int 10h             ;posiciona el cursor

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,26           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M2mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,30           ;columna
    int 10h             ;posiciona el cursor
    call matrix2imp

    call matrix3

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    call matrix3imp

    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret
multirm endp
```

Fig.17., ***Fig.18.*** y ***Fig.19.*** Código de proyecto (procedimiento para ejecutar la multiplicación)

Para la multiplicación es lo mismo que "*sumarm*" solo que el procedimiento para por cuestiones técnicas se cambio a 4 llamadas a un procedimiento externo que se analizara mas adelante. Pero en lo demás se ejecuta como "*sumarm*".

```
dprinrm proc
    ; imprime cadena ------------------------------------------------------------------------ FUNCIONA
    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opdx
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1
    mov ah,0ah
    int 21h

    call matrix1

    xor ax,ax           ;ax = 0000
    mov bx,offset M1r   ;bx = M1r
    mov cx,04           ;4 veces
    sumaDiag:
        add al,[bx]     ;al = al + [bx]
        add bx,10       ;bx = bx + 10
    loop sumaDiag
    mov SumDiag,al      ;SumDiag = resultado de suma

    call confirmacion

    cmp bl,53h; Comparar con S
    je dprinrmrp
    cmp bl,1Bh; Comparar con ESC
    je rind
    jmp sucesd

    dprinrmrp:
    call dprinrm
    rind:
    call inicio
```

```
    sucesd:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,Mlmnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrixlimp

    lea si,SumDiagcad   ;si = SumDiagcad
    xor ax,ax           ;Convertir a decimal - hexadecimal
    mov al,[SumDiag]
    mov bl,diez
    div bl              ;ax = SumDiag/10
    add al,30h          ;obteniendo su ASCII
    add ah,30h
    mov [si],al
    mov [si+1],ah       ;posicion de datos [al ah]

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina
```

```
    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    mueve si,offset SumDiagcad
    mov ah,09h
    mov dx,si
    int 21h

    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret
dprinrm endp
```

Fig.20., ***Fig.21.*** y ***Fig.22.*** Código de proyecto (procedimiento para ejecutar la diagonal principal)

En la suma de la diagonal principal, la estructura del procedimiento se parece mas a la de procedimiento "*tranptr*", solo cambia su procedimiento principal y casteo del dato resultante. Para este, se la matriz bx cada 10 posiciones y se suma su valor en hexadecimal a la parte baja del registro ax 4 veces, por cada renglón y al final la suma se guarda en la variable "*SumDiag*" y se castea el valor de hexadecimal a decimal para imprimirlo en la cadena "*SumDiagcad*" lo demás es solo la impresión.

```
sumcolrm proc
    ; imprime cadena ------------------------------------------------------------------ FUNCIONA
    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10h             ;posiciona el cursor
    lea dx,Opex
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1   ;dx
    mov ah,0ah
    int 21h

    call matrix1

    mov bx, offset M1r
    mov si, offset Columnas
    mov cx,04
    otraCol:
        inc si
        xor ax,ax
        mov ctaEXT,cx
        mov cx,04
        SumaCol:
            add al,[bx]
            add bx,8
        loop SumaCol
        mov [si],al
        inc si
        inc si
        sub bx,30
        mov cx,ctaEXT
    loop otraCol

    call confirmacion

    cmp bl,53h; Comparar con S
    je sumcolrmrp
    cmp bl,1Bh; Comparar con ESC
    je rine
    jmp sucese

    sumcolrmrp:
    call sumcolrm
    rine:
    call inicio

    sucese:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,M1mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrixlimp

    lea si,Columnas
    mov cx,04h
    mov dx,2Ch

    call retra

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    mueve si,offset Columnas
    mov ah,09h
    mov dx,si
    int 21h

    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret
sumcolrm endp
```

Fig.23., ***Fig.24.*** y ***Fig.25.*** Código de proyecto (procedimiento para ejecutar la suma de columnas)

Para la suma de columnas se parece a el procedimiento "*tranptr*", imprime los mensajes para introducir la matriz, después se castea de decimal a hexadecimal en "*M1r*", y se recorre esta con saltos en su posición como cadena de 8, y el dato de esa posición de mueve a cadena "*Columnas*" y al final de ejecutarlo 4 veces regresa a l siguiente digito del que comenzamos para sumar ahora esa comuna y meterla en la siguiente posición de la cadena "*Columnas*" todo esto 4 veces. AL final solo castean tanto la Matriz 1 como la matriz "*Columnas*", para columnas se manda a llamar al procedimiento "retra" ya que solo necesita recorrerse 4 veces, y se imprime las matrices.

```
sumrenrm  proc
    ; imprime cadena ------------------------------------------------------------------------ FUNCIONA
    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10h             ;posiciona el cursor
    lea dx,Opfx
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,18           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opamp
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,18           ;columna
    int 10h             ;posiciona el cursor

    mov dx, offset M1
    mov ah,0ah
    int 21h

    call matrix1

    mov bx, offset M1r
    mov si, offset Renglones
    mov cx,04
    otroRen:
        inc si
        xor ax,ax
        mov ctaEXT,cx
        mov cx,04
        SumaRen:
            add al,[bx]
            add bx,2
        loop SumaRen
        mov [si],al
        inc si
        inc si
        mov cx,ctaEXT
    loop otroRen

    call confirmacion

    cmp bl,53h; Comparar con S
    je sumrenrmrp
    cmp bl,1Bh; Comparar con ESC
    je rinf
    call inicio

    sucesf:

    call limpia

    mov ah,02
    mov bh,0            ;número de página
    mov dh,3            ;renglón
    mov dl,13           ;columna
    int 10H             ;posiciona el cursor
    lea dx,Opminsrt
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,6            ;columna
    int 10h             ;posiciona el cursor
    lea dx,M1mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,10           ;columna
    int 10h             ;posiciona el cursor
    call matrixlimp

    lea si,Renglones
    mov cx,04h
    mov dx,2Ch

    call retra

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,44           ;columna
    int 10h             ;posiciona el cursor
    lea dx,M3mnsj
    mov ah,09
    int 21h             ;imprime la esquina

    mov ah,02
    mov bh,0            ;número de página
    mov dh,5            ;renglón
    mov dl,57           ;columna
    int 10h             ;posiciona el cursor
    mueve si,offset Renglones
    mov ah,09h
    mov dx,si
    int 21h


    mov ah,08h
    int 21h
    xor ax,ax
    call limpia
    ret
sumrenrm endp
```

Fig.26., ***Fig.27.*** y ***Fig.28.*** Código de proyecto (procedimiento para ejecutar la suma de renglones)

Para la renglones el procedimiento es exactamente el mismo que el de las columnas, “sumcolrm”, las diferencias son que: en la operación para mover los datos en lugar de recorrer la matriz “*M1r*” 8 veces, aquí solo se recorre 2 posiciones y la otra diferencia es q se emplean variables diferentes para guardar los datos, esto se hizo así para no confundir las cadenas al momento de analizar el código.

```
multmmm proc
    xor ax,ax
    call multmmmr
    call multmmmr
    call multmmmr
    call multmmmr
    add bx,8
    mov si,recuerdb
    ret
multmmm endp

multmmmr proc
    xor ax,ax
    inc di
    mov [di],al
    call multmmmrt
    call multmmmrt
    call multmmmrt
    call multmmmrt
    inc di
    inc di
    sub si,30
    sub bx,8
    ret
multmmmr endp

multmmmrt proc
    xor ax,ax
    mov al,[bx];elemento de al = M1
    mul byte ptr[si];al = M1 * M2
    add [di],al;M3 = M1 * M2
    add bx,2
    add si,8
    ret
multmmmrt endp
```

Fig.29. Código de proyecto (procedimiento para obtener la multiplicación)

Como había comentado, para el procedimiento de multiplicación realice 3 procedimientos, el primero a "*multmmm*" recorre M1r al siguiente renglón y le recuerda a M2r su posición original, el segundo; "*multmmmr*" mueve M3 y la limpia, y posiciona M3 en el siguiente digito; regresa las posiciones en M2r al siguiente digito del que iniciamos y en M1r de donde comenzamos y "*multmmmrt*" multiplica el valor de M1r para con M2r y lo coloca en M3, después mueve a M1r en el siguiente digito del renglón y M2r en el siguiente digito de la columna.

Todo esto es, lo que se explicó en la filosofía de diseño.

```
matrixposicionar proc
    mov di, offset MTRx; di apunta a Maxwell
    inc si
    inc si
    mov cx,04h

    repetirdaaaa:           ;Convertir a hexadecimal
        mov ah,[si]
        mov [di],ah
        inc si
        inc di
        mov ah,[si]
        mov [di],ah
        inc si
        inc si
        inc di
        inc di
    loop repetirdaaaa
    mov recuerda,si

    mueve si,offset MTRx    ;imprimir MTRx
    mov ah,09h
    mov dx,si
    int 21h

    mov si,recuerda
    mov di, offset MTRx; di apunta a Maxwell
    mov cx,04h

    repetirdaaaab:          ;Convertir a hexadecimal
        mov ah,[si]
        mov [di],ah
        inc si
        inc di
        mov ah,[si]
        mov [di],ah
        inc si
        inc si
        inc di
        inc di
    loop repetirdaaaab
    mov recuerda,si

    mov ah,02
    mov bh,0            ;número de página
    mov dh,6            ;renglón
    mov dl,[recuerdaimp];columna
    int 10h             ;posiciona el cursor

    mueve si,offset MTRx    ;imprimir MTRx
    mov ah,09h
    mov dx,si
    int 21h

    mov si,recuerda
    mov di, offset MTRx; di apunta a Maxwell
```

```
    mov si,recuerda
    mov di, offset MTRx; di apunta a Maxwell
    mov cx,04h

    repetirdaaaac:          ;Convertir a hexadecimal
        mov ah,[si]
        mov [di],ah
        inc si
        inc di
        mov ah,[si]
        mov [di],ah
        inc si
        inc si
        inc di
        inc di
    loop repetirdaaaac
    mov recuerda,si

    mov ah,02
    mov bh,0            ;número de página
    mov dh,7            ;renglón
    mov dl,[recuerdaimp];columna
    int 10h             ;posiciona el cursor

    mueve si,offset MTRx    ;imprimir MTRx
    mov ah,09h
    mov dx,si
    int 21h

    mov si,recuerda
    mov di, offset MTRx; di apunta a Maxwell
    mov cx,04h

    repetirdaaaad:          ;Convertir a hexadecimal
        mov ah,[si]
        mov [di],ah
        inc si
        inc di
        mov ah,[si]
        mov [di],ah
        inc si
        inc si
        inc di
        inc di
    loop repetirdaaaad
    mov recuerda,si

    mov ah,02
    mov bh,0            ;número de página
    mov dh,8            ;renglón
    mov dl,[recuerdaimp];columna
    int 10h             ;posiciona el cursor
```

```
    mueve si,offset MTRx    ;imprimir MTRx
    mov ah,09h
    mov dx,si
    int 21h

    ret
matrixposicionar endp

end inicio
```

Fig.30., ***Fig.31.*** y ***Fig.32.*** Código de proyecto (procedimiento para imprimir matrices)

Y el ultimo procedimiento en el código es "*matrixposicionar*" que ubica los datos de las matrices cada 4 números en un renglón diferente con la cadena "*MTRx*", pero desde el mismo renglón para dar la forma en la impresión, el código es repetitivo, ya que lee la cadena donde meteremos los 4 elementos (previamente antes de llamar a este procedimiento hay que leer la matriz a imprimir en el registro si) y se recorre en "*MTRx*", se manda imprimir "*MTRx*" en una posición, para luego repetir el proceso con los demás renglones,

Al final considero que pude realizar un loop para ahorrar las líneas de código, pero por tiempos decidí simplemente copiar el código 4 veces.

CONCLUSION.

Tras desarrollar el código de la manera más acorde a lo solicitado, creo que el programa cumple con la mayoría de las opciones solicitadas. Su desarrollo fue estresante y me ayudo a reforzar y recordar algunos conocimientos durante mis vacaciones que fue donde lo empecé. Entre algunos conocimientos que recordé es colocar '$' al final de una variable tipo cadena y el comando ret al final de un procedimiento para mantener la ejecución de este y no imprimir resultados muy largos. También recordé la importancia de respetar los tamaños de registros y de memoria para no producir errores en los movimientos y operaciones con estos. Por lo tanto, después del estrés de desarrollar el código y las complicaciones para mantener su ejecución, y ver que las operaciones son correctas considero que el objetivo del proyecto se cumplió.

Nota: muchas gracias profesor fue un dolor de cabeza desarrollar el código pero me hizo usar mi cerebro para comprender las operaciones que tenia que realizar y creo que ahora si domino mas el tasm.