



UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO  
CENTRO UNIVERSITARIO UAEM ATLACOMULCO



### Unidad de aprendizaje: Lenguaje Ensamblador

Docente: M. en C. C. Juan Carlos Ambriz Polo

Evaluación: Segundo parcial

Fecha de asignación: 9/04/2019

Fecha límite de entrega: 30/04/2019

Alumno: Gustavo Blas Duran

**Nota1:** Recuerde que las primeras entregas obtiene el 100% del valor, posteriormente por cada clase que transcurra reduce el 10% esto hasta llegar a la fecha límite, ya que después de dicha fecha no se aceptan entregas. Lea detenidamente las instrucciones y evite omitir pasos, trabaje sobre este formato de lo contrario no se recibirá la práctica.

## Practica 2 Ciclos y Saltos

### Objetivo:

Implementar ciclos numéricos así como saltos y saltos condicionales en lenguaje ensamblador.

### Actividad 1: Resumir conceptos principales del tema

Redacte un resumen de los ciclos numéricos así como de los saltos, saltos condicionales, incrementos, decrementos y comparaciones. Nota2: extensión mínima dos cuartillas.

En los lenguajes de programación, es necesario hacer repeticiones de una instrucción o un bloque de instrucciones un número determinado de veces. Es para esto que se utilizan los ciclos. En el lenguaje ensamblador existen cinco tipos de ciclos, estos van dependiendo al valor de los registros.

El ciclo LOOP, decrementa el registro CX, cuando el valor de CX se encuentra en cero, se termina el ciclo y se ejecutan las instrucciones que se encuentran después. Para determinar el número de ciclos, se le tiene que mover a el registro CX, el valor de las repeticiones. La sintaxis es: *LOOP Destino*. Para la ejecución de la instrucción LOOP se requieren dos pasos: primero se resta 1 a el registro CX, después CX se compra con cero. Si se tiene alguna instrucción de incrementar a CX dentro del ciclo LOOP, este nunca llegará a cero, pues el ciclo no se detendrá nunca. Es por eso que para realizar movimientos dentro del ciclo se debe evitar el uso del registro CX.

El ciclo LOOPE, en esta función igualmente que en los ciclos LOOP, decrementa el registro CX, pero a la vez si el ZF es diferente de uno, se ejecuta la siguiente instrucción, es decir, se sale del ciclo. El ciclo LOOPNE, salta si no es igual.

La instrucción JMP es una transferencia incondicional hacia un destino, la cual se identifica mediante una etiqueta de código que el ensamblador traduce en un desplazamiento, la sintaxis es *JMP destino*. Cuando la CPU ejecuta una transferencia incondicional, el desplazamiento de destino (a partir del inicio del segmento de código) se mueve hacia el apuntador de instrucciones, cual provoca que la ejecución continúe en la nueva ejecución.

La instrucción JMP proporciona una manera sencilla de crear un ciclo, saltando a una etiqueta a la parte superior, es decir, puede funcionar casi como un ciclo LOOP, pero este nunca tendrá un fin, a menor que se le agregue una condición que haga salirse del ciclo. En esta instrucción podemos hacer uso del registro CX, podemos alterar su dato, pues no hace uso el ciclo de este registro para terminar.

Las instrucciones INC (incremento); suma uno y DEC (decremento) resta uno de un solo operando. Estas instrucciones se pueden usar para poder manipular variable que necesitamos que vayan incrementando, pueden ir dentro un ciclo, del cual podemos incrementar o decrementar valores para la solución a problemas. Si debemos incrementar mas de una vez alguna variable, el uso de esta instrucción no es recomendable, pues solo escribiríamos líneas de código que se pueden resumir con instrucciones como ADD: esta es una instrucción que funciona como una suma, pues a través de esta podemos añadir valores a una variable. La sintaxis de esta instrucción es la siguiente: *ADD destino, origen* pues podemos agregar el valor de un registro a otro, o una constante a un registro. Pues es una instrucción que funciona como una suma. Pero también, al igual que la instrucción DEC, existe la instrucción SUB, que es una instrucción de resta, que funciona de la misma manera que ADD, solo que en vez de suma es resta, su sintaxis es *SUB destino, origen*. Estas instrucciones son muy usadas en los programas, pues existen muchos problemas en lo que necesitamos sumar o restar valores para obtener una solución.

Una instrucción de la cual se hizo uso en esta practica fue XCHG la cual intercambia el contenido de dos operandos. Pues en ocasiones necesitamos intercambiar valores, entre dos campos o variables, pero necesitamos un auxiliar para no perder los datos, esta instrucción es muy practica pues nos ahorra instrucciones. De esta manera se puede optimizar mas aun el programa.

La instrucción CMP (comparar) resta implícitamente un operando de origen de un operando de destino, ninguno de los operandos se modifica, la sintaxis de esta instrucción es la siguiente *CMP destino, origen*. Es una valiosa herramienta para crear estructuras lógicas condicionales. Cuando se coloca después de CMP una instrucción de salto condicional, el resultado es el equivalente en lenguaje ensamblador de una instrucción IF.

Podemos aplicar saltos con base a especificaciones de CMP. Una vez que la instrucción CMP se le hayan pasado los valores, se aplica un salto según el estado de las variables, es decir estos saltos hacen la pregunta a CMP dependiendo de los valores que se le hayan pasado. Por ejemplo, la condición JE realiza el salto si las variables que contiene CMP son iguales; JNE si son diferentes; JL si el destino es menor al origen; JG si el destino es mayor al origen; JLE si el destino es menor o igual al origen y JGE si el destino es mayor o igual al origen.

Gracias a estos saltos de pueden crear instrucciones lógicas, pues uno de los principales usos es en los ciclos, pues podemos insertar una comparación y un salto condicional para poder salir de un ciclo y terminar lo que este se encontraba realizando.

Es un uso principalmente en el ciclo JMP, pues como este es un ciclo infinito nunca terminaría a menos que se le indique un salto, de acuerdo con ciertas condiciones. Estas condiciones son muy usadas dentro de esta práctica, en la cual hacen el uso de varios saltos condicionales.

La instrucción LEA es útil para inicializar un registro con una dirección de desplazamiento. De hecho, un nombre más descriptivo para esta instrucción seria 'Load Offset Address' Carga una dirección de desplazamiento. La sintaxis de esta instrucción es: *LEA registro/memoria*. El uso común de LEA es para inicializar el desplazamiento en el registro BX, DI, Si para indexa una dirección de memoria.

DB es el único formato que define una cadena de caracteres que excede a dos caracteres, y los almacena en la secuencia normal, de izquierda a derecha. Es un formato condicional para la definición de datos de caracteres de cualquier longitud.

El operador DUP asigna almacenamiento para varios elementos de datos, usando una expresión constante como contador. En especial, es útil cuando se asigna espacio para una cadena o un arreglo y pueden utilizarse con datos inicializados o sin inicializar. A través de este operador podemos generar un arreglo y podemos manipularlo, es decir podemos insertarle datos como números o letras

y después ordenarlos. En este caso podemos hacer uso de algoritmos como en otros lenguajes de programación. Por ejemplo, podemos hacer uso de el algoritmo de ordenamiento del método de la burbuja para ordenar este arreglo. Este tipo de operaciones es indispensable en distintos programas. La colocación del cursor es un requisito común en modo texto, ya que su posición determina en donde desplegado el siguiente carácter. La interrupción 10H es la operación del BIOS para manejo de la pantalla. Con frecuencia un programa tiene que desplegar mensajes al usuario que solicite datos o le indique que ejecute una acción. A través de esta interrupción tenemos diferentes funciones para el manejo de la pantalla. También existen funciones que pueden hacer cambio en el color de los caracteres que se despliegan en la pantalla. Como ya se dijo anteriormente también para fijar la posición del cursor. A través de la función 09H de la interrupción 21H, también podemos desplegar caracteres en la pantalla.

El uso del teclado es indispensable en la ejecución de un programa, por lo cual también existen funciones e interrupciones que permiten la manipulación del teclado, pues podemos insertar valores para poder ejecutar un programa. Tenemos que hacer uso de los incrementos y decrementos para poder obtener el valor original de la tecla, en caso de los números. Pues al insertar o presionar una tecla, el programa recibe valores en ASCII, los cuales se encuentran en sistema hexadecimal. Es por eso por lo que se tiene que manipular los valores de entrada.

## Actividad 2: Codificación de programas en ensamblador

**Instrucciones:** Codifique los siguientes programas en ensamblador

1. Programa en ensamblador que despliegue  $N$  veces un mensaje mediante el uso de ciclos, permitiendo al usuario ingresar el valor de  $N$

```

1  .MODEL SMALL
2  .STACK
3  .DATA                                ;SEGMENTO DE DATOS VARIABLES
4      MSG1 DB 10,13,"INSERTE EL NUMERO DE CICLOS: ","$"
5      MSG2 DB 10,13,"ERROR.... INSERTE CARACTER VALIDO",10,13,"$"
6      MSG3 DB 10,13,"HOLA MUNDO","$"
7  .CODE
8
9      MOV AX,@DATA
10     MOV DS,AX
11     JMP CLS
12     NEXTCLS:
13 INICIO:
14     ;se muestra el primer mensaje
15     MOV AH,09H
16     MOV DX,OFFSET MSG1
17     INT 21H
18
19     ;se inserta el caracter
20     MOV AH,01h                ;lee una tecla
21     INT 21H
22
23     ;se inserto una opcion invalida (caracter)
24     CMP AL,30H                ;si es menor
25     JLE DEFAULT
26     CMP AL,40H                ;si es mayor
27     JGE DEFAULT
28

```

```

29
30 MUESTRAMSG:
31     MOV AH,09H
32     MOV DX,OFFSET MSG3
33     INT 21H
34     CMP AL,31H
35     JE SALIR
36     DEC AL
37     LOOP MUESTRAMSG
38
39 SALIR:
40     ;se devuelve el control al sistema
41     MOV AH,4CH
42     INT 21H
43
44 DEFAULT:           ;cls screen y muestra error
45     MOV AH,00H
46     MOV AL,3
47     INT 10H
48     MOV AH,09H
49     MOV DX,OFFSET MSG2
50     INT 21H
51     JMP INICIO
52 CLS:
53     MOV AH,00H
54     MOV AL,3
55     INT 10H
56     JMP NEXTCLS
57
58 END

```

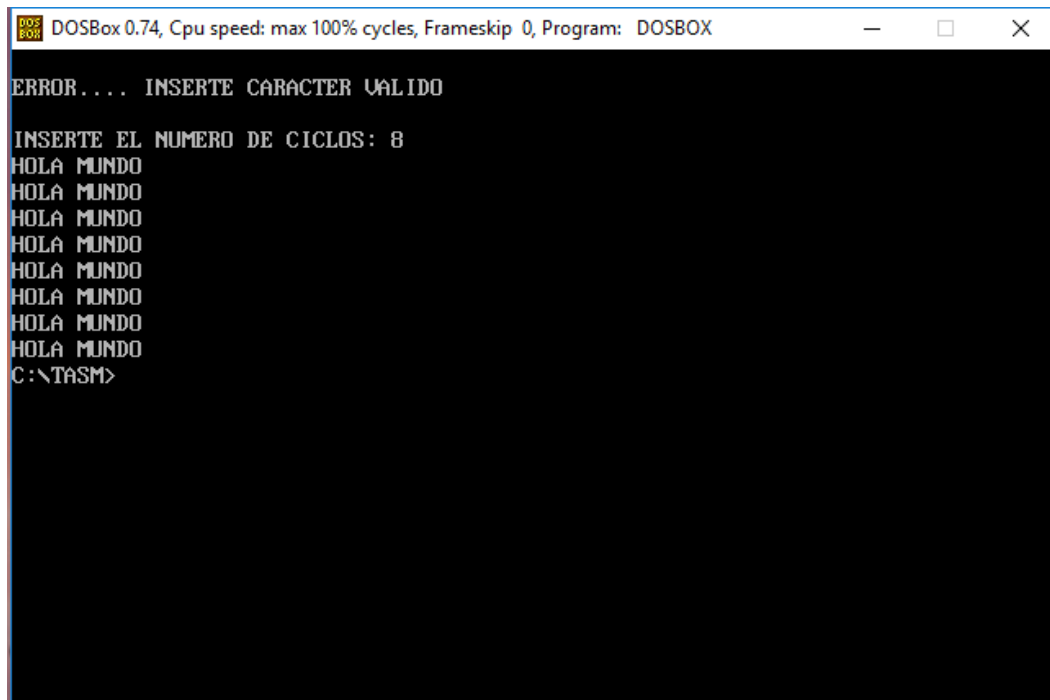


Ilustración 1 Programa 1: el usuario inserto 8, por lo tanto se muestran 8 veces el mensaje en pantalla

2. Programa en ensamblador que permita ingresar 5 números e imprima el número mayor de estos, haga uso de ciclos y de comparaciones.

```
1  .MODEL SMALL
2  .STACK
3
4  .DATA
5
6      STRING DB 5 DUP(?)           ;set string of 5
7
8      ;input prompts
9      MSG1 DB 10,13,"INSERTA VALOR 0-9: ","$"
10     MSG2 DB 10,13,"ERROR... INSERTA SOLO VALORES 0-9","$"
11     ;output prompts
12     V1 DB 10,13,36
13     MSG3 DB 10,13,"EL STRING LLENO ES: ",36
14     MSG4 DB 10,13,"STRING ORDENADO: ",36
15     MSG5 DB 10,13,"VALOR MAS GRANDE: ",36
16
17  .CODE
18     MOV AX,@DATA
19     MOV DS,AX
20     MOV CX,5                     ;initialize the count with 5
21     MOV SI,0                     ;initialize SI with 0
22     INPUTSTRING:                ;input value of the character
23         ;show message prompts input
24         INICIO:
25         MOV AH,09H
26         MOV DX,OFFSET MSG1
27         INT 21H
28         ;read key, return AL
29         MOV AH,01H
30         INT 21H
31
32         ;se inserto una opcion invalida (caracter)
33         CMP AL,2FH               ;si es menor
34         JLE DEFAULT
35         CMP AL,40H               ;si es mayor
36         JGE DEFAULT
37         JMP N1
38         DEFAULT:                 ;cls screen y muestra error
39         MOV AH,00H
40         MOV AL,3
41         INT 10H
42         MOV AH,09H
43         MOV DX,OFFSET MSG2
44         INT 21H
45         JMP INICIO
46     N1:
47         ;insert the value in the string
48         MOV STRING[SI],AL
49         MOV AH,09H
50         MOV DX,OFFSET V1
51         INT 21H
52         ;indecreease the count
```

```

53         INC SI
54
55     LOOP INPUTSTRING        ;end INPUTSTRING
56
57     ;show message
58     MOV AH,09H
59     MOV DX,OFFSET MSG3
60     INT 21H
61     ;initialize CX and SI, because the value change before
62     MOV CX,5
63     MOV SI,0
64
65     PRINTSTRING:            ;print the string
66         MOV AH,02H
67         MOV DL,STRING[SI]
68         INT 21H
69         MOV AH,02H
70         MOV DL,' '
71         INT 21H
72         INC SI
73     LOOP PRINTSTRING        ;END PRINTSTRING
74
75     ;*****INICIA ORDENAMIENTO BURBUJA*****
76
77     MOV CX,5
78     DEC CX                    ;DECREMENTA CUENTA
79     CICLO1:
80         PUSH CX                ;GUARDA CUENTA DEL CICLO EXTERNO
81         MOV SI,0                ;APUNTA AL PRIMER VALOR
82
83     CICLO2:
84         MOV AL,STRING[SI]        ;OBTIENE EL VALOR DEL ARREGLO
85         CMP STRING[SI+1],AL      ;COMPARA PAR DE VALORES
86         JLE CICLO3                ;SI [SI] <= [DI], NO INTERCAMBIA
87         XCHG AL,STRING[SI+1]    ;INTERCAMBIA EL PAR
88         MOV STRING[SI],AL
89
90     CICLO3:
91         INC SI                    ;MUEVE APUNTADORES HACIA ADELANTE
92         LOOP CICLO2                ;CICLO INTERNO
93
94         POP CX                    ;OBTIENE LA CUENTA DEL CICLO EXTERNO
95         LOOP CICLO1                ;CUALQUIER OTRO CASO REPITE CICLO EXTERNO
96
97     ;*****TERMINA ORDENAMIENTO BURBUJA*****
98     ;show message
99     MOV AH,09H
100    MOV DX,OFFSET MSG4
101    INT 21H
102    ;initialize value for the print
103    MOV SI,0
104    MOV CX,5
105    PRINTSTRINGINORDER: ;print the string
106        MOV AH,02H
107        MOV DL,STRING[SI]
108        INT 21H
109        MOV AH,02H

```

```

110         MOV DL, ' '
111         INT 21H
112         INC SI
113     LOOP PRINTSTRINGINORDER ;END PRINTSTRING
114     ;show message bigger value
115     MOV AH,09H
116     MOV DX,OFFSET MSG5
117     INT 21H
118     ;initialize the value
119     MOV SI,0 ;INDICE
120     MOV CX,5 ;
121     MOV AL,0 ;VALOR MAS GRANDE
122     PLUSVALUE:
123         CMP AL,STRING[SI]
124         JGE NOTSWAP
125         XCHG AL,STRING[SI]
126     NOTSWAP:
127         INC SI
128         LOOP PLUSVALUE
129     ;show the value bigger
130     MOV DL,AL
131     MOV AH,02H
132     INT 21H
133
134     ;return the control of the system
135     MOV AH,4CH
136     INT 21H
137
138     END

```

```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX
C:\TASM>p2e2.exe
INSERTA VALOR 0-9: 9
INSERTA VALOR 0-9: 1
INSERTA VALOR 0-9: 0
INSERTA VALOR 0-9: 5
INSERTA VALOR 0-9: 6
EL STRING LLENO ES: 9 1 0 5 6
STRING ORDENADO: 9 6 5 1 0
VALOR MAS GRANDE: 9
C:\TASM>

```

*Ilustración 2 Programa 2; el usuario inserta 5 números cualesquiera, el programa los ordena y muestra el valor más grande. Si es usuario inserta alguna otra tecla que no sea un número, el programa pide el valor en la posición i hasta que sea un numero valido.*

- Programa en ensamblador que imprima una figura de rombo mediante un carácter ingresado por el usuario, haga uso de ciclos para generar la figura.

```

*
***
*****
*****
*****
***
*

```

```

1  .model small
2  .stack
3  .data
4      msg1 db 10,13,"inserte numero de ciclos:  ",36
5      msg2 db 10,13,"Error ... solo numeros del 1-9",10,13,36
6      msg3 db 10,13,"inserte variable en figura: ",36
7      jumpLine db 10,13,36
8      maxima db 0,36
9      segmentoEspacio db 0,36
10     num db 0,36
11     var db 0,36
12 .code
13     mov ax,@data
14     mov ds,ax
15     mov ah,09h           ;mensaje de entrada de variable a imprimir
16     mov dx,offset msg3
17     int 21h
18     mov ah,01h           ;se inserta el caracter
19     int 21h
20     mov var,al           ;se guarda en num la variable de entrada al
21     mov segmentoEspacio,al
22     mov dl,var
23     add dl,30h
24     mov si,0
25 inicio:
26     mov ah,09h           ;mensaje de entrada de numero de ciclos
27     mov dx,offset msg1
28     int 21h
29     mov ah,01h           ;se inserta el caracter
30     int 21h
31
32     cmp al,39h
33     jg inicio
34     cmp al,32h
35     jl inicio
36
37     cmp al,30h
38     jle default
39     cmp al,40h
40     jg default
41     jmp correcto

```



```

42     default:
43         mov ax,0600h
44         mov bl,0ah
45         mov cx,0000h
46         mov dx,184fh
47         int 10h
48         mov ah,09h
49         mov dx,offset msg2
50         int 21h
51     jmp inicio
52     correcto:
53     sub al,30h
54     mov num,al
55     mov ah,09h             ;salto de linea
56     mov dx,offset jumpLine
57     int 21h
58 ;parte alta
59     mov al,num
60     mov cl,al
61     add cl,al
62     mov al,0
63     max:
64         inc al
65     loop max
66     dec al
67     mov bl,al
68     mov maxima,al
69     ;bl contienen el numero maximo de caracteres
70     mov si,1             ;iniciamos indice
71
72     figuraPA:
73     ;inicia impresion de espacios
74     dec segmentoEspacio
75     mov cl,segmentoEspacio
76     espacios:
77         mov ah,02h
78         mov dl,' '
79         int 21h
80     loop espacios
81     ;termina impresion de espacios
82     ; se le asigna el al el total de caracteres a imprimir
83     mov cx,si
84     mov al,0
85     maxC:
86         inc al
87     loop maxC
88     mov cl,al
89     sub cl,30h
90     cmp cl,maxima
91     je seguir
92     cmp cl,1
93     je cls
94     nextCls:
95         figuraPA1:             ;este ciclo imprime los caracteres
96         mov ah,02h
97         mov dl,var             ;var guarda el carcater a imprimir
98         int 21h

```

```

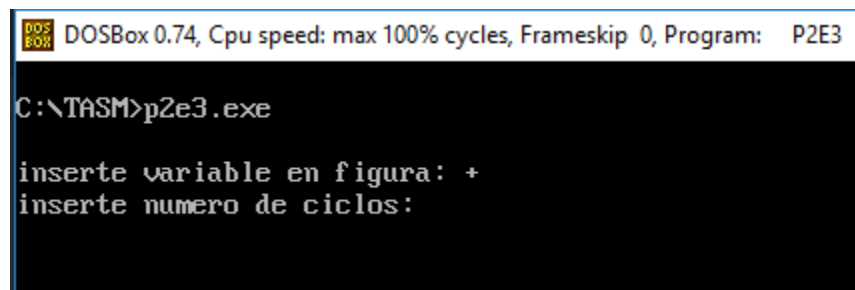
99         loop figuraPA1
100     mov  ah,09h
101     mov  dx,offset jumpLine
102     int  21h
103
104     add  si,02d
105     jmp  figuraPA
106
107     cls:                ;limpia pantalla para imprimir solo la figura
108     mov  cl,25
109     saltoCLS:
110         mov  ah,09h
111         mov  dx,offset jumpLine
112         int  21h
113     loop saltoCLS
114     ;inicia impresion de espacios
115     dec  segmentoEspacio
116     mov  cl,segmentoEspacio
117     espaciosCLS:
118         mov  ah,02h
119         mov  dl,' '
120         int  21h
121     loop espaciosCLS
122     ;termina impresion de espacios
123     mov  cl,1
124     jmp  nextCls
125
126
127     seguir:
128     ;parte maxima de caracteres y parte baja
129     mov  cl,02h
130     mov  al,num
131     mul  cl
132     mov  bx,ax
133     mov  cl,00h
134     mov  cl,bh
135     add  cl,bl
136     mov  cl,bl
137     figura:
138     dec  cl
139     figura2:            ;este ciclo imprime la variable
140         mov  ah,02h
141         mov  dl,var
142         int  21h
143     loop figura2
144     mov  ah,09h
145     mov  dx,offset jumpLine
146     int  21h
147     ;inicia impresion de espacios
148     inc  segmentoEspacio
149     mov  cl,segmentoEspacio
150     espacios2:
151         mov  ah,02h
152         mov  dl,' '
153         int  21h
154     loop espacios2
155     ;termina impresion de espacios

```

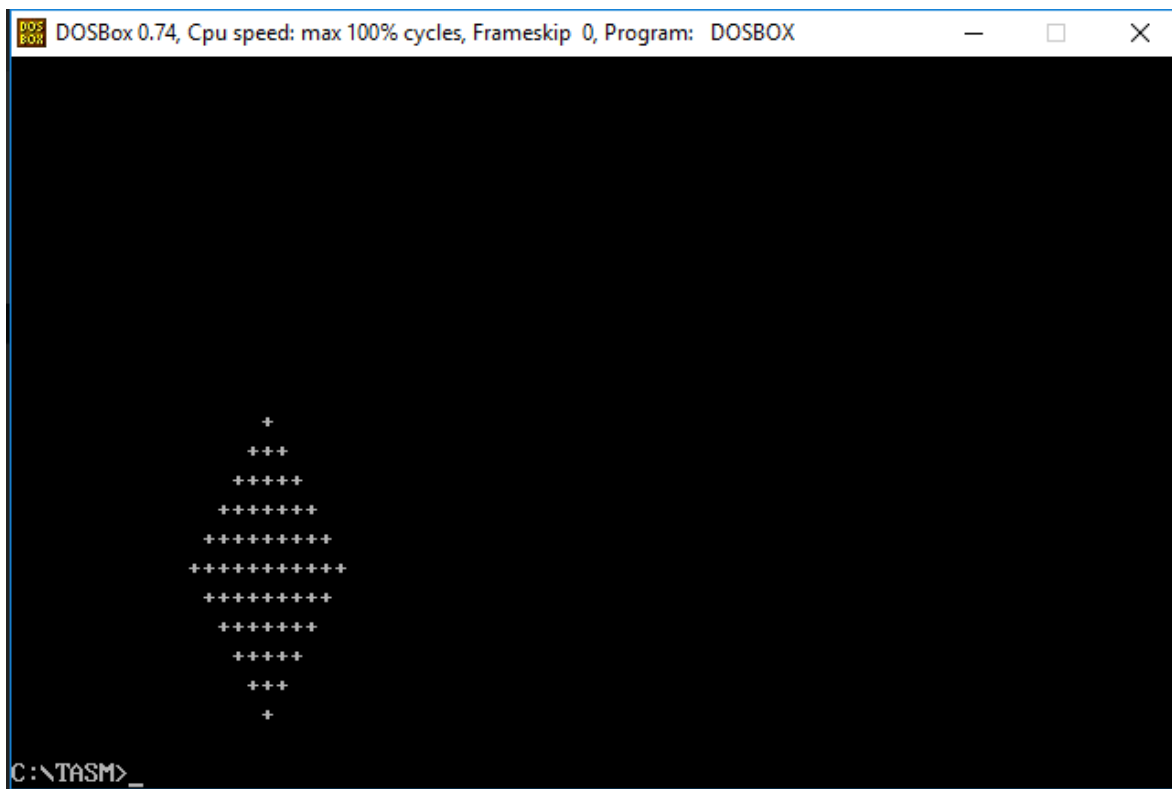
```

156         dec bl
157         dec bl
158         mov cl,bl
159         cmp cl,0
160         je salir
161     jmp figura
162 ;-----
163 salir:
164         mov ah,4ch
165         int 21h
166
167     end

```



*Ilustración 3 Programa 3; parte 1, el usuario inserta el carácter a imprimir.*



*Ilustración 4 Programa 3, el usuario inserta el numero de filas .*

4. Programa en ensamblador que genere la tabla de multiplicar del número que ingrese el usuario.

1x0=0

1x1=1

1x2=2

```
1  .model small
2  .stack
3  .data
4      v1 db 10,10,13,"inserte numero de tabla a mostrar: ",36
5      v2 db 10,13,"error... solo numeros del 1-9","$"
6      num db 0,36
7      saltodelinea db 10,13,36
8      espacio db 09,36
9  .code
10     mov ax,@data
11     mov ds,ax
12     mov cx,30
13     mov ah,09h
14     mov dx,offset saltodelinea
15     cls:
16         int 21h
17     loop cls
18
19 inicio:
20     ;mostramos mensaje para introducir numero
21     mov ah,09h
22     mov dx,offset v1
23     int 21h
24     ;lee el teclado
25     mov ah,01h
26     int 21h
27
28     cmp al,30h
29     jle default
30     cmp al,40h
31     jge default
32
33     sub al,30h
34     mov num,al
35     mov cl,00h
36     ;salto de linea
37     mov ah,09h
38     mov dx,offset saltodelinea
39     int 21h
40     int 21h
41 tabla:
42
43     mov ah,09h
44     mov dx,offset espacio
45     int 21h
46
47     mov al,num
48     add al,30h
```

```

49     mov ah,02h
50     mov dl,al
51     int 21h
52
53     mov ah,02h
54     mov dl,'x'
55     int 21h
56
57     add cl,30h
58     cmp cl,39h
59     jg diex
60
61     mov ah,02h
62     mov dl,'0'
63     int 21h
64     mov dl,cl
65     int 21h
66
67     afterdiex:
68     sub cl,30h
69
70     mov ah,02h
71     mov dl,'='
72     int 21h
73
74     mov al,num
75     mul cl      ;cl x al  R=al
76     aam        ;ajuste ASCII para multiplicaci3n
77     mov bx,ax   ;Se respalda la multiplicaci3n el BX
78     mov ah,02h
79     mov dl,bh
80     add dl,30h
81     int 21h
82     mov dl,bl
83     add dl,30h
84     int 21h
85     mov ah,09h
86     mov dx,offset saltodelinea
87     int 21h
88     inc cx
89     cmp cx,11
90     ja salir    ;salir
91     jb tabla    ;repetir
92
93     salir:
94     mov ah,4ch
95     int 21h
96     default:
97     mov ax,0600h
98     mov bl,0ah
99     mov cx,0000h
100    mov dx,184fh
101    int 10h
102
103    mov ah,09h
104    mov dx,offset v2
105    int 21h

```

```

106     jmp inicio
107 diex:
108     mov ah, 02h
109     mov dl, '1'
110     int 21h
111     mov ah, 02h
112     mov dl, '0'
113     int 21h
114     jmp afterdiex
115
116
117 end

```

The screenshot shows a DOSBox window titled "DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: DOSBOX". The window contains a text-based interface where the user has entered the number 6. The program displays a multiplication table for the number 6, ranging from 6x00 to 6x10. The prompt "C:\TASM>" is visible at the bottom of the window.

```

inserte numero de tabla a mostrar: 6

6x00=00
6x01=06
6x02=12
6x03=18
6x04=24
6x05=30
6x06=36
6x07=42
6x08=48
6x09=54
6x10=60

C:\TASM>

```

Ilustración 5 Programa 4: el usuario inserta 6, se muestra la tabla de multiplicar del número 6 desde 0 hasta 10

5. Genere un programa que despliegue un ciclo infinito de caracteres simulando un efecto en cascada (Efecto Matrix).

```
1  .MODEL SMALL
2  .STACK
3  .DATA
4      matrix DB 46 DUP ("
5      matrix4 DB 46 DUP (" ", "A", "B", "C", "D", "E", "
6      matrix2 DB 45 DUP ("10100110")
7      matrix3 DB 45 DUP (03H)
8      endl DB 10,13,36
9  .CODE
10     MOV AX,@DATA
11     MOV DS,AX
12     MOV SI,0
13     MOV CX,35
14     JMP L1
15     L1:
16         INC DH ;fila siguiente
17         L2:
18             ;color de fuente
19             MOV AH,9 ;funcion
20             MOV AL,matrix4[SI] ;caracter a mostrar
21             MOV BH,0 ;pagina de video
22             MOV BL,0AH ;atributos fondo/frente
23             MOV CX,2 ;repeticiones del caracter
24             INT 10H
25             INC SI
26             PUSH CX
27             MOV AH,02H
28             INT 10H
29             CMP SI,46 ;si llega el final de la pantalla
30             JE REINICIAR
31             CONTINUAR:
32             ;posicion del cursor
33             INC DL ;siguiente columna
34             MOV AH,02H
35             INT 10H
36             POP CX
37             CMP DL,35
38             JE L1
39             JMP L2
40     JMP L1
41     REINICIAR:
42         MOV SI,1
43     JMP CONTINUAR
44     END
```

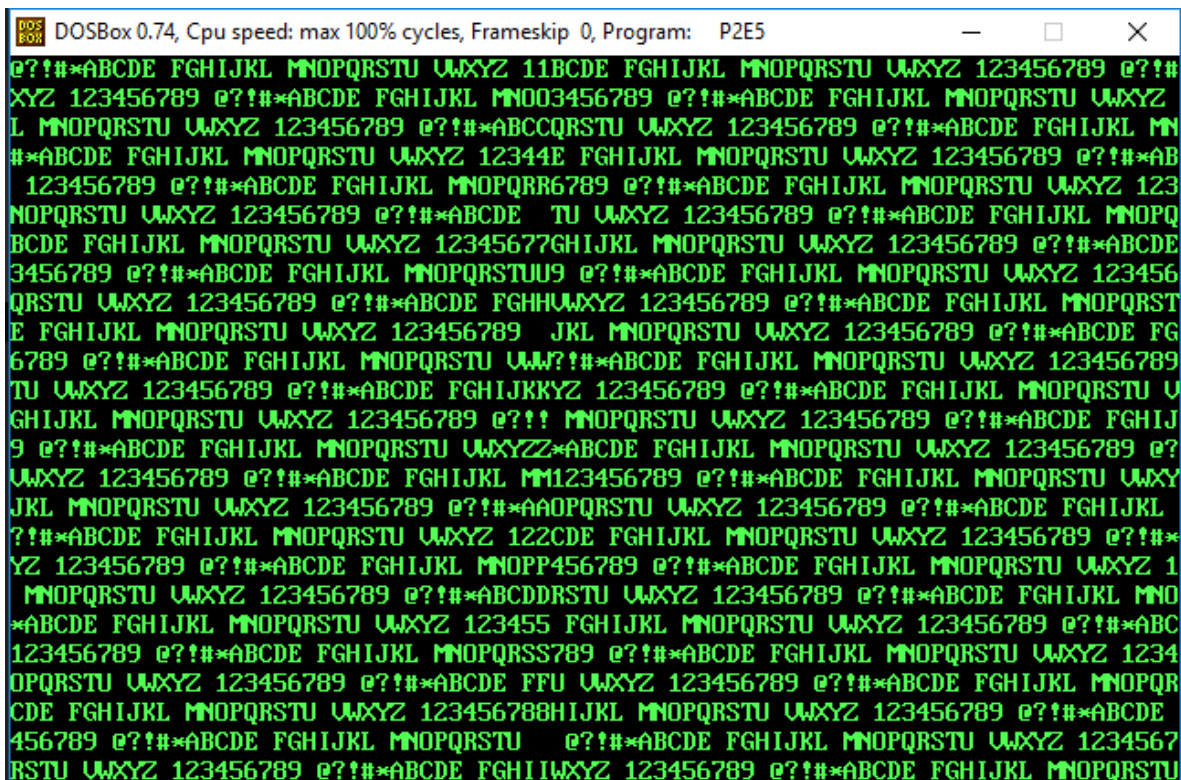


Ilustración 6 Programa 5: efecto MATRIX, se cambian los colores de fuente solo se cierra forzando el programa. Los caracteres se mueven hacia arriba. Pues se hace uso de la función 02H de la interrupción 10H en la cual se escribe en la posición del cursor, pues en este se va avanzando.

6. Programa en ensamblador que despliegue un menú como el siguiente, realice la función indicada y repita el menú hasta que se elija la opción salir:

- 1.- Limpiar pantalla
- 2.- Imprimir un mensaje
- 3.- Decrementar número ingresado
- 4.- Incrementar número ingresado
- 5.- Salir

```

1  .model small
2  .stack
3  .data
4      msgMenu db 10,10,13,"Menu",10,13,"1.- LIMPIAR
PANTALLA",10,13,"2.- IMPRIMIR UN MENSAJE",10,13,"3.- DECREMENTAR UN
NUMERO INGRESADO",10,13,"4.- INCREMENTAR UN NUMERO INGRESADO",10,13,"5.-
Salir",10,13,"inserte opcion: ",10,10,"$"
5      msgCLSScreen db 10,13,"se ha borrado la pantalla ",36
6      msgPrint db 10,13,"este es el mensaje",36
7

```



```

8      msgNDec db 10,13,"DECREMENTAR UN NUMERO",10,13,"inserta un
numero 1-9:      ",36
9      msgNDec1 db 10,13,"el numero ingresado es:      ",36
10     msgNDec2 db 10,13,"el numero decrementado es:   ",36
11
12     msgNInc db 10,13,"INCREMENTAR UN NUMERO",10,13,"inserta un
numero 0-8:      ",36
13     msgNInc1 db 10,13,"el numero ingresado es:      ",36
14     msgNInc2 db 10,13,"el numero ncrementados es:   ",36
15
16     msgDefault db 10,13,"Inserta caracter valido",10,13,36
17
18     .code
19     mov ax,@data
20     mov ds,ax
21     menu:
22     mov ah,09h          ;despliega una cadena de caracteres
23     mov dx,offset msgMenu ;la cadena entra a dx
24     int 21h            ;interrupcion
25     mov ah,01h          ;funcion para leer el teclado
26     int 21h            ;interrupcion
27     cmp al,31h          ;compara al con 1
28     je first            ;realiza el salto
29     cmp al,32h          ;compara al con 2
30     je second           ;realiza el salto
31     cmp al,33h          ;compara al con 3
32     je third            ;realiza el salto
33     cmp al,34h          ;compara al con 4
34     je fourth           ;realiza el salto
35     cmp al,35h
36     je fifth
37     jmp L1
38     fifth:
39         mov ah,4ch          ;se solicita la opcion FINALIZAR UN
PROCESO de 21h
40         int 21h            ;se devuelve el control al sistema
41     L1:
42     cmp al,36h
43     jge default
44     cmp al,30h
45     jle default
46
47
48     first:
49         mov ah,00h          ;(00h) modo de video 3
50         mov al,3            ;modo de video 3 (texto a color)
51         int 10h            ;interrupcion de serivioios de video
52                             ;una vez borrada la patalla se
muestra el mensaje
53         mov Ah,09h          ;09h despliega una cadena de
caracteres
54         mov Dx,offset msgCLSScreen ;la cadena v1 entra a Dx
55         int 21h
56         jmp menu
57
58     saltoMenu:
59         jmp menu

```

```

60
61         second:
62             mov Ah,09h           ;09h despliega una cadena de
caracteres
63             mov Dx,offset msgPrint ;la cadena v1 entra a Dx
64             int 21h             ;llamara a la interrupcion 21h con
la funcion 09h
65             jmp menu
66
67         third:
68             MOV AH,00H
69             MOV AL,3
70             INT 10H
71             mov Ah,09h           ;09h despliega una cadena de
caracteres
72             mov Dx,offset msgNDec ;la cadena v1 entra a Dx
73             int 21h             ;llamara a la interrupcion 21h con
la funcion 09h
74             mov ah,01h
75             int 21h
76             cmp al,40h
77             jge third
78             cmp al,30h
79             jle third
80
81         decrementarNumero:
82             dec al
83             mov ah,09h           ;09h despliega una cadena de
caracteres
84             mov dx,offset msgNDec2 ;la cadena v1 entra a Dx
85             int 21h             ;llamara a la interrupcion 21h
con la funcion 09h
86             mov ah,02h
87             mov dl,al
88             int 21h
89             cmp al,30h
90             je menu
91             loop decrementarNumero
92             jmp menu
93
94         fourth:
95             MOV AH,00H
96             MOV AL,3
97             INT 10H
98             mov Ah,09h           ;09h despliega una cadena de
caracteres
99             mov Dx,offset msgNInc ;la cadena v1 entra a Dx
100            int 21h             ;llamara a la interrupcion 21h con
la funcion 09h
101            ;el usuario inserta el numero
102            mov ah,01h
103            int 21h
104            cmp al,39h
105            jge fourth
106            cmp al,2Fh
107            jle fourth
108

```

```

109             incrementarNumero:
110                 inc al
111                 mov ah,09h             ;09h despliega una cadena de
caracteres
112                 mov dx,offset msgNInc2 ;la cadena v1 entra a Dx
113                 int 21h               ;llamara a la interrupcion 21h
con la funcion 09h
114                 mov ah,02h
115                 mov dl,al
116                 int 21h
117                 cmp al,39h
118                 je saltoMenu
119                 loop incrementarNumero
120             jmp menu
121
122
123             default:
124                 mov ah,00h             ;(00h) modo de video 3
125                 mov al,3               ;modo de video 3 (texto a color)
126                 int 10h               ;interrupcion de serivioios de video
127                 ;una vez borrada la patalla se
muestra el mensaje
128                 mov Ah,09h             ;09h despliega una cadena de
caracteres
129                 mov Dx,offset msgDefault ;la cadena v1 entra a Dx
130             jmp menu
131 end

```

```

DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: P2E6
C:\TASM>p2e6.exe

Menu
1.- LIMPIAR PANTALLA
2.- IMPRIMIR UN MENSAJE
3.- DECREMENTAR UN NUMERO INGRESADO
4.- INCREMENTAR UN NUMERO INGRESADO
5.- Salir
inserte opcion:

```

Ilustración 7 Programa 6: se muestra el menú

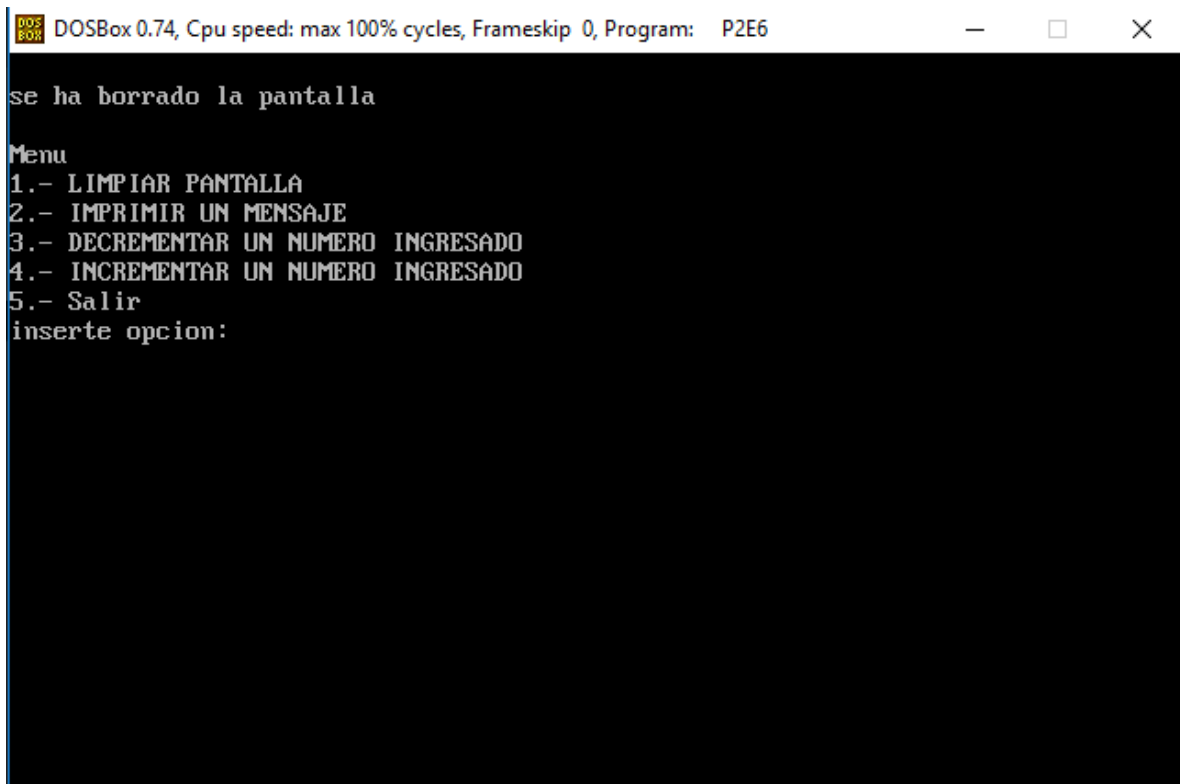


Ilustración 8 Programa 6, opción 1, se limpia la pantalla y se vuelve a mostrar el menú.

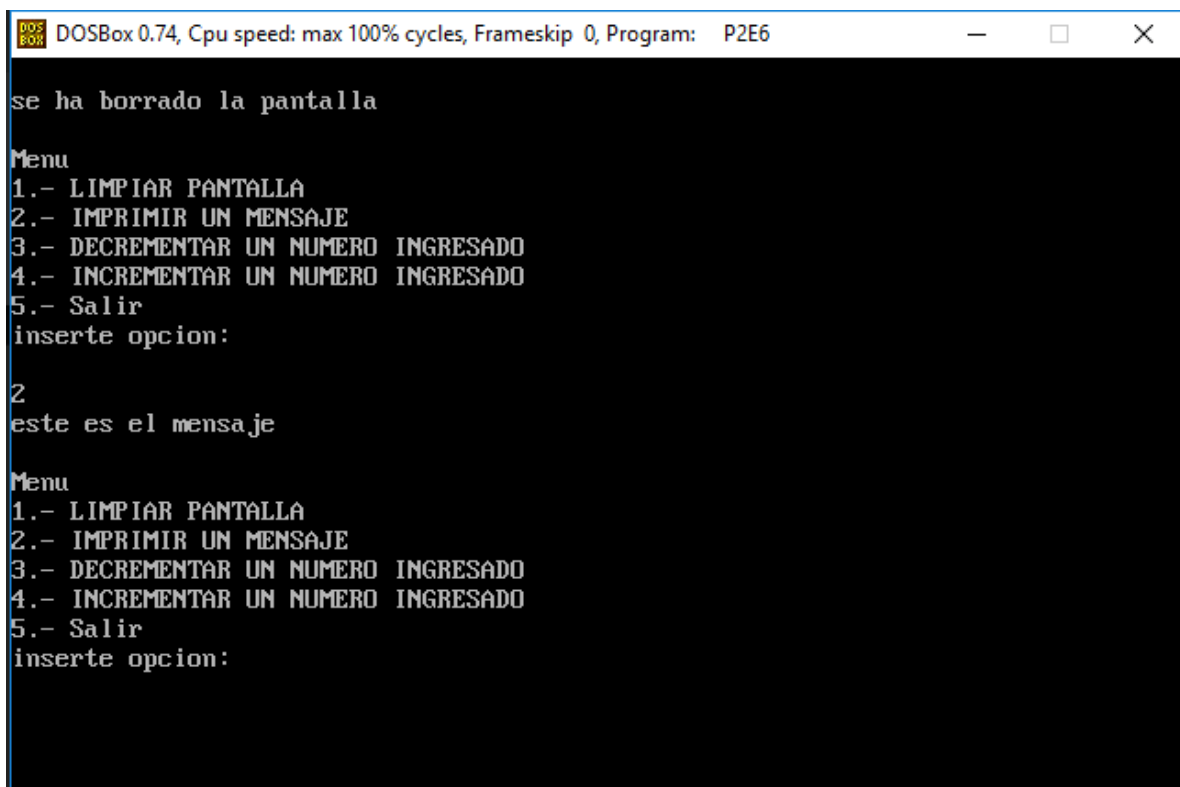
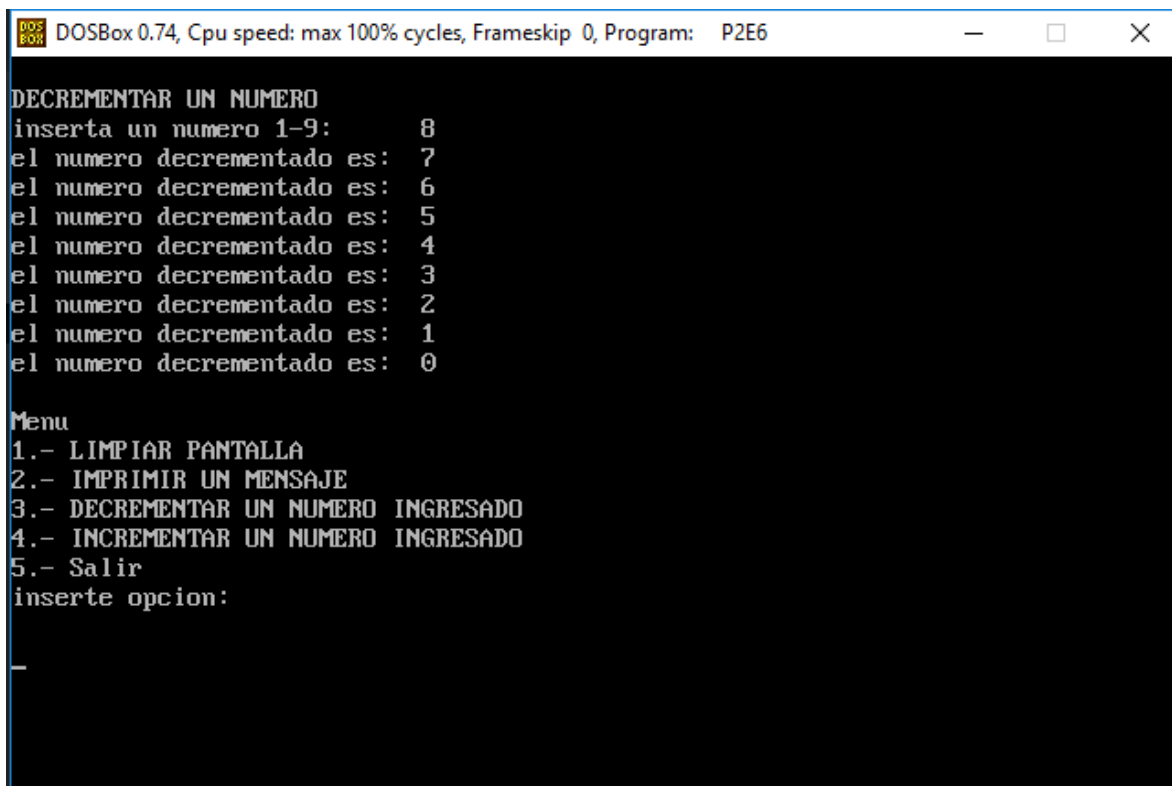


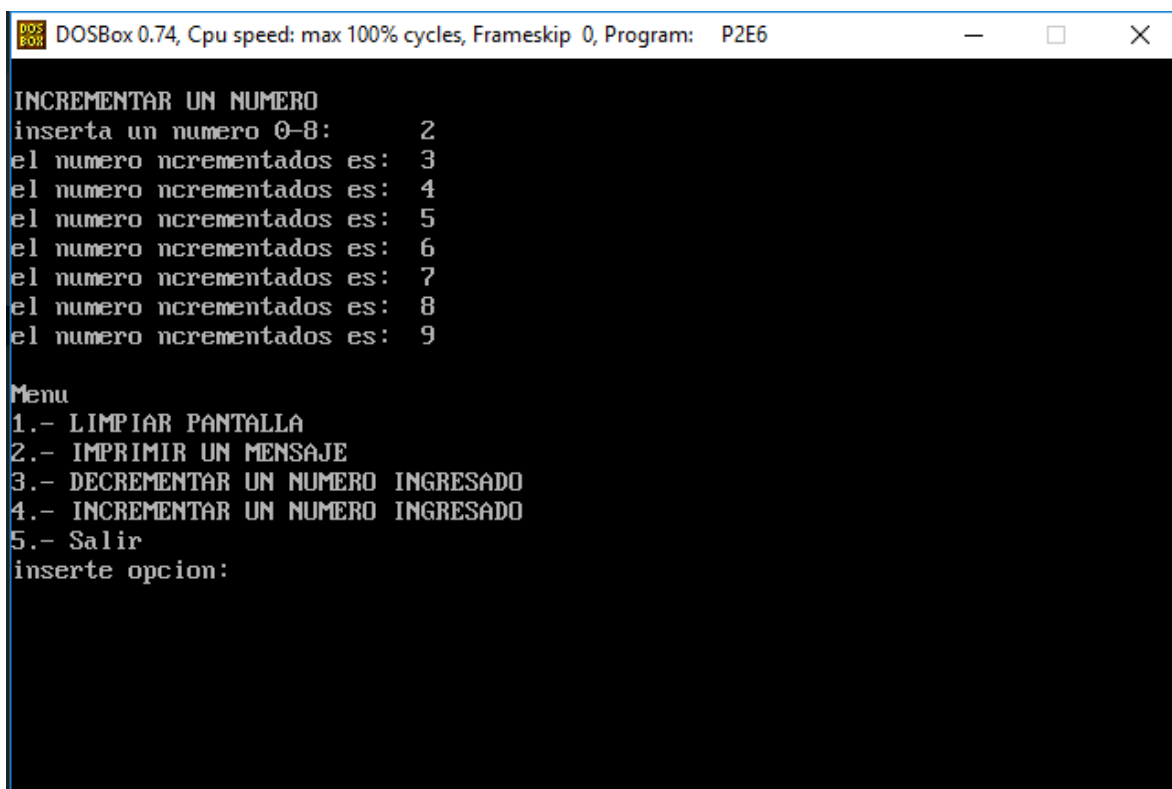
Ilustración 9 Programa 6; el usuario inserta la opción 2, se muestra el mensaje.



```
DOS BOX DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: P2E6
DECREMENTAR UN NUMERO
inserta un numero 1-9:      8
el numero decrementado es:  7
el numero decrementado es:  6
el numero decrementado es:  5
el numero decrementado es:  4
el numero decrementado es:  3
el numero decrementado es:  2
el numero decrementado es:  1
el numero decrementado es:  0

Menu
1.- LIMPIAR PANTALLA
2.- IMPRIMIR UN MENSAJE
3.- DECREMENTAR UN NUMERO INGRESADO
4.- INCREMENTAR UN NUMERO INGRESADO
5.- Salir
inserte opcion:
_
```

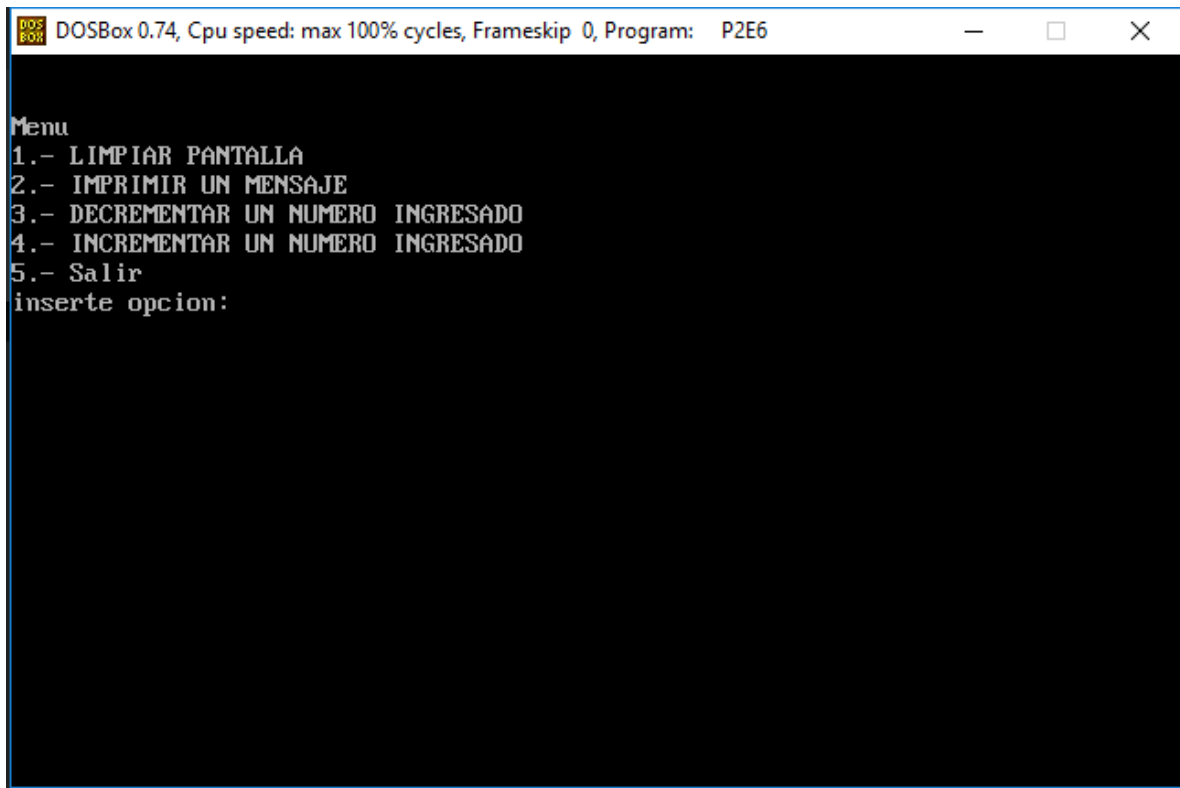
*Ilustración 10 Programa 6: opción 3, el usuario inserta el número 8, después se muestran sus decrementos.*



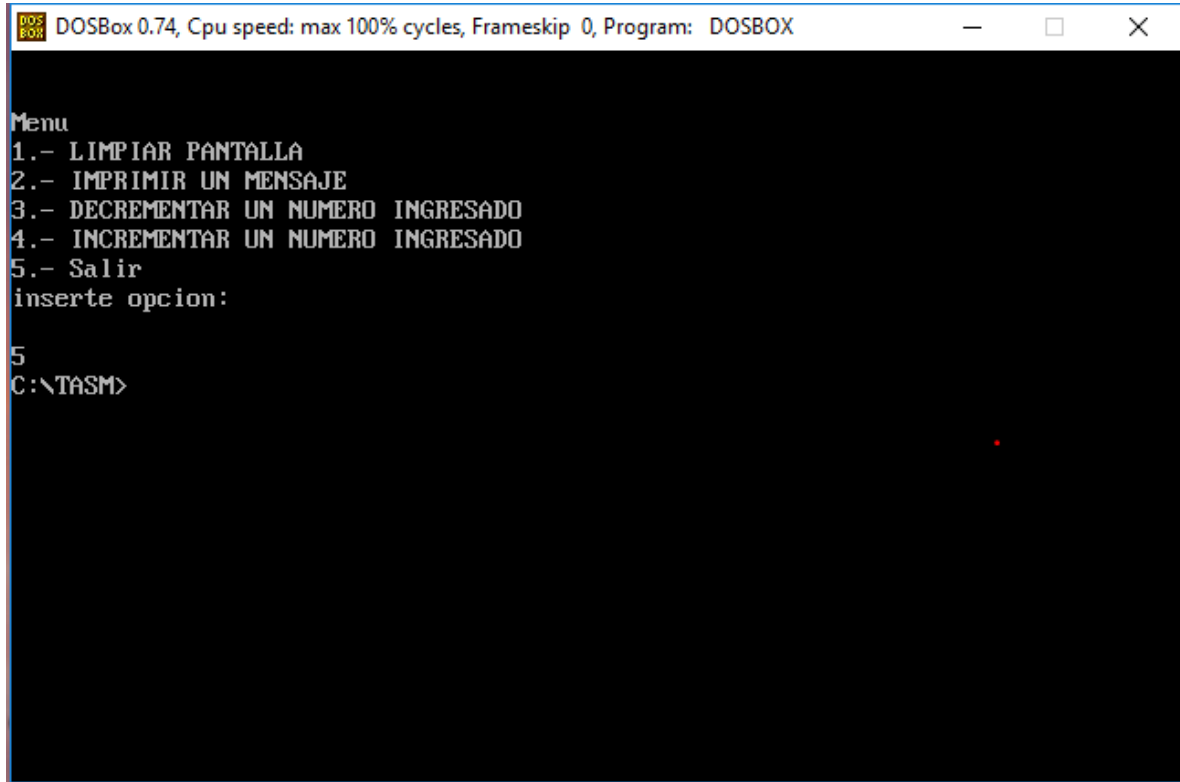
```
DOS BOX DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: P2E6
INCREMENTAR UN NUMERO
inserta un numero 0-8:      2
el numero ncrementados es:  3
el numero ncrementados es:  4
el numero ncrementados es:  5
el numero ncrementados es:  6
el numero ncrementados es:  7
el numero ncrementados es:  8
el numero ncrementados es:  9

Menu
1.- LIMPIAR PANTALLA
2.- IMPRIMIR UN MENSAJE
3.- DECREMENTAR UN NUMERO INGRESADO
4.- INCREMENTAR UN NUMERO INGRESADO
5.- Salir
inserte opcion:
```

*Ilustración 11 Programa 6: opción 4, el usuario inserta el numero 2, se muestran sus incrementos hasta 9*



*Ilustración 12 Programa 6: el usuario inserta cualquier otra tecla, el programa no responde hasta insertar un carácter válido.*



*Ilustración 13 Programa 6: se inserta el número 5, el programa termina.*

Nota<sub>3</sub>: Como evidencia íntegra, el código fuente de cada uno de sus programas correctamente comentado así como una impresión de pantalla de la ejecución del mismo, no olvide colocara pies de imagen debidamente referenciados.

**Conclusiones:** Redacte las conclusiones de su práctica (extensión mínima 20 renglones).

El uso de los ciclos en la programación es indispensable, pues en ocasiones se necesitan ejecutar una instrucción o un bloque de instrucciones un cierto número de veces. Dependiendo de la lógica del programa se utilizan diferentes ciclos, el ciclo loop hace uso del registro CX, como un contador, pues este ciclo va decrementando el valor de CX, hasta llegar a cero. Cuando llega a cero se terminan de ejecutar las instrucciones que se encuentran dentro, pero no solo va decrementando a CX, si no también podemos hacer uso de la parte baja o alta de C, CL o CH, pues el ciclo loop decrementa estos registros. Es muy útil porque cuando debemos repetir el ciclo las veces que el usuario elija, el valor que recibe la función 01h de la interrupción 21h, nos lo devuelve en el registro AL, con esto debemos mover este valor a CX que funciona como contador en decremento, pero la longitud de este registro es de 16 Bits, y AL es de solo 8 Bits, por lo que el programa nos marcaría un error. Es por eso por lo que debemos moverlo a un registro de igual longitud, es decir a CL. Con esto podemos hacer las repeticiones que el usuario desea.

Así como el uso de ciclos, también el uso de arreglos es indispensable, pues en ellos podemos insertar varios datos, con los cuales podemos trabajar, es por eso por lo que se utiliza el operador DUP, con el cual se define un arreglo, el cual puede o no inicializarse los valores. Con esto podemos también manipular los datos del arreglo, como en el programa 2, en el cual se ordenan los datos, se hace uso del algoritmo del método de la burbuja, el cual es un algoritmo muy utilizado para ordenar valores en arreglos.

Para este ordenamiento hacemos uso de ciclos, registros y comparaciones. Las comparaciones se realizan con el operador CMP, al cual se le asignan dos valores, después se les aplica una condición, dependiendo de los datos, es decir podemos compararlos si son iguales, mayores entre si o entre otros valores. Y con esto poder hacer un salto a otra parte del programa. La instrucción CMP es más utilizada en los ciclos JMP, lo cuales no tienen un fin, a menos que se realice un salto, con las instrucciones CMP y los saltos condicionales podemos salir de este ciclo infinito.

Una observación que se tuvo durante el uso de los saltos condicionales es que no podían saltar después de ciertas instrucciones, si realizábamos un salto desde el final del programa hasta el inicio, y las instrucciones que se encontraban entre estos saltos era demasiada, se marcaba un error como el siguiente: *Relative jump out range by 0011h bytes*. Para solucionar este error se tuvieron que mover los saltos a otra parte del código mas cerca, en ocasiones eliminar instrucciones o cambiar las condiciones.

Se realizó el uso de variables DB inicializadas en 0, las cuales después se le asignaba un valor, para respaldarlo o poder acceder a él después, ya que los valores de los registros cambiaban de acuerdo con las funciones, el principal valor que se guardaba en estas variables, son los valores de los registros AL, los cuales son los valores del teclado. Pues no siempre se podía tener guardado un valor en los registros.

El uso de incrementos resulto complejo cuando los incrementos pasaban de dos dígitos, pues al imprimirse estos valores se encontraba un error, ya que la función que se utilizaba para imprimir imprime caracteres ASCII, los cuales solo son de un dígito, para ello teníamos que utilizar el registro de 16 bits, para imprimir la parte alta y parte baja de este registro.

Con el uso del teclado, los valores que se recibían se encontraban en carácter ASCII, por lo que había que restar o sumar a los valores para que, tuvieran un valor real, en el caso de los números, se les restaba 30h, pues al insertar un número se insertan después de los 30h en carácter ASCII, ajustar las operaciones que se realizaban. Para los programas realizados se insertan teclas numéricas, en caso de que el usuario insertara una tecla con otro carácter, el programa arrojaba otro resultado, pues referenciaba al valor ASCII de la tecla, esto se restringía realizando comparaciones entre el valor de AL y el valor ASCII de los números, como el valor de los números del 0 al 9 se encontraba entre el 30h y el 39h, se realizaban comparaciones para ver si se encontraban entre este rango, si no se tendría que volver a insertar un valor. El uso de los colores para los caracteres el cursor no se desplazaba, se tenía que usar otra función, con la cual, se debía recorrer el cursor en fila y columna.

Nota final: La copia total o parcial de alguna práctica previamente entregada no será recibida y afectará la calificación otorgada a la persona que entrego previamente. Sea cuidadoso con sus reportes y productos de las actividades realizadas, evite pasarlos a sus compañeros. Si se basan en algún ejemplo de libros o de internet es preciso referenciarlo y hacer énfasis en las mejoras que usted genere a dicho trabajo.

## **Bibliografía**

- [1] K. R. IRVINE, LENGUAJE ENSAMBLADOR PARA COMPUTADORAS BASADAS EN INTEL, Quinta Edición ed., México: Prentice Hall, 2007.
- [2] P. Abel, LENGUAJE ENSAMBLADOR Y PROGRAMACION PARA IBM PC Y COMPATIBLES, Mexico: PEARSON Educacion, 1996.
- [3] B. B. Brey, MICROPROCESADORES INTEL, arquitectura, programación e interfaz, Séptima Edición ed., PEARSON Prentice Hall.