

5

NOMBRE DE LA PRÁCTICA	Ciclo Loop			No.	UNIDAD 2
ASIGNATUR A:	LENGUAJE INTERFAZ	CARR ERA:	ISIC	PLAN:	ISIC-2010-204

Nombre: Vanesa Hernández Martínez

Grupo: 3501

Objetivo: Realizar un programa utilizando la instrucción LOOP

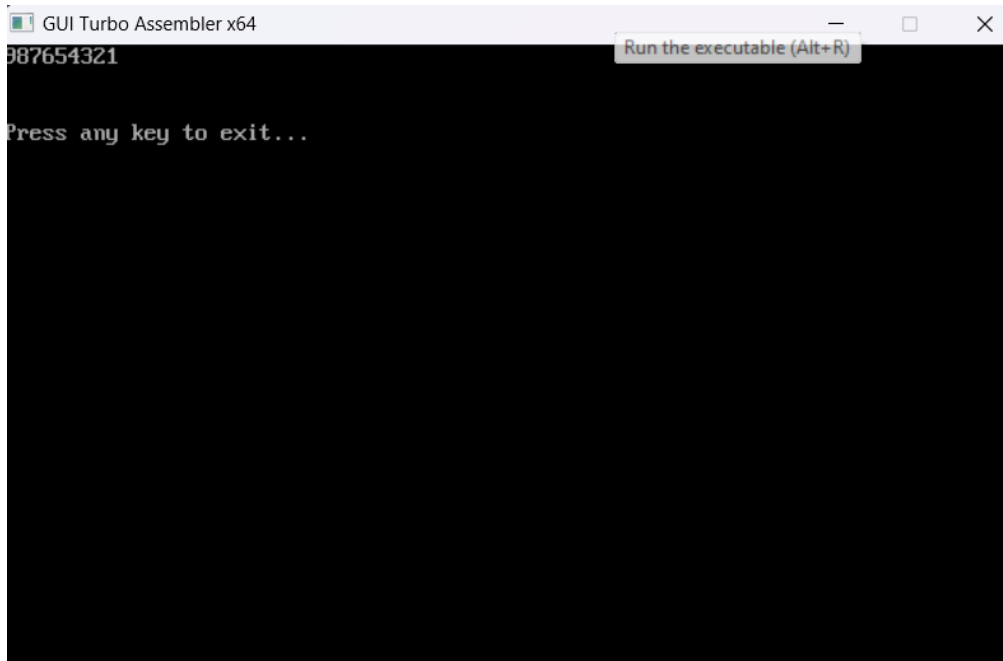
1. Realiza un programa en ensamblador que realice un conteo descendente de 9 a 0.
Anexa la captura de pantalla del código y la corrida del programa:

```

1  .model small ; se asigna el tamaño de memoria
2  .stack ; modelo de pila
3  .data ; inicio de datos
4
5  .code; inicio del código
6  main proc; inicia procedimiento principal
7  mov cx, 9
8  ciclo:
9
10 mov ah, 02
11 mov dx, cx
12 add dx, 30h
13 int 21h
14
15 loop ciclo
16
17 mov ax, 4c00h; salir del programa
18 int 21h
19 main endp; termina el procedimiento
20 end main
21

```

5



2. Ahora realiza un programa ascendente de 1 a 9:

Anexa captura de pantalla de código y corrida de programa:

```

1  .model small ; se asigna el tamaño de memoria
2  .stack ; modelo de pila
3  .data ; inicio de datos
4
5  .code; inicio del código
6  main proc; inicia procedimiento principal
7      mov cx,9 ; Inicializa CX con el número de iteraciones (9)
8      mov dl,1 ; Inicializa DL con el valor 1 (primer número a imprimir)
9
10 ciclo:
11
12     mov ah,02h ; Función para imprimir un carácter
13     add dl,30h ; Convierte el valor numérico a su representación ASCII
14     int 21h ; Llama a la interrupción para imprimir en pantalla
15     sub dl,30h ; Regresa el valor de DL a su estado numérico original
16     inc dl ; Incrementa DL para que en la próxima iteración imprima el siguiente número
17
18     loop ciclo ; Decrementa CX y repite el ciclo si CX no es 0
19
20     mov ax,4c00h; Salir del programa
21     int 21h
22 main endp; termina el procedimiento
23 end main
24

```

```
GUI Turbo Assembler x64
123456789
Press any key to exit...
```

- Ahora realiza los dos programas anteriores para números con dos dígitos, ascendente de 1 a 99 y descendente de 99 a 1.

Ascendente

```

1  ; Programa: Conteo Ascendente de 1 a 99
2  ; Ensamblador: TASM/MASM
3  ; Entorno: DOS o DOSBox
4
5  .model small
6  .stack 100h
7
8  .data
9      mensaje db 'Conteo ascendente:', 0Dh, 0Ah, '$'
10     numero db '00', 0Dh, 0Ah, '$' ; Variable para almacenar el número a imprimir
11
12 .code
13 main:
14     mov ax, @data          ; Inicializar segmento de datos
15     mov ds, ax
16
17     ; Mostrar el mensaje inicial
18     mov ah, 09h            ; Función 09h de DOS para imprimir cadena
19     lea dx, mensaje        ; Cargar la dirección del mensaje en DX
20     int 21h                ; Llamar a la interrupción de DOS
21
22     ; Configurar el contador
23     mov cx, 99             ; Número de iteraciones (1 a 99)
24     mov bx, 1              ; Valor inicial del contador (1)
25     mov si, 10             ; Divisor para obtener decenas y unidades
26
27 loop_conteo_ascendente:
28     ; Dividir BX entre 10 para obtener decenas y unidades
29     mov ax, bx              ; Mover el valor de BX a AX
30     xor dx, dx              ; Limpiar DX antes de la división
31     div si                  ; AX / 10 => AL = decenas, DL = unidades
32
33     ; Convertir las decenas a ASCII si es necesario
34     cmp al, 0               ; Verificar si hay decenas
35     je sin_decenas_ascendente
36     add al, '0'             ; Convertir decenas a carácter ASCII
37     mov [numero], al        ; Almacenar el carácter de decenas en 'numero'
38     jmp convertir_unidades_ascendente
39
40 sin_decenas_ascendente:
41     mov byte ptr [numero], ' ' ; Si no hay decenas, poner un espacio

```

```

40 sin_decenas_ascendente:
41     mov byte ptr [numero], ' ' ; Si no hay decenas, poner un espacio
42
43 convertir_unidades_ascendente:
44     add dl, '0' ; Convertir unidades a car?cter ASCII
45     mov [numero+1], dl ; Almacenar el car?cter de unidades en 'numero'
46
47     ; Mostrar el n?mero
48     mov ah, 09h ; Funci?n 09h de DOS para imprimir cadena
49     lea dx, numero ; Cargar la direcci?n de 'numero' en DX
50     int 21h ; Llamada a la interrupci?n de DOS
51
52     ; Incrementar el contador
53     inc bx ; Incrementar BX para el siguiente n?mero
54
55     ; Decrementar CX y repetir si no es cero
56     loop loop_conteo_ascendente
57
58     ; Finalizar el programa
59     mov ah, 4Ch ; Funci?n 4Ch de DOS para terminar el programa
60     int 21h ; Llamada a la interrupci?n de DOS
61
62 end main
63

```

GUI Turbo Assembler x64

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

Run the executable (Alt+R)

Press any key to exit...

Descendente

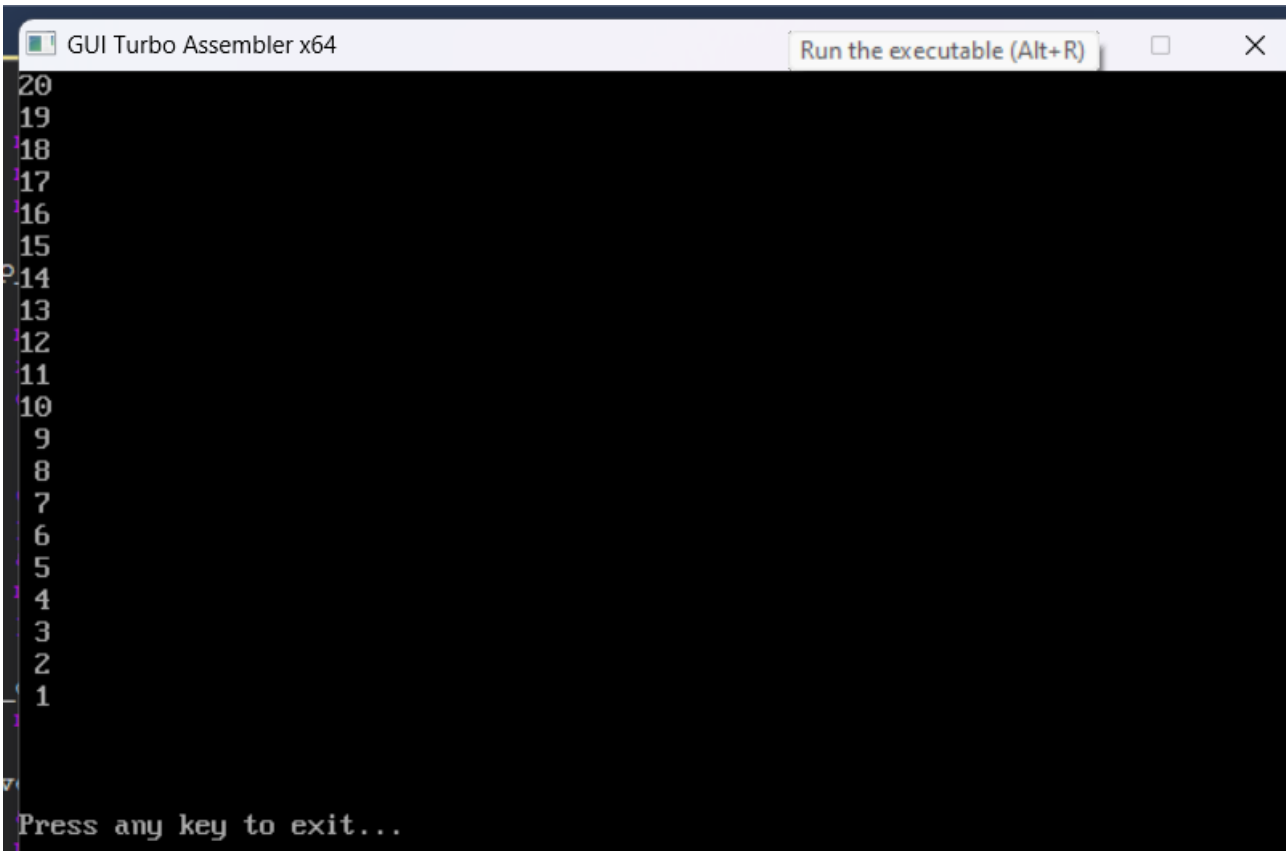
```
1 ; Programa: Conteo Descendente de 99 a 1
2 ; Ensamblador: TASM/MASM
3 ; Entorno: DOS o DOSBox
4
5 .model small
6 .stack 100h
7
8 .data
9     mensaje db 'Conteo descendente:', 0Dh, 0Ah, '$'
10    numero db '00', 0Dh, 0Ah, '$' ; Variable para almacenar el número a imprimir
11
12 .code
13 main:
14     mov ax, @data ; Inicializar segmento de datos
15     mov ds, ax
16
17     ; Mostrar el mensaje inicial
18     mov ah, 09h ; Función 09h de DOS para imprimir cadena
19     lea dx, mensaje ; Cargar la dirección del mensaje en DX
20     int 21h ; Llamada a la interrupción de DOS
21
22     ; Configurar el contador
23     mov cx, 99 ; Número de iteraciones (99 a 1)
24     mov bx, 99 ; Valor inicial del contador (99)
25     mov si, 10 ; Divisor para obtener decenas y unidades
26
27 loop_conteo_descendente:
28     ; Dividir BX entre 10 para obtener decenas y unidades
29     mov ax, bx ; Mover el valor de BX a AX
30     xor dx, dx ; Limpiar DX antes de la división
31     div si ; AX / 10 => AL = decenas, DL = unidades
32
33     ; Convertir las decenas a ASCII si es necesario
34     cmp al, 0 ; Verificar si hay decenas
35     je sin_decenas_descendente
36     add al, '0' ; Convertir decenas a carácter ASCII
37     mov [numero], al ; Almacenar el carácter de decenas en 'numero'
38     jmp convertir_unidades_descendente
39
40 sin_decenas_descendente:
41     mov byte ptr [numero], ' ' ; Si no hay decenas, poner un espacio
42
```



```

40 sin_decenas_descendente:
41     mov byte ptr [numero], ' ' ; Si no hay decenas, poner un espacio
42
43 convertir_unidades_descendente:
44     add dl, '0' ; Convertir unidades a car?cter ASCII
45     mov [numero+1], dl ; Almacenar el car?cter de unidades en 'numero'
46
47     ; Mostrar el n?mero
48     mov ah, 09h ; Funci?n 09h de DOS para imprimir cadena
49     lea dx, numero ; Cargar la direcci?n de 'numero' en DX
50     int 21h ; Llamada a la interrupci?n de DOS
51
52     ; Decrementar el contador
53     dec bx ; Decrementar BX para el siguiente n?mero
54
55     ; Decrementar CX y repetir si no es cero
56     loop loop_conteo_descendente
57
58     ; Finalizar el programa
59     mov ah, 4Ch ; Funci?n 4Ch de DOS para terminar el programa
60     int 21h ; Llamada a la interrupci?n de DOS
61
62 end main
63

```



The screenshot shows the GUI Turbo Assembler x64 window. The main window displays the assembly code from the previous block, with line numbers 1 through 20 visible on the left. The output window at the bottom shows the text "Press any key to exit..." and a cursor.

CONCLUSIONES

Este programa es un claro ejemplo de cómo el lenguaje ensamblador permite un control detallado y preciso sobre la ejecución de bucles, en este caso utilizando el ciclo LOOP para realizar un conteo tanto descendente como ascendente y nuevamente descendente. El uso del ciclo LOOP junto con registros permite realizar estas operaciones de manera directa y eficiente, sin la sobrecarga de estructuras de control complejas como las que se encuentran en lenguajes de alto nivel.

A través de esta implementación, se demuestra cómo ensamblador optimiza el manejo de ciclos repetitivos, minimizando el consumo de recursos y proporcionando una gran precisión en el flujo del programa. Además, el control explícito de las condiciones de fin de ciclo resalta la importancia de la correcta manipulación de los registros para garantizar un conteo exacto y evitar errores. En un contexto más amplio, este tipo de ejercicios refuerza la comprensión de la arquitectura interna del procesador y la interacción directa con los componentes del hardware, lo que hace que el programador tenga un control total sobre cada operación ejecutada.