
 GOBIERNO DEL ESTADO DE MÉXICO	<b>PRÁCTICA 7</b> Ing. y Esp. Rodolfo Guadalupe Alcántara Rosales	 <b>TESI</b> TECNOLÓGICO DE ESTUDIOS SUPERIORES ILOILOTEPEC
---	--	---

5

NOMBRE DE LA PRÁCTICA	Manipulación de cadenas			No.	UNIDAD 2
ASIGNATURA:	LENGUAJE INTERFAZ	CARRERA:	ISIC	PLAN:	ISIC-2010-204

Nombre: Jesús Navarrete Martínez  
Grupo: 3501



Objetivo: Realizar un programa que manipule cadena de caracteres.

1. Escribe el siguiente programa:

```
.model small
.stack 64
.data
msg db "mensaje$"
.code
inicio:
mov ax,@data
mov ds,ax
mov ah,09h
mov dx, offset msg
int 21h
mov ah,02
mov dl,0ah
int 21h

mov ah,02
mov dl,0dh
int 21h

mov SI,0
ciclo:
mov dl,msg[SI]
cmp DL, 24h
je fuera_ciclo
```

 GOBIERNO DEL ESTADO DE MÉXICO	<h2>PRÁCTICA 7</h2> <p>Ing. y Esp. Rodolfo Guadalupe Alcántara Rosales</p>	 TESI TECNOLÓGICO DE ESTUDIOS SUPERIORES IXILTEPEC
---	--	--

5

```
mov ah,02
mov dl,dl
int 21h
```

```
mov ah,02
mov dl,0ah
int 21h
```

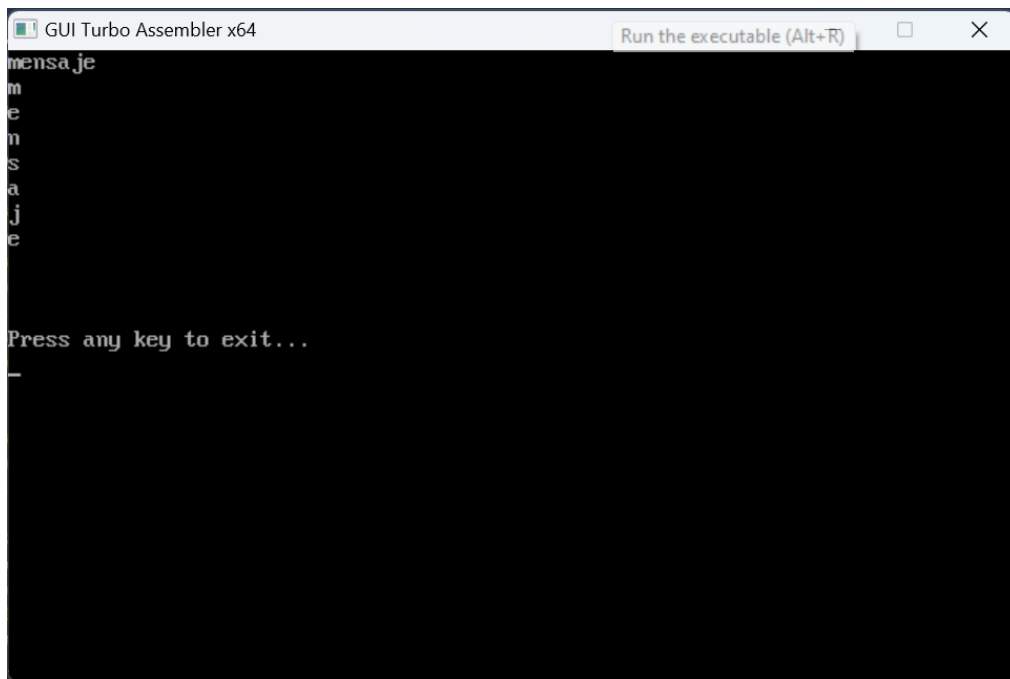
```
inc SI
jmp ciclo
fuera_ciclo:
```

```
mov ax,4c00h
int 21h
```

```
end inicio
```

### Indica que hace el programa:

1. **Introducción:** "Este programa en ensamblador para DOS muestra un mensaje completo en pantalla y luego imprime cada carácter del mensaje en una línea nueva."
2. **Configuración inicial:** "Primero, definimos el modelo de memoria y el tamaño de la pila. Luego, declaramos la cadena msg con el texto "mensaje\$", donde \$ indica el final de la cadena."
3. **Segmento de datos y visualización:** "Configuramos el segmento de datos para acceder correctamente a msg. Usamos la interrupción 21h con la función 09h para mostrar msg en pantalla hasta el \$."
4. **Salto de línea:** "Agregamos un salto de línea con 0Ah (nueva línea) y 0Dh (retorno de carro), así el cursor pasa a la siguiente línea."
5. **Bucle para imprimir cada carácter:** "Usamos un bucle que recorre msg carácter por carácter. Si encuentra \$, sale del bucle; si no, imprime el carácter y agrega un salto de línea. El bucle sigue hasta completar cada letra."
6. **Finalización:** "Finalmente, el programa termina y devuelve el control al sistema operativo."



- Realiza un programa en ensamblador que invierta el mensaje en forma vertical  
Anexa la captura de pantalla del código y la corrida del programa:

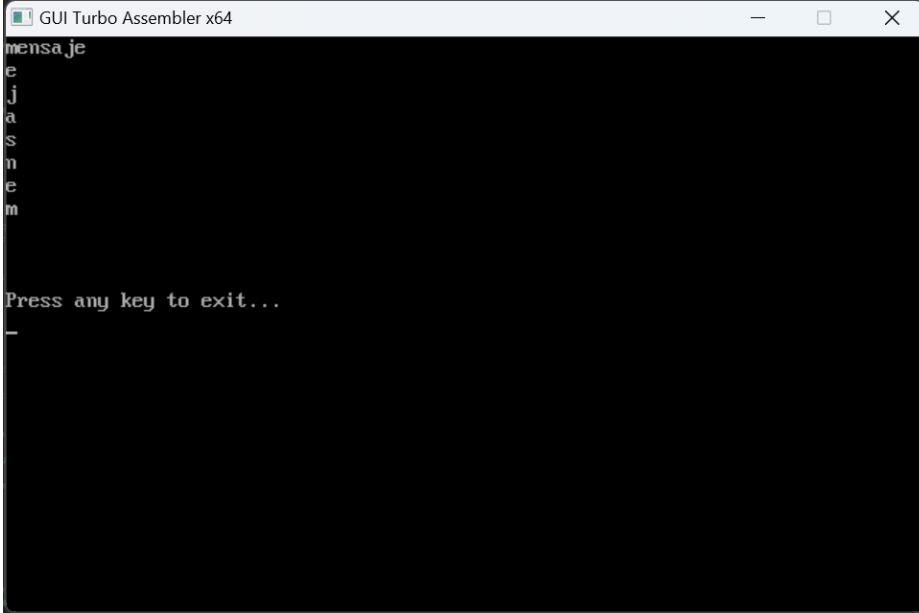
### Código del programa:

```

1  .model small
2  .stack 64
3  .data
4  msg db "mensaje$" ; Cadena original con terminaci?n de '$'
5  .code
6  inicio:
7      mov ax, @data
8      mov ds, ax
9
10     ; Imprimir el mensaje original
11     mov ah, 09
12     mov dx, offset msg
13     int 21h
14
15     ; Salto de 1?nas despu?s del mensaje original
16     mov ah, 02
17     mov dl, 0Ah
18     int 21h
19
20     mov ah, 02
21     mov dl, 0Dh
22     int 21h
23
24     ; Encuentra la longitud de la cadena
25     mov cx, 0 ; Usaremos CX para contar la longitud
26     mov si, offset msg
27 cuenta:
28     cmp byte ptr [si], '$' ; Verifica el final de la cadena
29     je imprime_al_reves
30     inc si
31     inc cx
32     jmp cuenta
33
34 imprime_al_reves:
35     dec si ; Retrocede a la ?ltima letra (antes del '$')
36 ciclo_inverso:
37     cmp cx, 0 ; Verifica si ya imprimi? todos los caracteres
38     je fin
39
40     mov dl, [si] ; Mueve el car?cter actual a DL
41     mov ah, 02
42     int 21h ; Imprime el car?cter
43
44     ; Salto de 1?nas despu?s de cada letra
45     mov ah, 02
46     mov dl, 0Ah
47     int 21h
48     mov ah, 02
49     mov dl, 0Dh
50     int 21h
51
52     dec si ; Mueve el puntero a la izquierda
53     dec cx ; Decrementa la longitud
54     jmp ciclo_inverso
55
56 fin:
57     ; Terminar el programa
58     mov ax, 4c00h
59     int 21h
60
61 end inicio
62

```

### Ejecución del programa:



```
GUI Turbo Assembler x64
mensaje
e
j
a
s
n
e
m

Press any key to exit...
```

### Conclusiones:

En estos programas en ensamblador para DOS, hemos explorado cómo manipular y mostrar cadenas en pantalla utilizando interrupciones y estructuras de bucle.

1. **Primer programa:** El primer programa muestra el mensaje completo y luego imprime cada uno de sus caracteres en una línea nueva. Esto se logró mediante un bucle que recorre cada carácter de la cadena y lo muestra individualmente. Este ejercicio permitió entender cómo recorrer una cadena y aplicar interrupciones para formatear la salida en pantalla.
2. **Segundo programa:** El segundo programa, además de mostrar la cadena original, cuenta sus caracteres y los imprime en orden inverso, cada uno en una nueva línea. Aquí, utilizamos un bucle adicional para determinar la longitud de la cadena y luego la recorremos hacia atrás. Esto muestra cómo podemos trabajar con índices y manipular el flujo de datos en la memoria, imprimiendo caracteres en un orden específico.

Ambos programas demostraron el uso eficiente de instrucciones de ensamblador como `mov`, `cmp`, `int`, y estructuras de control como bucles. Estos conceptos son fundamentales en el desarrollo en ensamblador, especialmente para operaciones de manejo de cadenas y salida en pantalla.