

5

NOMBRE DE LA PRÁCTICA	ESCRITURA DE UNA CADENA DE CARACTERES			No.	UNIDAD 1
ASIGNATURA:	LENGUAJE INTERFAZ	CARRERA:	ISIC	PLAN:	ISIC-2010-204

Nombre: Vanesa Hernández Martínez

Grupo: 3501

Objetivo: Desplegar una cadena de caracteres en lenguaje ensamblador, utilizando los registros de datos

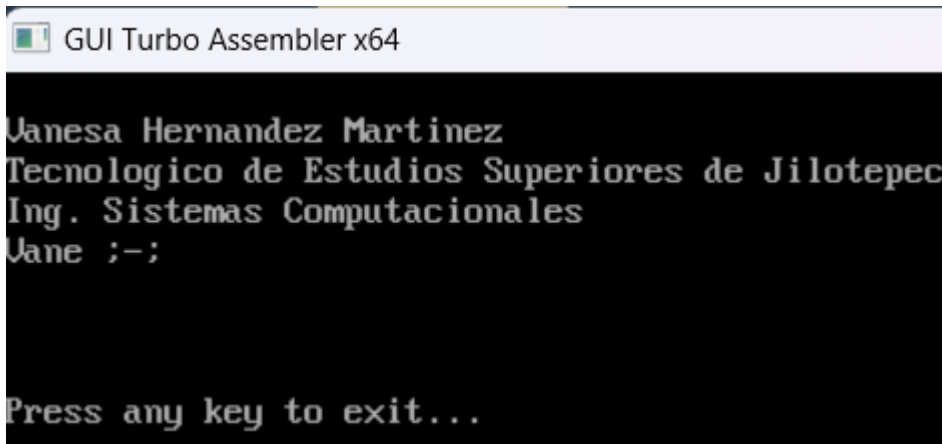
Utilizando los registros acumuladores y de datos, elabora un programa en ensamblador que permita desplegar una cadena de caracteres, utilizando las instrucciones respectivas para ubicar el inicio de texto. Indica lo que realiza cada renglón.

```

1 CR EQU 13      ; Definición del carácter de retorno de carro (Carriage Return)
2 LF EQU 10      ; Definición del carácter de salto de línea (Line Feed)
3
4 ■ DATOS SEGMENT
5   ; Cadenas de texto que se mostrarán en pantalla, cada línea finaliza con un '$'
6   LINEA1 DB CR, LF, 'Vanesa Hernández Martínez', CR, LF, '$' ; Primera línea a imprimir el nombre
7   LINEA2 DB 'Tecnológico de Estudios Superiores de Jilotepec', CR, LF, '$' ; Segunda línea para imprimir el nombre de la escuela
8   LINEA3 DB 'Ing. Sistemas Computacionales', CR, LF, '$' ; Tercera línea para imprimir la carrera
9   LINEA4 DB 'Vane ;-', CR, LF, '$' ; Cuarta línea para imprimir el apodo
10  ■ DATOS ENDS
11
12 ■ PILA SEGMENT STACK
13   DB 64 DUP('PILA') ; Reserva espacio para el stack (64 bytes)
14  ■ PILA ENDS
15
16 ■ CODIGO SEGMENT
17   ; Inicio del procedimiento principal
18   LN PROC FAR
19   ASSUME CS:CODIGO, DS:DATOS, SS:PILA ; Define los segmentos de código, datos y stack
20
21   ; Inicializa el segmento de datos
22   MOV AX, DATOS ; Carga la dirección del segmento de datos en AX
23   MOV DS, AX    ; Mueve la dirección del segmento de datos a DS
24
25   ; Mostrar cada línea de texto en la pantalla
26   LEA DX, LINEA1 ; Carga la dirección de LINEA1 en DX (primera línea)
27   CALL ESCRIBE   ; Llama a la subrutina ESCRIBE para mostrar la cadena
28
29   LEA DX, LINEA2 ; Carga la dirección de LINEA2 en DX (segunda línea)
30   CALL ESCRIBE   ; Llama a la subrutina ESCRIBE para mostrar la cadena
31
32   LEA DX, LINEA3 ; Carga la dirección de LINEA3 en DX (tercera línea)
33   CALL ESCRIBE   ; Llama a la subrutina ESCRIBE para mostrar la cadena
34
35   LEA DX, LINEA4 ; Carga la dirección de LINEA4 en DX (cuarta línea)
36   CALL ESCRIBE   ; Llama a la subrutina ESCRIBE para mostrar la cadena
37
38   ; Finaliza el programa
39   MOV AX, 4C00H ; Carga el código de finalización del programa en AX
40   INT 21H      ; Llama a la interrupción 21h para finalizar la ejecución del programa
41
42   LN ENDP      ; Fin del procedimiento principal
43
44   ; Subrutina para escribir una cadena en pantalla
45   ESCRIBE PROC
46   MOV AH, 9    ; Función 9 de la interrupción 21h: Imprimir cadena de texto
47   INT 21H      ; Llama a la interrupción 21h para mostrar la cadena en pantalla
48   RET         ; Regresa de la subrutina ESCRIBE
49   ESCRIBE ENDP ; Fin de la subrutina ESCRIBE
50
51 CODIGO ENDS    ; Fin del segmento de código
52 END LN        ; Indica el final del código y especifica el punto de entrada del programa
53

```

Escribe las instrucciones y captura de pantalla que demuestre que el programa si corrió



```
GUI Turbo Assembler x64

Janesa Hernandez Martinez
Tecnologico de Estudios Superiores de Jilotepec
Ing. Sistemas Computacionales
Jane ;-;

Press any key to exit...
```

CUESTIONARIO:

¿Qué dificultades encuentras al programar en ensamblador?

Para mí, programar en ensamblador es complicado principalmente porque es mucho más detallado que los lenguajes de alto nivel. Tengo que preocuparme por cada pequeño paso, como mover datos entre registros y la memoria de forma manual. Eso hace que cometa errores fácilmente, y me toma más tiempo depurar. Además, no tengo las facilidades que ofrecen otros lenguajes, como los bucles o condicionales claros. Todo esto hace que el código sea más difícil de leer y mantener a largo plazo. También el hecho de que cada procesador tenga su propio conjunto de instrucciones limita mucho la portabilidad del código.

¿Es importante el uso de ensamblador en la programación?

Yo creo que, aunque no es tan común como antes, el ensamblador sigue siendo importante en ciertos casos. Es clave cuando se trabaja a nivel muy bajo, por ejemplo, en la creación de sistemas operativos o controladores de dispositivos, donde se necesita un control absoluto sobre el hardware. Aunque normalmente uso lenguajes de alto nivel, entender ensamblador me permite optimizar el rendimiento en situaciones críticas y tener un conocimiento más profundo de cómo funciona el hardware.

¿Un lenguaje de alto nivel no requiere ensamblador?

Para ser honesto, en la mayoría de los casos no es necesario usar ensamblador cuando trabajo con lenguajes de alto nivel. Estos lenguajes están diseñados para abstraer todos esos detalles complicados, lo que facilita mucho el desarrollo. Sin embargo, creo que tener una base de ensamblador es útil para ciertos casos, como cuando quiero optimizar un programa al máximo o depurar errores complejos a nivel bajo. Además, me ayuda a entender mejor cómo funcionan internamente los lenguajes y los compiladores, lo cual me puede dar ventaja en problemas avanzados o sistemas más críticos.

Conclusión

En conclusión, aunque el lenguaje ensamblador puede parecer complicado y laborioso en comparación con los lenguajes de alto nivel, su importancia no debe ser subestimada. Programar en ensamblador me ha permitido conocer cómo funciona el hardware y cómo se gestionan los recursos a nivel más básico. Si bien no es necesario para la mayoría de los proyectos actuales, sigue siendo una herramienta invaluable en áreas donde el control sobre el rendimiento y el hardware es esencial. Además, entenderlo me ha dado una visión más profunda de la programación en general, lo que considero una ventaja para enfrentar problemas más complejos. Aunque no se use todos los días, es una habilidad que complementa y enriquece mi perfil como desarrollador.