PRÁCTICA 2

Ing. y Esp. Rodolfo Guadalupe Alcántara Rosales

GOBIERNO DEL ESTADO DE MÉXICO

TECNOLÓGICO DE ESTUDIOS SUPERIORES JILOTEPEC

5

NOMBRE DE LA PRÁCTICA	ESCRITURA DE DOS NÚMEROS DE DOS DÍGITOS			No.	UNIDAD 1
ASIGNATURA:	LENGUAJE INTERFAZ	CARRERA:	ISIC	PLAN:	ISIC-2010-204

Nombre: Vanesa Hernández Martínez

Grupo:3501

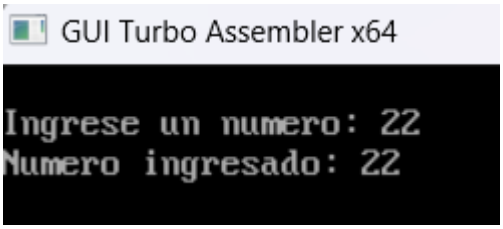
Objetivo: Desplegar números que contengan dos dígitos

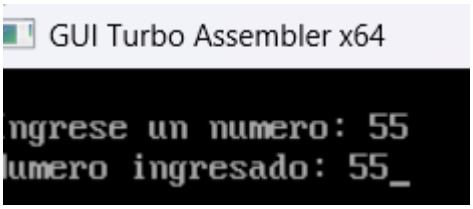
Utilizando los registros acumuladores y de datos, elabora un programa en ensamblador que permita desplegar dos números de dos dígitos cada uno. Indica lo que realiza cada renglón.

```
.model small
.stack
.data
; Declaración de variables donde se almacenarán los dígitos y el número final
u db 0        ; Unidad del número ingresado
d db 0        ; Decena del número ingresado
n db 0        ; Número final calculado a partir de los dígitos ingresados

; Mensajes a mostrar en pantalla
mensaje db 10,13,7, "Ingrese un numero: ", "$"    ; Mensaje de solicitud de entrada de número
mensaje2 db 10,13,7, "Numero ingresado: ", "$"    ; Mensaje para mostrar el número calculado

.code
main proc
    ; Inicializa el segmento de datos
    mov ax, SEG @data
    mov ds, ax

    ; Mostrar mensaje para ingresar un número
    mov ah, 09h             ; Función DOS para mostrar cadenas de texto
    lea dx, mensaje         ; Cargar la dirección del mensaje en DX
    int 21h                 ; Llamar a la interrupción 21h para mostrar el mensaje

    ; Leer el primer dígito (decena) del número ingresado
    mov ah, 01h             ; Función DOS para leer un carácter del teclado
    int 21h                 ; Llamar a la interrupción 21h para leer el carácter
    sub al, 30h             ; Convertir el carácter ASCII a número (0-9)
    mov d, al               ; Guardar el primer dígito en la variable 'd'

    ; Leer el segundo dígito (unidad) del número ingresado
    mov ah, 01h             ; Función DOS para leer un carácter del teclado
    int 21h                 ; Llamar a la interrupción 21h para leer el carácter
    sub al, 30h             ; Convertir el carácter ASCII a número (0-9)
    mov u, al               ; Guardar el segundo dígito en la variable 'u'

    ; Calcular el número completo (decenas y unidades)
    mov al, d               ; Mover el valor de la decena a AL
    mov bl, 10              ; Cargar 10 en BL para multiplicar por la base decimal
    mul bl                  ; Multiplicar decena por 10
    add al, u               ; Sumar el valor de la unidad
    mov n, al               ; Guardar el número completo en 'n'
```

```
    ; Calcular el número completo (decenas y unidades)
    mov al, d               ; Mover el valor de la decena a AL
    mov bl, 10              ; Cargar 10 en BL para multiplicar por la base decimal
    mul bl                  ; Multiplicar decena por 10
    add al, u               ; Sumar el valor de la unidad
    mov n, al               ; Guardar el número completo en 'n'

    ; Mostrar mensaje indicando el número ingresado
    mov ah, 09h             ; Función DOS para mostrar cadenas de texto
    lea dx, mensaje2        ; Cargar la dirección del mensaje en DX
    int 21h                 ; Llamar a la interrupción 21h para mostrar el mensaje

    ; Preparar para mostrar el número (convertir a caracteres ASCII)
    mov al, n               ; Cargar el número completo en AL
    AAM                     ; Ajuste para división de AL por 10: AH = decenas, AL = unidades
    mov bx, ax              ; Guardar las decenas y unidades en BX

    ; Mostrar el dígito de las decenas
    mov ah, 02h             ; Función DOS para mostrar un carácter
    mov dl, bh              ; Cargar las decenas (BH) en DL
    add dl, 30h             ; Convertir las decenas a carácter ASCII
    int 21h                 ; Mostrar el carácter de las decenas

    ; Mostrar el dígito de las unidades
    mov ah, 02h             ; Función DOS para mostrar un carácter
    mov dl, bl              ; Cargar las unidades (BL) en DL
    add dl, 30h             ; Convertir las unidades a carácter ASCII
    int 21h                 ; Mostrar el carácter de las unidades

main endp
end main
```

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





¿Cómo utilizaste los registros acumuladores y de datos para realizar el programa?

En este programa, se utilizan varios registros acumuladores y de datos para manejar la entrada y procesamiento de los números:

- **AL (parte baja de AX)**: Se utiliza para almacenar y manipular los dígitos ingresados por el usuario. Los dígitos son ingresados como caracteres ASCII, por lo que se convierten a su valor numérico restando 30h y se almacenan en **AL** antes de ser movidos a las variables de datos.
- **AH**: Este registro se usa junto con **AL** para llamar a interrupciones del sistema operativo (como la lectura del teclado y la impresión de mensajes). También se usa después de la operación **AAM** para almacenar el valor de las decenas.
- **BL**: Se usa como multiplicador para convertir el dígito de las decenas a su valor decimal al multiplicarlo por 10.
- **BX**: Al final, después de usar la instrucción **AAM**, los valores de decenas y unidades se almacenan en **BX** para ser procesados y mostrados como caracteres ASCII.

¿Por qué no se pueden escribir como en un lenguaje de alto nivel?

En lenguajes de bajo nivel como el ensamblador, se tiene un control directo sobre los registros y las operaciones del procesador, lo que significa que no hay abstracciones como en los lenguajes de alto nivel. Los lenguajes de alto nivel manejan automáticamente las operaciones aritméticas, el manejo de variables y la entrada/salida de datos, mientras que en ensamblador es necesario controlar manualmente cada paso de una operación, desde la lectura de datos hasta su conversión y procesamiento.

Además, en ensamblador se trabaja directamente con los registros y no con variables abstractas como en lenguajes de alto nivel, lo que complica el manejo de múltiples datos de manera simple y directa.

¿Si se requieren más dígitos, cómo lo realizarías? Explica.

Si se requiere procesar y mostrar números con más dígitos (por ejemplo, números de tres o cuatro dígitos), el enfoque debe modificarse para manejar los dígitos adicionales. A continuación los pasos:

1. **Capturar más dígitos**: Al igual que con los primeros dos dígitos (decenas y unidades), se necesitaría leer cada nuevo dígito adicional (centenas, millares, etc.) uno por uno utilizando

interrupciones de DOS (función 01h), y convertir cada uno de ellos de su representación ASCII a su valor numérico restando 30h.

2. **Expandir la multiplicación**: Para números con más dígitos, se debe multiplicar cada dígito por su posición en el sistema decimal. Por ejemplo, el tercer dígito (centena) se multiplicaría por 100, el cuarto por 1000, y así sucesivamente. Cada nuevo valor se sumaría para formar el número completo.
3. **Modificar el código para mostrar el resultado**: Después de calcular el número final, se puede usar la instrucción **AAM** o dividir manualmente el número para obtener cada dígito, como en la solución para dos dígitos. Si el número tiene más de 99, se necesitaría realizar divisiones sucesivas para descomponer el número en centenas, decenas y unidades.

Conclusión

En conclusión, el programa en ensamblador utiliza registros acumuladores y de datos para procesar y mostrar números ingresados manualmente, controlando directamente cada operación a nivel de hardware. A diferencia de los lenguajes de alto nivel, en ensamblador se requiere un manejo explícito de la entrada, conversión y salida de datos, lo que implica mayor complejidad y precisión en el código. Para manejar números de más dígitos, el enfoque se debe extender sumando y multiplicando cada dígito según su posición en el sistema decimal, lo que hace el código más complejo pero más flexible y funcional.