UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍA
DIVISION DE ELECTRONICA Y COMPUTACION

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

Programación de Bajo Nivel (IL358)

Reporte de Actividad Práctica

Práctica 4: "Manejo de Directorios y Archivos"

Alumno: Torres Rivera Rodrigo
Código: 397423431

Sección: D03

Profesor: Jose Manuel Espinoza

Fecha 19 de octubre de 2025
firma de revisado

Requerimientos de la práctica:

- Crear un directorio que se llame 1er_nobre_del_alumno
- Entrar al directorio 1er_nobre_del_alumno
- mostrar o verifica que estamos en ese directorio
- crear un archivo que se llame 1er_apellido_del_alumno.txt
- leer la fecha de la pc
- escribir la fecha dentro del archivo
- cerrar el archivo
- cambiar al primer directorio (fuera del directorio 1er_nobre_del_alumno)

Desarrollo:

El programa requiere llevar a cabo cada tarea en secuencia, ya que primero debe crear el directorio (de nombre Rodrigo)para poder entrar en él y ahí poder crear el archivo (de nombre Torres) una vez creado, debe escribir en él la fecha actual, lo cual implica que debe obtener la fecha del sistema, la cual estará en Hexadecimal y habrá que convertirla en ASCII, para que al escribir en el archivo la información sea legible, posteriormente debe regresar al directorio padre.

todas estos requerimientos se cumplen en la sección main del programa:

Creando el Directorio Rodrigo:

```
197     ; Crear directorio en C:\emu8086\MyBuild\ACTIVIAD05
198     lea dx, msgCrearDir
199     call ImprimirCadena
200     lea dx, rutaDirectorio
201     mov ah, 39h                     ; MKDIR DS:DX -> ASCIIZ
202     int 21h
203     jnc DIR_CREADO
204     ; CF=1 -> verificar si ya existe con Get File Attributes (43h)
205     lea dx, rutaDirectorio
206     mov ax, 4300h                   ; Get File Attributes of dirPath
207     int 21h
208     jc  DIR_ERROR_CF                ; no existe u otro error
209     ; Existe -> notificar CF activada por existencia
210     lea dx, msgDirExiste
211     call ImprimirCadena
212     jmp DESPUES_DIR
213 DIR_ERROR_CF:
214     lea dx, msgDirErrorCF
215     call ImprimirCadena
216     jmp DESPUES_DIR
217 DIR_CREADO:
218     lea dx, msgDirCreado
219     call ImprimirCadena
220 DESPUES DIR:
```

De la línea 198 a la línea 202, configuramos los parámetros para el uso de la INT 21H para crear un archivo, tomando en cuenta si la bandera CF falla o no, y tenemos las etiquetas DIR_ERRO_CF y DIR_CREADO.

Creando el Archivo:

```
222     ; Crear archivo y escribir contenido
223     lea dx, msgCrearArchivo
224     call ImprimirCadena
225
226     ; Construir contenido de fecha en dateBuf
227     call ConstruirFecha
228
229     ; Create (turnca si existe podria requerir 3Ch Entonces escribimos; 3Ch Falla si existe)
230     ; Intento 1: crear con 3Ch; si falla porque existe, abrimos con 3Dh modo escritura (2) y truncar no es directo en DOS clásico.
231     ; Para simplicidad: si 3Ch falla, intentamos abrir (3Dh, modo escritura=2) y sobreescribimos desde el principio.
232     mov cx, 0                       ; atributos normales
233     lea dx, rutaArchivo
234     mov ah, 3Ch                     ; CREATE
235     int 21h
236     jc  INTENTAR_ABRIR_ESCRITURA
237     mov [manejador], ax
238     jmp HACER_ESCRITURA
```

De la línea 223 a la 238 preparamos la creación del archivo, usamos en la línea 227 una procedimiento ConstruirFecha para preparar los datos que vamos a escribir en el archivo, tomamos en cuenta que la operación puede fallar, creamos las bifurcaciones INTENTAR_ABRIR_ESCRITURA e HACER_ESCRITURA, para el caso de que falle o sea un éxito.

Para leer la fecha de la PC y escribirla en el archivo en la línea 227 hacemos referencia al procedimiento de conversión de la fecha:

```
149  ConstruirFecha PROC
150      push ax
151      push bx
152      push cx
153      push dx
154      push di
155      mov ah, 2Ah         ; DOS Get Date
156      int 21h             ; CX=year, DH=month, DL=day
157      lea di, bufferFecha
158      mov al, dl          ; day
159      call DosDigitos
160      mov byte ptr [di], '/'
161      inc di
162      mov al, dh          ; month
163      call DosDigitos
164      mov byte ptr [di], '/'
165      inc di
166      mov ax, cx          ; year
167      call CuatroDigitos
168      ; CRLF
169      mov byte ptr [di], 0Dh
170      inc di
171      mov byte ptr [di], 0Ah
172      inc di
173      ; dateLen = DI - dateBuf
174      mov ax, di
175      sub ax, offset bufferFecha
176      mov [longitudFecha], ax
177      pop di
178      pop dx
179      pop cx
180      pop bx
181      pop ax
182      ret
183  ConstruirFecha ENDP
```

Usando la INT 21H opción 2A obtenemos la fecha como indica el comentario:

; CX=año, DH=mes, DL=dia

Usamos los procedimientos DosDigitos y CuatroDigitos para convertir cada uno de los datos de formato Hexadecimal a ASCII, el dato convertido lo guardamos en la variable BufferFecha la cual tiene suficiente espacio para guardar el dato.

Con el archivo ya creado o abierto en caso de que ya existía con anterioridad (etiqueta INTENTAR_ABRIR), procedemos a escribir los datos (HACER_ESCRITURA), como muestra la imagen:

```
INTENTAR_ABRIR_ESCRITURA:
    mov al, 2                       ; modo 2 = read/write
    lea dx, rutaArchivo
    mov ah, 3Dh                     ; OPEN
    int 21h
    jc  ARCHIVO_ERROR
    mov [manejador], ax
    ; Si llegamos aqui, el CREATE fallo pero el OPEN funciono => el archivo ya existia
    lea dx, msgArchivoExiste
    call ImprimirCadena
    ; situarse al inicio (por defecto al abrir está al inicio)

HACER_ESCRITURA:
    mov bx, [manejador]
    mov ah, 40h                     ; WRITE
    mov cx, [longitudFecha]
    lea dx, bufferFecha
    int 21h
    jc  ARCHIVO_ERROR_CERRAR
    ; Verificar bytes escritos
    cmp ax, cx
    jne ARCHIVO_ERROR_CERRAR

    ; Cerrar
    mov bx, [manejador]
    mov ah, 3Eh
    int 21h
    jc  ARCHIVO_ERROR

    lea dx, msgArchivoOk
    call ImprimirCadena
```

Cerramos el archivo y mandamos un mensaje a pantalla de que hemos escrito en el archivo.

Procedemos a cambiar al directorio padre:

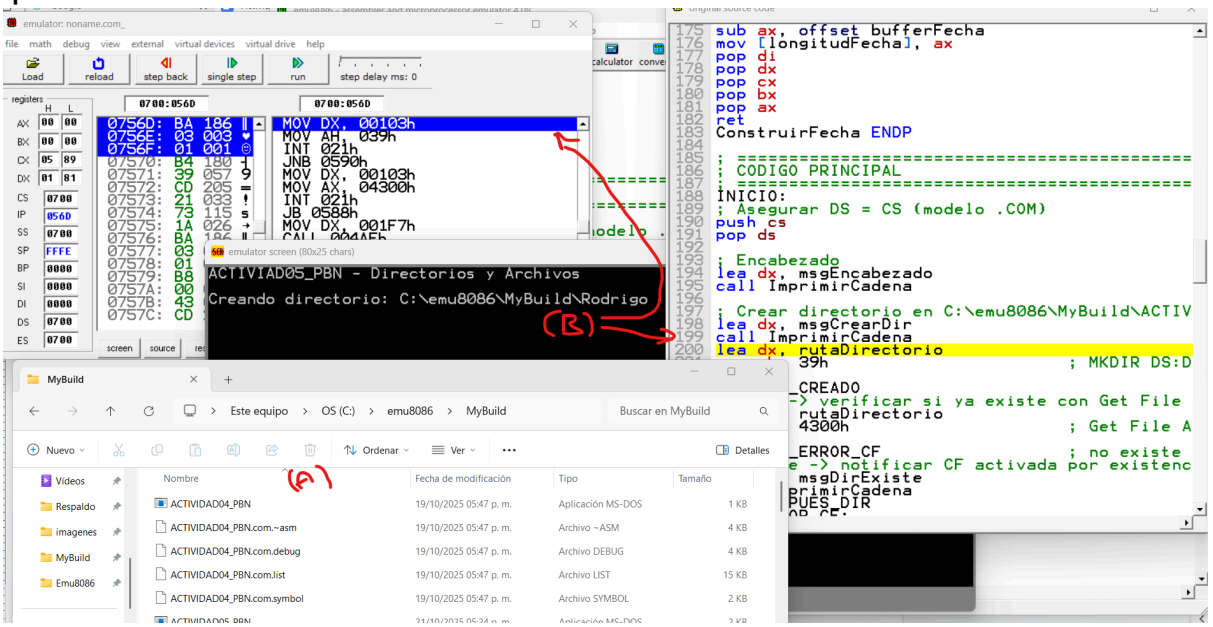
```
    ; Cambiar al directorio padre
    lea dx, msgCambiarDir
    call ImprimirCadena
    lea dx, rutaPadre
    mov ah, 3Bh                     ; CHDIR (Change Directory)
    int 21h
    jc  ERROR_CAMBIO_DIR
    lea dx, msgDirCambiado
    call ImprimirCadena
    jmp LEER_ARCHIVO
```

Si algo falla lo tratamos en la etiqueta ERROR_CAMBIO_DIR, y de ahí cualquier detalle adicional, si todo sale bien nos vamos hasta LEER_ARCHIVO, donde realizamos una lectura del contendio del archivo para comprobar que todo fue correcto, y si es asá saltamos a la etiqueta FIN, si existe un error se trata en las etiquetas correspondientes como ARCHIVO_VACIO

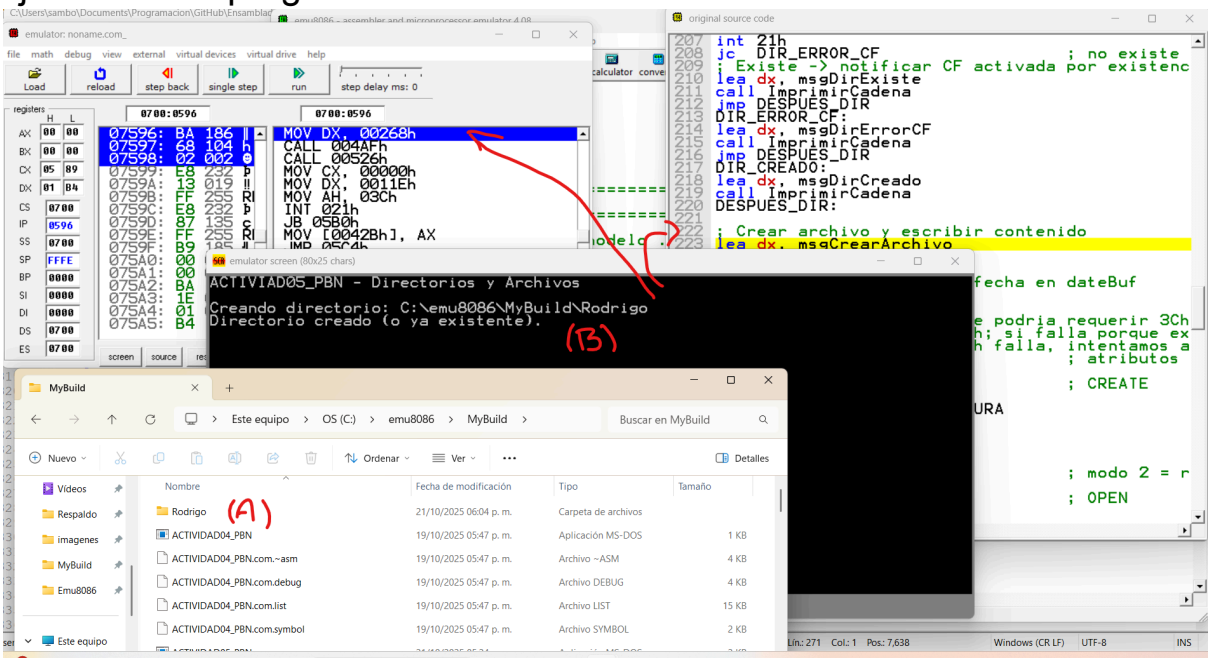
```
LEER_ARCHIVO:
    lea dx, msgLeerArchivo
    call ImprimirCadena

    ; Abrir solo lectura
    mov al, 0                           ; modo lectura
    lea dx, rutaArchivo
    mov ah, 3Dh
    int 21h
    jc  LECTURA_ERROR
    mov [manejador], ax

    ; Leer hasta 128 bytes
    mov bx, [manejador]
    mov cx, 128
    lea dx, bufferLectura
    mov ah, 3Fh
    int 21h
    jc  LECTURA_ERROR_CERRAR
    mov [bytesLeidos], ax

    ; Cerrar
    mov bx, [manejador]
    mov ah, 3Eh
    int 21h

    ; Mostrar contenido
    lea dx, msgContenidoLeido
    call ImprimirCadena
    mov ax, [bytesLeidos]
    cmp ax, 0
    je ARCHIVO_VACIO
    mov cx, [bytesLeidos]
    lea si, bufferLectura
    call ImprimirBuffer
    call ImprimirCRLF
    jmp FIN
```

Resultados Obtenidos:

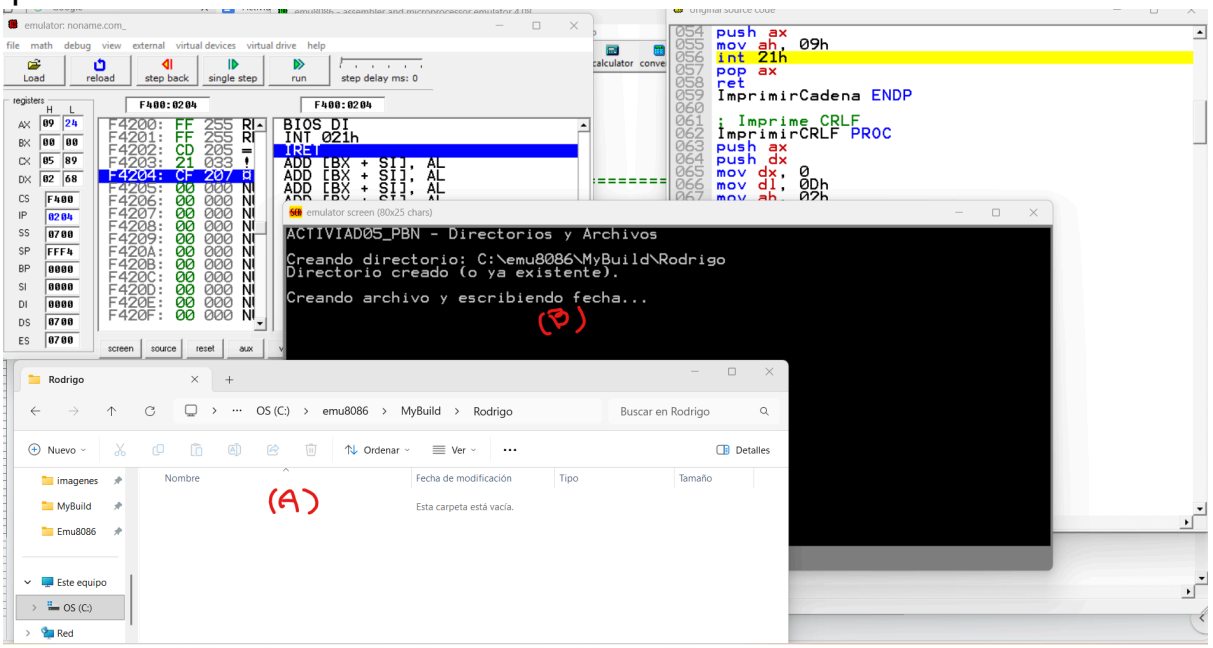
Durante la ejecución del programa en la siguiente imagen se nos informa en (B) que se va a crear el directorio, y podemos constar en (A) que aún no se realiza la acción.



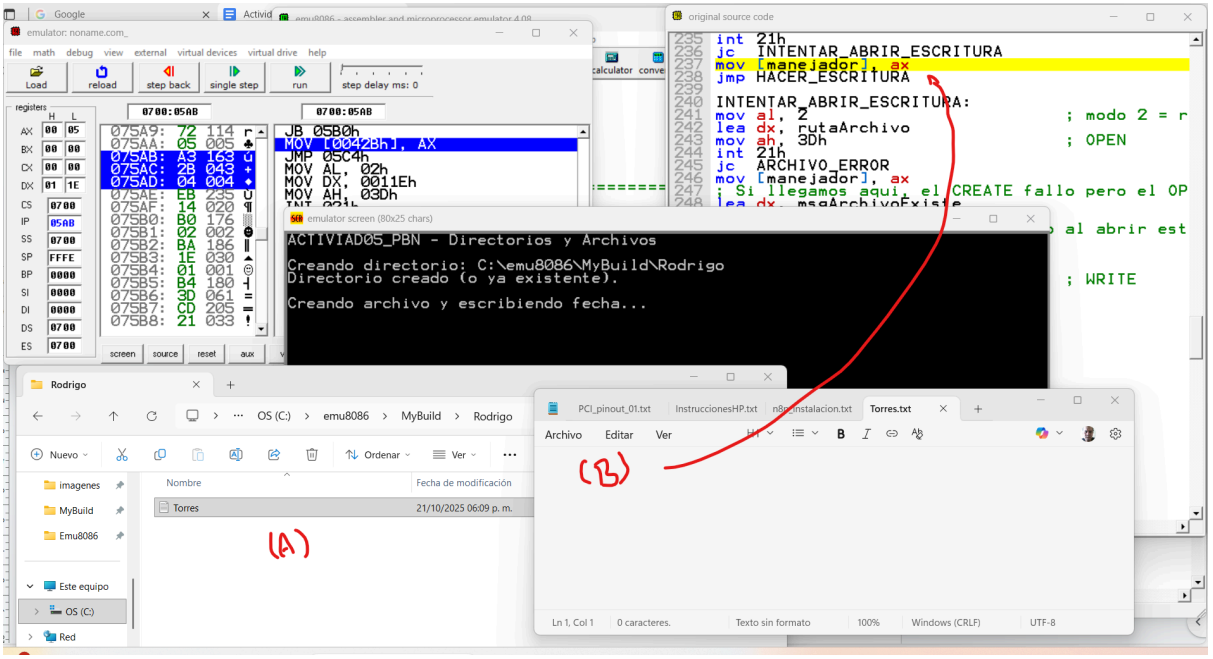
Como no hubo problemas, se crea el directorio (A) y vemos que el mensaje en (B) aun sigue en pantalla, pero hemos avanzado en la ejecución del programa.



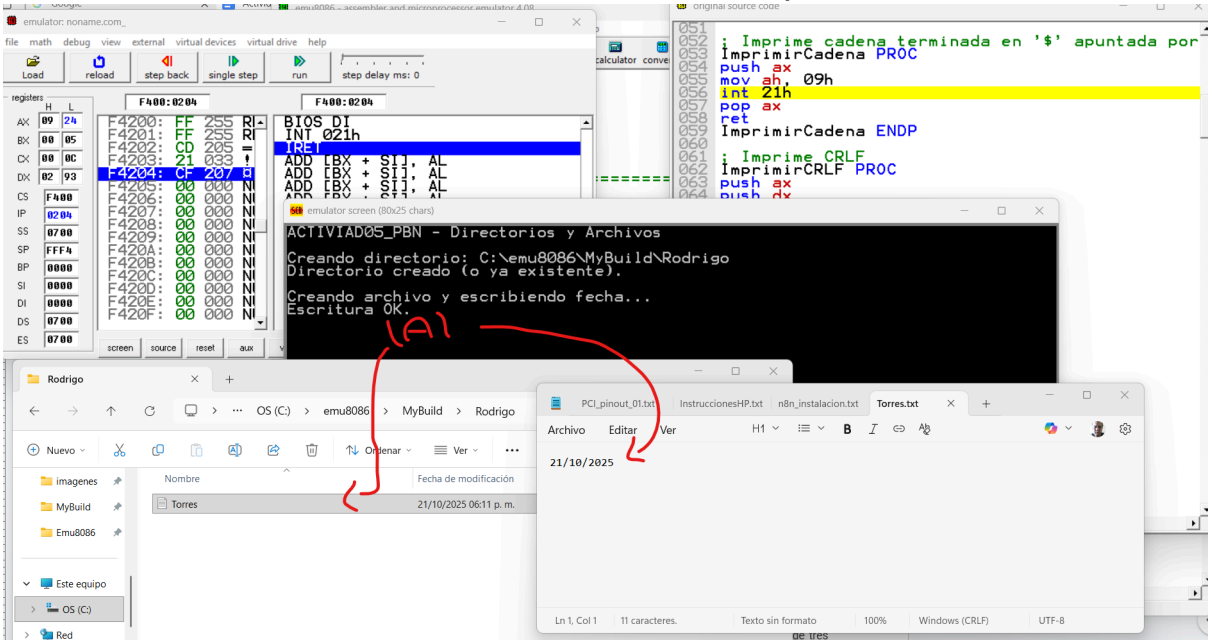
Ingresamos al directorio y vemos que en (A) esta vacío, (B) nos informa que vamos a crear el archivo:



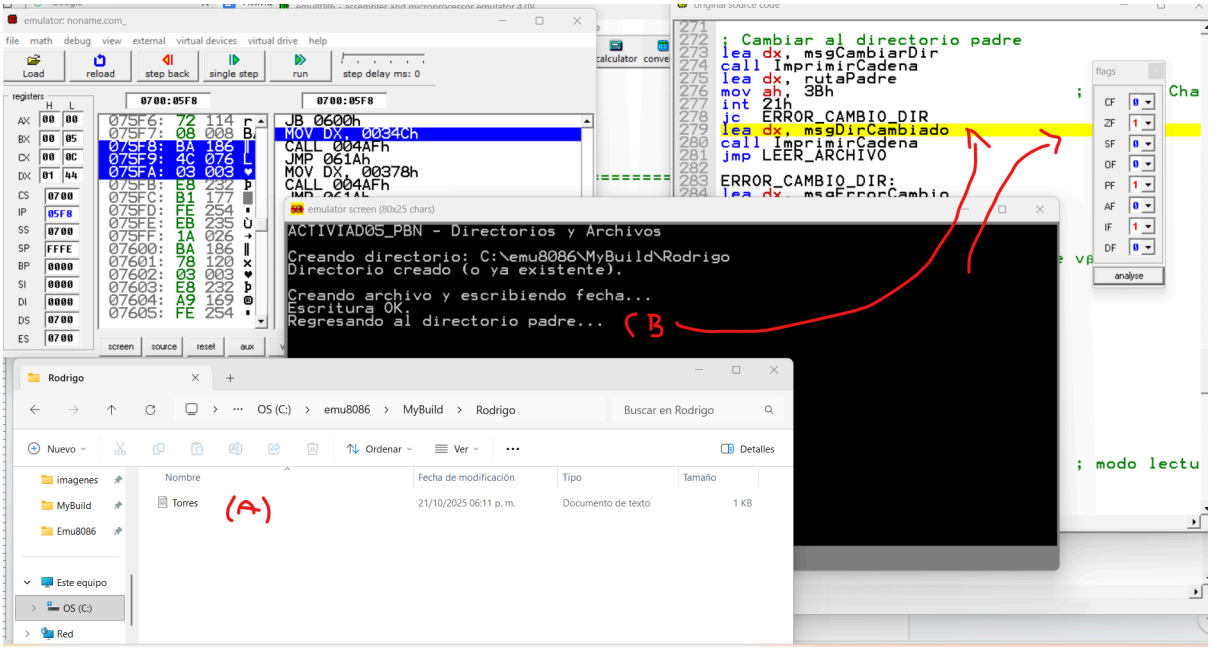
El archivo se crea (A), ingresamos al mismo en (B) y vemos que esta vacío, va a comenzar a escribir la fecha en el.



La operación de escritura se realiza con éxito (A) ahora vemos el contenido del archivo. observamos que esto implica que se realizó la conversión de Hex a ASCII en la función correspondiente



Como hemos terminado, la operación de cambio al directorio padre, no se ve en expresamente, pero vemos que al ejecutar la INT esta no falló debido a que CF no está activada indicando que el cambio de directorio se realizó correctamente.



El programa se ha comportado como se esperaba.

Conclusiones:

No lo parece pero el ensamblador debe poder efectuar todas las operaciones que ofrecen las instrucciones de lenguajes de alto nivel, esto es debido a que todos los programas escritos en dichos lenguajes son traducidos a código máquina para su ejecución y hemos visto que el ensamblador es una representación uno a uno de esos códigos de operación, ahora resulta obvio que cualquier programa escrito en Lenguajes de Alto nivel puede traducirse a Lenguaje Ensamblador.