**LABORATORIO NO. 02**

“Ensamblador, DEBUG y Sistemas Numéricos”

**Ejercicio 1: Utilización del Ensamblador y el Enlazador**

Utilizando los archivos “Ejemplo1.asm” y “Ejemplo2.asm” genere el código objeto y el programa ejecutable utilizando el Ensamblador “TASM” y el Enlazador “TLINK”.

Conteste las siguientes preguntas:

1. Cuando se genera el ejecutable del archivo “Ejemplo2.asm”, ¿cuál es la advertencia que se muestra en pantalla? ¿Por qué muestra esa advertencia?
2. Modificado el programa para que no muestre la advertencia, ¿cuál es el resultado del programa, es decir, por qué se imprime ese carácter y no un 30?
3. Modifique el código del archivo “Ejemplo2.asm” y utilizando la tabla de códigos ASCII, imprima en pantalla una letra “Z”.

**Ejercicio 2: Utilización del Modo “DEBUG”**

Comandos del Modo “DEBUG”:

* N Nombrar un programa.
* L Se encarga de cargar el programa.
* U        "Desensamblar" código máquina y pasarlo a código simbólico.
* A        Ensamblar instrucciones simbólicas y pasarlas a código máquina.
* D        Mostrar el contenido de un área de memoria.
* E        Introducir datos en memoria, iniciando en una localidad específica.
* G       Correr el programa ejecutable que se encuentra en memoria.
* P        Proceder o ejecutar un conjunto de instrucciones relacionadas.
* Q       Salir de la sesión con DEBUG.
* R       Mostrar el contenido de uno o más registros.
* T        Rastrear la ejecución de una instrucción.
* W       Escribir o grabar un programa en disco.

Utilizando el Modo “DEBUG” de DOS cargue el programa “Ejemplo2.exe” y responda las siguientes preguntas:

1. ¿En qué dirección de memoria inicia el código del programa?

**0CFD:0000**

1. ¿En qué dirección de memoria termina el código del programa?

**0CFD:001D CD21**

1. Aparecen los comentarios en pantalla ¿Sí? ¿No? ¿Por qué?

**No, debido a que no conforman como parte de las palabras reservadas, por lo que después de un ; todo lo escrito será un comentario.**

1. Para cada una de las instrucciones del programa, escriba la dirección de memoria que tiene asignada:

|  |  |
| --- | --- |
| Dirección de memoria | Instrucción |
| B8FF0C | Mov AX,@DATA |
| 0CFD:0003 8ED8 | Mov DS,AX |
| 0CFD:0005 B80000 | Mov AX,0000h |
| 0CFD:0008 B80000 | Mov BX,0000h |
| 0CFD:000B B015 | Mov AL,15h |
| 0CFD:000D B315 | Mov BL,15h |
| 0CFD:000F 02C3 | Add AL,BL |
| 0CFD:0011 B130 | Mov DL,AL |
| 0CFD:0013 02C1 | Mov AH,02 |
| 0CFD:0015 8AD0 | Int 21h |
| FE2E:6038 7204 | Mov AH,4CH |
| FE2E:289B 3CFF | int 21h |

1. ¿Cuál es la dirección del segmento de código?

Es una de las secciones de un programa en un fichero objeto o en memoria, que contiene instrucciones ejecutables.

1. Antes de iniciar la ejecución por pasos del programa, ¿cuáles son los valores de los registros de propósito general?

AX: El registro AX, el acumulador principal, es utilizado para operaciones que implican entrada/salida y la mayor parte de la aritmética. Por ejemplo, las instrucciones para multiplicar, dividir y traducir suponen el uso del AX. También, algunas operaciones generan código más eficiente si se refieren al AX en lugar de a los otros registros.

BX: El BX es conocido como el registro base ya que es el único registro de propósito general que puede ser índice para direccionamiento indexado. También es común emplear el BX para cálculos.

DX: El DX es conocido como registro de datos. Algunas operaciones de entrada/salida requieren uso, y las operaciones de multiplicación y división con cifras grandes suponen al DX y al AX trabajando juntos.

CX: El CX es conocido como el registro contador. Puede contener un valor para controlar el número de veces que un ciclo se repite o un valor para corrimiento de bits, hacia la derecha o hacia la izquierda. El CX también es usado para muchos cálculos.

1. El valor del IP, ¿coincide con la dirección de inicio del programa?

Contiene el desplazamiento de dirección de la siguiente instrucción que se ejecuta. El registro IP está asociado con el registro CS en el sentido de que el IP indica la instrucción actual dentro del segmento de código que se está ejecutando actualmente.

1. Utilice el comando para el rastreo instrucción por instrucción y, por cada línea del código, escriba el contenido de los registros internos del CPU.

