USO DE DEBUG

Con el sistema operativo DOS (MS-DOS = Microsoft Disk Operating System) se incluye un programa para observar el comportamiento de los registros del CPU 80386.
Recuérdese que los registros de esta arquitectura son un subconjunto elemental de aquellos presentes en modelos más modernos de la familia '86.
Usando DEBUG es posible observar el comportamiento de las instrucciones, la forma cómo éstas afectan a las banderas, los contenidos de la memoria, el código de las instrucciones; además permite ensamblar código elemental usando los mnemotécnicos del 8086.

Todos los comandos de DEBUG se invocan usando una sola letra y son los siguientes:

01) A (assemble)
02) C (compare)
03) D (dump)
04) E (enter)
05) F (fill)
06) G (go)
07) H (aritmética hexadecimal)
08) I (input)
09) L (load)
10) M (move)
11) N (name)
12) O (output)
13) Q (quit)
14) R (register)
15) S (search)
16) T (trace)
17) U (unassemble)
18) W (write)

En particular, obsérvese lo siguiente:
a) DEBUG opera bajo DOS.
b) Cuando se invoca (como en el ejemplo) sin argumentos, el contenido de la memoria es arbitrario.

En las siguientes secciones de hace un breve resumen de los comandos de DEBUG. El lector interesado en detalles puede consultar, por ejemplo, "USING ASSEMBLY LANGUAGE" de Allen L. Wyatt, QUE Corporation, 1987.

2.1.1. ASSEMBLE (A)

El comando A se usa para introducir mnemotécnicos de ensamblador y que éstos se traduzcan directamente a lenguaje de máquina en memoria.
La sintaxis es

A <dirección>

Prácticamente cualquier mnemotécnico es soportado por DEBUG, incluyendo los especificadores de "override" de segmento (CS:, DS:, ES:, SS:).
Una excepción es que DEBUG no puede diferenciar entre NEAR y FAR returns; asume que RET es "near" y RETF es "far".

2.1.2. COMPARE (C)

Este comando compara y reporta diferencias entre los contenidos de dos bloques de memoria.
La sintaxis es:

C <bloque> <dirección>

<bloque> es la dirección de inicio y fin de un bloque o, si se preceden con "L", la dirección de inicio y la longitud del bloque; <dirección> es el inicio de otro bloque. Se presupone que la longitud de ambos bloques es la misma.

2.1.3. DUMP (D)

Este comando despliega el contenido de una serie de localidades de memoria-
La sintaxis es:

D <dirección1> <dirección2>

Ambas direcciones son opcionales. La 1a es la dirección de inicio de despliegue; la 2a es la dirección de fin.

2.1.4. ENTER (E)

Este comando permite cambiar los contenidos de localidades específicas de memoria.
La sintaxis es:

E <dirección> <cambios>

<dirección> es el inicio de los cambios y <cambios> es una lista opcional de los cambios deseados. Los cambios pueden ser especificados en la línea de comandos en cualquier combinación de números hexadecimales o caracteres ASCII; los caracteres ASCII deben estar entre comillas simples o dobles.
Por ejemplo:

E 100 'Buenas Tardes'

Establece el patrón "42 75 65 6E 61 73 20 54 61 72 64 65 73" en memoria a partir de la localidad 100H. Cuando no se especifica <cambios> se entra en un modo especial en el que DEBUG despliega los valores de <dirección>. Entonces es posible teclear nuevos valores que reemplacen a los que se muestran. Si se teclea "-" DEBUG regresa a la localidad anterior. Si se activa la barra espaciadora DEBUG pasa a la siguiente localidad.

2.1.5. FILL (F)

Este comando llena un bloque de memoria con un valor específico o una serie de valores.
La sintaxis es:

F <bloque> <valor de relleno>

<bloque> es la dirección de inicio y final o , si se preceden con "L", la dirección de inicio y la longitud del bloque; <valor de relleno> es(son) el(los) valor(es) con los que debe de llenarse el bloque. Si <valor de relleno> representa menor bytes que los que se necesitan para llenar el bloque, la serie se repite hasta llenar el bloque.
Por ejemplo, cualquiera de las siguientes dos líneas llena (con 0s) el bloque DS:00FF:

F DS:0000 DS:00FF 0

F DS:0000 LFF 0

2.1.6. GO (G)

Este comando ejecuta el código en memoria. Si se está depurando un programa, permite ejecutar el código cargado en memoria. También permite establecer puntos de quiebre (breakpoints) que son direcciones en las que se detiene la ejecución del programa.
La sintaxis es:

G =<inicio> <quiebre1> <quiebre2> ... <quiebre10>

<inicio> es la dirección de inicio de ejecución; <quiebre1> hasta <quiebre10> son direcciones opcionales de paro del programa. Si no se especifica <inicio>, Go inicia con la dirección contenida en CS:IP. Para lograr los quiebres, DEBUG reemplaza el código en las direcciones de quiebre por el valor hexadecimal CC, que es el código de interrupción. Si DEBUG llega a CC todos los puntos de quiebre son restituidos, los registros se despliegan (como con el comando R [véase adelante]) y se para la ejecución.

2.1.7. Aritmética Hexadecimal (H)

Este comando ejecuta restas y suma hexadecimales.
La sintaxis es:

H <valor1> <valor2>

Como resultado de lo anterior, DEBUG regresa dos valores: la suma y la resta de los argumentos en hexa.

2.1.8. INPUT (I)

Este comando "jala" un byte de un puerto.
La sintaxis es:

I <puerto>

<puerto> es la dirección del puerto a leer. Se lee el dato y se despliega en pantalla.

2.1.9. LOAD (L)

Este comando se usa para cargar un archivo o sectores de disco a memoria.
La sintaxis es:

L <buffer> <numdisco> <sectorini> <numsector>

<buffer> es la dirección en donde se carga la información; <numdisco> es el número (opcional) del disco de donde se leerá la información (0=A, 1=B, 2=C, etc.); <sectorini> es el sector de disco absoluto (en hexadecimal) a leer; <numsector> es la cantidad de sectores a leer. No se pueden leer más de 80H (128) sectores. Si no se suministra la combinación <numdisco> <sectorini> <numsector> DEBUG presume que se desea leer un archivo. En este caso <buffer> es opcional. Debe usarse el comando N (ver más adelante) para especificar el archivo a leer. Éste se carga en CS:0100.

2.1.10. MOVE (M)

Este comando mueve un bloque de memoria de una localidad a otra.
La sintaxis es:

M <bloque> <dirección>

<bloque> es como arriba (ver 2.1.5.); <dirección> es la dirección destino. El bloque de origen y la dirección destino pueden traslaparse.

2.1.11. NAME (N)

Este comando se usa para especificar el nombre del archivo usado por los comandos LOAD y WRITE.
La sintaxis es:

N <nomarch1< <nomarch2>

<nomarch1> es la especificación de archivo completa que será "parseada" y colocada en el bloque de control en CS:005C. <nomarch2> es la especificación de archivo que será colocada en CS:006C.
La expresión tal cual se tecleó se almacena en CS:0081, precedida por el número de bytes tecleados.

2.1.12. OUTPUT (O)

Este comando pone un byte en el puerto especificado.
La sintaxis es:

O <puerto< <valor>

<valor> es el byte hexadecimal a escribir en <puerto>.

2.1.13. QUIT (Q)

Este comando se usa para salir de DEBUG.

2.1.14. REGISTER (R)

Este comando despliega los registros del CPU y los valores de las banderas.
La sintaxis es:

R <registro>

<registro> es el nombre opcional y puede ser alguno de los siguientes: AX, BX, CX, DX, SP, BP, SI, DI, DS, ES, SS, CS, IP, PC o F. IP y PC son sinónimos.

2.1.15. SEARCH (S)

Este comando permite buscar en un bloque de memoria una secuencia específica de valores.
La sintaxis es:

S <bloque> <valor_a_buscar>

<bloque> se define como antes (ver la sección 2_1_1).
<valor_a_buscar> es(son) el(los) valor(es) que deseamos buscar en el bloque.

2.1.16. TRACE (T)

Este comando permite ejecución paso-a-paso de las instrucciones de máquina. Después de cada instrucción se muestra el estado de los registros.

La sintaxis es:

T =<inicio> <cuenta>

<inicio> es la dirección de inicio de la traza
<cuenta> es el número de instrucciones a trazar

2.1.17. UNASSEMBLE (U)

Este comando decodifica los valores de un grupo de localidades de memoria a mnemotécnicos de 8086.

La sintaxis es la siguiente:

U <alcance>

<alcance>, que es opcional, es ya sea un par de direcciones de inicio y fin o, si se precede con "L", la dirección de inicio y la longitud del área a desensamblar.

2.1.18. WRITE (W)

Este comando se usa para escribir un archivo a sectores individuales de disco a disco.

La sintaxis es:

W <buffer> <numdisco> <sectorini> <numsector>

<buffer> es la dirección de donde se carga la información; <numdisco> es el número (opcional) del disco en donde se escribirá la información (0=A, 1=B, 2=C, etc.); <sectorini> es el sector de disco absoluto (en hexadecimal) en donde empieza la escritura; <numsector> es la cantidad de sectores a leer. No se pueden escribir más de 80H (128) sectores. Si no se suministra la combinación <numdisco> <sectorini> <numsector> DEBUG presume que el inicio de archivo es CS:100. En este caso <buffer> es opcional. Debe usarse el comando N (ver arriba) para especificar el archivo a escribir.

Antes de escribir BX:CX debe ajustarse al número de bytes que desean grabarse.

W no puede escribir a archivos con la extensión EXE o HEX.

Ejemplo:

N <nomarchivo> <cr>
BX:CX <--- 2000 <cr>
W <cr>