APÉNDICE B

DEBUG

- Para manejar el lenguaje ensamblador, es aconsejable conocer básicamente la arquitectura de la computadora, en especial cómo funciona el CPU.
- Por tanto es necesario saber utilizar la herramienta DEBUG.COM que ofrece el MS-DOS.
- DEBUG es un programa que permite introducir o escribir pequeños programas en ensamblador o lenguaje máquina.
- Se pudenden ejecutar instrucciones una a una (step by step o paso a paso), de esta manera se pueden localizar y corregir errores en los programas.
- Un "bug" (chinche, bicho, microbio, etc.) se denomina en informática a un error de programa (se dice que los errores de un programa "chinchan").
- El proceso de buscar, localizar y capturar "bugs" o errores, se denomina debugging (depuración), y de ahí el nombre de DEBUG.
- El origen del término debugging se remonta a los primeros días de la informática, cuando, en la universidad de Harvard, un ordenador MARK I se paró.
- Tras una intensa búsqueda, los técnicos encontraron una polilla atrapada en los contactos de un relé. Tras quitarla, anotaron en el libro de registro que habían "debugged" (quitado la polilla) del ordenador.
- Nuestro objetivo con la herramienta DEBUG es introducir instrucciones en lenguaje máquina e ir viendo cómo se van ejecutando, cómo se almacenan los resultados, dónde deben estar los datos y el desarrollo de los procesos que determinan el funcionamiento del computadora.
- Esta herramienta nos ayudará a entender cómo funciona la máquina y, sobre todo, para entender cómo funcionan las órdenes y las operaciones antes de incluirlas definitivamente en un programa.

Entrar en DEBUG

- Para ejecutar DEBUG se pueden utilizar dos métodos desde el prompt del DOS:
  - 1- DEBUG.- Ejecuta DEBUG en una nueva sesión
  - 2- DEBUG nombre_archivo_EXE_ó_COM.- Ejecuta DEBUG cargando en memoria el archivo .EXE ó .COM que indiquemos como parámetro.
- Al entrar en DEBUG, aparecerá un guión en la parte izquierda, y, a su derecha, un cursor. Este es el prompt de DEBUG, desde donde introduciremos las órdenes correspondientes.

Comandos del DEBUG

- El uso de DEBUG se rige bajo unos comandos especiales, los cuales se escriben desde el prompt de DEBUG:
- ? (help) Ayuda.- muestra una lista con los comandos de DEBUG y su uso. Sintaxis: ?

- Q (Quit).- salir de DEBUG.
  Sintaxis: Q
- H (Hex).- Suma y resta los números indicados como parámetros.
  DEBUG trabaja siempre en formato hexadecimal, por lo que los números a introducir y los resultados se expresarán en dicho formato.
  Sintaxis: H <número> <número>
  Ejemplo:

```
-H 3 2
0005 0001
```

- R (Register).- Muestra el contenido actual de los registros, permitiendo su modificación. Si no se especifica ningún parámetro visualiza el contenido de todos los registros y banderas:
  Sintaxis: R [nombre_registro]
  Ejemplo:

```
-R
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
DS=3BD0 ES=3BD0 SS=3BD0 CS=3BD0 IP=0100 NV UP EI PL NZ NA PO NC
3BD0:0100 0000          ADD     [BX+SI],AL
-
```

  - En la primera línea se visualiza el contenido o el valor de los registros de trabajo - o generales - (AX, BX, CX y DX), de los registros punteros de pila (SP y BP) y de los registros índice (SI yDI).
  - En la segunda línea se visualiza el contenido de los registros de segmento (DS, ES, SS y CS), del registro puntero de instrucciones (IP) y el valor de las banderas o indicadores.
  - Como se puede apreciar en el ejemplo, todos los segmentos tienen el mismo valor o apuntan al mismo segmento de memoria.
  - Esto se debe a que DEBUG trabaja normalmente con archivosCOM, y estos ocupan un único segmento solapado de 64 K, en el que los segmentos de datos, el de pila y el de código se encuentran aglutinados en el mismo segmento.
  - El registro puntero de instrucción vale 0100 (256 en decimal), se dice que está apuntando a la primera instrucción de un programa, puesto que los archivos COM poseen una cabecera de 255 bytes (de 0000 a 0099).
  - Los indicadores aparecen aquí codificados mediante la siguiente tabla:

Flag	OF	DF	IF	SF	ZF	AF	PF	CF
0	NV	UP	DI	PL	NZ	NA	PO	NC
1	OV	DN	EI	NG	ZR	AC	PE	CY

  - Si se especifica el nombre de un registro, muestra el valor o contenido actual de ese preciso registro, y debajo el signo : (dos puntos), invitando a introducir un nuevo valor para dicho registro. Si no se introduce ningún valor, conserva su valor anterior.

  Ejemplo:

```
-R AX
AX 0000
:
```

- Para mostrar y/o modificar el valor de los indicadores se indica una F (mayúscula) como parámetro.

Ejemplo:

```
-R F
NV UP EI PL NZ NA PO NC -
```

- E (Enter).- Muestra el contenido de la posición de memoria indicada (se introduce siempre la dirección relativa dentro del segmento actual), permitiendo modificar dicho contenido con un nuevo valor.
  Sintaxis: E<dirección>
  Ejemplo:

```
-E 100
3BD0:0100 E4.01

-E 101
3BD0:101 85.D8
```

- Los valores E4 y 85 son los antiguos valores que contenían las direcciones 0100 y 0101, tras el punto, aparece un cursor parpadeante y es aquí donde introduciremos el nuevo valor del byte en esa posición de memoria. En este caso, la secuencia de bytes 01D8 corresponden al código de la instrucción ADD AX,BX, la cual añade a AX el valor de BX, o lo que es lo mismo: AX=AX+BX.

- T (Trace).- Ejecuta una instrucción en la posición actual (CS:IP), a no ser que se indique otra. Tras la ejecución, muestra el contenido de los registros (como en la orden R), con el fin de analizar los resultados y los efectos de la instrucción.
  Sintaxis: T [<dirección>]

- G (go).- Ejecuta un programa, deteniéndose en la dirección especificada como parámetro.
  Sintaxis: G [<dirección>]

- U (unassemble).- Desensambla los códigos de las direcciones de memoria especificadas, es decir, que muestra un listado en ensamblador equivalente a los códigos encontrados. Por defecto, desensambla desde la posición actual (CS:IP), de 32 en 32 bytes.
  Sintaxis: u [<rango>]

- A (assemble).- Permite introducir directamente instrucciones o sentencias en ensamblador, con el fin de crear rutinas o programas en lenguaje máquina. El juego de sentencias del lenguaje máquina se denomina mnemotecnia y, a las sentencias, mnemónicos.
  Sintaxis: A[<dirección>]
  Ejemplo:

```
-A
3DB0:0100 ADD AX,BX
3DB0:0102
-
```

  - Tras la dirección ofrece el cursor para introducir las instrucciones. Al pulsar Return pasa a la siguientes, y así hasta que no introduzcamos ninguna orden (directamente Return), volviendo al prompt.
- N (name).- Asigna un nombre al archivo del programa. Esta orden NO GUARDA EL PROGRAMA, SI NO QUE SOLO ASIGNA EL NOMBRE.
  Sintaxis: N <nombre_archivo.COM>

```
-N PRUEBA.COM
```

- W (write).- Guarda el programa en el archivo indicado con la orden N. Hay que tener en cuenta que hay que asignar el número de bytes que ocupa el programa en el par de registros BX:CX, lo cual se puede hallar rápidamente con la orden H, indicando la dirección donde termina el programa y la dirección donde comienza.
  Sintaxis: W
  Ejemplo:

```
-N PRUEBA.COM           Aquí se asigna el nombre
-R BX                   A continuación se asigna una lon-
BX 0000                 gitud de archivo de 8 bytes
:0
-R CX
CX 0000
:8
-W                      Por último, se guarda el archivo
Writing 0008 bytes
-
```

- D (dump).- Vuelca o muestra el contenido de la memoria en pantalla. No muestra instrucciones, sólo muestra el contenido de las celdas de memoria (valor de sus bytes).
  Sintaxis: D [<intervalo>]

```
-D
3DB0:0100  01 D8 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
3DB0:0170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
-------------------------------------------------------------------------------
DIRECCION              VOLCADO HEXADECIMAL                  VOLCADO ASCII
```

  - Los caracteres ASCII no imprimibles se representarán con un punto (.).

BIBLIOGRAFÍA

- http://club.telepolis.com/mydream/Asm/
  Rafael Hernampérez Martín