Laboratorio de Sistemas Basados en Microprocesadores

Práctica 0b: Tutorial del entorno de desarrollo/depuración del 80x86 (II)

Estructura Básica de Programas en Lenguaje Ensamblador 80x86

Abrir el programa fuente "factor.asm" en un editor de texto. Podemos observar que básicamente contiene las definiciones de los diferentes segmentos del programa (datos, pila, extra y código). El orden en que aparecen las definiciones de los diferentes segmentos dentro del programa fuente es indiferente. No siempre estarán presentes los 4 segmentos en todos los programas. Por ejemplo, el segmento extra podría no estar definido.

Al comienzo del segmento de código hay que utilizar la directiva ASSUME para relacionar el nombre dado a los segmentos con el registro de segmento a través del cual van a ser accedidas las posiciones de memoria existentes en cada segmento. Sin embargo, esta directiva no carga el valor de los registros de segmento (DS, SS, ES) con los valores asignados a los segmentos. Las instrucciones que cargan estos valores deben ser las primeras de cualquier programa.

```
MOV AX, DATOS
MOV DS, AX
MOV AX, PILA
MOV SS, AX
MOV AX, EXTRA
MOV ES, AX
```

Las instrucciones sólo pueden existir en el segmento de código. Sin embargo, los datos pueden estar definidos en cualquier segmento. La situación más usual es que aparezcan todos en los segmentos de datos/extra, pero en este ejemplo vemos definida la variable FACT_DATO_1 dentro del segmento de código. Esta es la razón por la cual las instrucciones del programa comienzan en la dirección cs:0002 y no en la cs:0000 (que será la habitual durante el curso).

Para finalizar el programa es necesario utilizar la interrupción INT 21H con AX=4C00h, que devuelve el control al sistema operativo.

La última línea debe tener la directiva END seguida por el nombre del procedimiento donde empieza a ejecutarse el programa, en nuestro caso será el procedimiento START. Al arrancar el TD, la flecha apuntará justo al comienzo de dicho procedimiento, y el registro "ip" contendrá la dirección de memoria correspondiente.

Inspección de variables

En ocasiones puede sernos de gran utilidad ver el contenido de las variables de nuestro programa en memoria mientras estamos en el proceso de depuración con el “td”.

Para ello es necesario modificar ligeramente el makefile que creamos anteriormente. En concreto, vamos a modificar la última línea de la siguiente forma:

Antes:

```
tasm /zi factor.asm
```

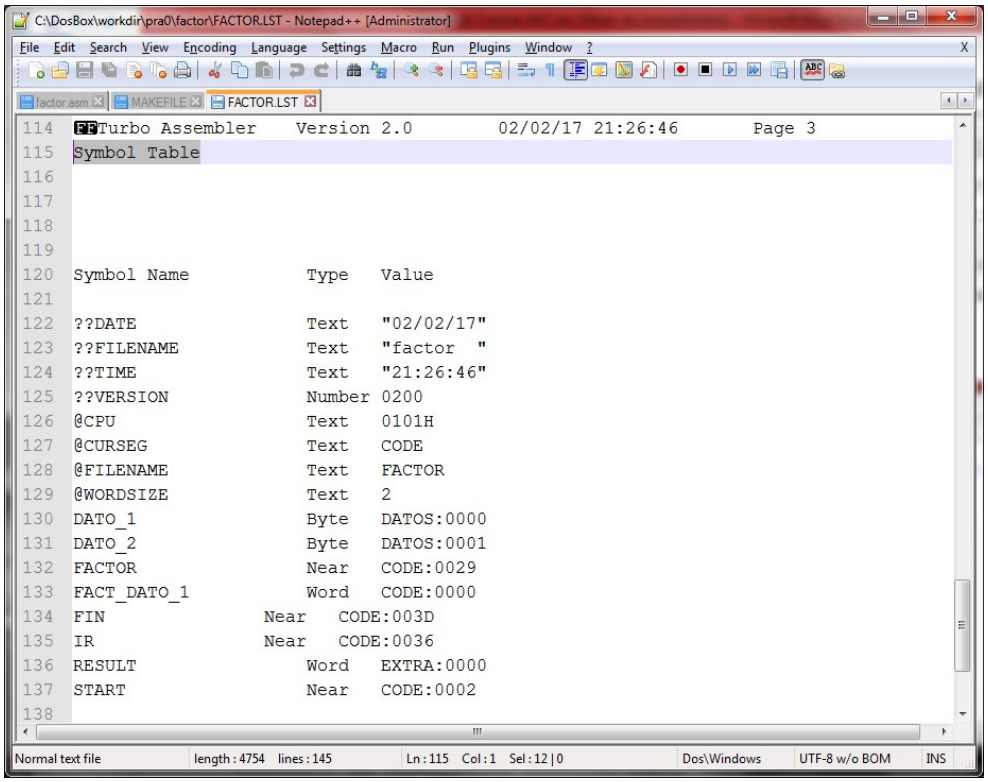
Ahora:

```
tasm /zi factor.asm,,factor.lst
```

Si ejecutamos “make” tras estos cambios (borrando los ficheros factor.obj y factor.exe previamente), podemos observar tecleando “dir” que se ha creado un fichero adicional, llamado factor.lst.

En este nuevo fichero podemos encontrar en primer lugar la correspondencia entre las instrucciones en ensamblador de nuestro programa y las instrucciones en lenguaje máquina generadas (obsérvese que es una información similar a la que vemos en la ventana 1 del td).

Sin embargo, la información útil si queremos ver el contenido de las variables del programa se encuentra hacia el final del fichero factor.lst: la tabla de símbolos del programa.



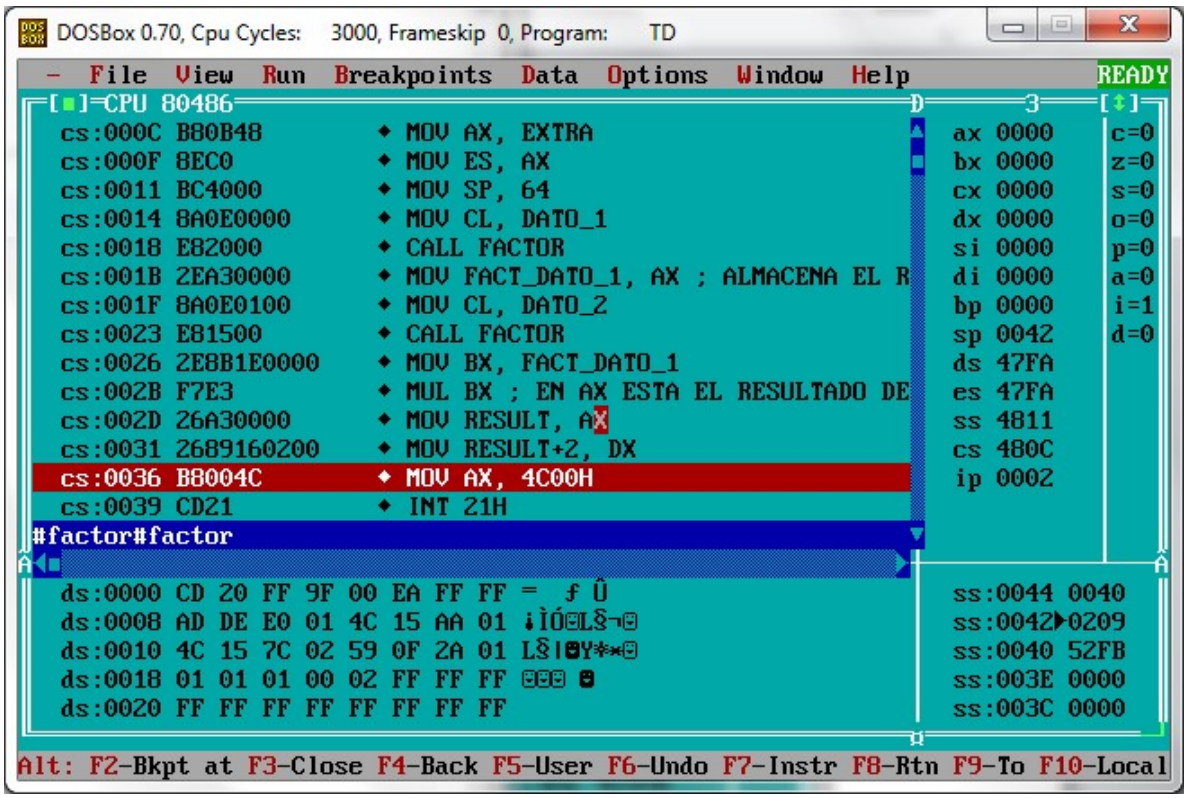
A falta de adquirir los conocimientos teóricos que nos permitan entender completamente esta información, avanzaremos aquí que las direcciones lógicas de las variables se componen de dos campos: segmento y offset. Así, por ejemplo:

```
DATO_1: segmento DATOS, offset 0

DATO_2: segmento DATOS, offset 1

FACT_DATO_1: segmento CODE, offset 0

RESULT: segmento EXTRA, offset 0
```

Con esta información en mente, vamos a ejecutar nuestro programa en el "td". Esta vez, en lugar de ejecutar paso a paso con F7, podemos colocar un *breakpoint* en la instrucción

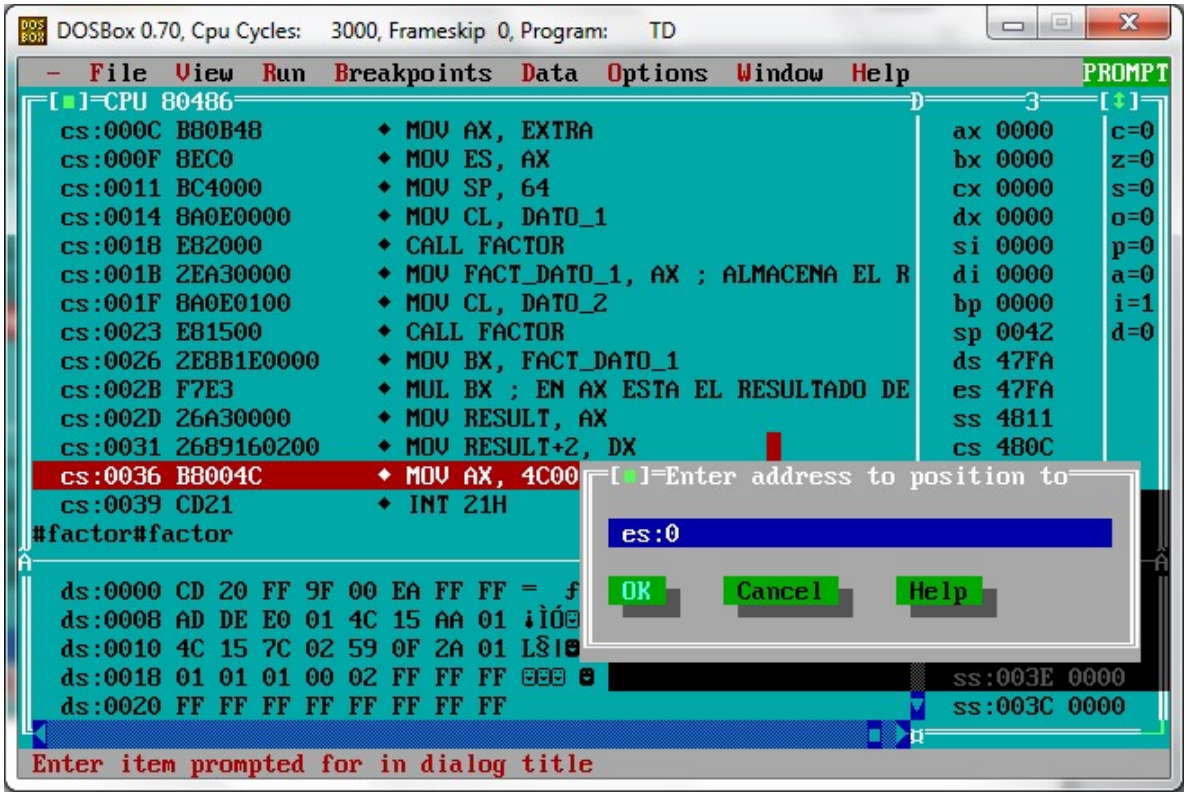
MOV AX, 4C00H

Para ello, nos situamos en la instrucción con el cursor y pulsamos F2. Si volvemos con el cursor hacia arriba comprobamos que el breakpoint se ha activado, lo cual se indica mediante una línea de color rojo.

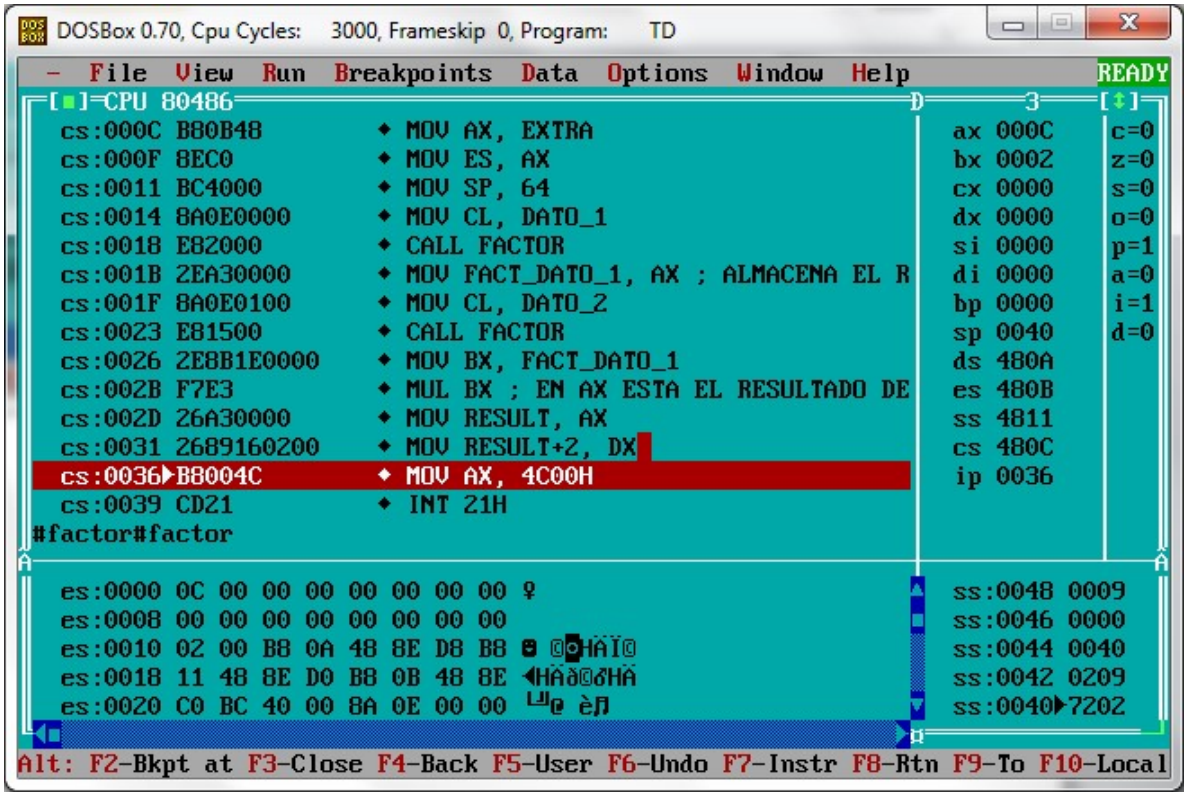


Seguidamente ejecutamos el programa pulsando F9. El programa será ejecutado hasta detenerse en el breakpoint. En este momento la variable RESULT ya debe contener su valor final. Podemos inspeccionar su valor en la ventana número 5, mediante "Goto →ES:0".

Recordemos que hemos asociado el segmento EXTRA al registro ES mediante la directiva ASSUME.



Podemos ver como la ventana muestra el valor **0C**, es decir, el valor 12 decimal (= 2! x 3!).



De manera análoga podríamos inspeccionar el resto de variables del programa (DATO_1, DATO_2 y FACT_DATO_1), mediante “Goto →DS:0”, “Goto →DS:1” y “Goto →CS:0” respectivamente.

Ejercicio 1: Cálculo de factoriales

El programa “factor.asm” ha sido diseñado para calcular el producto del factorial de un número por el factorial de otro número. Realizar las modificaciones oportunas en el código fuente para calcular los siguientes productos, con la ayuda del TD. Anotar los resultados en una hoja, tanto en formato hexadecimal como decimal, incluyendo una captura de pantalla de DOSBOX y una breve explicación de las modificaciones y los resultados obtenidos.

1. 4! x 5! =
2. 8! =
3. 9!=
4. 8! x 7! =

¿Cuál es el valor de la variable FACT_DATO_1 en cada uno de los casos?

Ejercicio 2: Modificación del Programa “Factor”

Modificar el programa “factor.asm” de forma que calcule factoriales de productos en lugar de productos de factoriales. Una vez modificado, calcular las expresiones mostradas a continuación con la ayuda del TD.

Anotar los resultados en una hoja, tanto en formato hexadecimal como decimal, incluyendo una captura de pantalla de DOSBOX y una breve explicación de las modificaciones y los resultados obtenidos.

1. (2 x 3) ! =
2. (2 x 4) ! =
3. (3 x 3) ! =
4. (5 x 2) ! =

Ejercicio 3: Modificación del Programa "Alumno"

El programa "alumno.asm", situado dentro del subdirectorio "alumnos" del directorio "pra0", ha sido diseñado para solicitar al usuario la introducción del nombre por teclado e imprimir por pantalla una línea de texto incluyendo en ella el nombre introducido.

Se pide repetir todo el proceso seguido para el programa "factor", es decir, ensamblar, enlazar y ejecutar utilizando el TD. Este programa puede ser ejecutado también desde línea de comandos, ya que incluye entrada/salida de datos por teclado/pantalla.

Realizar las modificaciones oportunas en el código fuente para que pregunte por separado por el nombre, primer apellido y segundo apellido del usuario, y a continuación imprima por pantalla una única línea de texto incluyendo los 3 datos anteriormente introducidos.

Por ejemplo: "DON FERNANDO ALONSO GARCIA ES ALUMNO DE 2 CURSO DE INGENIERIA INFORMATICA".

Desde dentro del programa TD, durante el proceso de depuración, podemos ver lo que el programa "alumno" va escribiendo en pantalla mediante la combinación de teclas Alt+F5.

Una vez realizado el ejercicio añadir comentarios en el fichero fuente detallando los cambios realizados.

ENTREGA DE LA PRÁCTICA: Fecha y contenido

Ejercicios 1 y 2 : se deberá subir a Moodle una memoria que incluya los resultados, tanto en formato hexadecimal como decimal, incluyendo una captura de pantalla de DOSBOX y una breve explicación de las modificaciones realizadas y los resultados obtenidos.

Ejercicio 3: Se deberá subir a Moodle un fichero comprimido zip que contenga el fichero fuente comentado y el Makefile. El fichero fuente, deberá incluir los comentarios con las modificaciones realizadas y el nombre del alumno en la cabecera.

El límite de fecha de subida de los ficheros para cada grupo es:

Ejercicios 1 y 2:

Grupos del Jueves:4 de Marzo a las 20:15

Grupos del Viernes: 5 de Marzo a las 19:15h

Ejercicio 3:

Grupos del Jueves:10 de Marzo a las 23:55

Grupos del Viernes: 11 de Marzo a las 23:55h