



# Laboratorio de Sistemas Basados en Microprocesadores

# Práctica Ob: 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

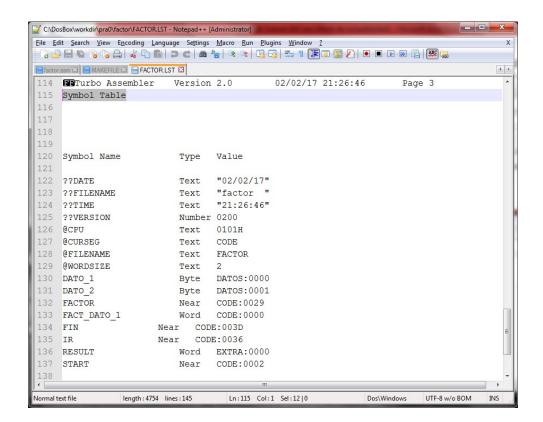
Ahora:

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.

```
X
DOSBox 0.70, Cpu Cycles: 3000, Frameskip 0, Program:
     File
           View Run Breakpoints Data Options
                                                     Window
                                                                                READY
  [ • ]=CPU 80486=
                                                                                 ‡1=
                                                                     ax 0000
  cs:000C B80B48

    MOV AX, EXTRA

                                                                                 C=0
                          . MOV ES, AX
                                                                     bx 0000
  cs:000F 8EC0
                                                                                 z=0
                          ◆ MOV SP, 64
  cs:0011 BC4000
                                                                     cx 0000
                                                                                 s=0
  cs:0014 8A0E0000
                          MOV CL, DATO_1
                                                                     dx 0000
                                                                                 0=0
  cs:0018 E82000

    CALL FACTOR

                                                                        0000
                                                                     si
                                                                                 p=0

    ◆ MOV FACT_DATO_1, AX ; ALMACENA EL R

  cs:001B ZEA30000
                                                                     di 0000
                                                                                 a=0
  cs:001F 8A0E0100

    MOV CL, DATO_2

                                                                     bp 0000
                                                                                 i=1

    CALL FACTOR

                                                                                 d=0
  cs:0023 E81500
                                                                     sp 0042
  cs:0026 ZE8B1E0000

    MOV BX, FACT_DATO_1

                                                                     ds 47FA

    MUL BX ; EN AX ESTA EL RESULTADO DE

  cs:002B F7E3
                                                                     es 47FA
  cs:002D 26A30000
                          ◆ MOV RESULT, AX
                                                                     ss 4811
   cs:0031 2689160200

    MOV RESULT+2, DX

                                                                     cs 480C

    MOV AX, 4C00H

  cs:0036 B8004C
                                                                     ip 0002
  cs:0039 CD21

    INT 21H

 #factor#factor
  ds:0000 CD 20 FF 9F 00 EA FF FF
                                                                     ss:0044 0040
  ds:0008 AD DE E0 01 4C 15 AA 01 iÌÓ@L§¬@
                                                                     ss:004210209
   ds:0010 4C 15 7C 02 59 0F 2A 01 L§I@Y**®
                                                                     ss:0040 52FB
                                                                     ss:003E 0000
  ds:0018 01 01 01 00 02 FF FF FF 999 8
  ds:0020 FF FF FF FF FF FF FF
                                                                     ss:003C 0000
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local
```

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.

```
×
DOSBox 0.70, Cpu Cycles:
                      3000, Frameskip 0, Program:
    File View Run Breakpoints Data Options
                                                   Window
                                                                            PROMPT
  [ ]=CPU 80486=
                                                                             :[‡]=
                         • MOV AX, EXTRA
  cs:000C B80B48
                                                                   ax 0000
                                                                              c=0
                         • MOV ES, AX
                                                                   bx 0000
  cs:000F 8EC0
                                                                              z=0
                         • MOV SP, 64
• MOV CL, DATO_1
                                                                   cx 0000
  cs:0011 BC4000
                                                                              s=0
  cs:0014 8A0E0000
                                                                   dx 0000
                                                                              0=0
                                                                   si 0000
  cs:0018 E82000

    CALL FACTOR

                                                                              p=0
  cs:001B ZEA30000
                         ◆ MOV FACT_DATO_1, AX ; ALMACENA EL R
                                                                   di 0000
                                                                              a=0
  cs:001F 8A0E0100
                         • MOV CL, DATO_2
                                                                   bp 0000
                                                                              i=1

    CALL FACTOR

  cs:0023 E81500
                                                                   sp 0042
                                                                              d=0
  cs:0026 ZE8B1E0000
                         MOV BX, FACT_DATO_1
                                                                   ds 47FA
  cs:002B F7E3
                         ◆ MUL BX ; EN AX ESTA EL RESULTADO DE
                                                                   es 47FA
                         • MOV RESULT, AX
  cs:002D 26A30000
                                                                   ss 4811
  cs:0031 2689160200

    MOV RESULT+2, DX

                                                                   cs 480C
                         ◆ MOV AX, 4C00
                                         =[ ]=Enter address to position to=
  cs:0036 B8004C
   cs:0039 CD21
                         • INT 21H
 #factor#factor
                                          es:0
  ds:0000 CD 20 FF 9F 00 EA FF FF =
  ds:0008 AD DE E0 01 4C 15 AA 01 ilú@
  ds:0010 4C 15 7C 02 59 0F 2A 01 L§18
  ds:0018 01 01 01 00 02 FF FF FF @@@ 8
                                                                   ss:003E 0000
  ds:0020 FF FF FF FF FF FF FF
                                                                   ss:003C 0000
 Enter item prompted for in dialog title
```

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

```
X
DOSBox 0.70, Cpu Cycles: 3000, Frameskip 0, Program:
     File View Run Breakpoints Data Options Window Help
                                                                              READY
  [ ] CPU 80486=
                         • MOV AX, EXTRA
                                                                    ax 000C
  cs:000C B80B48
                                                                               c=0
                         . MOV ES, AX
  cs:000F 8EC0
                                                                    bx 0002
                                                                               z=0
                         • MOV SP, 64
                                                                    cx 0000
                                                                               s=0
  cs:0011 BC4000
  cs:0014 8A0E0000

    MOV CL, DATO_1

                                                                    dx 0000
                                                                               0=0
                                                                    si 0000
  cs:0018 E82000

    CALL FACTOR

                                                                               p=1
                                                                    di 0000
  cs:001B ZEA30000
                         ◆ MOV FACT_DATO_1, AX ; ALMACENA EL R
                                                                               a=0
  cs:001F 8A0E0100
                         MOV CL, DATO_2
                                                                    bp 0000
                                                                               i=1
                                                                    sp 0040
  cs:0023 E81500

    CALL FACTOR

                                                                               d=0
                         • MOV BX, FACT_DATO_1
  cs:0026 ZE8B1E0000
                                                                    ds 480A
  cs:002B F7E3
                         ◆ MUL BX ; EN AX ESTA EL RESULTADO DE
                                                                    es 480B
                         . MOV RESULT, AX
  cs:002D 26A30000
                                                                    ss 4811
  cs:0031 2689160200

    MOV RESULT+2, DX

                                                                    cs 480C
  cs:0036 B8004C

    MOV AX, 4C00H

                                                                    ip 0036
  cs:0039 CD21

    INT 21H

 #factor#factor
                                                                   ss:0048 0009
  es:0000 0C 00 00 00 00 00 00 00 9
  es:0008 00 00 00 00 00 00 00 00
                                                                   ss:0046 0000
  es:0010 02 00 B8 0A 48 8E D8 B8 8 0 0HAIC
                                                                    ss:0044 0040
  es:0018 11 48 8E DO B8 0B 48 8E ◀HÃð©♂HÃ
                                                                    ss:0042 0209
   es:0020 C0 BC 40 00 8A 0E 00 00 <sup>山</sup>@ è月
                                                                    ss:0040 7202
Alt: F2-Bkpt at F3-Close F4-Back F5-User F6-Undo F7-Instr F8-Rtn F9-To F10-Local
```





De manera análoga podríamos inspeccionar el resto de variables del programa (DATO\_1, DATO\_2 y FACT\_DATO\_1), mediante "Goto  $\rightarrow$  DS:0", "Goto  $\rightarrow$  DS:1" y "Goto  $\rightarrow$  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