

# **Manual de Copro II**

**Curso 2007 - 2008**  
**Asignatura: Laboratorio de Computadores**  
**3º Ingeniería en Informática**  
**Departamento de Informática e Ingeniería de Sistemas**  
**Centro Politécnico Superior**

# Manual de Copro II

## 1. El monitor

Nada más iniciar la aplicación *Monitor*, se abre la ventana que aparece en la figura 1. Esta ventana contiene, a su vez, otras subventanas que muestran el estado del procesador “Copro II”: registros, memorias, la pila. . .

Algunas de las ventanas permiten modificar su contenido. Dichas ventanas son las del banco de registros, macromemoria, *flags* y registros especiales. Para modificar un dato basta con seleccionar el campo en cuestión con el ratón y teclear el nuevo valor en hexadecimal. Conforme vayamos escribiendo, cada dígito introducido irá apareciendo en el dígito de menor peso, provocando que el de mayor peso antiguo desaparezca.

### 1.1. Las ventanas

#### Registros

La ventana muestra el valor de los 64 registros de uso común de la tarjeta (figura 2).

#### Flags

La ventana de *flags* muestra el valor de los micro y macroflags. Se puede modificar el valor de cualquiera de ellos pinchando con el ratón (figura 2).

- Z. Cero.
- N. Negativo.

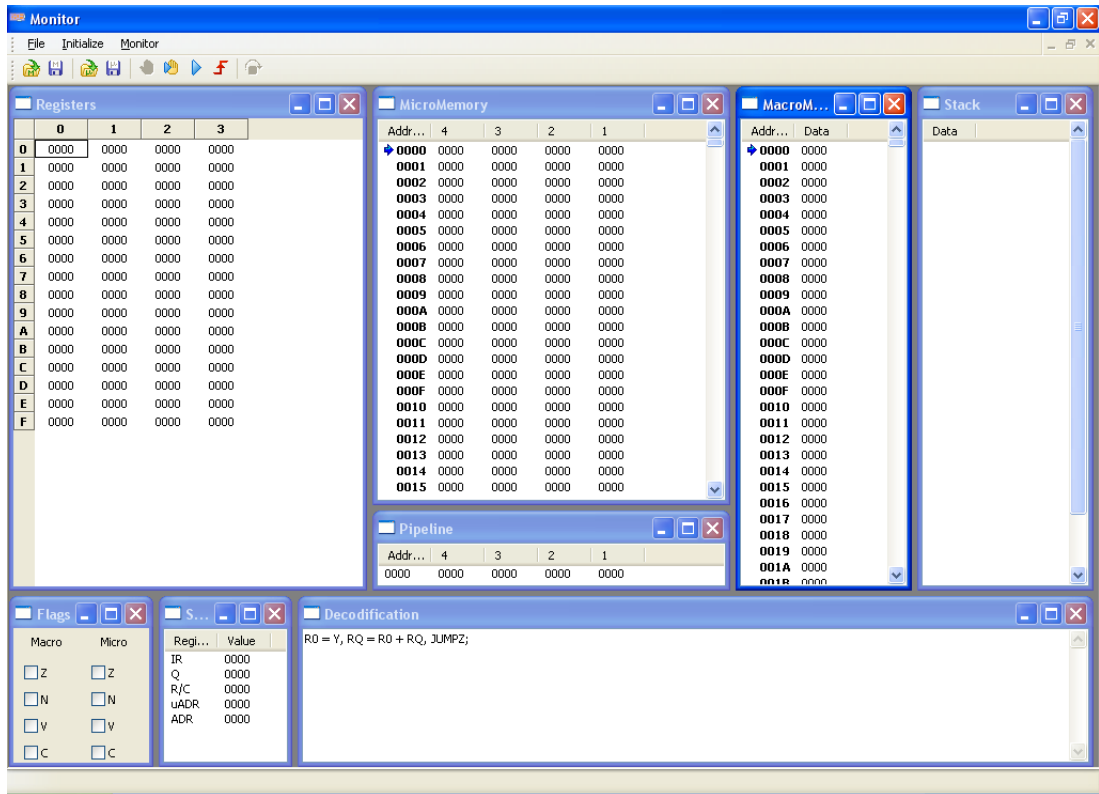


Figura 1: Ventana del monitor

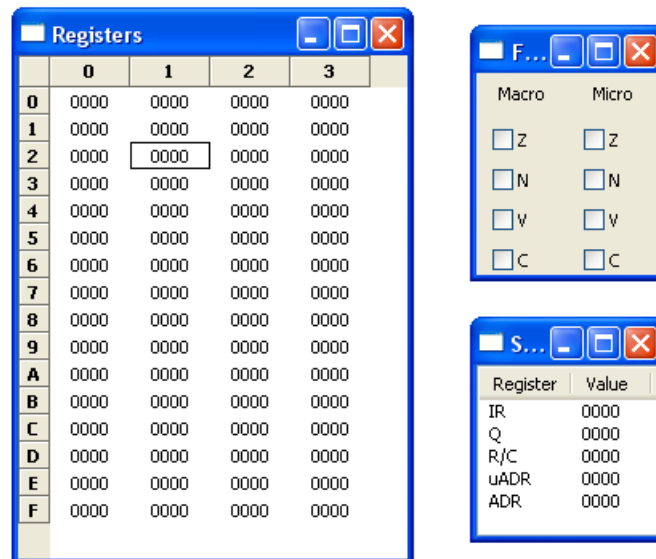


Figura 2: Banco de registros, ventana de *flags* y ventana de registros especiales

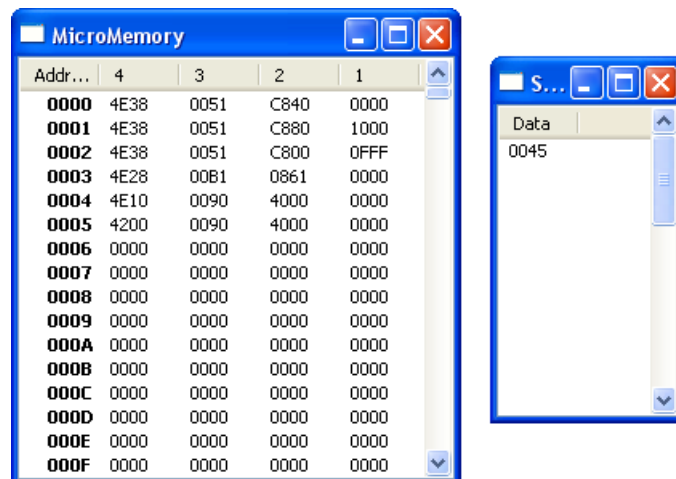


Figura 3: Micromemoria y pila

- V. Desbordamiento.
- C. Acarreo.

### Registros especiales

Esta ventana muestra los registros especiales del procesador (figura 2).

- **IR.** Registro de instrucción.
- **Q.** Registro Q de la ALU.
- **R/C.** Registro contador del secuenciador.
- **$\mu$ ADR.** Dirección de la micromemoria.
- **ADR.** Registro que direcciona la macromemoria (mar).

El campo  $\mu$ ADR refleja el valor de la dirección de la micromemoria y no corresponde a ningún registro de la tarjeta real, por lo que no puede ser modificado. Direcciona la siguiente dirección a la que se está ejecutando y cuyo contenido ya está cargado en el *pipeline*.

### Pila

Esta ventana (figura 3) muestra el contenido de la pila de la tarjeta. Tiene un máximo de 33 niveles. La cima de la pila es el último valor que aparece en la ventana.

Address	4	3	2	1
0000	4E38	0051	C840	0000

Figura 4: *Pipeline*

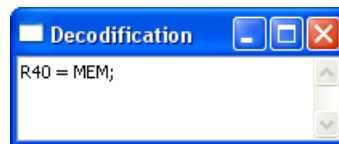


Figura 5: Ventana de decodificación

## Micromemoria

Esta ventana (figura 3) muestra el contenido de la micromemoria, que tiene 4k entradas de 64 bits cada una. También indica con una flecha la palabra que está direccionando el campo  $\mu$ ADR.

## Pipeline

Esta ventana muestra el contenido del registro *pipeline*. El campo dirección y el contenido del registro no son modificables. Este registro contiene la siguiente microinstrucción que se va a ejecutar, la cual aparece desensamblada en la ventana de decodificación.

## Decodificación

La ventana de decodificación muestra el contenido del registro *pipeline* desensamblado. Algunas microinstrucciones de “Copro II” no se pueden expresar con el ensamblador, por lo que dichas microinstrucciones son decodificadas mostrando la directiva `INLINE` seguida del valor binario que tiene un determinado campo de la microinstrucción.

## Macromemoria

Esta ventana muestra el contenido de la macromemoria. Para modificar cualquier posición de la misma basta con seleccionarla y teclear en hexadecimal el nuevo valor.

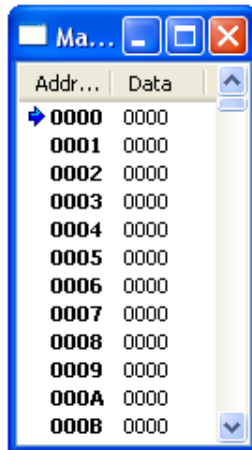


Figura 6: Macromemoria

## 1.2. La barra de menús

### Menú Archivo

Este menú permite cargar cualquiera de las dos memorias con el contenido de un fichero de texto. Asimismo, también hace posible guardar en un fichero de texto el contenido que muestran en pantalla. El formato de los ficheros debe ser el siguiente:

- **Micromemoria:** `xxxx yyyyyyyyyyyyyyy<CR>`. Las *x* indican en hexadecimal la dirección de micromemoria y la *y*, el valor de esa palabra de memoria, también en hexadecimal. En cuanto al retorno de carro, es válido tanto el de Windows como el de Macintosh.
- **Macromemoria:** `xxxx yyyy<CR>`. El formato es el mismo que en el caso anterior.

El fichero a cargar en la micromemoria lo produce la aplicación *MicroAss*, aunque también lo podemos escribir a mano si queremos. Por contra, el fichero a cargar en la macromemoria, lo obtendremos como resultado de guardar la macromemoria en la propia aplicación *Montador*, tras haber introducido el programa sobre la propia ventana, tecleando en hexadecimal el valor de cada palabra.

### Menú Inicializar

- **Nueva ventana micro.** Abre una nueva ventana de micromemoria, análoga a la que aparece inicialmente. El propósito de esta opción es el de poder ver simultáneamente dos regiones distintas de la memoria.

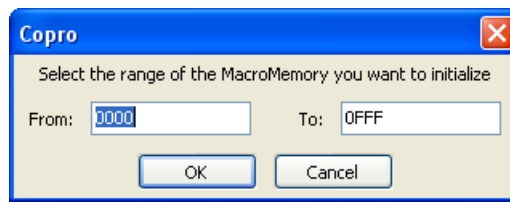


Figura 7: Rango a inicializar

- **Nueva ventana macro.** Hace lo mismo que la opción anterior, salvo que con la macromemoria.
- **Registros, Flags y Pila.** Pone a cero el valor de todos los registros, pone a falso todos los *flags* y deja la pila vacía.
- **Pila.** Vacía la pila.
- **Macro Memoria. . . .** Permite poner a cero el contenido de un rango de direcciones de la macromemoria. Al elegir esta opción aparece la ventana de la figura 7. Introducimos en esa ventana el rango de direcciones a poner a cero y presionamos Aceptar. Tras hacer esto, las direcciones especificadas habrán quedado a cero.
- **Actualizar Ventanas.** Actualiza el contenido de todas las ventanas. No debería ser necesario utilizar esta opción, ya que la aplicación actualiza las mismas automáticamente.

## Menú Monitor

- **Punto de ruptura.** Pone un punto de ruptura en la posición seleccionada de la micromemoria. Al hacer esto, cambiará el valor del bit *breakpoint* de la palabra correspondiente. Asimismo, la aplicación resaltará de color rojo dicha microinstrucción. Si la posición seleccionada ya tiene un punto de ruptura, entonces esta opción lo quita.
- **Saltar.** Permite cargar el campo  $\mu$ ADR con la dirección de la microinstrucción seleccionada. Asimismo, el *pipeline* quedará cargado con el contenido de esa dirección de memoria, que será la siguiente instrucción en ejecutarse.
- **Ejecución con puntos de ruptura.** Ejecuta el microprograma hasta encontrar un punto de ruptura. El usuario también puede detener la ejecución voluntariamente, pulsando cancelar en la ventana que aparece al ejecutar esta opción.



Figura 8: Barra de herramientas

- **Ejecución.** Ejecuta el microprograma hasta que el usuario detenga la ejecución.
- **Paso a paso.** Ejecuta una microinstrucción.

### Menú Idioma










Permite cambiar el idioma de la interfaz de usuario. Para que el cambio haga efecto es necesario reiniciar la aplicación.

### Menú Ayuda

- **Contenidos...** (F1). Abre la ayuda.
- **Acerca de...** Abre un diálogo con información acerca del programa.

## 1.3. La barra de herramientas

La figura 8 muestra la barra de herramientas de la aplicación. Proporciona acceso directo a algunas opciones de los menús. Estas opciones, por orden de aparición en la barra de herramientas, son:

Icono	Descripción
	Cargar la macromemoria
	Salvar la macromemoria
	Cargar la micromemoria
	Salvar la micromemoria
	Punto de ruptura
	Ejecución con puntos de ruptura
	Ejecución
	Ejecución de una microinstrucción
	Saltar a una microinstrucción



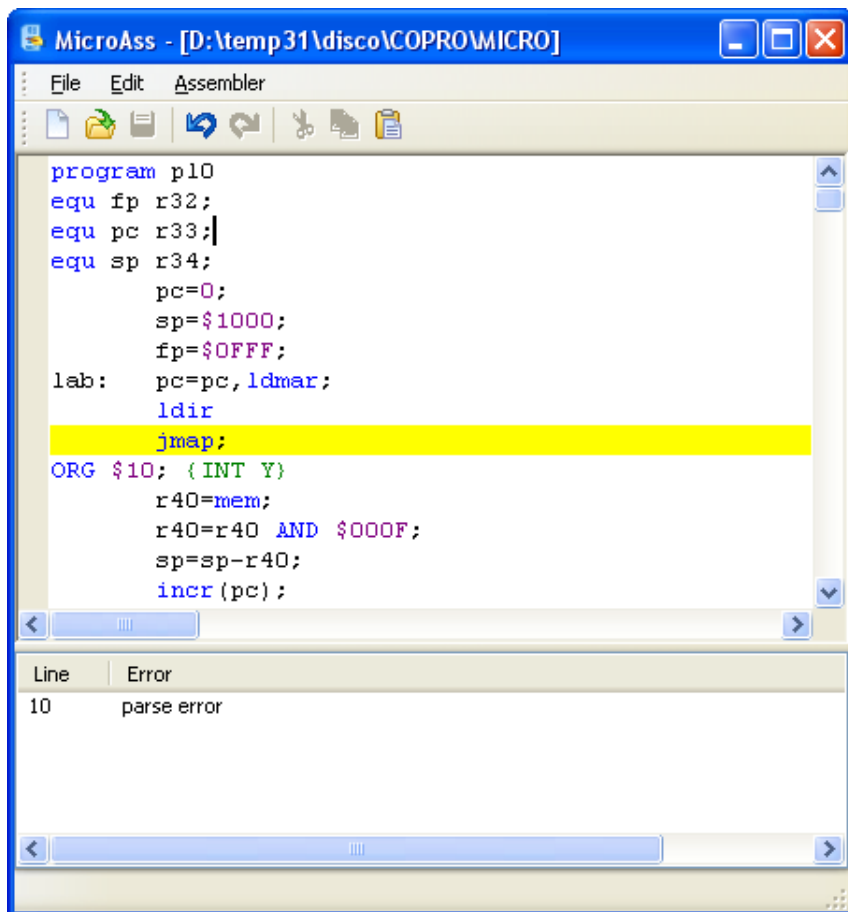


Figura 9: Aplicación *MicroAss*

## 2. MicroAss

El propósito de esta aplicación es el de actuar de compilador de microprogramas, proporcionando una interfaz de usuario agradable. Para ello cuenta con las opciones básicas que todo editor tiene (deshacer, rehacer, cortar, copiar, pegar...).

Esta aplicación hace uso del ejecutable “coproAsm.exe”, que es un compilador que funciona bajo línea de comandos.

La figura 9 muestra el aspecto de la aplicación tras compilar un fuente. En caso de que haya errores, el entorno realzará con color amarillo la línea en la que está el primero de ellos. Para resaltar cualquier otro error basta con hacer doble clic sobre el mensaje de error que aparece en el listado de la parte inferior de la ventana.

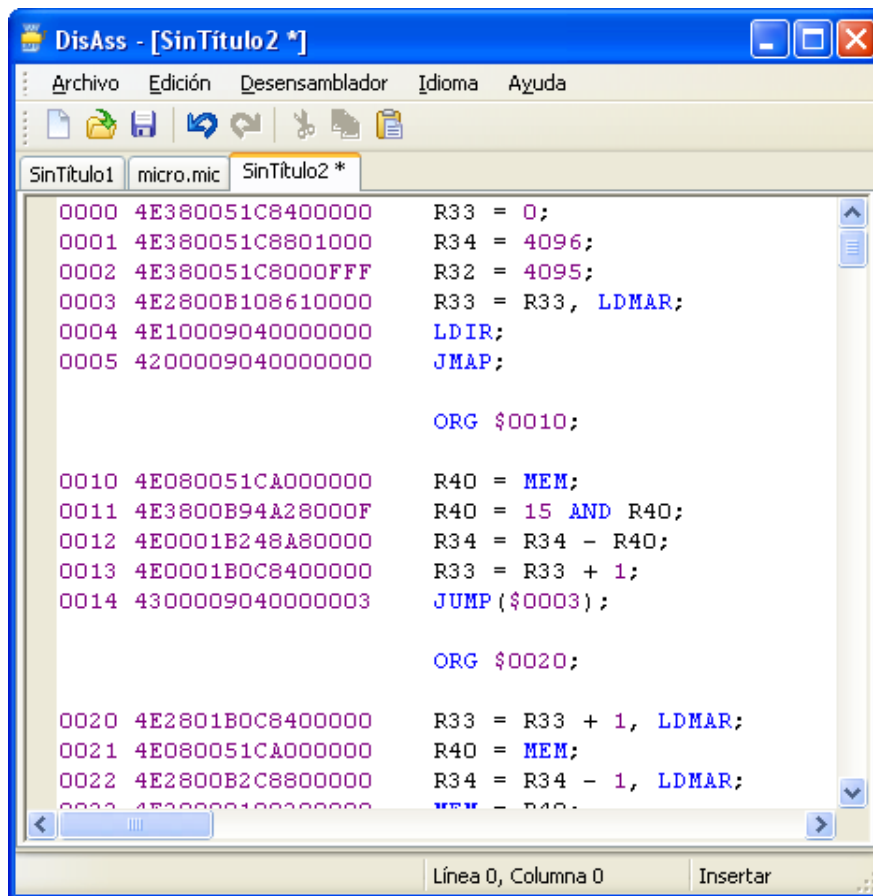


Figura 10: Aplicación *DisAss*

### 3. DisAss

Esta aplicación (figura 10) se encarga del trabajo contrario al que realiza *MicroAss*, es decir, se encarga de desensamblar programas. Para llevar a cabo dicha tarea dispone de un editor de texto que permite tener abiertos varios ficheros a la vez y acceder a ellos a través de solapas. Al igual que *MicroAss* cuenta con todas las opciones básicas de un editor: deshacer, rehacer, copiar, pegar...

Para desensamblar un programa basta con abrir un microprograma generado por la aplicación *MicroAss* y después ejecutar **Desensamblador** | **Desensamblar**. Al hacer esto, si no hay ningún error sintáctico en el código, se abrirá una nueva solapa con el microprograma desensamblado.