Manual de usuario del simulador BIRD

La arquitectura y el lenguaje máquina de los procesadores comerciales son, en general, demasiado complicados para que puedan ser utilizados con fines didácticos. Por ello, hemos desarrollado el computador BIRD y su lenguaje ensamblador TXORI, para tratar de aclarar de una forma sencilla los conceptos acerca de la arquitectura de computadores y la programación en bajo nivel estudiados a lo largo de este libro. Obviamente, BIRD no es un computador real, sino que se simula su funcionamiento mediante el simulador BIRD.

Este simulador toma como entrada un programa escrito en lenguaje ensamblador TXORI, realiza su análisis sintáctico y, en caso de no existir errores, simula la ejecución de ese programa, obteniendo una serie de datos relativos a la ejecución de ese programa en la máquina BIRD. Este manual de usuario tiene como objetivo guiar al usuario en el proceso de instalación y utilización del simulador BIRD.

1 Instalación del simulador BIRD

El simulador de la máquina BIRD se puede descargar en la siguiente página: **www.sc.ehu.es/acweb/BIRD.html**. En esta página el lector encontrará el propio simulador (*BIRD.zip*), este manual de usuario (*manual.pdf*) y el conjunto de programas que se han presentado como ejemplos a lo largo de este libro (*ejemplos.zip*).

Para la utilización del simulador BIRD es necesario un equipo PC con sistema operativo Windows, sin ninguna necesidad especial en cuanto a cantidad de memoria RAM o tamaño de disco duro. La instalación resulta sencilla, ya que únicamente hay que elegir la ubicación deseada en el disco duro y copiar en una nueva carpeta los ficheros que forman parte del simulador. En concreto, el entorno del simulador lo componen los siguientes ficheros:

- *bird.exe*: simulador BIRD estudiado en el libro.
- *errores.txt / erroreak.txt:* contienen el texto que el simulador ofrecerá por la pantalla en el caso de que exista algún error sintáctico en el tratamiento del fichero *.asm* por parte del analizador sintáctico. Como se puede ver, existen dos versiones de este fichero en función del idioma en el que se trabaje: castellano o euskara.

- *menucastellano.txt / menueuskara.txt:* contienen los literales correspondientes a todas las opciones del menú del simulador. Existen dos ficheros distintos en función del idioma en el que se utilice el simulador (castellano/euskara).
- *default.cfg:* es un fichero de configuración de opciones de simulación (direcciones de carga del programa y de los datos, así como tiempos de funcionamiento asociados a los distintos componentes de la arquitectura). Este fichero contiene unos valores por defecto, que podrán ser modificados y guardados utilizando, como se explicará en este manual, una opción del menú del simulador.

Es necesario que estos ficheros estén siempre en un mismo directorio, para que no se produzca ningún error en el uso del simulador.

Figura 1. Diálogo para el cambio de Idioma.

2 Manual de usuario

Tras iniciar la ejecución del simulador (ejecutable *bird.exe*), lo primero que tendremos que hacer es configurar el idioma de funcionamiento del simulador. La Figura 1 presenta el diálogo en el que se nos pide la selección del **Idioma**: *Euskara* o *Castellano*. Si seleccionamos Castellano, obtendremos el menú principal del simulador (Figura 2). A lo largo de la simulación, en cualquier momento podremos seleccionar el idioma en que queremos trabajar mediante la opción **Idioma** del menú principal.

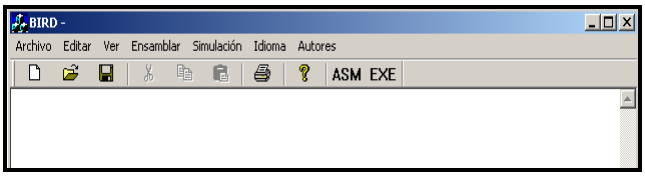


Figura 2. Menú principal del simulador (versión en castellano).

En este momento estamos listos para trabajar con el simulador. Distinguiremos tres partes importantes: (a) edición de un fichero en lenguaje ensamblador TXORI (*.asm*), (b) ensamblaje de este fichero, y (c) simulación de la ejecución del programa en la máquina BIRD. Veamos cada una de estas partes por separado, siguiendo un ejemplo. En concreto, vamos a simular el programa que suma los elementos de dos vectores A y B elemento a elemento, dejando el resultado en su correspondiente elemento de otro vector C. Este ejemplo ya ha sido desarrollado previamente en el Capítulo 3.

2.1 Edición de un programa *asm*

El primer paso para simular la ejecución de un programa en lenguaje ensamblador en la máquina BIRD es editar un fichero *.asm* siguiendo la sintaxis del lenguaje ensamblador TXORI. Es importante seguir exactamente la sintaxis descrita en el Anexo 1 para el lenguaje ensamblador TXORI, de lo contrario el ensamblaje no será correcto.

Para editar un fichero con el programa que se desea simular, podemos recurrir a cualquier editor de texto y generar un fichero ASCII con la extensión *.asm*. También podemos utilizar el entorno de edición que proporciona el simulador. Para ello, utilizaremos los submenús de las opciones **Archivo**, **Editar** y **Ver** del menú principal. Estas opciones no difieren de las encontradas en otros editores sencillos que proporciona un equipo Windows. Presentan todos los comandos necesarios para el trabajo con ficheros de texto:

abrir, guardar y cerrar un fichero, copiar, cortar o pegar un texto seleccionado, mostrar y ocultar barras de herramientas, etc. En concreto, la Figura 3 presenta el ejemplo que estamos trabajando.

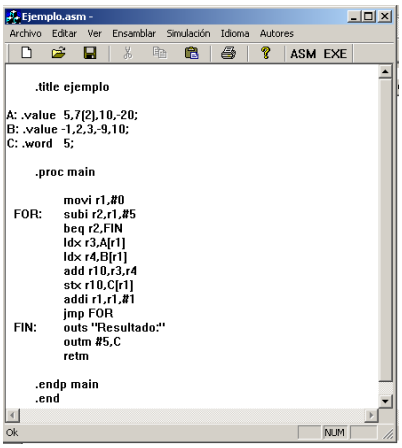


Figura 3. Programa ejemplo.

Es importante repasar algunas restricciones del lenguaje ensamblador TXORI a la hora de poner el título del programa o a la hora de nombrar las variables. En todos los casos, sólo se admiten caracteres numéricos y letras; no están permitidos, por ejemplo, los signos de puntuación como nombre del programa o de las variables. Cualquier otro carácter dará como resultado del proceso de ensamblaje un mensaje de error. Por ejemplo, los siguientes nombres de variables serían erróneos: *VAR_1; VAR.1; VAR@1; VAR1+2*. Además, el nombre del programa, que acompaña a la directiva *.title* no puede empezar por un dígito (por ejemplo, no se puede utilizar *5EJEMPLO* como nombre de programa). Así mismo, en los strings asociados a instrucciones como *OUTS* los únicos caracteres que se permiten fuera de los anteriormente mencionados son el espacio en blanco y el carácter ':'. Finalmente, también hay que tener en cuenta que el ensamblador distingue entre letras minúsculas y mayúsculas en los nombres de las variables (por ejemplo, *VAR1* y *var1* serían dos variables distintas).

Una vez editado el programa, lo guardaremos en el disco y pasaremos a la siguiente opción dentro del entorno del simulador: **Ensamblar**.

2.2 Ensamblaje de un programa

Para ensamblar el programa escrito en lenguaje ensamblador TXORI tenemos la opción **Ensamblar** del menú principal (ver Figura 4). También podemos realizar en ensamblaje del programa pulsando el botón **ASM** existente en la barra de herramientas de la pantalla principal del simulador. El simulador desplegará una ventana de diálogo en la que se nos pide el nombre del fichero que se quiere ensamblar. Para facilitar la tarea de selección de este fichero, se muestran únicamente los ficheros con la extensión correcta (*.asm*).

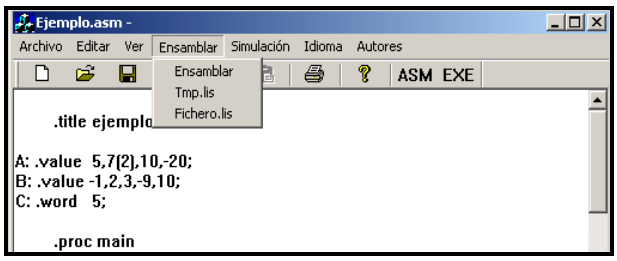
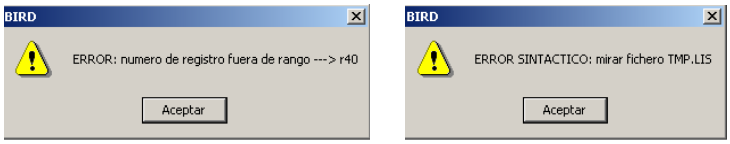


Figura 4. Opciones de ensamblaje dentro del menú **Ensamblar**.

Como ya se ha explicado en el Capítulo 4, debido a la posible existencia de referencias adelantadas, el proceso de ensamblaje se realiza en dos fases. En la primera se examina la sintaxis del programa en ensamblador (fichero *.asm*). Si ésta es correcta, la primera fase finaliza y comienza la segunda. Si, por el contrario, existe algún error sintáctico, el simulador nos ofrecerá un mensaje de error indicándonos tal circunstancia (ver Figura 5). Este mensaje de error puede indicar de forma clara la fuente de error (ver Figura 5a), o bien, hacer referencia de forma genérica a la necesidad de examinar el fichero *tmp.lis* (ver Figura 5b). Este fichero nos va a

servir para depurar a nivel sintáctico el código escrito en la fase de edición: habrá que observar la última línea del fichero ya que en este punto se ha detenido el proceso de ensamblaje al detectar algún error en la siguiente línea de programa. Para visualizar de forma cómoda el fichero *tmp.lis* podemos utilizar la opción **Tmp.lis** de submenú correspondiente a la opción **Ensamblar**.



(a) (b)

Figura 5. Ejemplo de mensajes de error en la fase de ensamblaje.

Si este primer paso de ensamblaje es correcto, el fichero *tmp.lis* muestra el código ensamblado a excepción de las etiquetas de salto. En la segunda fase de ensamblaje, fundamentalmente, se resuelven los saltos comprobando la concordancia del uso de las etiquetas de salto con su definición. En el momento en que no existan errores en estas fases, el simulador muestra un mensaje confirmando que la etapa de ensamblaje ha sido correcta. También se obtiene como resultado el fichero de ensamblaje final, que contiene el código binario del programa origen. La nomenclatura de este fichero es *fichero.lis* (se mantiene el nombre del fichero origen en ensamblador y únicamente se cambia la extensión de *.asm* a *.lis*). Para visualizar este fichero puede utilizarse la opción **Fichero.lis** del submenú correspondiente a la opción **Ensamblar** del menú principal. Así mismo, genera un fichero con la extensión *.bin*, que pretende simular el código ejecutable generado en tiempo de ensamblaje. La fase de simulación requiere la existencia de este fichero. En este momento estamos en condiciones de pasar a la fase de simulación del programa.

2.3 Simulación de un programa

Esta es sin duda la parte más importante del entorno del simulador. Permite la simulación en la máquina didáctica BIRD de los efectos que tiene en la arquitectura de la máquina (memoria, banco de registros, PC,

IR, pila, etc.) la ejecución de cada una de las instrucciones del programa, analizando el mecanismo de ejecución de las instrucciones definidas a partir del lenguaje ensamblador TXORI. También permite obtener una serie de estadísticas, entre las que destacamos las relativas al tiempo de ejecución de cada una de las instrucciones del programa, a partir de los parámetros definidos previamente (fundamentalmente, tiempos de ejecución) para cada uno de los componentes de la máquina.

Para comenzar la simulación es necesario haber ensamblado correctamente un programa *.asm*. Una vez hecho esto, seleccionamos la opción **Simulación** en el menú principal de la aplicación o pulsamos el botón **EXE** de la barra de herramientas, con lo que obtendremos el diálogo de simulación. A continuación habrá que seleccionar un fichero con extensión *.bin*, tras lo que comenzará la simulación de ese programa.

Veamos a continuación los pasos a seguir para la simulación del programa, los resultados obtenidos, así como la forma de definir los parámetros de la máquina didáctica.

2.3.1 Definición de los parámetros de BIRD

La simulación del programa se realiza en un entorno previamente definido para la máquina virtual: tiempos de ejecución y direcciones de comienzo en memoria para los datos y las instrucciones. Para acceder a la definición de estos parámetros deberemos seleccionar la opción **Parámetros** de este diálogo de simulación (ver Figura 6), y dentro de él seleccionaremos su única opción, **Máquina**. Obtendremos un submenú compuesto por las opciones **Tiempos de Ejecución**, **Direcciones de Inicio**, **Guardar Descripción** y **Cargar Descripción**. Veamos a continuación la descripción de estas opciones.

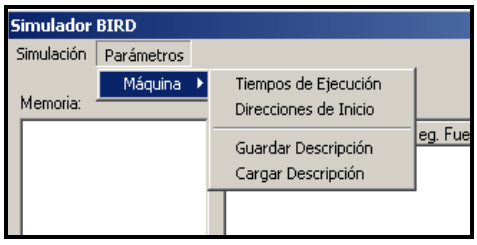


Figura 6. Opciones para la definición de parámetros en la máquina BIRD.

En lo referente a la definición de tiempos de ejecución, **Tiempos de Ejecución**, vemos que aparece un diálogo secundario de definición de tiempos asociados a los componentes del computador, con el objetivo de calcular el tiempo asociado a la ejecución de cada una de las instrucciones del programa (ver Figura 7). Se considera que los tiempos están medidos en nanosegundos. De esta manera se definen los tiempos de acceso a memoria, para lectura y escritura, tiempos de acceso al banco de registros, tiempos para la etapa de descodificación y tiempos de ejecución en la ALU dependientes de la operación a realizar: suma/resta, multiplicación, división, operaciones lógicas (and, or, etc.), desplazamiento y comparación.

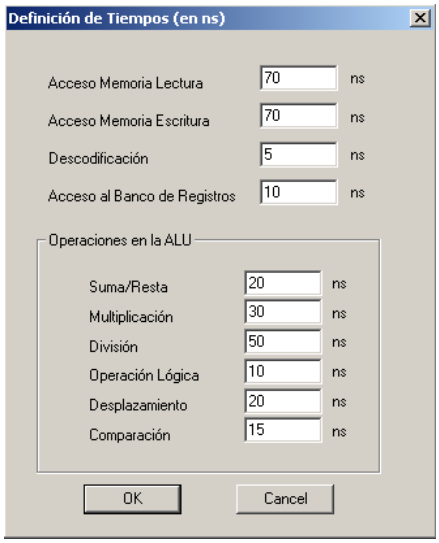


Figura 7. Diálogo de definición de tiempos de ejecución.

En cuanto a la definición de las direcciones de comienzo de los bloques de datos y de instrucciones del programa, ésta se puede realizar seleccionando la opción **Direcciones de Inicio**. Se pretende simular el funcionamiento del programa cargador del sistema operativo, que localiza una zona libre de memoria para cargar un programa. En nuestro caso, únicamente se definen dos bloques: datos e instrucciones. El diálogo del la Figura 8 permite la introducción en la máquina virtual de estas direcciones de comienzo. El simulador comprobará la validez de los datos introducidos (los valores no pueden ser negativos, el espacio de

direcciones para datos e instrucciones no debe solaparse ni deben superar el tamaño máximo de memoria [0, 65535], etc.), mostrando un mensaje de error en el caso de existir alguna anomalía.

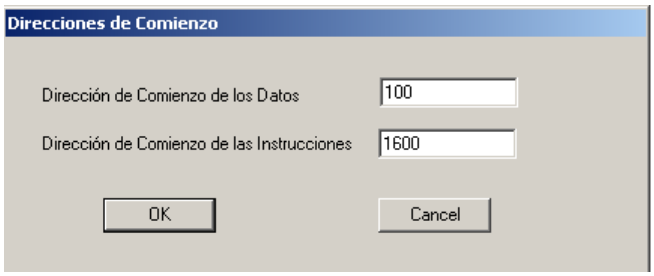


Figura 8. Diálogo de definición de direcciones de comienzo del programa.

Las dos últimas opciones de este submenú, **Guardar Descripción** y **Cargar Descripción**, permiten guardar en un fichero y cargar desde un fichero los parámetros definidos para la máquina virtual. De esta manera, podremos tener diferentes versiones de la misma máquina virtual en cuanto a direcciones de comienzo y tiempos de ejecución se refiere. Mediante al opción **Guardar Descripción**, como su nombre indica, podremos especificar un fichero con extensión *.cfg* para guardar los parámetros actuales. Mediante la opción **Cargar Descripción** podremos restaurar los parámetros de descripción de una anterior configuración desde un fichero *.cfg*. Por defecto, el simulador mantiene una configuración base en el fichero *default.cfg*, que siempre es la última que se ha utilizado en las simulaciones realizadas.

2.3.2 Componentes de la pantalla de simulación

Una vez definidos los parámetros de la simulación se puede pasar a simular la ejecución del programa en la máquina virtual. El menú de simulación en este diálogo, **Simulación**, permite dos tipos de simulaciones posibles: **Paso a Paso** y **Completa** (ver Figura 9). En cualquier caso, estas opciones piden al usuario la introducción, a través del correspondiente diálogo, del fichero binario (extensión *.bin*) que se desea simular. En la simulación paso a paso, **Paso a Paso**, el programa se ejecuta instrucción por instrucción, visualizando en este diálogo los

efectos en la ejecución de cada una de las instrucciones una a una. Para avanzar en la ejecución de cada instrucción, se pedirá al usuario que pulse sobre un botón que al efecto aparecerá en la parte superior (**Siguiente Paso**). En este tipo de simulación, el botón **Detener Simulación** permite abortar la simulación en cualquier instante a petición del usuario. Una vez seleccionada la simulación paso a paso, no se puede simular otro programa hasta que el que está en ejecución no finalice o sea abortado. La opción **Paso a Paso** tiene como objetivo poder depurar cualquier error lógico en el programa de una manera más sencilla.

Por el contrario, si se utiliza la opción **Completa** no existe ninguna interacción con el usuario en la ejecución de cada instrucción: éstas se ejecutan de forma continua hasta la finalización del programa. No obstante, en ese momento podremos ver los efectos de la ejecución de cada una de las instrucciones.

Figura 9. Opciones del menú de simulación.

Una vez elegida cualquiera de las opciones y el correspondiente fichero binario, comienza la simulación de la ejecución del programa en la máquina BIRD. La Figura 10 muestra el diálogo principal del simulador para el ejemplo que estamos tratando en este manual. Veamos a continuación la descripción de cada uno de los componentes de este diálogo. El componente etiquetado como "A" es una lista que va a contener el vuelco de memoria correspondiente a las direcciones del programa: direcciones de los datos definidos en el programa, direcciones de los strings y direcciones correspondientes a las instrucciones del programa. En el caso de los datos, el contenido de cada posición de memoria se ofrece en decimal. En el caso de los strings y de las instrucciones, el contenido de cada posición de memoria se muestra en hexadecimal (4 dígitos). Al inicializar la ejecución del programa, los

datos definidos como *.word* se inicializan de forma aleatoria. Finalmente, hay que recordar que, de acuerdo a las características de la máquina BIRD, los datos ocupan una única posición de memoria, mientras que las instrucciones se almacenan en dos posiciones consecutivas de memoria.

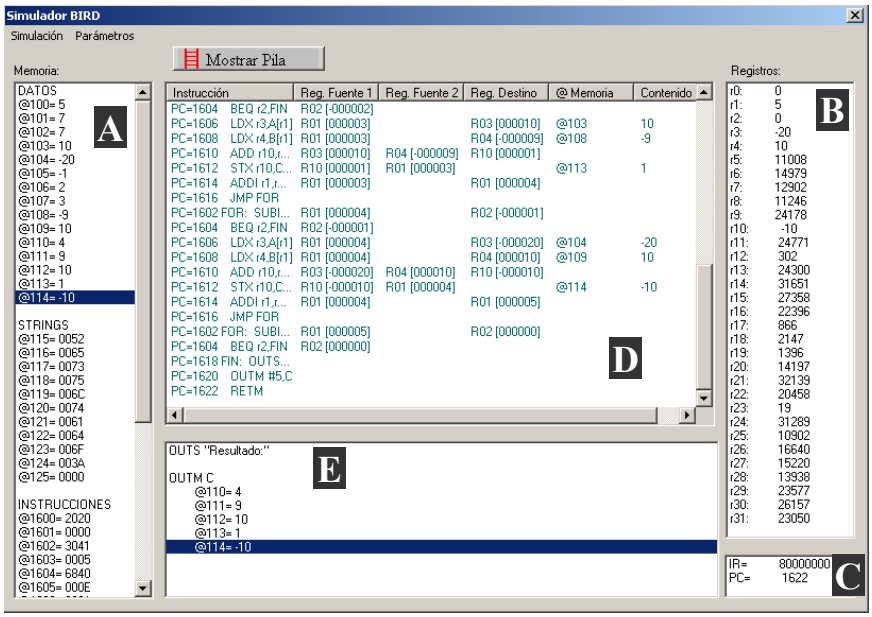


Figura 10. Diálogo principal del simulador.

Si durante la simulación se pulsa el botón **Mostrar Pila**, en este misma lista desplegable se mostrarán las posiciones de memoria correspondientes a la pila junto con su contenido, para que el usuario pueda visualizar el estado de la pila de una forma más directa, aislada del resto de posiciones de memoria. Este botón sólo se habilitará en el caso en que el programa inicialice el puntero de pila (*sp* o *r31*). En el caso en que el puntero de pila no haya sido inicializado se muestra un mensaje de error indicando esta circunstancia. En el caso de visualizar el estado de la pila, el botón **Ocultar Pila** oculta la ventana de la pila y volveremos otra vez a ver el contenido de toda la memoria.

Así mismo, en la ejecución paso a paso se resaltará de forma automática la dirección de memoria involucrada en la ejecución de la instrucción en caso de que esta instrucción acceda a memoria para leer o escribir un dato.

La lista etiquetada como "B" representa el banco de registros de la máquina. BIRD tiene 32 registros generales de 16 bits. En esta lista podremos ver la situación del banco de registros conforme se vayan ejecutando las instrucciones. El formato de los elementos en esta lista es el siguiente: nombre de registro (*rxx*, siendo *xx* un valor en el rango [0, 31]) y contenido del registro, expresado este valor en decimal. Al comienzo de la ejecución del programa, todos los registros se inicializan con valores aleatorios, excepto el registro *r0* que como se sabe siempre almacena el valor 0.

Conforme se va ejecutando el programa paso a paso, cuando una instrucción hace referencia a alguno de los registros de la máquina, al igual que lo que ocurría en el acceso a memoria, estos registros se resaltan de forma automática en el banco de registros con el objetivo de identificar de una forma más rápida los registros involucrados en la ejecución de esa instrucción. En el caso en que la instrucción modifique algún registro, esta modificación se verá en el contenido del registro en este desplegable.

Para completar la información del conjunto de registros de la máquina, la ventana etiquetada como "C" presenta el contenido de los dos registros específicos de BIRD: PC (*Program Counter*) e IR (*Instruction Register*). El registro PC contiene la dirección de la instrucción que se está ejecutando en este momento. Se trata de un registro de 16 bits cuyo contenido, una dirección de memoria, se muestra en decimal. Por su parte, el registro IR contiene la instrucción que está en ejecución. Se trata, por tanto, de un registro de 32 bits cuyo contenido se muestra en hexadecimal.

Siguiendo con la descripción de componentes, la ventana principal de simulación aparece en la Figura 10 etiquetada como "D". Es aquí donde se muestra la información referente a la ejecución de las instrucciones del programa. Esta información se divide en seis columnas. La columna con título "*Instrucción"* indica la instrucción en lenguaje ensamblador que se está ejecutando. Esta columna va precedida por el valor del PC, en decimal, de la instrucción. Las tres siguientes columnas, etiquetadas como *"Reg. Fuente 1"*, *"Reg. Fuente 2"* y *"Reg. Destino"* contienen los valores de los registros a los que hace referencia la instrucción en caso de que ésta tenga alguno de sus operandos en registros (sea fuente o destino); en caso contrario, aparece un hueco en blanco. El formato de estas columnas es el siguiente: nombre del registro y contenido del

registro en decimal, formateando el valor en un campo de 6 dígitos más el signo. Finalmente, las dos últimas columnas hacen referencia a la posición de memoria accedida por la instrucción y a su contenido, en el caso en que la instrucción acceda a algún operando en memoria. La columna etiquetada como *"@ Memoria"* especifica la dirección accedida, mientras que la columna *"Contenido"* hace referencia al contenido de esta posición de memoria después de la ejecución de la instrucción (caso de que la instrucción sea una escritura en memoria). Este valor está expresado en decimal.

Finalmente, el último componente de la ventana principal de simulación es la ventana de entrada/salida del simulador, etiquetada como "E" en la Figura 10. Como ya se ha explicado, el simulador BIRD no tiene instrucciones de entrada/salida. Sin embargo, sí permite la introducción de valores numéricos como datos de entrada en el programa, así como la visualización de unos resultados mínimos. Para ello, proporciona al usuario las instrucciones IN, OUT, OUTS, OUTM y OUTRM, cuyo funcionamiento ya ha sido explicado a lo largo de este libro. Los efectos de estas instrucciones de E/S se reflejan en esta ventana del simulador. Con el objetivo de que el usuario se percate de que se está ejecutando una de estas instrucciones y pueda ver los datos visualizados, cada vez que se ejecute una de estas instrucciones la simulación se detiene hasta que el usuario pulse sobre el botón **Continuar** que aparecerá en la ventana de entrada/salida. Una vez pulsado, la simulación continua normalmente. En el caso de la instrucción IN, aparecerá una caja de diálogo (ver Figura 11) que requiere la introducción de un valor numérico. Este valor debe ser correcto, es decir, debe mantenerse en el rango de representación de BIRD: 16 bits en C2, [-32768, 32767]. En caso contrario, un mensaje nos avisará del error cometido.

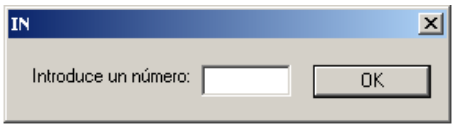


Figura 11. Diálogo correspondiente a la instrucción IN.

También hay que comentar que si la ejecución del programa genera alguna situación de error (por ejemplo, *overflow*, división por cero, etc.)

aparecerá un mensaje de error indicándonos tal circunstancia y la simulación abortará. En caso de que la simulación finalice sin ningún problema, el simulador nos lo indicará con el mensaje *"Simulación completada con EXITO"*. Al finalizar la ejecución es posible visualizar cualquiera de las ventanas de simulación desde el comienzo gracias a las barras de *scroll* que permiten desplazarse a lo largo de toda la ventana.

2.3.3 Trazas y estadísticas

Aparte de la traza de ejecución presentada en el apartado anterior, el simulador obtiene un conjunto de estadísticas que presentaremos en este apartado. En concreto, el simulador obtiene una traza con las direcciones de memoria a las que se ha accedido en la ejecución del programa y un conjunto de estadísticas de tiempos de ejecución y frecuencia de instrucciones que pueden ser obtenidas seleccionando la opción **Estadísticas** en el menú principal de la ventana de simulación (ver Figura 12). Obtendremos un submenú con tres opciones: **Tiempos de Ejecución**, para visualizar estadísticas relacionadas con el tiempo de ejecución de las instrucciones; **Traza de Memoria**, traza con las direcciones de memoria; y **Frecuencia de Instrucciones**, estadísticas de frecuencia de instrucciones, tanto dinámica como estática.

La opción **Traza de Memoria** presenta un diálogo como el de la Figura 13 en el que se pueden observar los accesos que se han realizado a la memoria del computador durante la ejecución del programa. Estos accesos pueden ser de dos tipos: (a) acceso para lectura de una instrucción (formato *I* @), lo que conlleva la lectura de dos posiciones consecutivas de memoria; y (b) acceso para lectura/escritura de un dato (formato *D L* @ para lectura o *D E* @ para escritura), que implica un único acceso.

Figura 12. Opciones del menú de estadísticas del simulador.

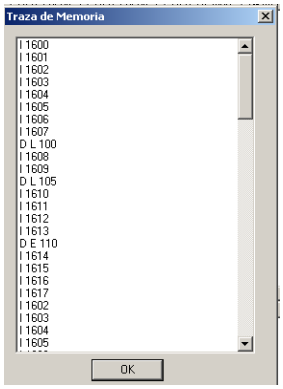


Figura 13. Diálogo de la traza de memoria.

La opción **Tiempos de Ejecución** obtiene las estadísticas de tiempo de ejecución tanto en su versión detallada (opción **Detallada**) como en su versión resumida (opción **Resumida**). En el primer caso, obtendremos una tabla como la de la Figura 14, en el que se muestran los tiempos de ejecución asociados a cada una de las fases de ejecución de las instrucciones, tal y como se ha descrito en los correspondientes capítulos del libro. El usuario puede desplazarse por la tabla de estadísticas hasta encontrar la información de su interés utilizando los botones de navegación localizados en la parte derecha del diálogo y cuyos iconos son autoexplicativos.

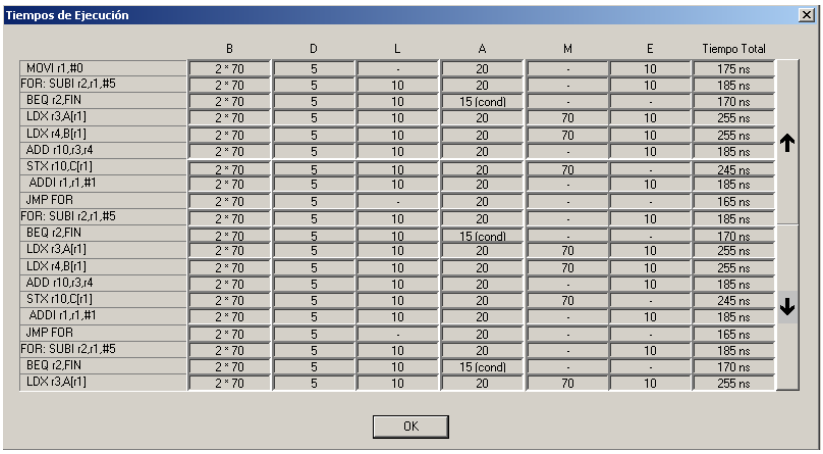


Figura 14. Diálogo de tiempos de ejecución detallados.

En el segundo caso, con el objetivo de tener una vista global de la ejecución de todas las instrucciones del programa, la opción resumida muestra las estadísticas de tiempo de ejecución por fases para el global de las instrucciones (ver Figura 15).

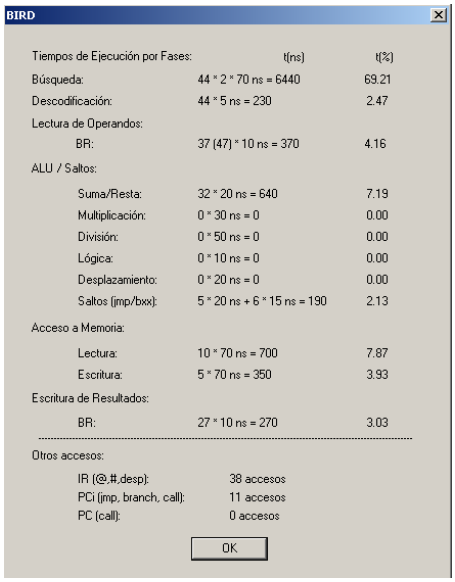


Figura 15. Diálogo de tiempos de ejecución resumidos.

En este caso, por cada fase de ejecución se muestra el número de instrucciones que pasan por dicha fase junto con el tiempo empleado en su ejecución. De esa forma, se obtiene el tiempo total de ejecución para esa fase teniendo en cuenta todas las instrucciones del programa, junto con un porcentaje que indica la relación entre el tiempo de ejecución en esa fase frente al tiempo de ejecución total del programa.

Hay que señalar que el cálculo de estos tiempos se realiza teniendo en cuenta los tiempos que se han asociado al acceso de cada uno de los componentes en la definición de los parámetros de la máquina virtual, tal y como se ha explicado en el apartado 2.3.1 de este apéndice. Cabe destacar la no inclusión de los tiempos de las instrucciones de entrada/salida (IN, OUT, etc.) dado su carácter meramente auxiliar para la introducción o visualización de resultados, tal y como ya se ha comentado.

La opción **Frecuencia de Instruccione**s permite obtener información relativa a la frecuencia de aparición de una instrucción en el programa (opción **Estática**) o al número de veces que se ejecuta una instrucción (opción **Dinámica**). Vamos a describir este tipo de información conjuntamente, ya que el formato de presentación de resultados es el mismo aunque su interpretación dependa del tipo de estadística. Al igual que en el caso anterior, aquí también es posible obtener estadísticas detalladas (opción **Detallada**) o resumidas (opción **Resumida**). En el caso de estadísticas detalladas, Figura 16, se presentan los datos, frecuencia y porcentaje sobre el total, para todas las instrucciones del juego de instrucciones de la máquina BIRD.

En el caso de estadísticas resumidas, Figura 17, las instrucciones se agrupan en función del tipo de instrucción y se obtienen las estadísticas relativas a estos grupos definidos. En concreto, se definen los siguientes grupos de instrucciones: aritmético-lógicas, de acceso a memoria, saltos, tratamiento de subrutinas (llamada/retorno), entrada/salida y la instrucción de fin de programa. Algunos de los tipos, como el acceso a memoria, se subdividen para poder hacer un estudio con mayor profundidad. Estas estadísticas sí tienen en cuenta las instrucciones de entrada/salida definidas en la máquina BIRD (IN, OUT, etc.).

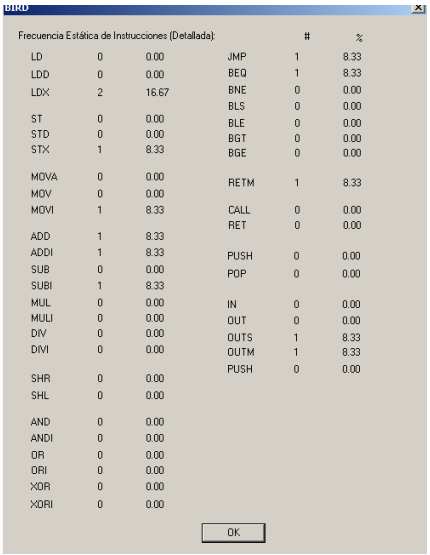


Figura 16. Diálogo de frecuencia estática de instrucciones detalladas.

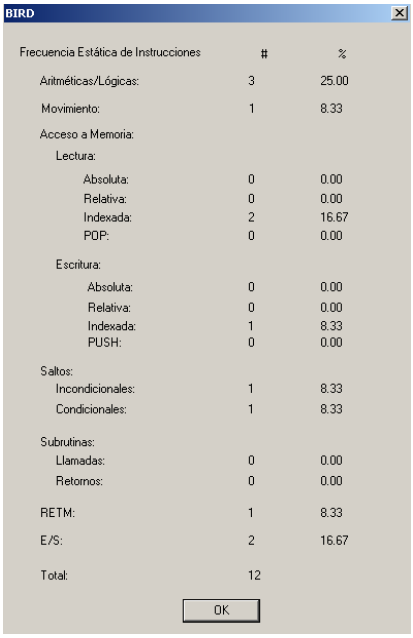


Figura 17. Diálogo de frecuencia estática de instrucciones resumidas.

2.3.4 Guardar la simulación

El simulador también ofrece la posibilidad de almacenar todos los datos generados en la simulación en varios ficheros con distintas extensiones en función de la información almacenada. Para guardar la simulación, dentro del menú **Simulación** de la ventana principal tenemos la opción **Guardar Simulación**. Por ejemplo, si el nombre de fichero a simular es *ejemplo.bin* (generado tras ensamblar el fichero *ejemplo.asm*), los ficheros que se generan son los siguientes:

- *ejemplo.pro*. Contiene la simulación en sí, es decir, la traza de instrucciones ejecutadas. Esta información es la misma que se puede visualizar en la ventana de traza del simulador (parte "D" de la Figura 10) cuyo contenido ya ha sido explicado. La extensión de este fichero (*.pro*) recuerda la nomenclatura habitual para este tipo de ficheros de traza en los computadores comerciales (ficheros de *profiling*).

- *ejemplo.mem*. Contiene la información de los accesos a memoria representada tal y como se ha descrito en la Figura 13.
- *ejemplo.est*. Este fichero contiene las estadísticas de frecuencia de instrucciones del programa simulado, tanto dinámicas como estáticas. Se almacenan tanto la versión resumida como la versión detallada de estas estadísticas (ver Figuras 16 y 17).
- *ejemplo.tim*. Finalmente, este fichero almacena las estadísticas relativas a tiempo de ejecución (ver Figura 14).

2.3.5 Eliminación de los ficheros de la simulación

Además de estos ficheros, tal y como se ha explicado, en la fase de ensamblaje se generan tres ficheros más: dos de ellos con el listado de ensamblaje resultante de cada una de las dos fases del proceso de ensamblaje (ficheros *tmp.lis* y *ejemplo.lis* en nuestro caso de ejemplo) y el tercero con el código binario que utiliza el simulador, *ejemplo.bin*. Como se puede ver el simulador genera varios ficheros por cada uno de los programas que se simulan. Para eliminar estos ficheros de forma automática se adjunta un fichero de comandos, *limpia.bat*, que elimina de forma automática todos los ficheros asociados al entorno de simulación, salvo los ficheros fuente *.asm*.

ATENCIÓN: En caso de ejecutar los comandos de este fichero, hay que tener cuidado de ejecutarlo sólo en el directorio donde está instalado el simulador, para no borrar por equivocación otros ficheros de usuario de forma accidental.