Contenido

1 Introducción

SiCoMe es un simulador de la arquitectura mejorada de una computadora sencilla, que diseñado para un uso docente. Proporciona los mecanismos necesarios para que el usuario comprenda como funciona una computadora por dentro.

SiCoMe es un sistema interactivo que permite al usuario programar el simulador con repertorios de instrucciones y programas elaborados por él. El usuario podrá, una vez programado el simulador, ejecutar los programas ciclo a ciclo, instrucción a instrucción o de manera continuada, facilitando así la observación del comportamiento de la computadora.

Este manual intenta explicar con el máximo detalle posible los fundamentos teóricos en los que se basa **SiCoMe**, su entorno de trabajo, su programación, etc., en definitiva todo lo que el usuario necesita saber sobre esta aplicación

2 Instalación

2.1 Requisitos software

El sistema **SiCoMe** está basado en la arquitectua *Java®* de *Sun Microsystems*. Debido a esto, el sistema donde se desee montar la aplicación debe poseer instalada la llamada *máquina virtual Java* o *Java Runtime Environment*. En concreto, la versión requerida de este software es la *J2RE 1.3.0*. Esta versión del *JRE* está disponible de forma gratuita en la sección de descargas del web oficial de Java de *Sun Microsystems* que es http://java.sun.com. Desde este sitio debe bajarse e instalarse el software en la máquina para posteriormente realizar la instalación del servidor.

SiCoMe ha sido realizado para ser soportado en cualquier plataforma donde pueda ser instalada la máquina virtual Java. Así pues, la instalación puede realizarse bajo Unix, bajo Linux o bajo Windows, por poner algunos ejemplos. Basta con descargar e instalar la versión de *JRE* pertinente al sistema operativo y después instalar **SiCoMe**. Practicamente no existe ningún otro requisito software además de éste.

2.2 Requisitos hardware

Se hablará aquí de los requisitos hardware necesarios para un funcionamiento satisfactorio del simulador de la computadora mejorada cuando éste es instalado en un sistema Linux o Windows:

- Procesador Intel Pentium II/AMD K-6II a 350 Mhz.
- 64 MB de memoria RAM. Se recomiendan 128 MB.
- Conexión a internet.

Se han realizado pruebas del sistema en un equipo de dichas características con resultados satisfactorios. El último requisito citado es inherente a la naturaleza de la

aplicación, pues qué sentido tiene pretender ser servidor de ficheros si no se posee conexión alguna con la Internet.

2.3 Instalación del software

A partir de este momento se va a suponer que el equipo donde se desea instalar **SiCoMe** cumple los requisitos hardware mínimos y tiene instalado *JRE 1.3.0*. Si el computador posee unidad de CD-ROM, puede elegir dicho soporte para la instalación. De lo contrario sírvase a obtener la versión en diskettes.

2.3.1 Instalación bajo Windows

El fichero que contiene la aplicación se denomina *sicome.zip*. Al abrir el fichero con el descompresor adecuado, se puede apreciar que en su interior están todos los directorios que forman el programa. Hay que descomprimirlo todo en un directorio de destino. Por consonancia con los demás programas de Windows, se recomiendo descomprimirlo en:

C:\Archivos de programa\sicome

Una vez hecho esto, bastará con ejecutar el fichero de procesamiento por lotes que se incluye en la aplicación: *sicome.bat*. Este fichero contiene la línea de comandos completa que habría que escribir cada vez que se quisiera arrancar el simulador. Como esto sería muy engorroso, se ha insertado en un fichero .bat para que baste un simple *click* de ratón. A partir de aquí se deja a la voluntad del usuario el realizar un acceso directo en el escritorio o en el menú de *Inicio*.

Sin embargo, haciendo sólo esto la aplicación no está lista para funcionar. En el directorio donde se ha instalado la aplicación existe una carpeta llamada *security*. En dicha carpeta hay dos ficheros llamados *java.policy* y *java.security*. Estos ficheros son ficheros de configuración de java. Necesita copiar estos ficheros en el directorio pertinente dentro de la carpeta donde instaló *JRE*. Este directorio puede cambiar, pero por defecto es el siguiente:

C:\Archivos de programa\JavaSoft\JRE\1.3\lib\security

Solamente debe copiar los ficheros mencionados a esta carpeta, sobreescribiendo los existentes. Una vez hecho esto, ya podrá ejecutar sin problemas el simulador.

Nota: Si usted ha modificado estos ficheros para que funcione correctamente algún otro programa basado en Java, entonces no sobreescriba los ficheros, edítelos e incluya las líneas de los ficheros suministrados que no aparezcan en los originales.

2.3.2 Instalación bajo Linux

La instalación del producto para el sistem operativo Linux no se diferencia mucho de la descrita para el sistema Windows. En este caso el fichero que habra que elegir como fichero de instalación será el denominado *sicome.tgz*. Este fichero se descomprime usando los comando *tar* y *gunzip* y se obtiene el mismo contenido comentado en la sección anterior. Este contenido se situa en el directorio deseado. Dentro de ese directorio estará situado un fichero de *shell* que contendrá la línea de comando completa para ejecutar la aplicación, de manera análoga a l fichero .bat de Windows.

El usuario que desee arrancar el programa podrá hacer accediendo al directorio de instalación y ejecutan el fichero de shell o bien creando un enlace simbólico a dicho fichero en su directorio de usuario, pero esto se deja ya a voluntad del usuario.

La operación de sobreescritura/modificación de los ficheros de política y seguridad del *JRE* se realiza de la misma manera descrita en la sección anterior. Igualmente la nota anterior también es extensible a esta situación.

3 La computadora

En este capítulo se explica la arquitectura simulada y su microprogramación. La lectura y comprensión de este capítulo es fundamental a la hora de poder usar el simulador ya que en él se explica la base teórica de éste.

3.1 Esquema de la arquitectura de la Computadora Mejorada

En la figura siguiente podemos observar el esquema de la arquitectura de la computadora mejorada que se va a simular con este proyecto. Como se puede observar, ésta trabaja con datos de 16 bits, de los cuales 5 son para codificar operaciones, y 11 para direccionar. A continuación se procederá a la descripción de cada uno de los componentes de la arquitectura.

3.1.1 Memoria

En este caso, la unidad de memoria puede almacenar hasta 2048 palabras de 16 bits. La única microoperación que soporta es **GPR → M**, la cual transfiere el contenido del GPR a la dirección de memoria señalada por el MAR.

3.1.2 PC (Program Counter)

El registro **PC** es el *contador de programa*, su función es indicar la dirección de memoria donde se encuentra la próxima instrucción a ejecutar. Su tamaño es de **11 bits**. Las microoperaciones que se puede realizar sobre él son las siguientes:

- **PC+1 → PC.** Se incrementa en 1 el contenido del PC.

- **GPR → PC**. Se transfiere el contenido del GPR al PC.

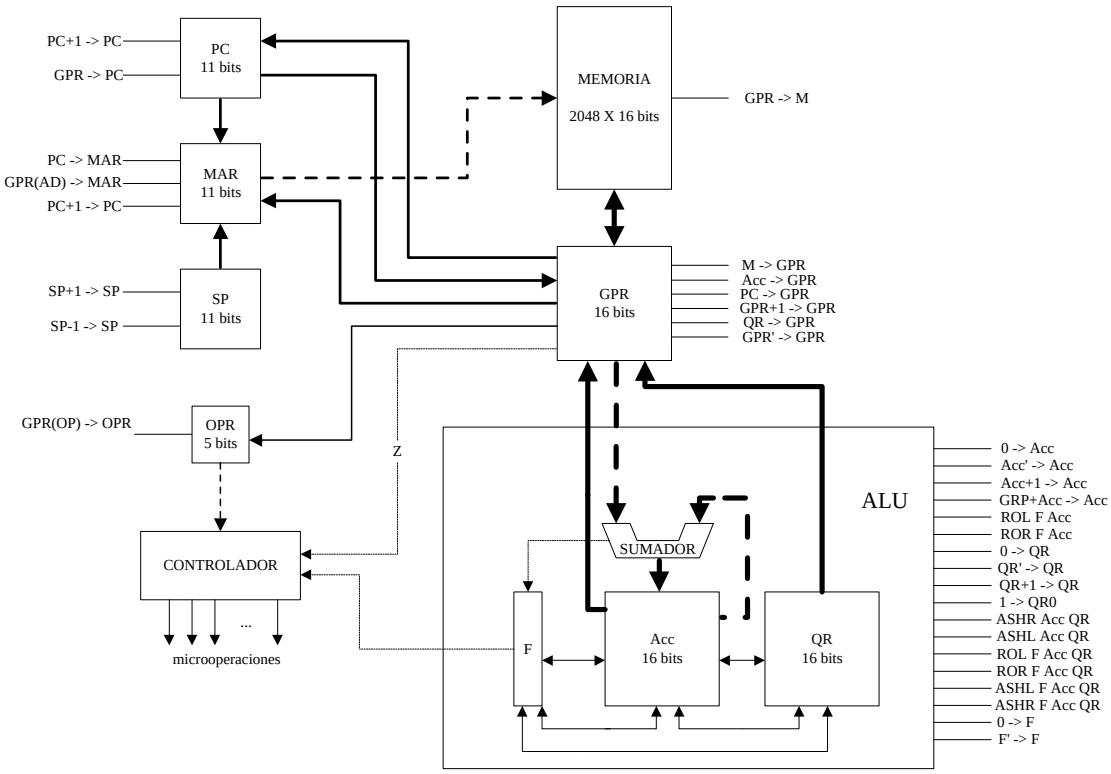

Figura 1. Computadora mejorada.

3.1.3 MAR (Memory Address Register)

El **MAR** es el *registro de dirección de memoria* y tiene un tamaño de **11 bits**. Su cometido es almacenar la dirección de memoria a la que se quiere acceder. La memoria "sabe" en todo momento el valor del MAR. Las microoperaciones que soporta son las siguientes:

- **PC → MAR.** Transfiere el contenido del PC al MAR.
- **GPR(AD) → MAR.** Transfiere los 11 bits menos significativos del valor almacenado en el GPR al MAR. Estos 11 bits especifican una dirección de memoria.
- **SP → MAR.** Transfiere el contenido del SP al MAR.

3.1.4 GPR (General Purpose Register)

Las siglas **GPR** corresponden al *registro de propósito general*. Puede almacenar un dato de **16** bits. Como su propio nombre indica, este registro tiene diversos usos. Las microoperaciones que se pueden realizar sobre él son las siguientes:

- **M → GPR.** Transfiere el contenido de la posición de memoria señalada por el MAR al GPR.
- **Acc → GPR.** Transfiere el contenido del acumulador al GPR.
- **QR → GPR.** Transfiere el contenido del registro QR al GPR.
- **PC → GPR.** Transfiere el contenido del PC al GPR.
- **GPR+1 → GPR.** Incrementa en 1 el valor almacenado en el GPR.
- **GPR' → GPR.** Niega el contenido del GPR.

El registro GPR incorpora también una señal denomida **Z**. Esta nos indica si el valor almacenado en el GPR es cero (**Z = 1**) o es distinto de 0 (**Z = 0**). Esta señal es usada por el controlador para realizar saltos condicionales.

3.1.5 OPR (Operation Register)

El **OPR** o *registro de operación*, tiene como función almacenar el código de operación de la proxima instrucción a ejecutar, que es usado por el controlador para generar las señales de control necesarias. Su tamaño es de **5 bits**, lo cual le permite definir hasta **32 instrucciones** distintas. La única microoperación que se puede realizar sobre el OPR es **GPR(OP) → OPR**, que transfiere los 5 bits más significativos del valor almacenado en el GPR, correspondientes a un código de operación, al OPR.

3.1.6 SP (Stack Pointer)

El registro **SP** es el *puntero de pila*. Su tamaño es de **11 bits**, y nos permite recorrer la memoria RAM como si se tratase de una memoria de pila. Las microoperaciones que soporta el registro SP son las siguientes:

- **SP+1 → SP.** Incrementa en 1 el puntero de pila.
- **SP-1 → SP.** Decrementa en 1 el puntero de pila.

3.1.7 ALU (Arithmetic-Logical Unit)

La **ALU** es la *unidad arimético-lógica*. Esta ALU permite realizar cualquier tipo de operación arimética o lógica mediante el uso de **un sumador, un complementador** y **un conjunto de registros**, que a continuación se describirán uno a uno. En este caso, la ALU nos permite trabajar con datos de **16 bits**.

3.1.7.1 Sumador

El sumador utilizado por la ALU es **un sumador de 16 bits en complemento a 2.** Uno de sus operandos se encuentra directamente conectado a la salida del complementador,

y el otro se encuentra conectado a la salida del acumulador. La salida de acarreo del sumador está directamente conectada a la entrada del registro F. La salida del sumador se encuentra conectada a la entrada del acumulador.

3.1.7.2 Complementador

Su función es complementar la salida del GPR si se activa las señal COMP.

3.1.7.3 Acc (Accumulator)

El acumulador es un registro de **16 bits**, que se utiliza para almacenar los valores resultantes de las operaciones de la ALU. Su entrada se encuentra conectada a la salida del sumador, y está conectado bidireccionalmente por la izquierda con el registro F, y por la derecha con el registro QR. Estas conexiones bidireccionales permiten realizar desplazamientos bidireccionales a través de estos tres registros.

3.1.7.4 QR

El **QR** es un *registro de desplazamiento con realimentación* de **16 bits**. Se usa cuando la ALU tiene que realizar operaciones aritméticas que requerien desplazamientos hacia la izquierda o la derecha, como por ejemplo la multiplicación o la división. Se encuentra conectado bidireccionalmente por la izquierda con el acumulador, y por la derecha con el registro F.

3.1.7.5 F (Flag)

El registro **F** o *bandera*, almacena el valor del acarreo que resulto de la última suma realizada por el sumador. Su tamaño es de **1 bit**, y le sirve al controlador para realizar saltos condicionales.

3.1.7.6 Microoperaciones de la ALU

Las microoperaciones soportadas por la ALU son las siguientes:

- **0 → Acc.** Coloca el valor del acumulador a 0.
- **Acc' → Acc.** Niega el contenido del acumulador.
- **Acc+1 → Acc.** Incrementa en 1 el contenido del acumulador.
- **GPR+Acc → Acc.** Suma el contenido del GPR con el del acumulador y almacena el resultado en el acumulador.
- **ROL F Acc.** Realiza un desplazamiento cíclico hacia la izquierda del registro F con el acumulador.
- **ROR F Acc.** Realiza un desplazamiento cíclico hacia la derecha del registro F con el acumulador.
- **0 → QR.** Pone a 0 el valor de QR.

- **QR' → QR.** Niega el contenido de QR.
- **QR+1 → QR.** Incrementa en 1 el contenido de QR.
- **GPR → QR.** Transfiere el contenido del GPR a QR.
- **1 → QR_0.** Pone a 1 el bit menos significativo del registro QR.
- **ASHL Acc QR.** Realiza un desplazamiento aritmético a la izquierda de Acc con QR.
- **ASHR Acc QR.** Realiza un desplazamiento aritmético a la derecha del Acc con QR.
- **ROL F Acc QR.** Realiza un desplazamiento cíclico hacia la izquierda del registro F con el acumulador y el registro QR.
- **ROR F Acc QR.** Realiza un desplazamiento cíclico hacia la derecha del registro F con el acumulador y el registro QR.
- **ASHL F Acc QR.** Realiza un desplazamiento aritmético hacia la izquierda de los registros F, Acc y QR.
- **ASHR F Acc QR.** Realiza un desplazamiento aritmético hacia la derecha de los registros F, Acc y QR.
- **0 → F.** Pone F a 0.
- **F' → F.** Niega el contenido de F.

3.1.8 El controlador

El controlador se ocupa básicamente de establecer el flujo de ejecución de la computadora. En éste caso el controlador es microprogramado, es decir, mantiene en una memoria ROM una serie de micropalabras en las que se codifican las microoperaciones a ejecutar y el flujo de ejecución .

3.1.8.1 CROM (ROM de control)

Esta memoria ROM es de **256 x 28 bits**, y se encarga de almacenar las micropalabras que corresponden a la codificación de las distintas instrucciones. Las micropalabras tienen una longitud de **28 bits** en nuestro caso, y poseen la siguiente estructura:

s15 s14 ... s0 b3 ... b0 d7 d6 ... d0

Los bits **s15** ... **s0** corresponden a las señales de microoperaciones. Los bits **b3, b2**, **b1** y **b0** nos permiten controlar el flujo de la ejecución. Los bits **d7 ... d0** corresponden a la dirección de bifurcación a la que se debe saltar en caso de que las

señales de control lo indiquen. Estos últimos también pueden ser utilizados para cargar el contador C con un determinado valor que sea usará en la ejecución de bucles.

3.1.8.2 CMAR (Control Memory Address Register)

El registro CMAR (**8** bits) se encarga de determinar cual es la dirección de la CROM donde se encuentra almacenada la micropalabra que debe ser accedida. Este circuito tiene las siguientes señales de entrada:

- *Reloj*.Es el reloj del sistema.
- *Reset.* Permite colocar a 0 el CMAR.
- *R (carga de subrutina).* Señal que indica que debe cargarse en el CMAR la dirección de comienzo de una subrutina.
- *B (salto).* Señal que indica al CMAR que debe cargarse con la dirección indicada por los bits **d7 ... d0**.
- *I (incremento).* Señal que indica al CMAR que incremente en 1 su contenido.
- *Salida de la lógica de correspondencia.* Este conjunto de señales indican al CMAR la dirección donde comienza el conjunto de micropalabras correspondientes a la codificación de la instrucción indicada por el registro OPR.

3.1.8.3 Lógica de correspondencia

La lógica de correspondencia se encarga de indicar cual es la dirección de comienzo del microprograma correspondiente al código de operación almacenado en el registro OPR.

3.1.8.4 Contador (C)

El contador es un registro de 16 bits que se usa para la ejecución de bucles dentro del microcontrolador. Almacena el valor del contador del bucle que se irá decrementado en cada iteración, y mediante la señal “C=0”, que indica si el contador está a 0 (“C=0” = 1) o no (“C=0” = 0), sabremos si ha terminado la ejecución del bucle. Este registro es útil a la hora de realizar operaciones repetitivas, como por ejemplo los sucesivos desplazamientos que se realizan en la multiplicación y en la división.

3.1.8.5 Lógica de control

La lógica de control se encarga de indicar si se debe pasar a la siguiente dirección de la ROM (se activa I y se incrementa en 1 el contenido del CMAR), se debe cargar una subrutina (se activa la señal R y se pasa la salida de la lógica de correspondencia al CMAR), ó se debe realizar una bifurcación (se activa la señal B y se carga en el CMAR la dirección indica por d7 d6 d5 d4 d3 d2 d1 d0). La salida de la lógica depende de las señales de control (b0, b1,b2 y b3), de las señales Z , F y de la señal “C=0”.

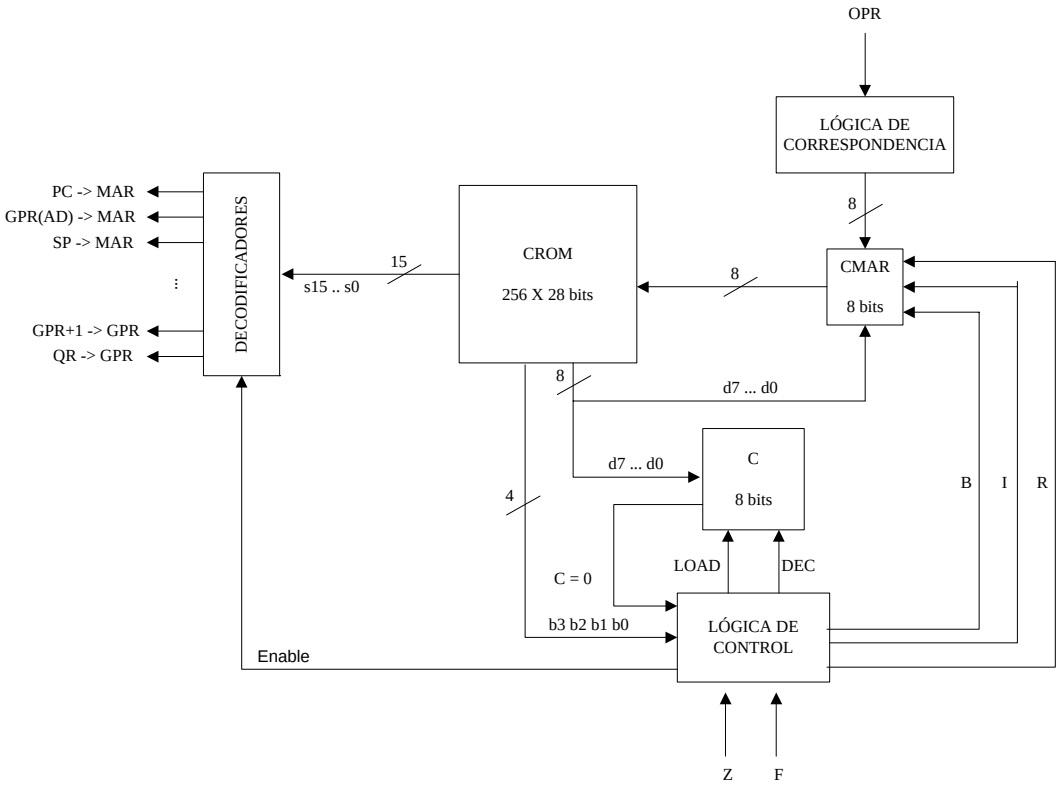

Figura 2. Controlador microprogramado.

3.2 Microprogramación

La microprogramación consiste en almacenar los microprogramas correspondientes a las instrucciones del repertorio de la computadora en una memoria ROM, a la cual se accede según lo requiera el programa que se esté ejecutando. En nuestro caso, las micropalabras que almacena la CROM tienen una longitud de 28 bits, de los cuales los 16 primeros son usados como señales de microoperaciones , los 4 siguientes como señales de control, y los 8 últimos como señales de direccionamiento. A continuación se expone la codificación de las distintas señales de microoperaciones:

- **Señales del MAR.** Son las señales *s15 y s14*. En la siguiente tabla se puede observar su codificación:

s15	s14	Microoperación
0	0	Ninguna
0	1	PC → MAR
1	0	GPR(AD) → MAR
1	1	SP → MAR

Tabla 1. Señales del MAR.

- **Señales del OPR.** Sólo hay una, la *s13*. En la siguiente tabla se puede ver su codificación:

s13	Microoperación
0	Ninguna
1	GPR(OP) → OPR

Tabla 2. Señales del OPR.

- **Señales de control de la memoria.** Sólo hay una, la *s12*. En la siguiente tabla se puede ver su codificación:

s12	Microoperación
0	Ninguna
1	GPR → M

Tabla 3. Señales de la memoria.

- **Señales de control del SP.** Son las señales *s11* y *s10*. Su codificación se expone en la siguiente tabla:

s11	s10	Microoperación
0	0	Ninguna
0	1	SP+1 → SP
1	0	SP–1 → SP

Tabla 4. Señales del SP.

- **Señales de control del PC.** Son las señales *s9 y s8*. La codificación correspondiente a esta señales se puede ver en la siguiente tabla:

s9	s8	Microoperación
0	0	Ninguna
0	1	PC+1 → PC
1	0	GPR → PC

Tabla 5. Señales del PC.

- **Señales de control de la ALU.** Son 5 las señales que se encargan de controlar la ALU: *s7, s6, s5, s4* y *s3*. La codificación de estas se puede observar en la siguiente tabla:

s7	s6	s5	s4	s3	Microoperación
0	0	0	0	0	Ninguna
0	0	0	0	1	0 → Acc
0	0	0	1	0	Acc' → Acc
0	0	0	1	1	Acc+1 → Acc
0	0	1	0	0	GPR+Acc → Acc
0	0	1	0	1	ROL F Acc
0	0	1	1	0	ROR F Acc

0	0	1	1	1	$0 \rightarrow QR$
0	1	0	0	0	$QR' \rightarrow QR$
0	1	0	0	1	$QR+1 \rightarrow QR$
0	1	0	1	0	$GPR \rightarrow QR$
0	1	0	1	1	$1 \rightarrow QR_0$
0	1	1	0	0	ASHL Acc QR
0	1	1	0	1	ASHR Acc QR
0	1	1	1	0	ROL F Acc QR
0	1	1	1	1	ROR F Acc QR
1	0	0	0	0	ASHL F Acc QR
1	0	0	0	1	ASHR F Acc QR
1	0	0	1	0	$0 \rightarrow F$
1	0	0	1	1	$F' \rightarrow F$

Tabla 6. Señales de la ALU.

- **Señales de control del GPR.** Las señales de control del registro de propósito general son *s2*, *s1* y *s0*. Su codificación se muestra en la tabla siguiente:

s2	s1	s0	Microoperación
0	0	0	Ninguna
0	0	1	$M \rightarrow GPR$
0	1	0	$Acc \rightarrow GPR$
0	1	1	$PC \rightarrow GPR$
1	0	0	$GPR+1 \rightarrow GPR$
1	0	1	$QR \rightarrow GPR$
1	1	0	$GPR' \rightarrow GPR$

Tabla 7. Señales del GPR

Una vez visto el reparto de las señales de control y su codificación, pasamos a las señales de bifurcación y de direccionamiento. La codificicación de las señales de bifurcación es la siguiente:

b3	b2	b1	b0	Z	F	C=0		I	B	R	LOAD	DEC	ENABLE
0	0	0	0	x	x	x		0	0	0	0	0	0
0	0	0	1	x	x	x		1	0	0	0	0	1
0	0	1	0	x	x	x		0	1	0	0	0	1
0	0	1	1	x	x	x		0	0	1	0	0	1
0	1	0	0	0	x	x		1	0	0	0	0	0
0	1	0	0	1	x	x		1	0	0	0	0	1
0	1	0	1	x	0	x		1	0	0	0	0	1
0	1	0	1	x	1	x		1	0	0	0	0	0
0	1	1	0	x	x	x		1	0	0	1	0	1
0	1	1	1	x	x	0		0	1	0	0	1	1
0	1	1	1	x	x	1		1	0	0	0	0	1
1	0	0	0	0	x	x		1	0	0	0	0	1
1	0	0	0	1	x	x		0	1	0	0	0	1
1	0	0	1	x	0	x		1	0	0	0	0	1

1	0	0	1	x	1	x		0	1	0	0	0	1
1	0	1	0	0	x	x		0	1	0	0	0	1
1	0	1	0	1	x	x		1	0	0	0	0	1
1	0	1	1	x	0	x		0	1	0	0	0	1
1	0	1	1	x	1	x		1	0	0	0	0	1
1	1	0	0	0	x	x		0	1	0	0	0	0
1	1	0	0	1	x	x		0	1	0	0	0	1
1	1	0	1	x	0	x		0	1	0	0	0	1
1	1	0	1	x	1	x		0	1	0	0	0	0
1	1	1	0	0	x	x		0	1	0	0	0	1
1	1	1	0	1	x	x		0	1	0	0	0	0
1	1	1	1	x	0	x		0	1	0	0	0	0
1	1	1	1	x	1	x		0	1	0	0	0	1

Tabla 8. Señales de control.

4 El entorno de trabajo

En este capítulo se describirán cada una de las ventanas, botones, menús, etc. de los que está dotado **SiCoMe**. Por lo tanto, el objetivo de este capítulo es la familiarización del usuario con la interfaz gráfica del simulador.

4.1 Ventana principal

Esta ventana aparece nada más arrancar la aplicación y en ella se desarrollan la mayor parte de las funciones del simulador. Esta pantalla consta de una barra de menú en la parte superior junto a una barra de herramientas, la representación del contenido de la memoria de la computadora, así como de la computadora en sí misma y una zona de estado. Por último se puede ver una barra de estado en la parte inferior. Esta ventana puede verse en la *Figura 3.*

En la barra de menú se encuentran los siguientes menús:

- *Archivo.* En este menú, se encuentran las opciones de carga de ficheros de programa y de repertorio, tanto ubicados en la máquina local, como en el servidor (por supuesto, se ha de estar conectado al servidor para disponer de dichos archivos), además de la configuración de los parámetros del simulador y del servidor. El control de ficheros de la cuenta en el servidor también está disponible en este menú, a través de una opción para borrar los ficheros no deseados. Por último se da la opción de salir del programa.Ver *Figura 4.*

- *Simulador*. Es el menú que proporciona las opciones relacionadas con la simulación en la computadora. Ya que los controles de la ejecución se han colocado en una barra de herramientas que se comentará más adelante, en este menú se han introducido opciones como reiniciar la computadora cada vez que se desea realizar una nueva ejecución o descargar el programa que haya en memoria (reset). Cuando se quiere borrar todo el contenido de la memoria (programa y datos), también puede

hacerse a través de este menú. En adición, controla si se muestran o no las ventanas del controlador y la del repertorio actual. Ver *Figura 4.*

- *Servidor*. En el menú Servidor se encuentran únicamente las opciones de conexión y desconexión del servidor de prácticas. Ver *Figura 5.*

- *Ayuda.* A través de este menú se accede a la ayuda del simulador, además de a la pantalla de créditos. Ver *Figura 5.*

La barra de herramientas se ha ubicado justo debajo de la barra de menú, y en ella se encuentran las opciones predeterminadas a ser las más usadas de la aplicación. Aquí se incluyen, además de las opciones de control de la ejecución (ejecutar siguiente ciclo, ejecutar siguiente instrucción, ejecución contínua o parar ejecución), las opciones de cargar fichero y cargar repertorio, para ficheros ubicados en la máquina local.

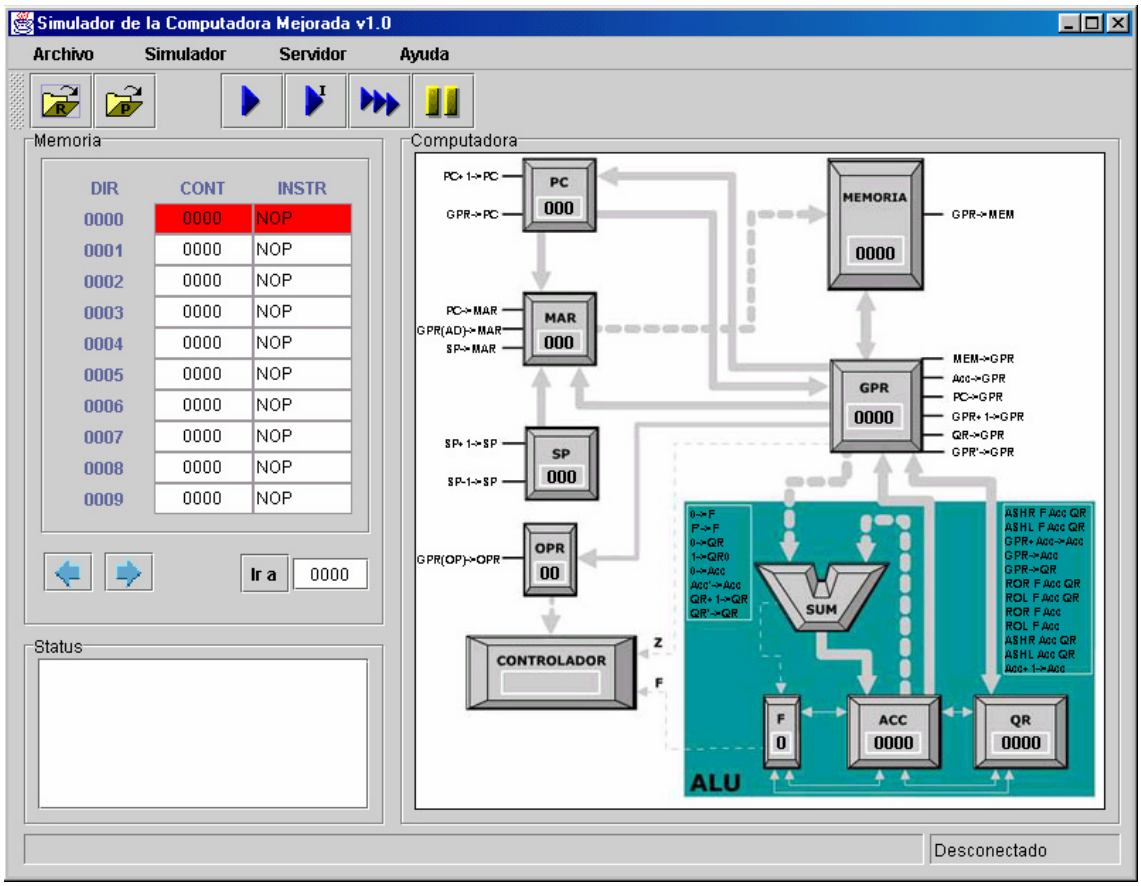

Figura 3. Pantalla principal del simulador

La representación de la memoria consta de una seria de campos que indican la dirección, el contenido de dicha dirección y la instrucción a la que equivale dicho contenido según el repertorio de instrucciones que esté cargado en cada momento. Para facilitar la navegación a través de todos los contenidos de la memoria se dispone de dos controles. A través del primero se puede avanzar y retroceder para ver las diez direcciones siguientes o anteriores respectivamente y mediante la segunda puede saltarse a una dirección determinada directamente.

En la representación gráfica de la computadora se pueden apreciar durante la simulación todos los cambios que se producen dentro de ella, tanto buses o señales activas, como contenido de todos los registros y memoria. Los buses activos aparecen en azul, y las señales activas en rojo.

Por último, la zona de estado y la barra de estado permiten al usuario saber lo que ocurre en cada momento del uso del programa. Así, en la zona de estado aparece el ciclo y la instrucción actual y los ficheros de programa y repertorio que se encuentran cargados en este momento. La barra de estado esta dividida en dos secciones, la primera y de mayor longitud que indica o informa de sucesos puntuales, como resultado de la carga de un programa, etc., y la segunda, de menor longitud y situado en el lado derecho, que informa en cada instante si se encuentra conectado o no con el servidor.

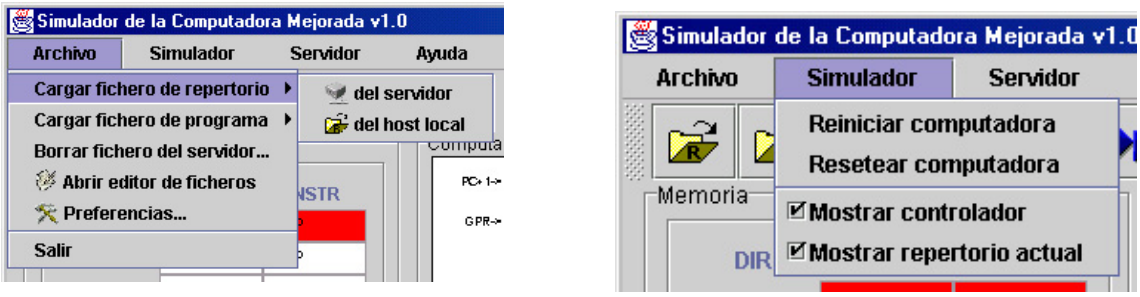

Figura 4. Menú *Archivo* y menú *Simulador.*

Figura 5. Menú *Servidor* y menú *Ayuda.*

4.2 Pantalla de edición de ficheros

Como el funcionamiento del simulador se basa en ficheros de entrada (programas y repertorios), la aplicación incluye un pequeño editor que permite la confección de los ficheros de usuario sin tener que recurrir a herramientas externas al programa. Esta pantalla consta una barra de menú y del panel contenido del editor propiamente dicho. La barra de menú consta de dos menús. El menú *Archivo* permite crear un nuevo fichero, o bien cargar o guardar el contenido de un fichero cualquiera, ya se encuentre ubicado en el servidor de prácticas o en la máquina local (como en ocasiones anteriores se ha mencionado, para poder realizar estas operaciones con ficheros almacenados en el servidor, antes ha de conectarse con éste). También ofrece la opción de salida del editor. El segundo menú de la barra da pie a las distintas opciones de edición, de sobra conocidas, como cortar, copiar y pegar. En la *Figura 6* se puede apreciar la presentación de esta ventana.

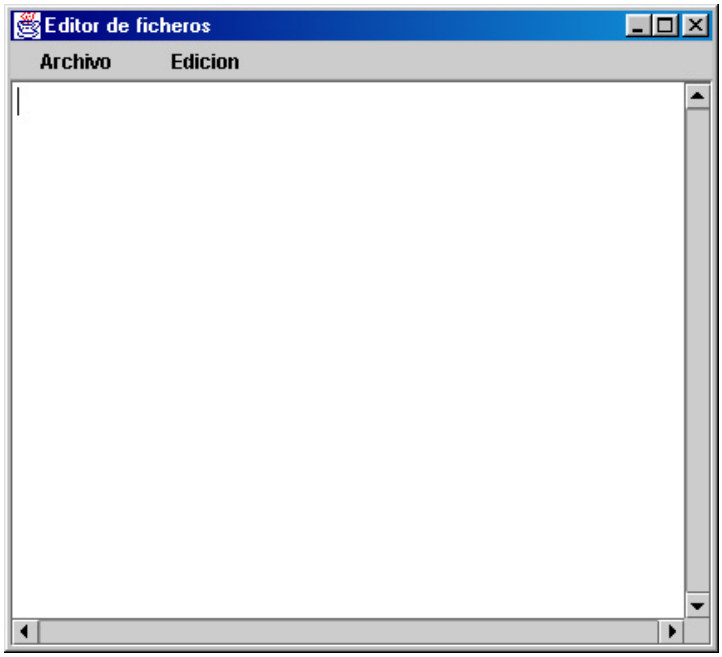

Figura 6. Editor de ficheros de la aplicación.

4.3 Pantalla de representación del controlador

Esta ventana permite visionar el funcionamiento del controlador durante la ejecución. Para mostrarla basta con pulsar sobre la opción del menú *Simulador/Mostrar controlador*. Esta ventana puede verse en la *Figura 7*.

4.4 Pantalla del repertorio de instrucciones

Para que el usuario pueda tener presente el repertorio de instrucciones que se ha cargado en la computadora, y por consiguiente, las posibles instrucciones que puede usar en sus programas, SiCoMe consta de una ventana donde éste es mostrado. Para poder visualizarla basta con pulsar sobre la opción del menú *Simulador/Mostrar repertorio*

actual. Entonces aparece una ventana con dos paneles. En el panel superior se pueden apreciar las instrucciones que constituyen el repertorio que en ese momento haya cargado en la computadora. Cuando se selecciona de esta lista de instrucciones algún nemónico, la decodificación de su microprograma, en forma de microinstrucciones aparece en el panel inferior, permitiendo al usuario conocer el cometido de cada nemónico a la hora de confeccionar sus programas. Esta ventana aparece en la *Figura 8*.

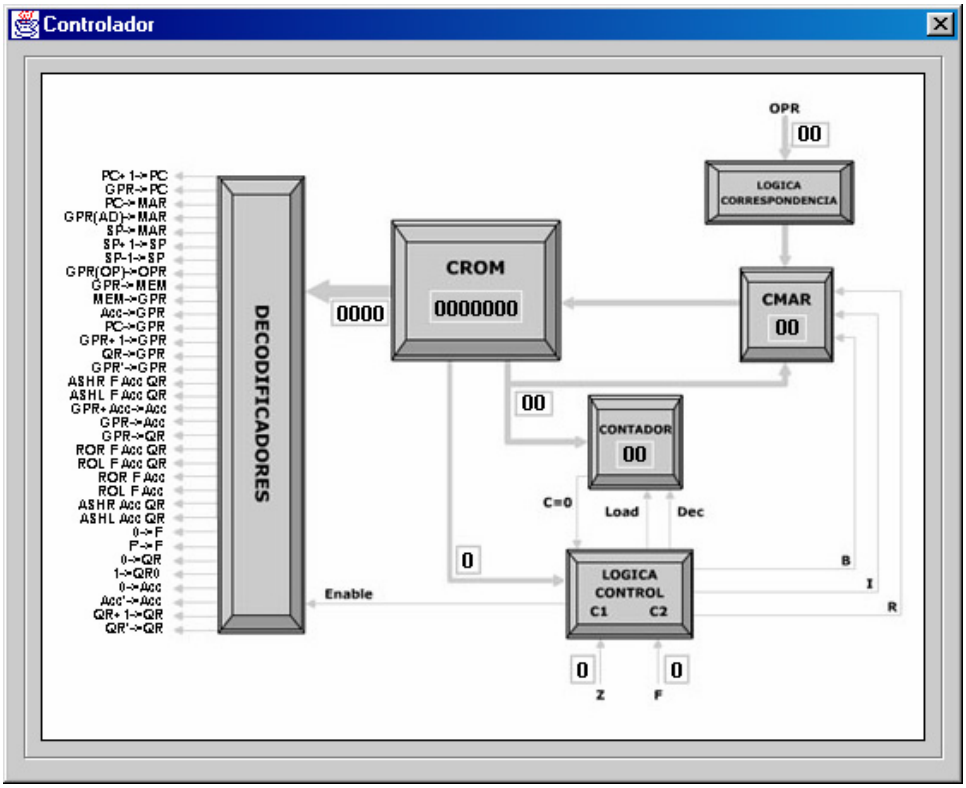

Figura 7. Ventana que muestra el controlador de la computadora.

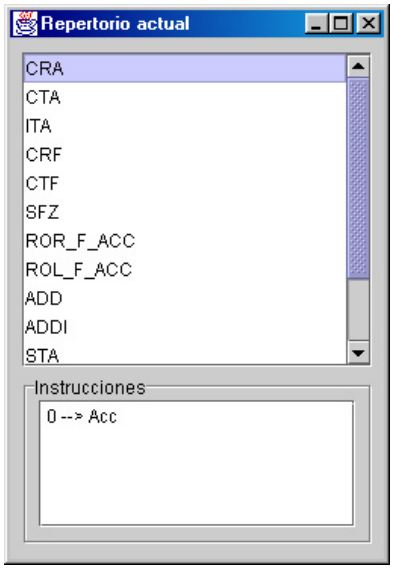

Figura 8. Ventana del repertorio de instrucciones.

4.5 Pantalla de autentificación

A la hora de conectar con el servidor, éste pide una autentificación para comprobar que el usuario puede o tiene permiso para acceder a dicho servidor de prácticas. Los datos necesarios para la autentificación son el *login* y el *password* del usuario, y la petición de estos datos se realiza a través de la pantalla de autentificación. Se trata de una pequeña ventana con dos campos para introducir los datos y dos botones para confirmar la petición de conexión o cancelarla. El resultado de la operación sea cual fuere, se mostrarán en la barra de estado y/o la barra de conexión de la pantalla principal. Esta ventana se puede apreciar en la *Figura 9***.**

4.6 Pantalla de configuración

Esta pantalla consta de dos pestañas que diferencian la sección de parámetros del simulador de la sección de parámetros de conexión. En cada pestaña existen los siguientes parámetros:

- *Pestaña de ejecución.* En esta pestaña se encuentra solamente la variable configurable que controla el tiempo que transcurre entre la ejecución de un ciclo en la simulación, y el comienzo del siguiente. La duración ha de establecerse en segundos. Ver *Figura 10*.

- *Pestaña de conexión.* Aquí se encuentran tres parámetros que configuran el acceso al servidor, como son el nombre de la máquina servidora, su dirección IP y el puerto al que realizar las peticiones. De estas tres variables, sólo pueden configurarse el nombre de la máquina y el puerto, puesto que la dirección IP la obtiene la aplicación a través del nombre de la máquina, lo cual sirve de comprobación de la corrección del host introducido. Ver *Figura 11*.

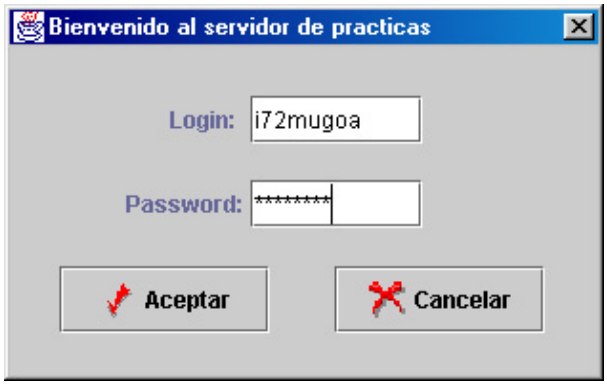

Figura 9. Ventana de autentificación para la conexión con el servidor.

4.7 Pantalla del explorador del servidor

Cuando el usuario se encuentra conectado al servidor, éste puede realizar una serie de operaciones con los ficheros que posee en dicho servidor. En estas operaciones siempre ha de producirse con anterioridad una selección del fichero. Para este necesidad se ha creado esta pantalla, que se asemeja mucho a un diálogo de ficheros, pero en el que se han eliminado las utilidades de navegación de directorios.

En la parte superior aparece en nombre de la máquina que sirve los ficheros. En la parte central se encuentra un panel con los ficheros que contiene la cuenta de usuario que se está explorando. Ya en la parte inferior se encuentra el campo donde aparece en cada momento en nombre del fichero seleccionado y la lista de los filtros de fichero. Por último y también en la parte inferior, los botones para confirmar o cancelar la operación en curso. Todos estos detalles se pueden observar en la *Figura 12*.

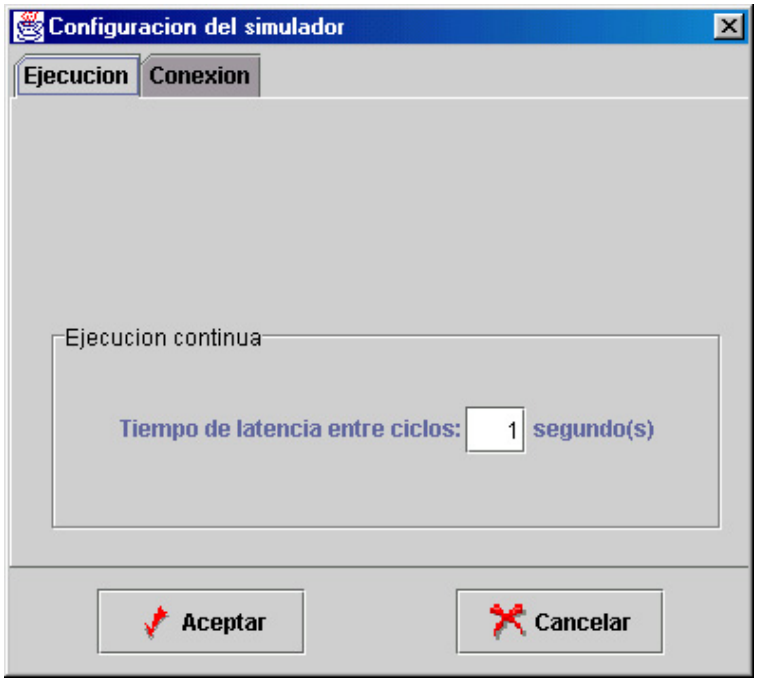

Figura 10. Pestaña de *Ejecución* de la ventana de configuración.

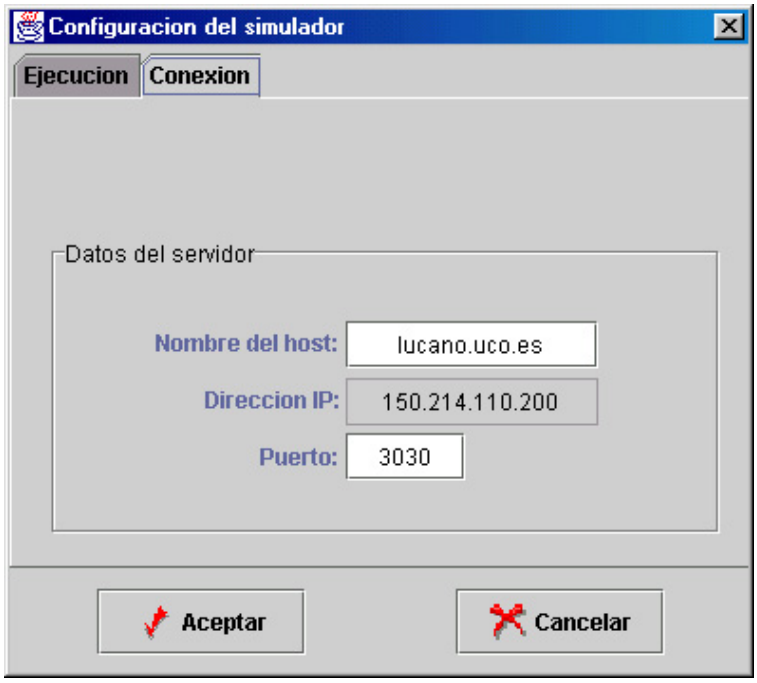

Figura 11. Pestaña de *Conexión* de la ventana de configuración.

4.8 Pantalla del explorador de la máquina local

Esta ventana tiene la misma función que la anterior, pero aplicada a la máquina local. Se trata pues de un diálogo de selección de ficheros en toda regla, en el cual ya no falta ninguna de las herramientas de navegación de directorios. En la parte superior de la pantalla ahora aparece el directorio actual, y a su lado todas las herramientas mencionadas (subir de nivel, crear directorio, etc.). El panel de ficheros es similar al de la pantalla del explorador del servidor, y todos los objetos de la parte inferior también. Su apariencia puede verse en la *Figura 14*.

4.9 Pantalla de créditos de la aplicación

A esta pantalla se accede a través del menú de *Ayuda* y en ella están reflejados los autores de la aplicación así como otros datos de interés sobre ésta. Consta de un elemento gráfico ornamental en la parte superior y la información citada en forma textual en la parte inferior. Un botón permite cerrar la ventana. Se puede ver en la *Figura 13*.

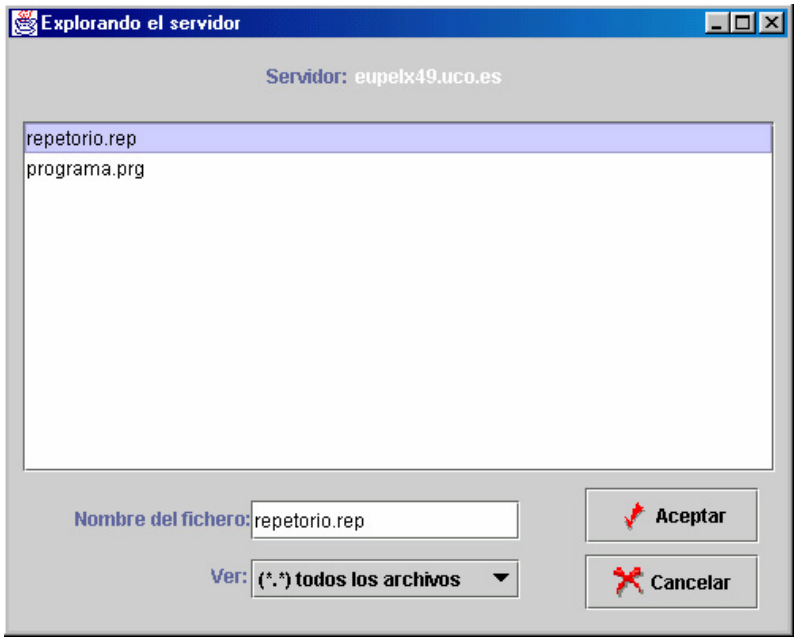

Figura 12. Pantalla de exploración del servidor.

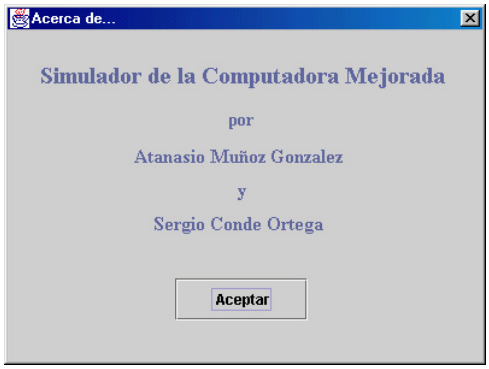

Figura 13. Ventana de *Acerca de...*

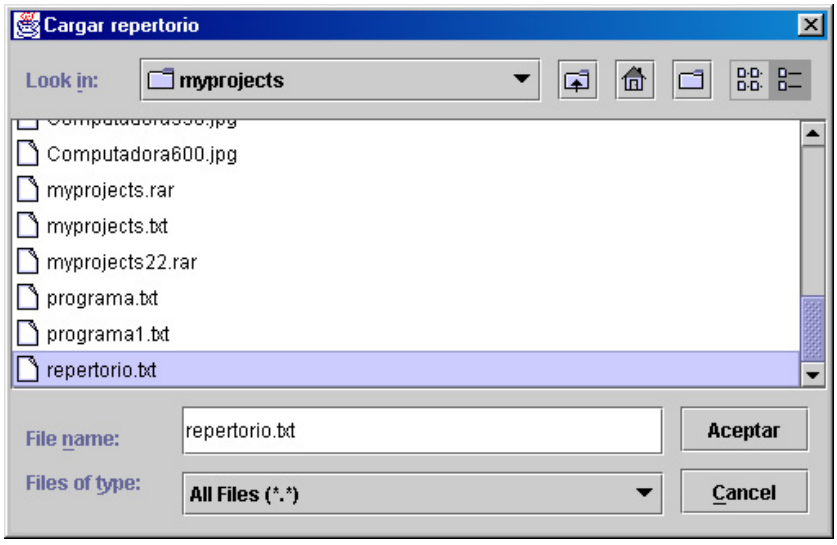

Figura 14. Explorador de fichero de la máquina local.

4.10Ayuda

La ayuda no se muestra en una ventana de **SiCoMe** sino que es cargada por un navegador de Internet, puesto que está desarrollada en HTML.

5 Repertorios y programas

Este capítulo se ocupa de describir la estructura de los repertorios de instrucciones y de los programas que pueden ser usados con éste simulador. Para poder entender bien la programación del simulador es necesarios haber comprendido antes la microprogramación de la computadora, la cual está descrita en el capítulo 3.

5.1 Repertorios de instrucciones

Un repertorio de instrucciones es el conjunto de nistrucciones que posee un computador. Estas instrucciones son las que se pueden usar luego para realizar programas para la computadora.

Los repertorios se programa en un fichero de texto plano. Un repertorio está formado por una secuencia de líneas con la siguiente estructura:

ADD	true|false	8000100 1100 20300
nemónico	bandera	microprograma

El nemónico es el nombre de la instrucción. La bandera indica si la instrucción lleva parámetro o no. Por último, el microprograma está constituido por una secuencia de micropalabras escritas en hexadecimal[1]. Estas definen la función de la instrucción.

La computadora permite hasta un máximo de 32 instrucciones, por lo que si se definen más en un repertorio, éste sólo reconocerá las 32 primeras.

[1] El simulador tiene programado por defecto el ciclo de búsqueda de instrucciones en la CROM (0x0h - 0x3h), por lo que la última micropalabra del programa de una instruccón debe incorporar un salto a la dirección de comienzo del mismo (0x0h).

Es muy importante no dejar líneas en blanco dentro del fichero, pues esto daría lugar a un error a la hora de interpretar el repertorio.

5.2 Programas

Al igual que los repertorios de instrucciones, los programas son editados en ficheros de texto planos. Un programa se divide en tres secciones claramente diferenciadas, cada una de ellas separadas por el carácter @:

declaración de variables
@
dirección de comienzo
@
sentencias
@

En la sección de declaración de variables se declaran las variables que van a ser usadas por el programa. Para declarar una variable simplemente hay que especificar su dirección de almacenamiento seguido de su valor, ambos en hexadecimal. Por ejemplo:

1 EF
FF 30
...

En la siguiente sección del programa se especifica la dirección de comienzo del programa.

Por último, en la zona de sentencias se desarrolla el cuerpo del programa. Una sentencia está constituida por el nemónico de una instrucción y por el parámetro de esta (si es que tiene que llevarlo). Un ejemplo:

ADD 8
JMP
...

Dentro de un programa también se pueden insertar líneas en blanco y comentarios. Los comentarios deben ir precedidos por el carácter #.

6 Ejemplo práctico

A continuación se expone un ejemplo de uso del simulador **SiCoMe** que abarca la mayoría de los aspectos de la aplicación.

El primer que hay que realizar es configurar el simulador. Para ello elegimos la opción **Preferencias** del menú **Archivo**. Ante nosotros aparecerá la ventana de configuración, la cuál consta de dos pestañas: una para configurar la duración de los ciclos de reloj, y otra para configurar el host y el puerto de escucha del servidor de prácticas.

Una vez configurado el simulador, el siguiente paso es la definición de un repertorio de instrucciones para la computadora. En este caso nuestro repertorio tendrá las siguientes instrucciones:

- CRA. Limpia el acumulador.
- CTA. Complementa el acumulador.
- ITA. Incrementa el acumulador.
- CRF. Limpia F.
- CTF. Complementa F.
- SFZ. Salta a la siguiente instrucción si F es cero.
- ROR_F_ACC. Realiza el desplazamiento cíclo a la derecha de F Acc.
- ROL_F_ACC. Realiza el desplazamiento cíclo a la izquierda de F Acc.
- ADD *dir*. Suma el contenido de la dirección *dir* al acumulador.
- ADDI *dir*. Suma el contenido de la dirección almacenada en la posición *dir* al acumulador.
- STA *dir*. Almacena el contenido del acumulador en la dirección *dir*.
- JMP *dir*. Salta a la direción *dir*.
- JMPI *dir*. Salta a la posición alamacenada en la diercción *dir*.
- HALT. Para la ejecución.

Para escribir el repertorio usaremos el editor de ficheros del **SiCoMe**. Para ello, pulsamos sobre la opción **Abrir editor de ficheros** dentro del menú **Archivo**.
Una vez abierto el editor debemos insertar el siguiente texto sin dejar líneas en blanco.

```
CRA false 8300
CTA false 10300
ITA false 18300
CRF false 90300
CTF false 98300
SFZ false 100500 300
ROR_F_ACC false 30300
ROL_F_ACC false 28300
ADD true 8000100 1100 20300
ADDI true 8000100 1100 8000100 1100 20300
STA true 8000100 2100 1000300
JMP true 2000300
JMPI true 8000100 1100 2000300
HALT false 0000000
```

Cuando hayamos escrito el repertorio sólo queda guardarlo en la máquina local o en el servidor. Para guardarlo en el servidor hay que conectarse previamente a este. Para realizar esto hay que insertar el login y el password de usuario en la ventana de autentificación y pulsar **Aceptar**. A esta ventana se accede a través de la opción **Conectar con el servidor** del menú **Servidor**.

Cuando salvemos nuestro repertorio podremos entonces cargarlo en el simulador. Podemos cargarlo bien desde el servidor o bien desde la máquina local, dependiendo, claro está, de donde lo hayamos guardado. Supongamos que lo hemos guardado en el servidor. Lo primero es seleccionar la opción **Cargar fichero de repertorio del servidor** del menú **Archivo**. Cuando aparezca la lista de ficheros que poseemos, sólo tendrmos que seleccionar el correspondiente a nuestro repertorio y pulsar sobre **Abrir**.

Una vez cargado el repertorio, podremos verlo en la ventana de repertorio. Sólo hay que pulsar la opción **Mostrar repertorio** en el menú **Simulador**.

Teniendo ya un repertorio de instrucciones podemos desarrollar programas para la computadora. Vamos a diseñar un programa que sume los números 17, B y 1C, y almacene el resultado en la posición FF.

Como en el caso del repertorio, usamos el editor para insertar el texto del programa.

```
0 17
1 B
2 1C
@
3
@
CRA
```

```
ADD 0
ADD 1
ADD 2
STA FF
HALT
@
```

Guardamos el fichero del programa en la máquina local y lo cargamos en la computadora al igual que hicimos con el repertorio. Cuando esté cargado, aparecerá en el marco correspondiente a la memoria.

El último paso de todos es ejecutar el programa. El simulador ofrece tres modos de ejecución:

- Ejecución ciclo a ciclo.
- Ejecución instrucción a instrucción.
- Ejecución contínua.

Para poder observar el funcionamiento del controlador durante la ejecución sólo tenemos que pulsar sobre la opción **Mostrar controlador** del menú **Simulador** y aprecerá ante nosotros la ventana del controlador.

Si deseamos volver a ejecutar el programa con las condiciones iniciales, tendremos que elegir la opción **Reiniciar computadora** del menú **Simulador**. Y si queremos borrar el programa de la computadora, elegiremos la opción **Resetear computadora** del mismo menú.

Bibliografía

[1] Hennessy, J.L.; Patterson, D.A. *Arquitectura de Computadores – Un enfoque cuantitativo*. Traducido por Sánchez, J.M. Revisión Técnica de González, A.;Valero, A.;Vaquero, A. Madrid. McGraw-Hill. 1993. ISBN: 1-55860-069-8.

[2] Morris Mano M. *Arquitectura de Computadores*. Traducido por Franco C. Méjico. Prentice-Hall Iberoamericana, S.A. 1983. 508 p. ISBN: 968-880-054-6.

Listado de figuras

Listado de tablas