Manual de usuario

Entorno de programación y simulación para Algorítmez

Entorno de programación para Símplez

Revisión 1.0
Mayo de 2003

Autor: F. Javier Rodríguez Dantart

Índice de contenidos

Capítulo 1

Prefacio

Algorítmez y Símplez son ordenadores didácticos diseñados por Gregorio Fernández. Están descritos en su libro de texto "Conceptos básicos de arquitectura y sistemas operativos. Curso de ordenadores" (Fernández 1998).

La aplicación descrita en este manual consta un **entorno de programación** que permite escribir programas **de Símplez** o **de Algorítmez** en lenguaje ensamblador, así como ensamblar y montar dichos programas.

Dispone también de un **simulador de Algorítmez** con el que se pueden cargar y ejecutar programas ensamblados y montados previamente, pudiéndose observar la evolución de la simulación y la interacción con los dispositivos de entrada y salida.

Por último, se le ha añadido un **intérprete de ficheros binarios de Algorítmez** para poder examinar los contenidos de los ficheros binarios generados, que son los módulos objeto y los programas ejecutables.

El programa está **desarrollado en Java**, por lo que el mismo binario puede ejecutarse en cualquier plataforma. Esto supera una de las limitaciones de los simuladores de Algorítmez desarrollados hasta la fecha.

1.1 Usuarios de este manual

Este manual está destinado a los usuarios del simulador, que normalmente serán estudiantes que deseen hacer prácticas para profundizar en los conceptos que aparecen el libro de texto

(Fernández 1998). Por esto, es imprescindible el estudio previo de los temas correspondientes en el libro para poder comprender y utilizar la aplicación

El manual está disponible en varios formatos en la página web del de la asignatura **Fundamentos de ordenadores**, **Departamento de Ingeniería Telemática** de la **Escuela Técnica Superior de Ingenieros de Telecomunicación de Madrid** perteneciente a la **Universidad Politécnica de Madrid**. La URL de la citada escuela es:

```
http://www.etsit.upm.es
```

1.2 Requisitos mínimos

Como requisito previo a la instalación y utilización del simulador, es necesario tener disponible en el sistema el conjunto de herramientas JDK, con una versión igual o superior a la 1.4. Éste se puede obtener de la URL:

```
http://www.sun.com
```

Una vez instalado el JDK se tendrá un directorio:

```
C:\j2sdk1.4.0\bin
```

en el caso de Windows 95/98/NT, y algo parecido a:

```
/usr/j2sdk1.4.0/bin
```

en el caso de Unix/Linux.

Para independizar las explicaciones respecto de la plataforma, se utilizará `$JAVAHOME` para referirse a estos directorios.

A lo largo del manual, para hablar de directorios, se usará la notación de Unix, por lo que la barra inclinada utilizada para separarlos será el carácter '/'. Si el usuario utiliza Windows, tendrá que utilizar '\'.

El simulador se ejecuta adecuadamente en un Pentium 133MHz con 32MB de memoria RAM, pero es aconsejable utilizar uno de velocidad superior y 64MB de memoria.

Para un correcto funcionamiento de las ventanas se recomienda tener instalados los diferentes tamaños de letra de la fuente `Courier`.

1.3 Instalación

El programa, que se presenta bajo licencia GPL, viene comprimido en la forma `alg.jar`, y debe ser copiado en un directorio. En este directorio, deben extraerse como mínimo los ficheros `alg.gif`, `ARRANQUE.LNA` y `ARRANQUE.MAP`. Para ello se ejecuta:

```
$JAVAHOME/jar xf alg.jar alg.png ARRANQUE.LNA ARRANQUE.MAP
```

Para arrancar el simulador:

```
$JAVAHOME/java –jar alg.jar
```

Puede verse una lista de todos los ficheros comprimidos con:

```
$JAVAHOME/jar tf alg.jar
```

Si se quieren extraer los ficheros:

```
$JAVAHOME/jar xf alg.jar
```

Nota 1: La ruta a los ficheros binarios, a la que aquí nos hemos referido como `$JAVAHOME`, no habrá que especificarla si se ha incluido en la variable de sistema `PATH`.

Nota 2: Para más información sobre el comando `jar`, puede ejecutarse sin argumentos, lo que proporciona una ayuda resumida.

1.4 Organización del manual

El manual comienza describiendo el entorno de programación y simulación así como sus componentes. Esto permite al usuario tener una visión global del mismo.

En los siguientes capítulos se profundiza en las distintas utilidades que incluye el entorno explicándolas en el orden en el que normalmente se van a ir utilizando. A saber: programación, ensamblaje, montaje, interpretación de ficheros binarios, carga, ejecución y comunicación con los periféricos.

Por último se incluyen a modo de apéndices los mensajes que pueden aparecer así como conceptos teóricos como ayuda a la comprensión de los procesos que se llevan a cabo en la aplicación.

1.5 Convenciones tipográficas

A lo largo del manual tanto comandos de sistema como secciones de código se muestran con caracteres de anchura fija:

```
abdefg12345
```

Las notas u observaciones se resaltarán con el siguiente formato:

Nota: Esto es una nota u observación.

1.6 Bibliografía

Gregorio Fernández (1998). *Conceptos básicos de arquitectura y sistemas operativos. Curso de Ordenadores*. (2ª edición). Ed Syserso. Madrid.

Iván Peña (2000). *Simulador e interfaz gráfica de usuario para un ordenador didáctico*. E.T.S.I. Telecomunicación. Madrid.

Jesús A. Pérez Garcilópez (1997). *Proyecto de Fin de Carrera.* E.T.S.I. Telecomunicación. Madrid.

Capítulo 2

El entorno de programación y simulación

El entorno de programación y simulación que constituye esta aplicación permite escribir y ejecutar programas de Algorítmez en el nivel de máquina convencional. Los programas se escriben en lenguaje ensamblador a través del entorno de programación, que consta de un editor, un ensamblador y un montador.

La parte del entorno de programación se ha diseñado conjuntamente para Algorímez y para Simplez y es común para ambos. Al ser común la interfaz gráfica, su utilización es igual para los dos ordenadores con la salvedad del tipo de ficheros que cada uno carga y genera. Si este entorno quiere usarse para programar en lenguaje ensamblador de Símplez hay que arrancarlo desde la aplicación gráfica que desarrolló Iván Peña y que llamó simulador-depurador de Símplez (Peña 2000).

Durante la simulación, para la interacción con el usuario, la aplicación proporciona, como mecanismos de Entrada/Salida, la pantalla y el teclado de Algorítmez.

También se dispone de un intérprete de ficheros binarios de Algorítmez para hacer un análisis de los ficheros objeto y ejecutables generados por el ensamblador y el montador respectivamente.

2.1 La ventana principal

Al arrancar el programa aparece la ventana principal que permite mostrar el entorno de programación, el intérprete de ficheros binarios o el entorno de simulación.

La ventana principal lleva el título de "Simulador de Algorítmez" y desde ella se accede a toda la funcionalidad del simulador. Ofrece una barra de menús y un conjunto de tres paneles superpuestos que contienen sendos grupos de herramientas.

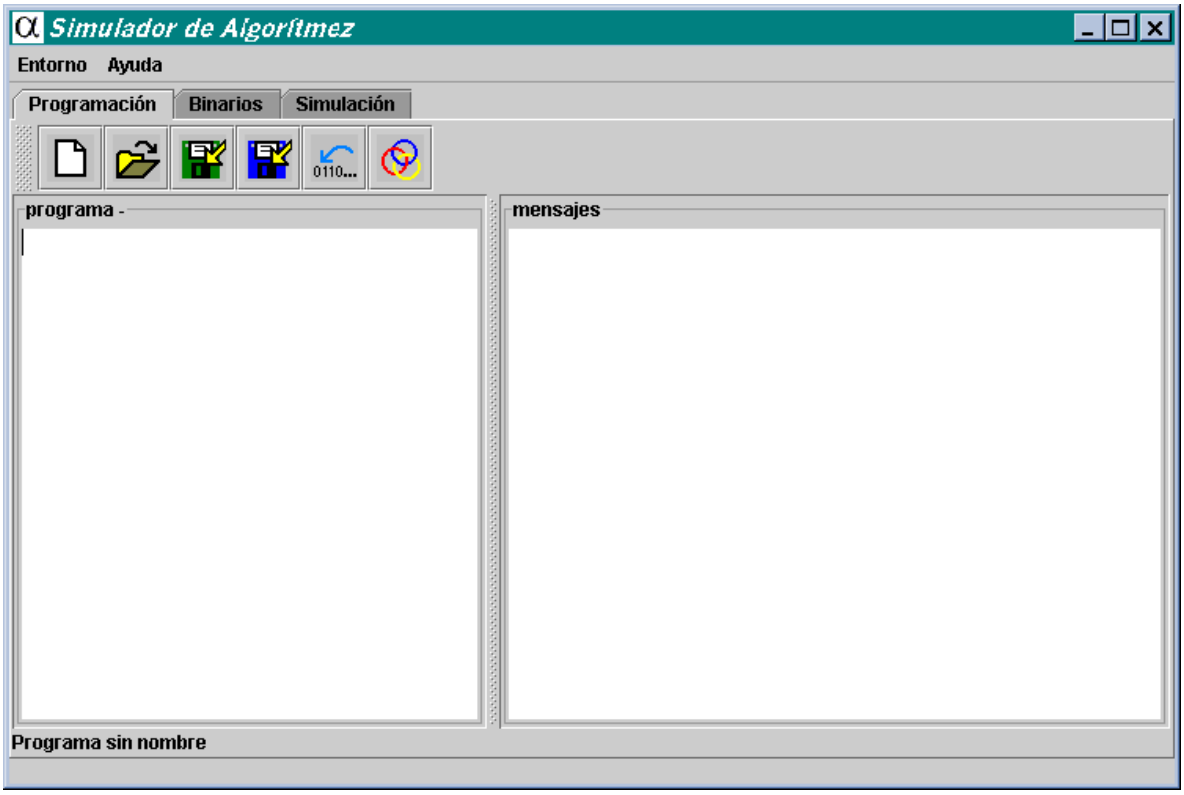


Figura 2.1. Ventana principal.

Barra de Menús

Menú de "**Entorno**"

- "**Salir**": Para salir del programa.

Menú de "**Ayuda**"

- "**Acerca De**": Muestra información sobre el simulador. Versión, Copyright, etc.

Paneles de herramientas

Además del menú, la pantalla principal muestra una de las tres herramientas que incluye la aplicación; a saber:

El entorno de programación.
El intérprete de ficheros binarios.
El entorno de simulación.

Las tres herramientas se encuentran en la pantalla en todo momento en tres paneles independientes, sobre la ventana principal y superpuestos entre sí.

El acceso a cada una de ellas se hace a través de la pestaña que incluye cada panel en la parte superior, mostrándose una sola de las herramientas en cada momento.

Figura 2.2. Detalle de las pestañas de los paneles.

2.2 El Entorno de programación

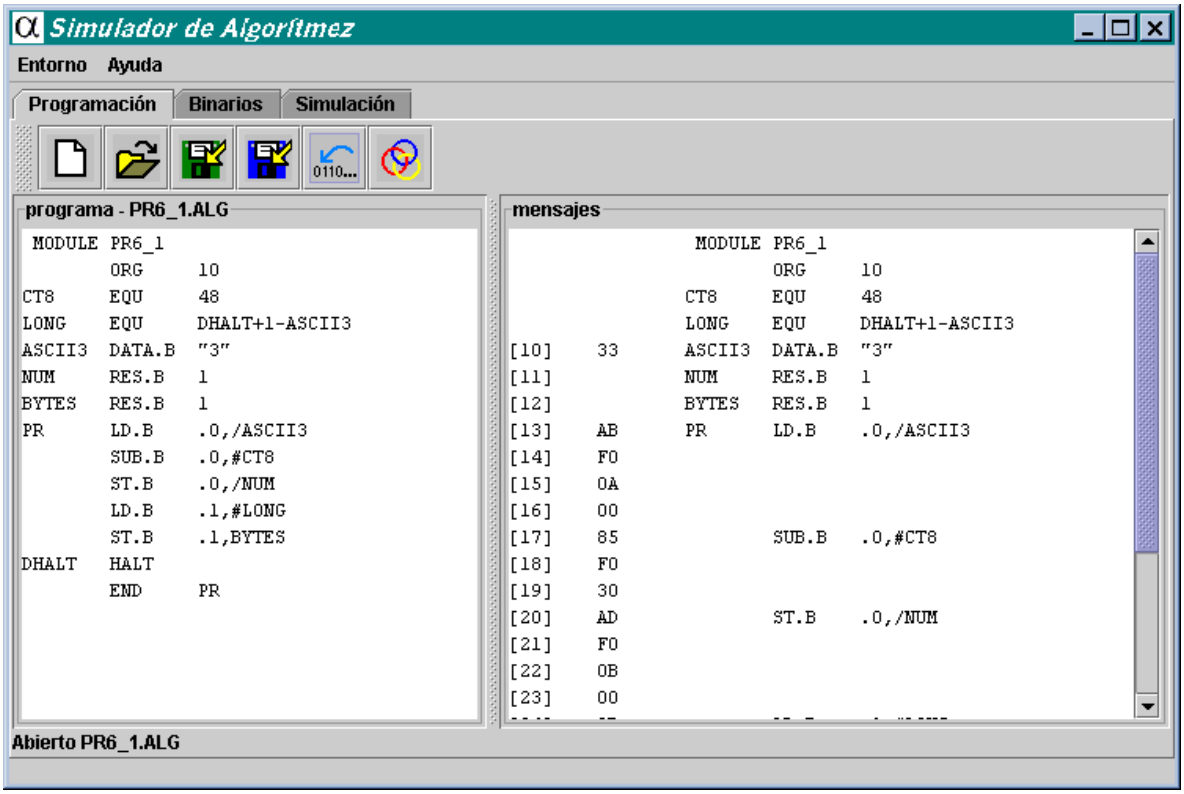


Figura 2.3. El entorno de programación.

El entorno de programación es la herramienta que permite escribir, ensamblar, montar y depurar programas escritos en ensamblador de Algorítmez.

El panel dispone de una barra de herramientas y de un área para texto que se divide en dos partes de tamaño configurable con el ratón. La parte izquierda es el editor de programas y la derecha es una zona para mensajes, que es donde se pueden ver los resultados del proceso de ensamblaje.

La barra de herramientas

Figura 2.4. Barra de herramientas del entorno de programación.

Consta de seis botones que realizan las siguientes funciones:

 "**Nuevo**": Borra los paneles para empezar la edición de un nuevo programa.

 "**Abrir...**": Abre para edición un programa previamente grabado.

 "**Guardar**": Graba el programa a disco.

 "**Guardar como...**": Graba el programa a disco pudiendo cambiar su nombre y localización.

"**Ensamblar**": Ensambla el programa escrito en el editor de programas.

 "**Montar...**": Toma como entrada un módulo principal y realiza el montaje de éste con los módulos de los que importa símbolos.

2.3 El intérprete de ficheros binarios

El intérprete de ficheros binarios es la herramienta que permite interpretar el contenido de los ficheros binarios generados por el ensamblador y el montador de Algorítmez.

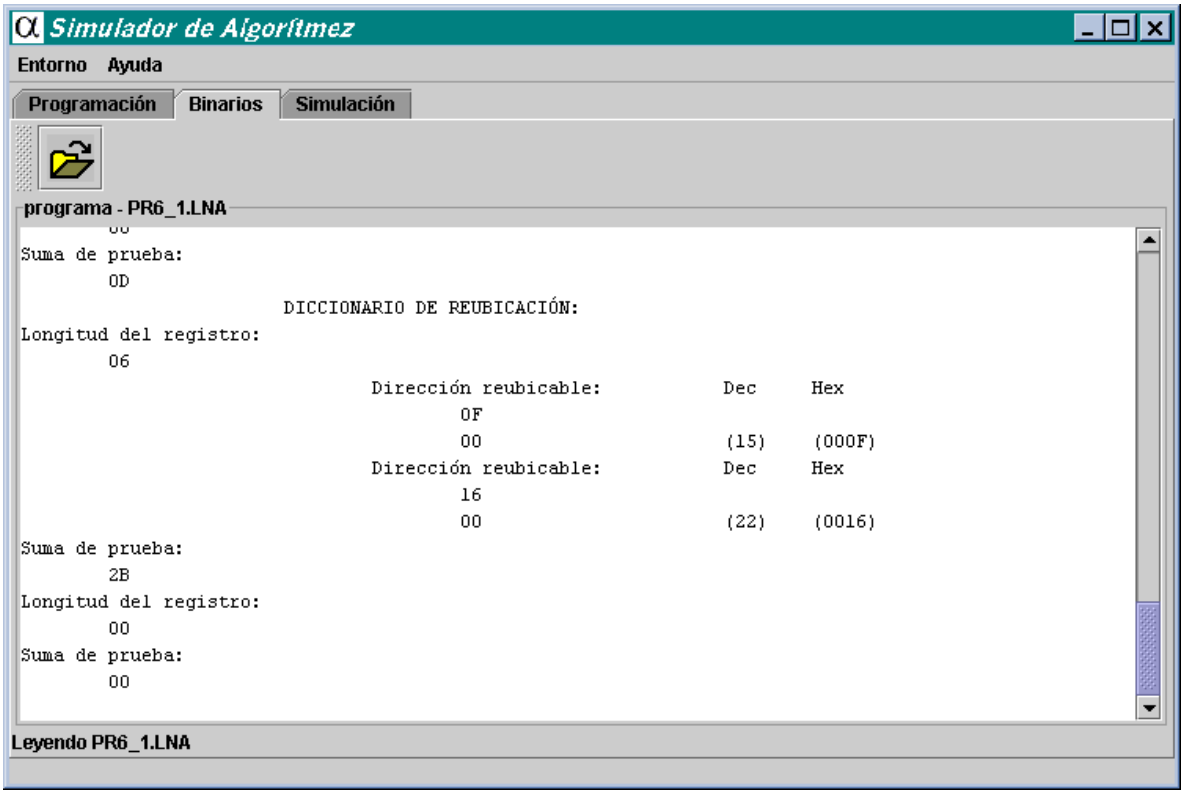


Figura 2.5. El intérprete de ficheros binarios.

Dispone de una barra de herramientas y de un área de texto.

La barra de herramientas

Figura 2.6. Barra de herramientas del intérprete de ficheros binarios.

Incorpora un único botón:

 "**Abrir...**": Abre un fichero binario para su interpretación.

El área de texto

En el área de texto se muestra la estructura interna del fichero binario así como el contenido y el significado de cada uno de los bytes que lo integran.

2.4 El Entorno de simulación

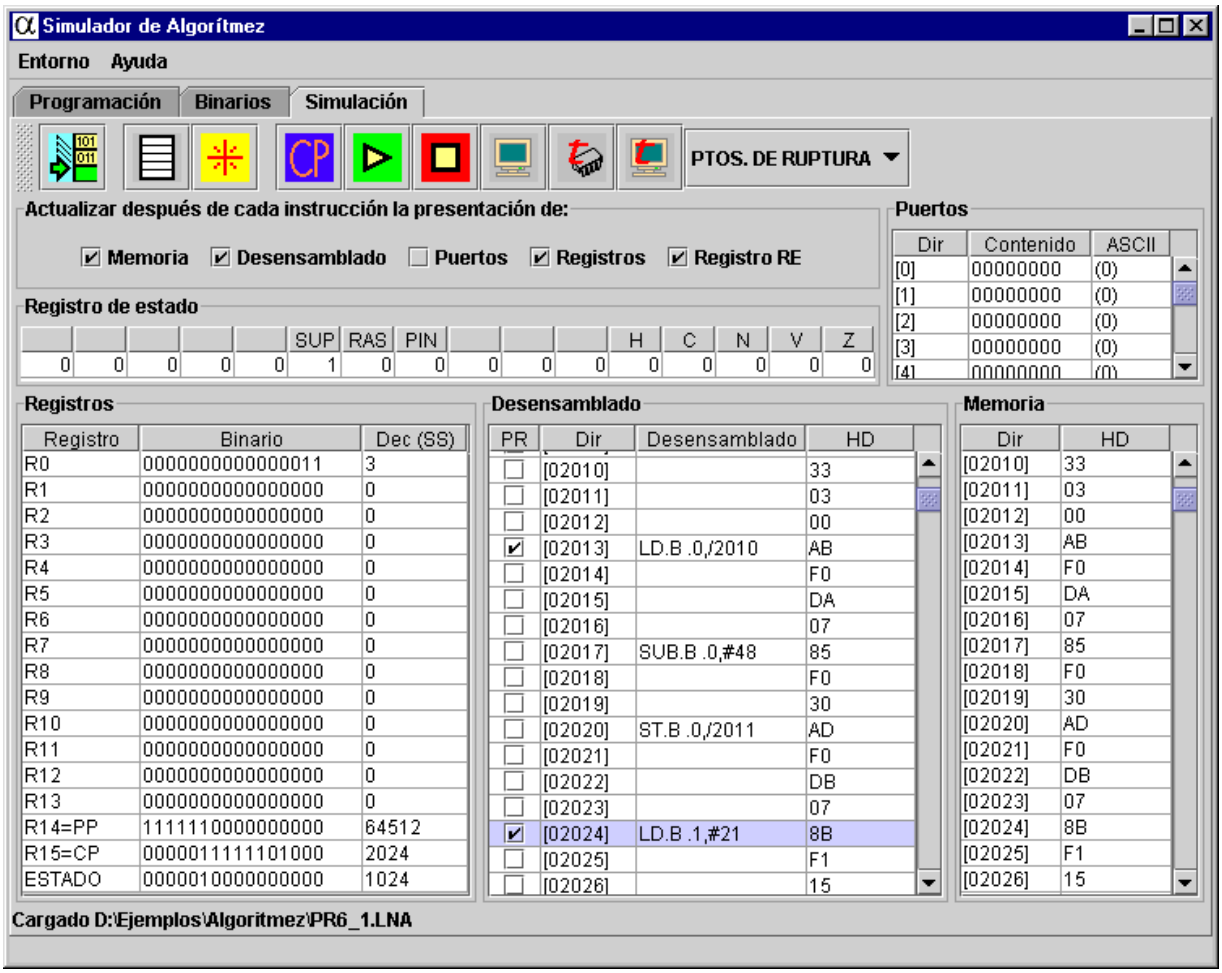


Figura 2.7. El entorno de simulación.

El entorno de simulación es la herramienta que permite ejecutar programas en código binario de Algorítmez.

El panel dispone de una barra de herramientas y un área de monitorización que permite conocer el estado de los principales componentes de Algorítmez en el nivel de máquina convencional.

La barra de herramientas

Figura 2.8. La barra de herramientas del panel de simulación.

Dispone de las siguientes funciones:


"**Cargar...**": Carga en la memoria un programa ejecutable.


"**Borrar memorias y puertos**".


"**Arrancar**": Igual que el anterior, cargando un programa de arranque.


"**Modificar el contador de programa...**".

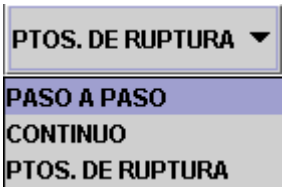

"**Avanzar en la simulación**".


"**Parar la simulación**".


"**Abrir/Cerrar la pantalla de E/S**".


"**Tiempo de UCP**": Permite configurar el tiempo de ejecución de una instrucción.


"**Tiempo de ES**": Permite configurar los tiempos de entrada/salida.



"**Selector del modo de ejecución**".

Área de monitorización

El área de monitorización permite conocer el estado de los principales componentes de Algorítmez a nivel de máquina convencional.

Se compone de los siguientes elementos representados en sendas tablas:

Registros: Muestra el contenido de los 16 registros de la memoria local y del registro de estado. Representa los contenidos en binario, hexadecimal, decimal sin signo y decimal con signo.

Registro de estado: Muestra el contenido del registro de estado especificando el valor y el significado de cada uno de sus bits.

Puertos: Muestra el contenido de los 256 puertos que puede direccionar Algorítmez. Representa los contenidos en binario y ASCII.

Memoria: Muestra el contenido de los 65.536 bytes de la memoria principal de Algorítmez en hexadecimal y ASCII.

Desensamblado: Igual que la anterior pero incluyendo las casillas para marcar los "puntos de ruptura" y el desensamblado de las instrucciones en aquellos programas en que sea posible.

Selectores de refrescos: En la parte superior del área de monitorización se encuentran cinco casillas para seleccionar qué tablas actualizan la representación de sus contenidos después de cada instrucción ejecutada en modo continuo o con puntos de ruptura. Cuando se simula en modo "paso a paso" o se llega a un punto de ruptura, se actualiza el contenido de todas las tablas independientemente del estado de los selectores de refrescos.

2.5 La ventana de entrada/salida

Esta ventana simula la pantalla de Algorítmez a la vez que permite controlar la entrada de datos por teclado. Cuando la ventana está activa (cuenta con el foco de acción) aparece con fondo negro y letras blancas. En esta situación, el hecho de pulsar una tecla, se considera como que se ha hecho sobre el teclado de Algorítmez. Si no tiene el foco, se invierten los colores, y al simulador no le afecta ninguna acción sobre el teclado.

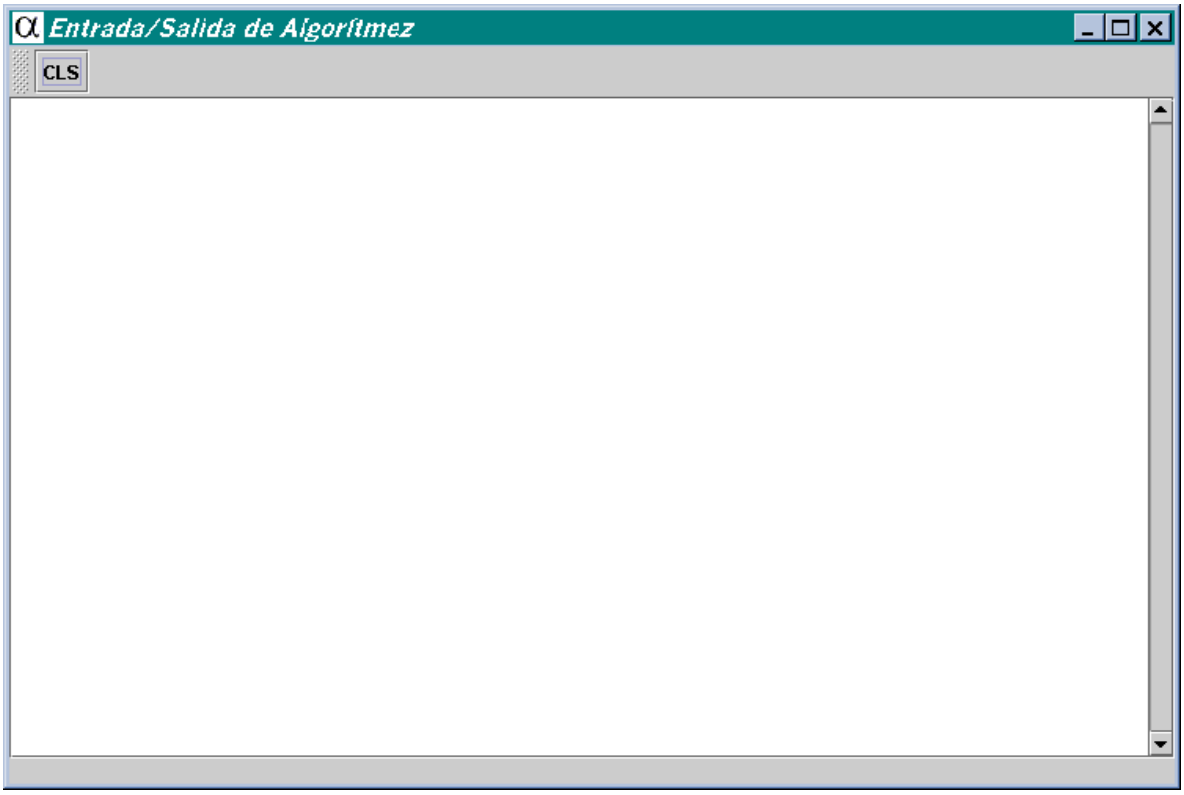


Figura 2.9. La pantalla de entrada/salida.

La pantalla se borra con el botón superior con indicativo CLS (Clear Screen).

2.6 Diálogos modales

Con determinadas operaciones o ante ciertas situaciones se abre otro tipo de ventana, los Diálogos Modales. Son pequeñas ventanas de información, llamadas modales porque bloquean la entrada de usuario al resto de ventanas del simulador. A continuación se muestran los cuatro tipos de diálogos modales que pueden aparecer.

- **Informativos:**

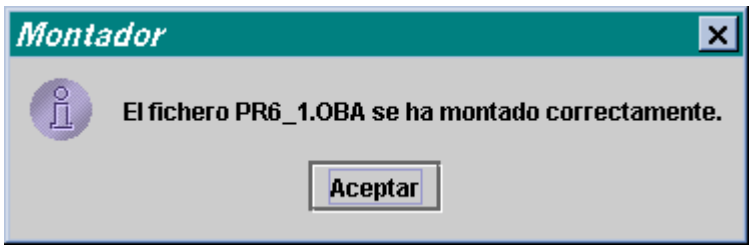


Figura 2.10. Diálogo modal informativo.

- **Advertencias:** Previenen al usuario acerca de las consecuencias indeseables que puede acarrear la acción que ha solicitado. Normalmente sugieren una solución.

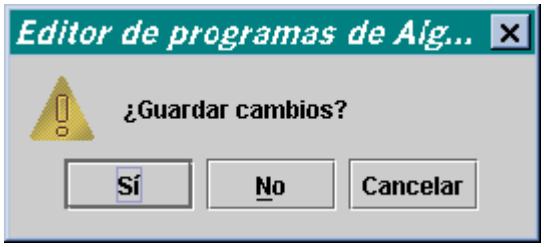


Figura 2.11. Diálogo modal de advertencia.

- **Errores:** Informan de una situación anómala provocada por una determinada acción.

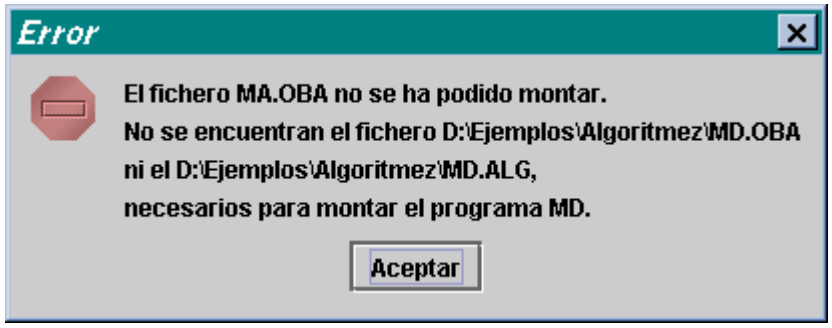


Figura 2.12. Diálogo modal de error.

- **Entrada de datos:** Permiten recoger datos del usuario.

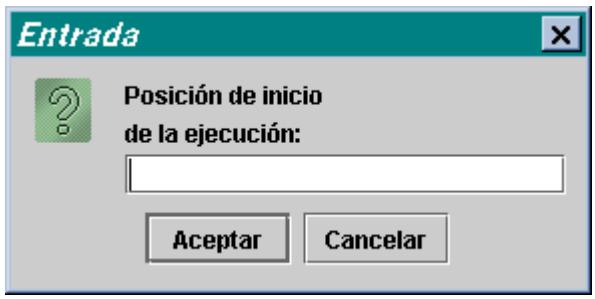


Figura 2.13. Diálogo modal de entrada de datos.

2.7 Ventanas de selección de ficheros

Estas ventanas tienen el formato que aparece en la figura 2.14. Hay dos tipos según la operación, abrir o guardar, que se vaya a realizar. Permiten navegar por los directorios del sistema para localizar un determinado fichero o guardar con un nombre y en un punto concreto.

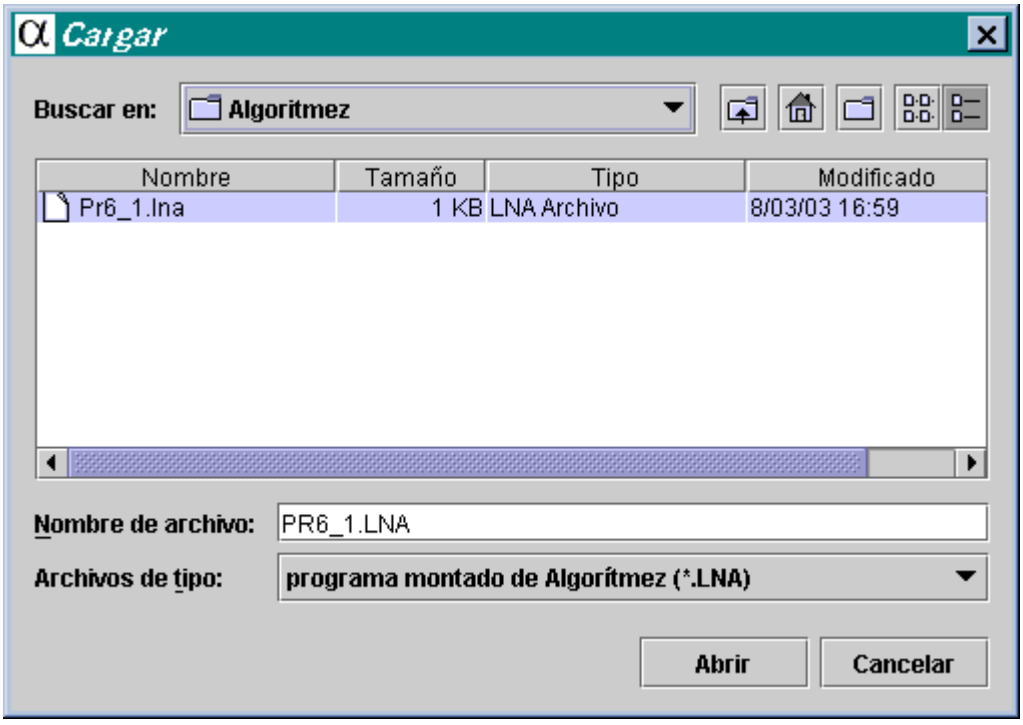


Figura 2.14. Ventana de selección.

2.8 Flexibilidad

La interfaz gráfica del entorno de programación y simulación se ha diseñado tratando de conseguir la mayor flexibilidad y sencillez de manejo posibles. Para optimizar el espacio disponible en el monitor, se diseñó la ventana principal como un conjunto de tres paneles superpuestos, ya que normalmente no se necesitará trabajar más que con uno de los tres en cada momento.

En el caso de la ventana de entrada/salida, que siempre está fuera de la ventana principal, puede minimizarse o cerrarse si no se está usando en ese momento.

Además, las barras de herramientas pueden desprenderse del resto de la ventana pinchando en la zona punteada de las mismas y arrastrándolas fuera.

Otras características de la interfaz gráfica son:

- Textos de ayuda al situar el cursor del ratón sobre los elementos que componen la interfaz.
- Organización de los controles en barras de herramientas y menús de fácil acceso.
- Manejo de todos los elementos con el ratón.
- Posibilidad de cambiar el tamaño de los elementos de la interfaz.
- Utilización de varios formatos numéricos en la representación de los contenidos de registros, memorias y puertos.

Capítulo 3

Programación

El entorno de programación es la herramienta que permite escribir, ensamblar, montar y depurar programas escritos en ensamblador de Algorítmez o de Símplez, según el caso. Para acceder al entorno de programación basta con hacer *click* con el ratón en la pestaña de la ventana principal que lleva el nombre "**Programación**".

Debajo de la barra de herramientas hay dos áreas de texto. El tamaño de una respecto de la otra se puede cambiar desplazando la columna central que las separa. Para ello basta con pulsar el botón izquierdo del ratón cuando el cursor está sobre ella y, manteniéndolo pulsado, arrastrarla a la posición deseada.

El área con la etiqueta "programa" es la correspondiente a la edición. En ella se pueden escribir los programas fuente utilizando el lenguaje ensamblador descrito en el libro de texto (Fernández 1998). El manejo de ficheros se hace a través de los botones de la barra de herramientas, pudiéndose en cualquier momento guardar en disco el contenido del área de edición, cargar un programa fuente desde un fichero o comenzar a trabajar en un nuevo programa.

El área con la etiqueta "mensajes" mostrará el resultado de la operación de ensamblar cuando sea invocada. Si se detectan errores en el proceso de ensamblaje, se muestran en esta área de mensajes. Si por el contrario el proceso de ensamblaje termina con éxito, mostrará el código fuente acompañado del código binario generado y de la dirección relativa en memoria correspondiente a cada byte. El contenido del área de mensajes se explicará con más detalle en el capítulo correspondiente al ensamblaje.

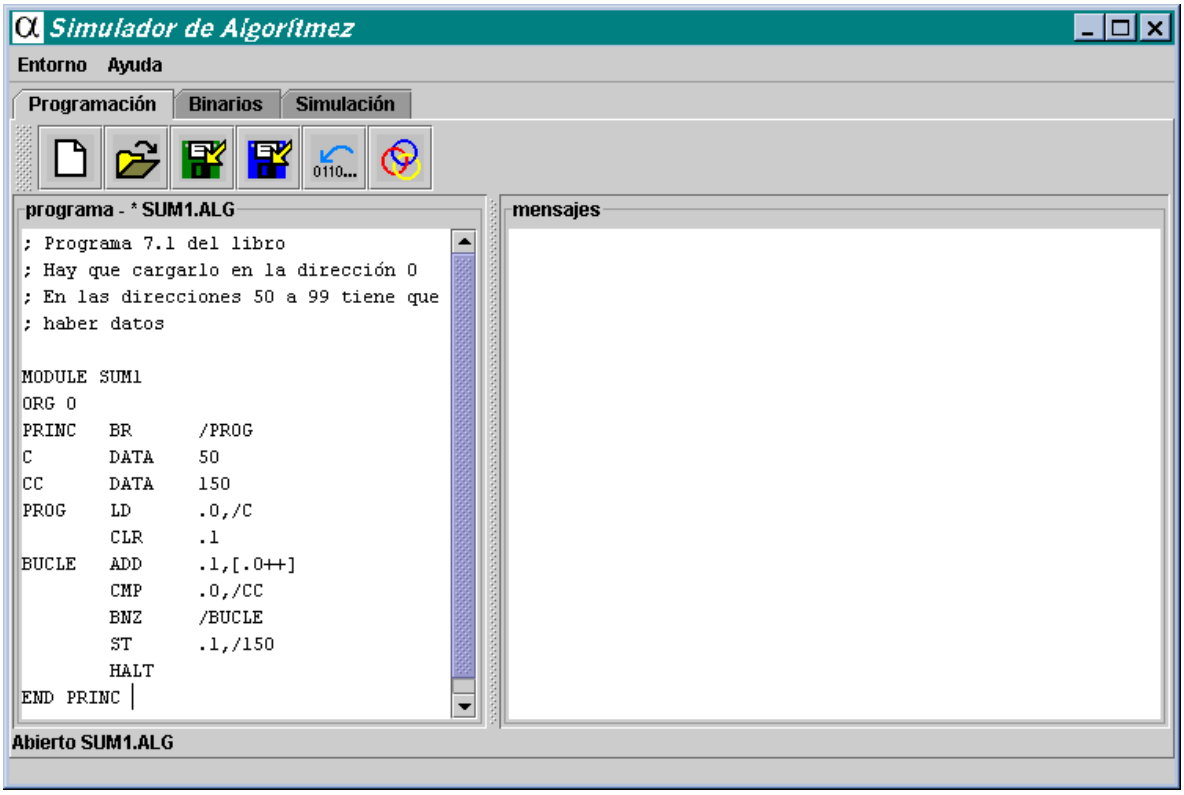


Figura 3.1. El entorno de programación.

3.1 Escritura de un programa

Los programas pueden escribirse directamente sobre el área de edición, o bien utilizar cualquier otro editor de texto plano, guardarlo a disco y luego cargarlo desde el entorno de programación. El fichero que almacene el programa fuente debe tener extensión "`.ALG`" en el caso de Algorítmez y "`.SIM`" en el caso de Símplez (las extensiones en letras mayúsculas).

Nota: Puesto que son muchos los sistemas operativos que distinguen en sus sistemas de ficheros entre letras mayúsculas y minúsculas, la aplicación también hace esta distinción.

Figura 3.2. Barra de herramientas.

Con el botón “Nuevo”  se limpian las áreas de texto para empezar a escribir un nuevo programa.

Pero se usa el botón “Abrir”  si lo que se quiere es abrir para edición un programa previamente grabado.

Con “Guardar”  o con “Guardar como...”  se graban en disco los cambios efectuados en los programas a través del área de edición. El último permite especificar el nombre y localización del fichero mientras que “Guardar” guarda el contenido del área de edición con los valores actuales de nombre y directorio. Sólo si el fichero no ha sido grabado en disco anteriormente, “Guardar” se comporta igual que “Guardar como...”.

La aplicación permite guardar los ficheros de texto con la extensión que el usuario desee. Pero si no se especifica extensión alguna (“.<extensión>”), se supondrá que se está editando un fichero en lenguaje ensamblador y al nombre indicado por el usuario se le añadirá la terminación “`.ALG`” en el caso de Algorítmez y “`.SIM`” en el caso de Símplez.

Una vez que el programa residente en el área de edición se corresponde con un fichero —bien porque se ha guardado o bien porque ha sido abierto desde disco — aparecerá el nombre del fichero acompañando a la etiqueta “programa –” de la parte superior del área de edición.

Siempre que se haya modificado el texto contenido en el área de edición, la aplicación advierte de que esos datos pueden diferir de los guardados en disco, y lo hace colocando un asterisco (`*`) entre la etiqueta “`programa -`” y el nombre del fichero, como puede verse en la figura 3.1. En esta circunstancia, si se intenta realizar alguna operación que suponga la pérdida de esos datos no guardados, como puede ser tratar de salir de la aplicación o pulsar los botones “Nuevo” o “Abrir...”, un cuadro de diálogo preguntará si se desea guardar los cambios.

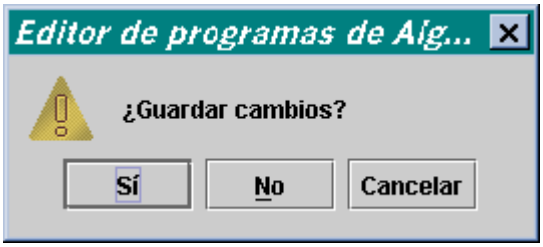


Figura 3.3. Diálogo modal de advertencia.

Como en cualquier editor existen una serie de operaciones asociadas a diferentes teclas o combinaciones de ellas, que permiten ejercer acciones sobre el texto escrito en el área de edición. Las disponibles se muestran en la siguiente tabla.

Tecla	**Operación**
Tabulador (→)	Avanza el cursor ocho posiciones respecto a la actual.
Retorno de carro (↵)	Termina la línea actual, crea una nueva y sitúa el cursor al comienzo de la misma.
Flecha arriba (↑)	Sitúa el cursor en la línea precedente, si existe.
Flecha abajo (↓)	Sitúa el cursor en la línea siguiente, si existe.
Flecha derecha (→)	Sitúa el cursor una posición a la derecha de la actual, si existe.
Flecha izquierda (←)	Sitúa el cursor una posición a la izquierda de la actual, si existe.
Inicio	Sitúa el cursor al comienzo de la línea actual.
Fin	Sitúa el cursor al final de la línea actual.
CTRL + Inicio	Sitúa el cursor al comienzo de la primera línea.
CTRL + Fin	Sitúa el cursor al final de la última línea.
Supr	Borra el carácter situado a la derecha del cursor.
Retroceso	Borra el carácter situado a la izquierda del cursor.
CTRL + A	Selecciona todo el texto presente en el área de edición.
CTRL + C	Copia el texto seleccionado al portapapeles.
CTRL + V	Pega en el área de edición, y a partir de la posición actual del cursor, el texto presente en el portapapeles.
CTRL + X	Corta el texto seleccionado y lo copia al portapapeles.

Tabla 3.1. Operaciones disponibles en el entorno de programación y teclas asociadas

Capítulo 4

Ensamblaje y montaje

4.1 Ensamblaje

El botón "Ensamblar"  del entorno de programación traduce a código binario el programa escrito en el área de edición, tal y como esté en ese momento. Se ensambla lo que hay escrito en área de texto y no lo grabado en fichero para evitar inconsistencias entre lo que hay en el área de edición y lo que aparece después de ensamblar en el área de mensajes. El programa se tiene que haber grabado en algún momento anterior como programa fuente de Algorítmez (con extensión "`.ALG`") o de Símplez (con extensión "`.SIM`"), según el caso. Esto es necesario porque el ensamblador necesita un nombre de referencia para utilizar en los ficheros que genera.

Si el programa tiene errores, aparecen indicados en la zona de mensajes. Si no se han detectado errores, se generan los ficheros:

- **Código objeto:** Fichero binario con extensión "`.OBA`" en el caso de Algorítmez y "`.OBS`" en el de Símplez, y cuya estructura se explica en el libro y en el capítulo siguiente, *interpretación de ficheros binarios*.

- **Programa ensamblado:** Fichero de texto con extensión `.ENA` en el caso de Algorítmez y `.ENS` en el de Símplez, y que es un volcado de lo que aparece en la zona de mensajes. Puesto que es un volcado de la zona de mensajes, en caso de haber errores, su contenido serán dichos errores.

- **Fichero mapa del módulo:** Fichero binario con extensión `.MP` que permite el desensamblado en la fase de simulación. La generación de este fichero se ha

suprimido en el caso de Símplez ya que la aplicación desarrollada para su simulación (Peña 2000) no contempla esta posibilidad.

Los mensajes de error que pueden aparecer en el proceso de ensamblaje están detallados y explicados en el capítulo 8.

Como puede verse en la figura 4.1, en Algorítmez el programa ensamblado se representa en hexadecimal, indicando la dirección de memoria relativa entre corchetes. La única diferencia con Símplez es que para este último se representa en octal el contenido de las posiciones de memoria correspondientes.

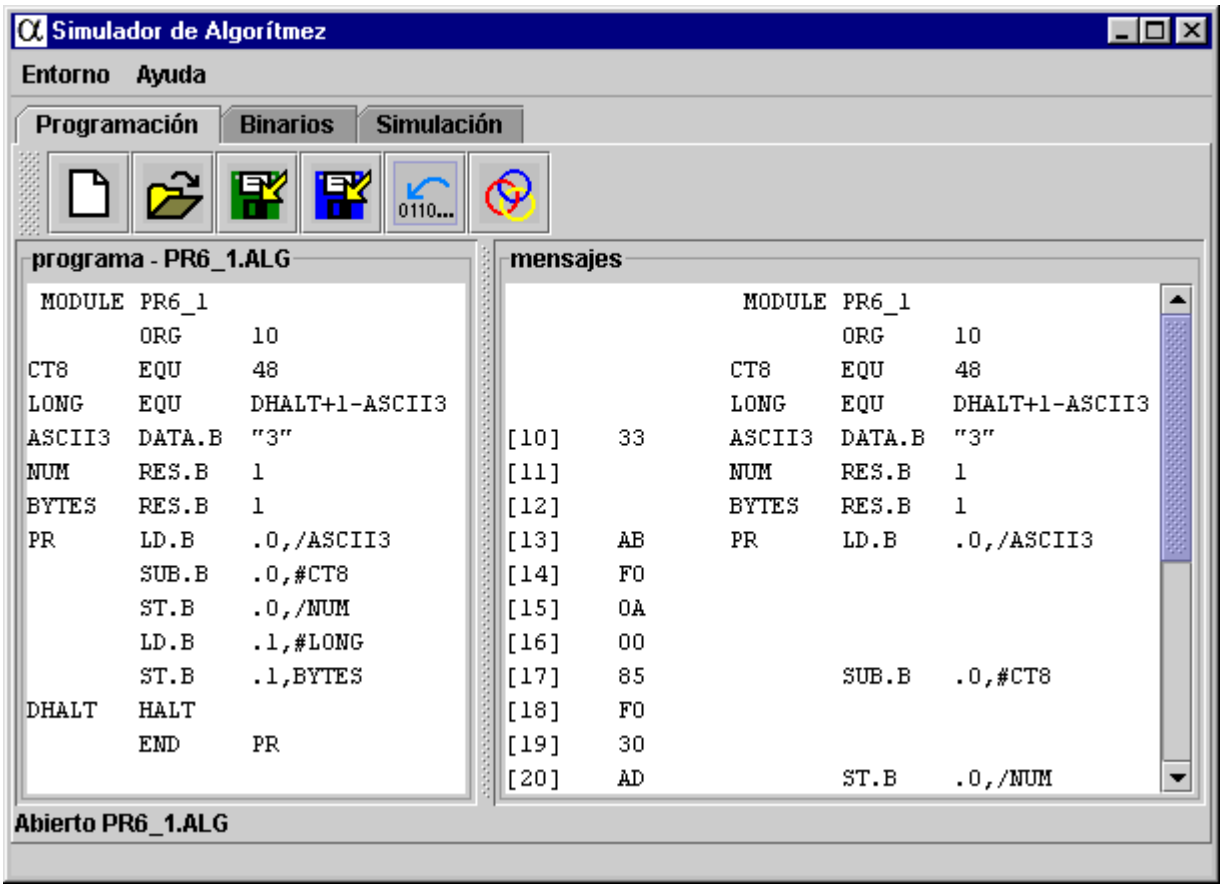


Figura 4.1. Entorno de programación de Algorítmez tras un ensamblado correcto.

Si a la hora de ensamblar se han producido errores, se indica línea, posición y tipo de error detectado, a fin de que pueda ser fácilmente subsanado.

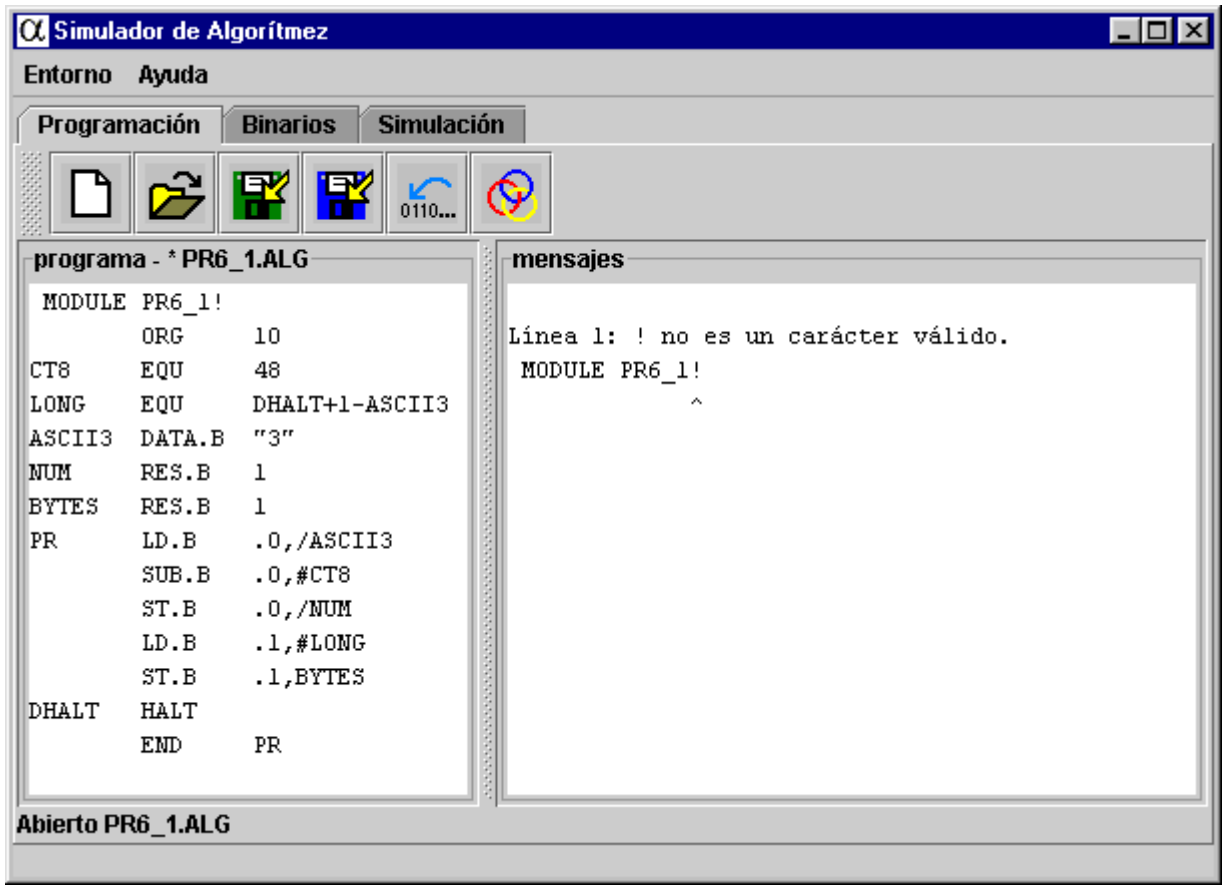


Figura 4.2. El entorno de programación tras detectar errores de ensamblado.

4.2 Montaje

El botón "Montar..."  del entorno de programación toma como entrada un módulo, que tiene que ser módulo principal — puede estar ya ensamblado ("`.OBA`" en el caso de Algorítmez y "`.OBS`" en el de Símplez) o en lenguaje ensamblador ("`.ALG`" en el caso de Algorítmez y "`.SIM`" en el de Símplez) — y realiza el montaje de éste con los módulos de los que importa símbolos.

Nota: El módulo a montar muchas veces no será aquel que se está editando, que no tiene porqué ser un módulo principal. Por eso, el módulo a montar se elige a través de una ventana de selección de ficheros.

Si se le da como entrada un programa fuente de Algorítmez (`.ALG`) — o de Símplez (`.SIM`) si es el caso —, o si no encuentra alguno de los módulos objeto que necesita, recurre al ensamblador para que los genere. Si el ensamblador encuentra errores, no se completa el montaje, y estos errores se pueden consultar en el fichero de extensión `.ENA` (o `.ENS` en el

caso de Símplez) que se genera para cada módulo. La aplicación avisa del evento a través de una ventana modal.

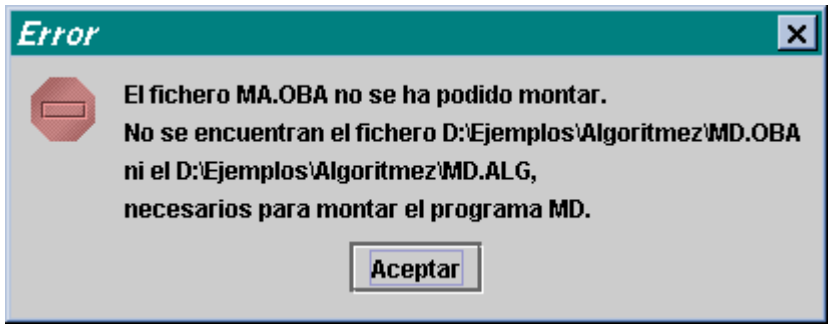


Figura 4.3. Ventana modal. Error en el montaje.

Si no hay errores y se pueden localizar o generar los ficheros objeto, la aplicación informa de ello a través de una ventana modal y se crean los ficheros:

- **Código montado:** Fichero binario ejecutable con extensión `.LNA` en el caso de Algorítmez y `.LNS` en el caso de Símplez; y cuya estructura se explica en el libro y en el capítulo siguiente, *interpretación de ficheros binarios*.

- **Fichero mapa del programa de Algorítmez:** Fichero binario con extensión `.MAP` que permite el desensamblado en la fase de simulación. Se construye a partir de los ficheros mapa del módulo (`.MP`) de cada uno de los módulos que intervienen. Si no se dispone del fichero mapa de alguno de los módulos no se generará el fichero mapa del programa y no será posible el desensamblado en la fase de simulación.

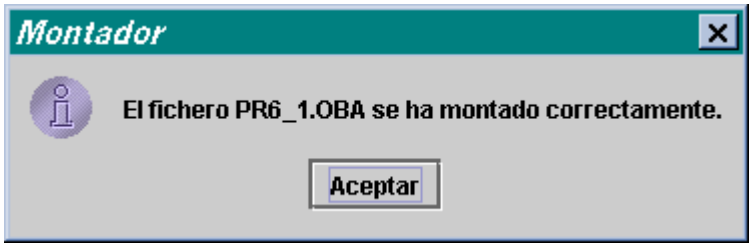


Figura 4.4. Ventana modal. Montaje correcto.

Capítulo 5

Interpretación de ficheros binarios

El intérprete de ficheros binarios dispone de un único botón,  , "**Abrir...**", con el que se puede elegir un fichero binario y abrirlo para su interpretación.

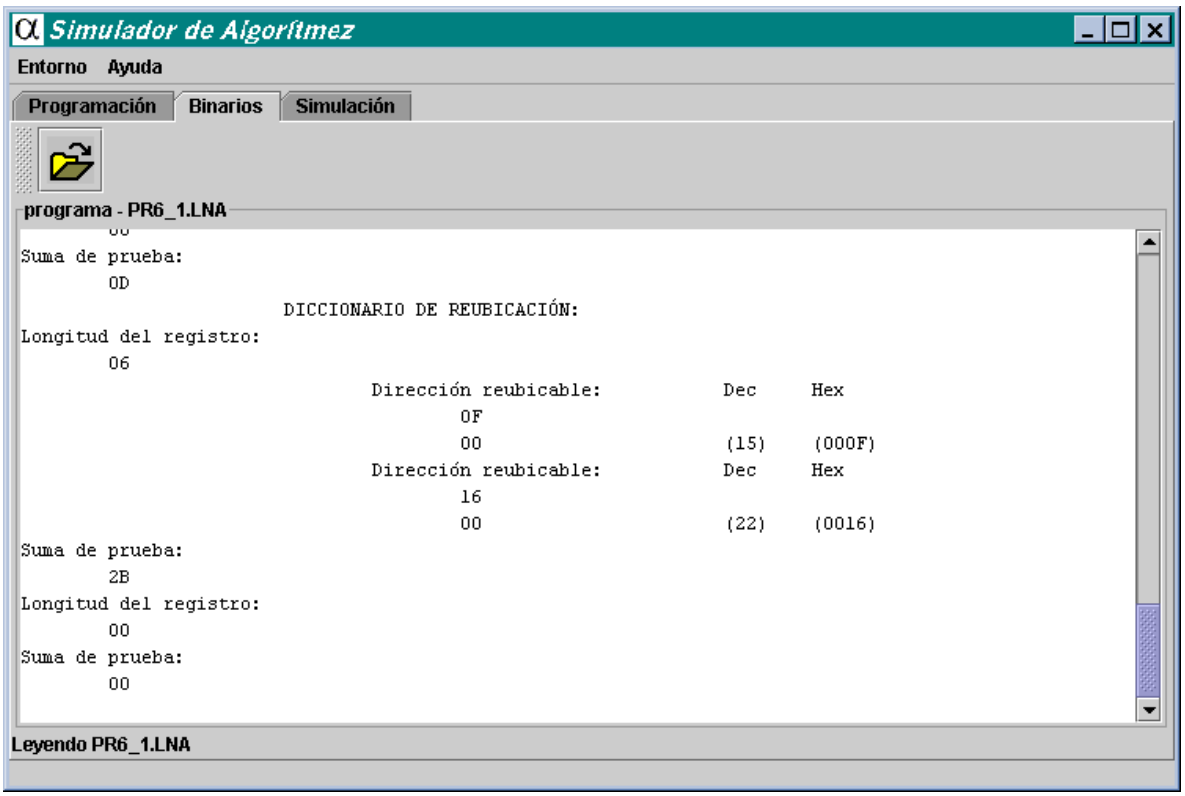


Figura 5.1. El intérprete de ficheros binarios.

En el área de texto se muestra la estructura interna del fichero binario así como el contenido y el significado de cada uno de los bytes que lo integran.

Los valores aparecen expresados en hexadecimal, y siempre que se estima útil, también en decimal. Los bytes correspondientes a código van acompañados de las posiciones relativas de memoria que ocuparán cuando el programa sea cargado.

A continuación se explica la estructura interna de los módulos objetos y de los módulos de carga, que son los dos tipos de ficheros binarios que la herramienta puede interpretar.

5.1 Módulos objeto

Dado un módulo fuente, el ensamblador de Álgorítmez genera un módulo objeto que consta de cuatro componentes:

1. *Código binario* de las instrucciones y datos, con indicación de la dirección de carga relativa de cada uno, y, si se trata de un módulo principal, de la dirección relativa de la primera instrucción a ejecutar.

2. *Diccionario de reubicación*; lista de las direcciones cuyo contenido puede ajustarse durante la carga (sumándoles el cargador reubicador una *constante de reubicación*).

3. *Tabla de símbolos externos*.

4. *Tabla de símbolos de acceso*.

Estas informaciones se almacenan en una secuencia de **registros** de longitud variable constituyendo un **fichero** que sirve de entrada para el montador. Estos *registros software* son estructuras de datos formadas por una sucesión de *campos*, donde cada campo consta de uno o más bytes.

Código binario de las instrucciones

Los registros de código binario tienen cuatro campos (figura 5.2):

- **NB** (un byte): número total de bytes del registro, *n*.

- **DIR** (dos bytes): dirección relativa en la MP del primero de los bytes de código.

- **Bytes de código** (*n*-4 bytes, siendo *n* el contenido del campo NB): traducción binaria de las instrucciones y datos del código fuente.

- **SUMPR** (un byte): suma de prueba. Es la suma binaria de todos los bytes precedentes, considerados como números sin signo, e ignorando desbordamiento. (El montador y el cargador vuelven a hacer la suma, para comprobar que no ha habido errores en la escritura o lectura del fichero en el disco).

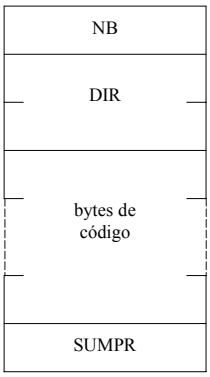


Figura 5.2. Formato de registro del código.

Como NB tiene un byte, si su contenido se interpreta como un número sin signo, el registro puede tener hasta 255 bytes. Sin embargo, con fines didácticos, la aplicación utiliza registros de 16 bytes para que al usuario le resulte más cómodo estudiarlos.

Cuando en el código fuente aparecen seudoinstrucciones RES se utilizan, aunque haya menos de 16 bytes, varios registros: en lugar de dejar bytes con contenido indiferente en el registro, se crean varios registros con los valores adecuados en los campos DIR para que en las direcciones de la MP entre uno y otro no se cargue nada.

En cualquier caso, y como a los registros del código les van a seguir otros, es preciso indicar dónde terminan estos registros. El convenio es que tras el último se pone un registro "vacío", caracterizado por (NB)=0. Este registro se aprovecha también para poner, en su caso, la dirección relativa de comienzo de la ejecución en el campo DIR.

Diccionario de reubicación

El diccionario de reubicación es, simplemente, una lista de las direcciones que el cargador reubicador deberá ajustar. El formato de registro tiene tres campos:

- **NB** (un byte): número total de bytes del registro, *n*.
- **DIRS** (*n*-2 bytes): con (*n*-2)/2 direcciones, siendo *n* el contenido del campo NB.
- **SUMPR** (un byte): suma de prueba.

El final del diccionario se indica añadiendo un registro vacío con dos campos: NB=0 y SUMPR=0.

Tabla de símbolos externos

La tabla de símbolos externos es una lista de nombres simbólicos en el programa fuente cuya traducción no ha sido posible acompañados de los nombres de los módulos de los que se importan y de las direcciones que ocupan dentro del módulo que se está ensamblando.

El formato adoptado por la tabla de símbolos externos tiene cuatro campos:

- **SÍMBOLO**: sucesión de bytes con las codificaciones ASCII del símbolo, terminados con H'00 (que indica el final de la cadena de caracteres del símbolo).

- **MÓDULO**: sucesión de bytes con las codificaciones ASCII del módulo del que se importa, terminados con H'00 (que indica el final de la cadena de caracteres del módulo).

- **ND** (1 byte): número de direcciones relativas dentro del módulo en las que aparece el símbolo importado.

- **DIRS** ((ND)*2 bytes): sucesión de direcciones relativas dentro del módulo en las que aparece el símbolo importado.

Esta estructura se repetirá tantas veces como símbolos se hayan importado. La tabla de símbolos externos podrá ocupar varios registros, que en el caso de esta aplicación son de 16 bytes, siguiendo las reglas de los demás registros que se recuerda que es:

- **NB** (un byte): número total de bytes del registro, *n*.

- **DATOS**: *n*-2 bytes con los datos que quepan de la tabla de símbolos externos, siendo *n* el contenido del campo NB.

- **SUMPR** (un byte): suma de prueba.

El final de la tabla de símbolos externos se indica añadiendo un registro vacío con dos campos: NB=0 y SUMPR=0.

Tabla de símbolos de acceso

La tabla de símbolos de acceso es una lista de etiquetas a las que pueden hacer referencia otros módulos, junto con sus valores.

El formato adoptado por la tabla de símbolos de acceso tiene dos campos:

- **SÍMBOLO**: sucesión de bytes con las codificaciones ASCII del símbolo, terminados con H'00 (que indica el final de la cadena de caracteres del símbolo).

- **DIR**: valor que el ensamblador ha asignado a ese símbolo en su tabla de de símbolos interna.

Esta estructura se repetirá tantas veces como símbolos se hayan exportado. La tabla de símbolos de acceso podrá ocupar varios registros, que en el caso de esta aplicación son de 16 bytes, siguiendo las reglas de los demás registros que se recuerda que es:

- **NB** (un byte): número total de bytes del registro, *n*.

- **DATOS**: *n*-2 bytes con los datos que quepan de la tabla de símbolos de acceso, siendo *n* el contenido del campo NB.

- **SUMPR** (un byte): suma de prueba.

El final de la tabla de símbolos de acceso se indica añadiendo un registro vacío con dos campos: NB=0 y SUMPR=0.

Cabecera y número mágico

El fichero en el que se almacena el módulo objeto contiene los bytes de todos estos registros, precedidos de una **cabecera** de catorce bytes. Los dos primeros bytes sirven para identificar el tipo de fichero, y su contenido se llama **número mágico**. Para los ficheros que contienen módulos objeto el ensamblador pone el número mágico H'A0A0, en el caso de Algorítmez, y H'B0B0, en el caso de Símplez. A cada tipo de fichero se le asigna un número mágico diferente. El montador comprueba que todos los ficheros que le entran contienen módulos objeto y tienen el número mágico adecuado.

En el caso de los módulos objeto, la cabecera contiene también el nombre del módulo, en doce bytes (si el nombre tiene menos de doce caracteres, se completa con módulos de carácter nulo H'00). Los doce primeros caracteres del nombre del fichero (que puede tener hasta catorce caracteres) deben coincidir con el nombre del módulo objeto almacenado en ese fichero.

5.2 Módulos de carga

El montador genera un módulo de carga a partir de uno o varios módulos objeto. Los módulos de carga contienen, como los módulos objeto, una serie de registros de código y un diccionario de reubicación. Las diferencias son:

- Los módulos de carga no contienen tablas de símbolos externos ni de símbolos de acceso.

- La cabecera es distinta: en los módulos de carga tiene cuatro bytes, dos con el número mágico H'0707 en el caso de Algorítmez, y H'0808, en el caso de Símplez, y otros dos con la longitud total (número de bytes) del código que ha de cargarse en la MP.

Por tanto, aunque sólo hubiese un módulo fuente sin referencias externas, el módulo objeto que genera el ensamblador debe pasar por el montador, para que éste cuente los bytes de código, añada la cabecera y prescinda de las tablas de símbolos externos y de acceso.

El montador recibe como dato de entrada el nombre del fichero donde se encuentra el **módulo objeto principal**. Éste es el módulo objeto cuya dirección de comienzo de ejecución (que debe figurar en la directiva `END` del programa fuente) será finalmente la dirección de comienzo de ejecución del módulo de carga. El módulo principal puede importar símbolos externos de otros módulos, que a su vez pueden importar símbolos de otros módulos, y así sucesivamente.

Capítulo 6

Carga y ejecución

En este capítulo se explican la mayor parte de las funciones que incorpora el entorno de simulación de Algorítmez. No obstante, se deja para el capítulo siguiente la explicación de la simulación de los procesos de entrada/salida.

El entorno de simulación de Algorítmez dispone de una barra de herramientas, donde se seleccionan las funciones que se quieren utilizar, y del área de monitorización, que permite conocer el estado de los principales componentes de Algorítmez a nivel de máquina convencional.

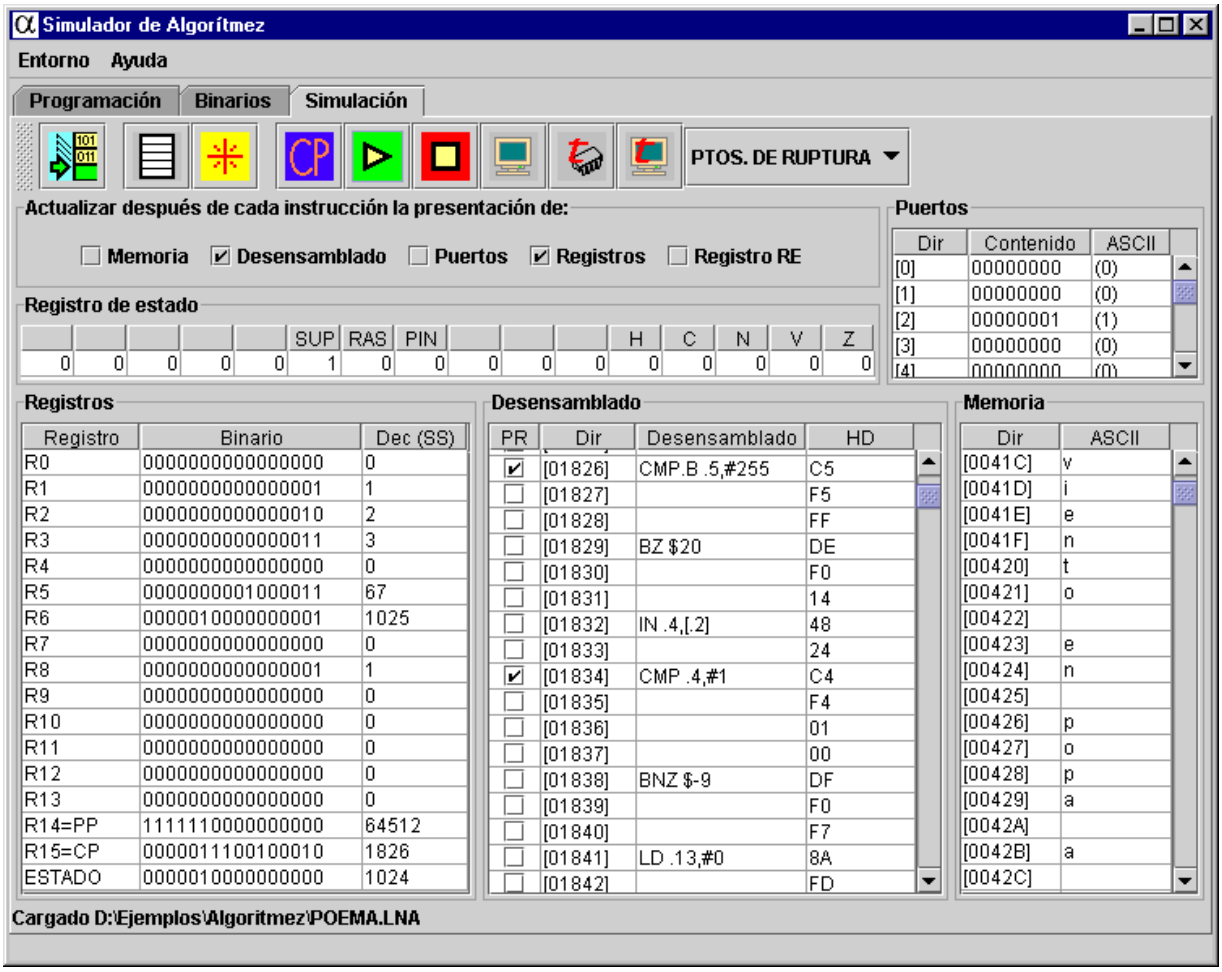


Figura 6.1. El entorno de simulación de Algorítmez.

6.1 Las funciones de la barra de herramientas

Como ya se adelantó en el capítulo de introducción, la barra de herramientas dispone de botones para cargar programas, borrar memorias y puertos, arrancar Algorítmez, modificar el contador de programa, avanzar en la ejecución, parar la ejecución, abrir y cerrar la ventana de entrada/salida, modificar los retardos de la UCP, modificar los retardos de entrada/salida y escoger el modo de ejecución.

Figura 6.2. La barra de herramientas del entorno de simulación.

Para simular un programa en el entorno de simulación de Algorítmez lo primero que hay que hacer es cargar dicho programa en la memoria del simulador. Para ello se dispone de un cargador reubicante.

Al pulsar el botón "**Cargar...**",  , el programa ofrece una ventana de selección de ficheros para moverse entre los directorios accesibles y escoger el programa — fichero de extensión "**.** LNA" — que se quiere cargar. Una vez seleccionado un programa, el **cargador reubicante** pregunta a partir de qué dirección de la memoria se desea que el programa sea cargado (a este dato se le llama **constante de reubicación**). El cargador verifica la coherencia del contenido del fichero y escribe entonces el programa a partir de dicha dirección, sumándole la constante de reubicación al contenido de todas las direcciones incluidas en el diccionario de reubicación.

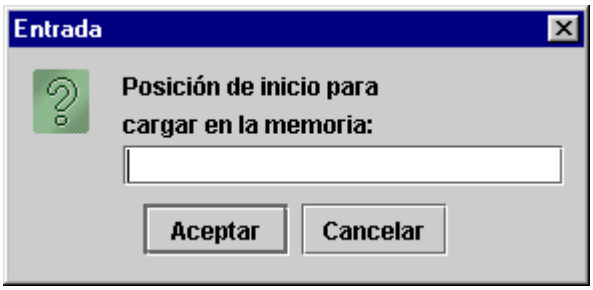


Figura 6.3. Ventana de diálogo para obtener la dirección de inicio.

Lo último que hace el cargador es modificar el registro contador de programa para que su contenido sea la dirección de memoria indicada como dirección de inicio en el programa recién cargado. Esto se habrá definido a través de la directiva END del módulo principal.

Nota: No todos los programas son reubicables; dependerá de cómo los haya escrito el programador.

En cualquier momento se puede hacer uso del botón  , "**Borrar memorias y puertos**", que pone a cero el contenido de todas las direcciones de la memoria principal y de los puertos. También borra el contenido de los registros de la memoria local, inicializando el puntero de pila al valor 64512 (H'FC00). Por ultimo, en el registro de estado escribe el valor 1024, que equivale a poner todos sus bits a cero salvo el bit "SUP", en el que se escribe un uno, indicando así que se entra en modo supervisor.

A través del botón "**Arrancar**",  , se accede a la siguiente opción disponible. Esta operación equivale al proceso que tendría lugar en un arranque normal del ordenador. En primer lugar borra las memorias y los puertos, e inicializa los registros puntero de pila y de estado como cuando se solicita "Borrar memorias y puertos" (explicado en párrafo anterior). A

continuación **carga un programa** para definir los vectores de interrupción y la tabla de periféricos, así como para cargar las rutinas de servicio y la de inicio. Por último, escribe en el registro contador de programa la dirección 268 e informa al usuario de la necesidad de ejecutar el programa cargado a partir de la posición indicada para que sea efectivo. La dirección de memoria 268 es la primera libre después de los vectores de interrupción.

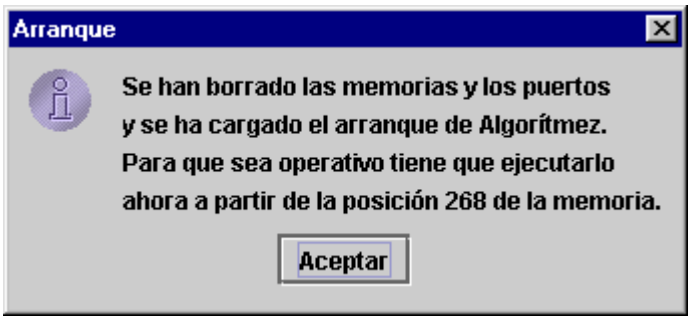


Figura 6.4. Ventana informativa para terminar el proceso de arranque.

El valor del CP puede modificarse a voluntad del usuario con el botón 

, "**Modificar el contador de programa...** ". El valor que se especifique será la dirección en la memoria de la siguiente instrucción a ejecutar. Este registro es el único componente de la simulación del hardware de Algorítmez que el usuario puede modificar en cualquier momento. El resto de registros así como la memoria y los puertos sólo se modifican a través de las instrucciones ejecutadas durante la simulación (no se están teniendo en cuenta los procesos de carga de programas y el uso de los botones de borrado e inicialización).

Se ha decidido impedir expresamente la modificación manual del contenido de la memoria y los registros por parte del usuario, porque se le ven más inconvenientes didácticos que ventajas, ya que puede crear confusiones y se aleja del proceso real en el que tampoco podría hacerse.

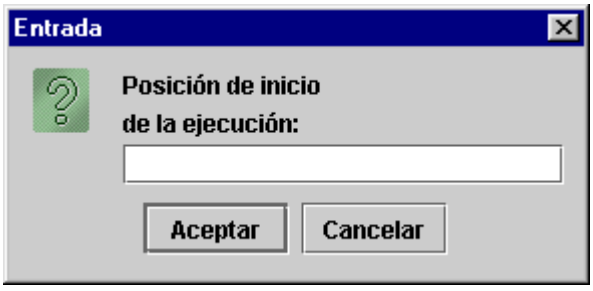
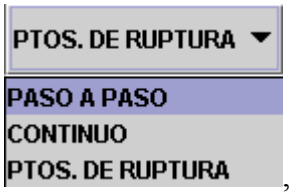


Figura 6.5. Ventana de diálogo para obtener el nuevo valor del contador de programa.

Con el botón  , "**Avanzar en la ejecución**", bien se inicia o bien se reanuda la simulación. Pone en marcha el secuenciador y comienza a ejecutar instrucciones.

El botón de parada,  , permite "**Parar la ejecución**" cuando se está ejecutando en modo continuo. La simulación se interrumpe dejando al secuenciador a la espera de una señal que le tiene que llegar del botón "Avanzar en la ejecución".

El "**Selector del modo de ejecución**", 

, es la lista desplegable que permite elegir uno de los tres modos de ejecución disponibles: paso a paso, continuo, y con puntos de ruptura.

- La ejecución de instrucciones en modo "**paso a paso**" se interrumpe después de cada instrucción ejecutada, esperando a que el usuario vuelva a pulsar el botón para reanudar el proceso.

- Cuando se selecciona el modo "**continuo**" la simulación tiene lugar hasta que el secuenciador encuentra una instrucción `HALT` o hasta que el usuario pulsa el botón de parada.

- La tercera posiblidad mencionada es el modo de ejecución "**con puntos de ruptura**", que ejecutan instrucciones en modo continuo hasta que llega a una instrucción en la que la primera dirección de memoria que ocupa ha sido marcada como "punto de ruptura". Allí se para, y al igual que en el modo de ejecución "paso a paso", queda a la espera de que el usuario vuelva a pulsar el botón de "Avanzar en la ejecución".

  Puede verse cómo establecer una instrucción como "punto de ruptura" en la descripción de la tabla de desensamblado.

El botón  , "**Abrir/Cerrar la pantalla de E/S**", muestra u oculta la ventana que simula la pantalla y el teclado de Algorítmez. De la simulación de los mecanismos de entrada y salida se hablará en detalle en el capítulo 7.

Normalmente será deseable que la simulación de los programas se haga lo más rápido que sea posible, pero puede haber momentos en que con fines didácticos sea necesario que la pantalla o el procesador simulen retardos. Así pueden apreciarse detalles que de otra manera pasarían desapercibidos a la vez que permite llevar a cabo simulaciones de las interrupciones de una forma más realista.

En Algorímez (como en cualquier otro ordenador) los procesos de entrada/salida, y en concreto la escritura en pantalla, son cientos de veces más lentos que la ejecución de instrucciones en la UCP. Como en este simulador, por su naturaleza gráfica, ambas operaciones tardan aproximadamente el mismo tiempo, se da la posibilidad de introducir retardos antes de escribir un carácter en la pantalla y antes de ejecutar una instrucción. Estos retardos se establecen en milisegundos de tiempo real y modifiarse en cualquier momento.

Con los botones  , "**Tiempo de UCP**", y  , "**Tiempo de ES**", es como se pueden establecer y modificar los retardos para la ejecución de instrucciones y la escritura en pantalla respectivamete. El proceso se realiza a través de sendas ventanas de diálogo.

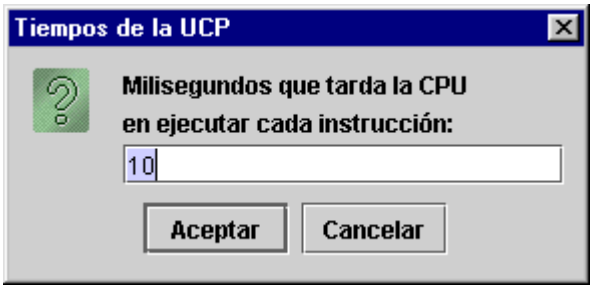


Figura 6.6. Ventana de diálogo para obtener el nuevo valor del retardo de la UCP.

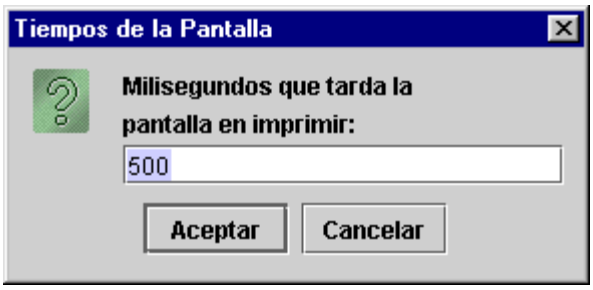


Figura 6.7. Ventana de diálogo para obtener el nuevo valor del retardo de la pantalla.

6.2 La tabla de registros

La tabla de registros muestra el contenido de los 16 registros de la memoria local y del registro de estado. La primera columna, "**Registro**" indica el nombre del registro, la segunda, "**Binario**" muestra su contenido en binario, y la tercera lo hace en decimal sin signo, en decimal con signo o en hexadecimal.

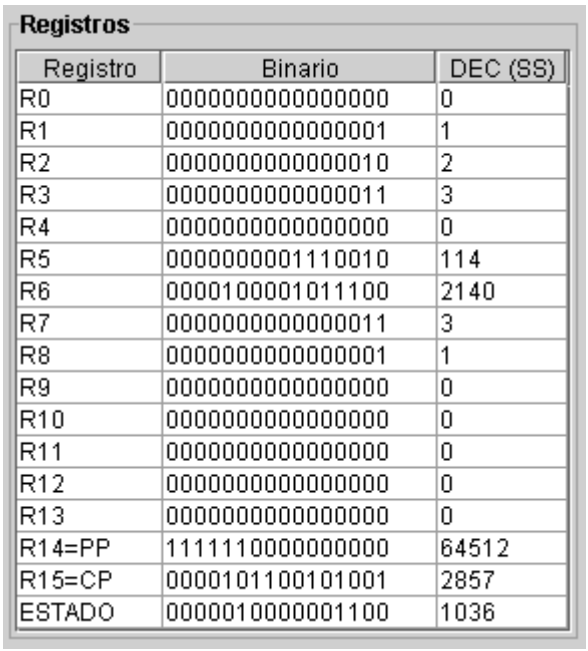


Figura 6.8. La tabla de registros.

Para que la tercera columna cambie el formato de representación del contenido del registro basta con ir haciendo *click* con el ratón sobre la cabecera de la dicha columna. La propia cabecera cambiará de nombre con esta operación, pudiéndose leer en ella "**DEC (SS)**" para la representación decimal sin signo, "**DEC (CS)**" para decimal con signo y "**HD**" para hexadecimal.

Nota: Los registros puntero de pila (R14) y contador de programa (CP) siempre hacen referencia a direcciones de la memoria principal; y el registro de estado (RE) sólo tiene sentido como conjunto de bits donde cada uno representa una situación o circunstancia. Por ello, los tres registros se representan siempre como valores positivos aunque se haya seleccionado para toda la tabla la representación en "decimal con signo".

6.3 La tabla del registro de estado

Dada la estructura especial del registro de estado se le ha dedicado una tabla individual aunque también aparece en la tabla que muestra los registros de la memoria local. La tabla del registro de estado desglosa el contenido de este registro en binario, especificando el valor y el significado de cada uno de sus bits.

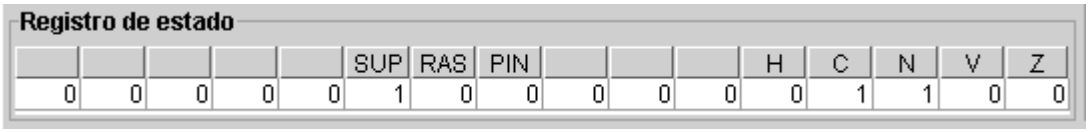


Figura 6.9. La tabla del registro de estado.

6.4 La tabla de puertos

La tabla de puertos muestra el contenido de los 256 puertos que puede direccionar Algorítmez en los formatos más utilizados. La primera columna, "**Dir**" indica el número de puerto, la segunda, "**Contenido**" muestra su contenido en binario y la tercera, "**ASCII**" lo hace en ASCII.

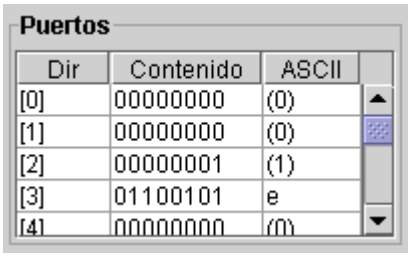


Figura 6.10. La tabla del puertos.

6.5 La tabla de la memoria

La tabla de la memoria muestra el contenido de los 65.536 bytes de la memoria principal de Algorítmez en hexadecimal y ASCII. La primera columna, "**Dir**" indica la dirección de memoria en decimal o en hexadecimal y la segunda muestra su contenido en decimal sin signo, en decimal con signo, en ASCII o en hexadecimal.

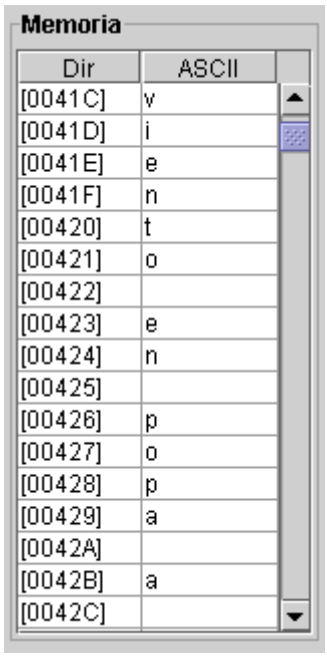


Figura 6.11. La tabla de memoria.

Para que la primera columna, que indica las direcciones de memoria, cambie el formato de representación basta con ir haciendo *click* con el ratón sobre la cabecera de la dicha columna y las direcciones de memoria se podrán ver en decimal o en hexadecimal.

Además, haciendo *doble-click* sobre la cabecera de esta columna de direcciones, se abre una ventana de diálogo para que el usuario introduzca la dirección de memoria que desea ver. La aplicación, entonces, desplaza la tabla de forma que la parte visible coincida con la dirección de memoria elegida.

Figura 6.12. Ventana de diálogo para obtener la dirección de memoria a mostrar.

Como con la columna de direcciones, para que la tercera columna cambie el formato de representación basta con ir haciendo *click* con el ratón sobre la cabecera de la dicha columna. La propia cabecera cambiará de nombre con esta operación, pudiéndose leer en ella "**DEC**

(SS)" para la representación decimal sin signo, "**DEC (CS)**" para decimal con signo, "**ASCII**" para ASCII y "**HD**" para hexadecimal.

6.6 La tabla de desensamblado

La tabla de desensamblado es como la tabla de memoria pero incluyendo las casillas para marcar los "puntos de ruptura" y el desensamblado de las instrucciones en aquellos programas en que sea posible. Además, la tabla de desensamblado **se desplaza automáticamente** de forma que la instrucción a ejecutar siempre esté en la parte visible de la tabla.

La primera columna, "**PR**", tiene en cada fila una casilla que se puede marcar si se quiere que esa dirección de memoria sea un "**punto de ruptura**"; la segunda columna, "**Dir**", indica la dirección de memoria en decimal o en hexadecimal; la tercera, "**Desensamblado**", muestra la instrucción desensamblada siempre que es posible; y en la cuarta puede verse el contenido de la posición de memoria en decimal sin signo, en decimal con signo, en ASCII o en hexadecimal.

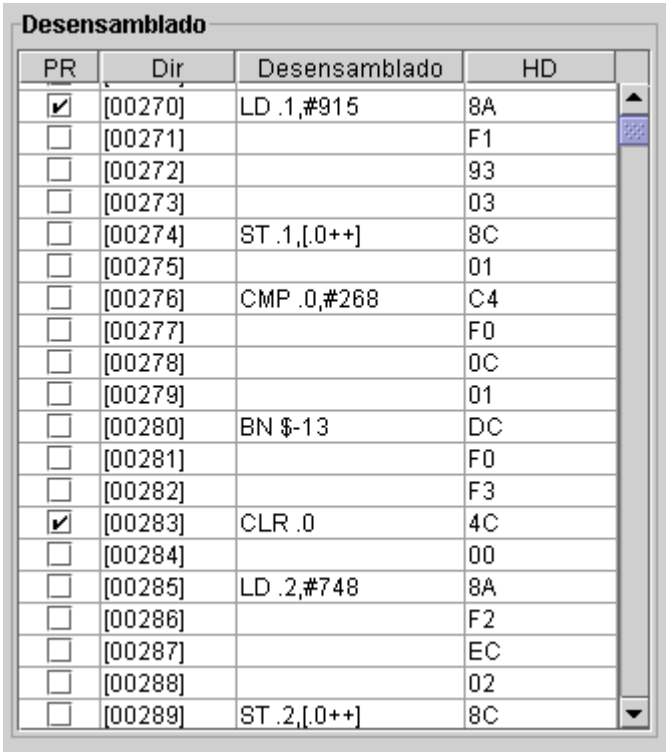


Figura 6.13. La tabla de desensamblado con puntos de ruptura marcados.

Para establecer que una instrucción sea "**punto de ruptura**" basta con marcar con el ratón el cuadro de la primera columna en la línea correspondiente a la primera dirección de memoria

que ocupa la instrucción. Para que deje de ser "punto de ruptura" se elimina la marca igualmente con el ratón. Para borrar de una vez todos los puntos de ruptura de la tabla se hace *click* en la cabecera de la primera columna (**PR**). Esto hace que aparezca una ventana de diálogo que pregunta si efectivamente se desea borrar todos los puntos de ruptura, pudiéndose aceptar o cancelar la acción en ese momento.

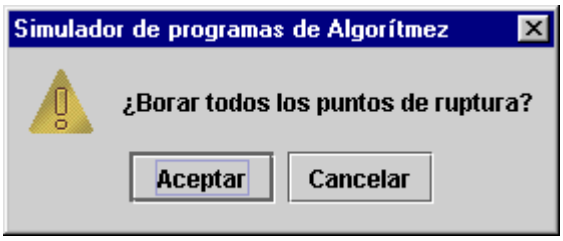


Figura 6.14. Ventana de diálogo para confirmar el borrado de todos los puntos de ruptura.

El simulador ejecuta instrucciones completas, así que sólo tendrá en cuenta los puntos de ruptura que se encuentren en direcciones de memoria que sean principio de una instrucción; las marcas que no cumplan esta condición serán simplemente ignoradas.

Por esto es por lo que se incluyen los "puntos de ruptura" en esta tabla, que es donde gracias a la representación desensamblada de las instrucciones puede saberse dónde empieza cada una.

Igual que la tabla de la memoria, para que la columna que indica direcciones cambie el formato de representación basta con ir haciendo *click* con el ratón sobre la cabecera de la dicha columna y las direcciones de memoria se podrán ver en decimal o en hexadecimal.

También, haciendo *doble-click* sobre la cabecera de la columna, se abre una ventana de diálogo para que el usuario introduzca la dirección de memoria que desea ver. La aplicación, entonces, desplaza la tabla de forma que la parte visible coincida con la dirección de memoria elegida.

Figura 6.15. Ventana de diálogo para obtener la dirección de memoria a mostrar.

Igualmente, para que la cuarta columna cambie el formato de representación basta con ir haciendo *click* con el ratón sobre la cabecera de la dicha columna. La propia cabecera cambiará de nombre con esta operación, pudiéndose leer en ella "**DEC (SS)**" para la representación decimal sin signo, "**DEC (CS)**" para decimal con signo, "**ASCII**" para ASCII y "**HD**" para hexadecimal.

Puede parecer que la **tabla de memoria** no es necesaria al no aportar ninguna ventaja sobre la **tabla de desensamblado**. Sin embargo, como es normal que el programa tenga una zona de instrucciones y otra de datos, es muy útil disponer de las dos tablas y usarlas a la vez de la siguiente manera: Como la tabla de desensamblado se desplaza automáticamente para mostrar la instrucción en curso y además permite la lectura de la instrucción desensamblada, se puede usar esta tabla para seguir el flujo del programa. Mientras tanto en la tabla de memoria (cuya parte visible queda estática salvo que el usuario utilice la barra de desplazamiento) se puede monitorizar el área de datos y ver el contenido de las posiciones de memoria de las que se lee y en las que se escribe.

6.7 Selectores de refrescos

Actualizar la presentación de las tablas es lo que más ralentiza la simulación. Por eso, cuando se simula en modo continuo, se puede elegir no actualizar en pantalla todas las tablas después de cada instrucción. Para ello, en la parte superior del área de monitorización se encuentran cinco casillas para seleccionar qué tablas actualizan la representación de sus contenidos después de cada instrucción ejecutada en modo continuo o con puntos de ruptura. Cuando se simula en modo "paso a paso" o cuando se llega a un punto de ruptura, se actualiza el contenido de todas las tablas independientemente del estado de los selectores de refrescos, cosa que también sucede cuando se pulsa el botón de "parar la ejecución" y después de ejecutar la instrucción HALT.

Figura 6.16. Selectores de refrescos en la parte superior del área de monitorización.

Capítulo 7

Entrada/Salida

En Algorítmez el espacio de direccionamiento de la entrada/salida es independiente de la MP. Concretamente, pueden direccionarse 256 puertos (direcciones de ocho bits, comprendidas entre D'0 y D'255), y cada uno de estos puertos es de un byte.

El entorno de simulación supone que como dispositivos de entrada/salida sólo están conectados el teclado y la pantalla. Aunque estos periféricos están presentes en todo momento para la aplicación, su representación puede mostrarse o no, a voluntad del usuario.

Con el botón  , "**Abrir/Cerrar la pantalla de E/S**" se muestra u oculta la ventana que simula la pantalla de Algorítmez a la vez que permite controlar la entrada de datos por teclado.

Con el botón superior 

, Clear Screen, la pantalla se borra. Mientras la ventana está activa (cuenta con el foco de acción) aparece con fondo negro y letras blancas. En esta situación, el hecho de pulsar una tecla, se considera como que se ha hecho sobre el teclado de Algorítmez. Si no tiene el foco, se invierten los colores, y al simulador no le afecta ninguna acción sobre el teclado.

La aplicación permite que le lleguen a la simulación de la UCP de Algorítmez las interrupciones que la pantalla y el teclado generan, siempre que no estén inhibidas a través del registro de estado.

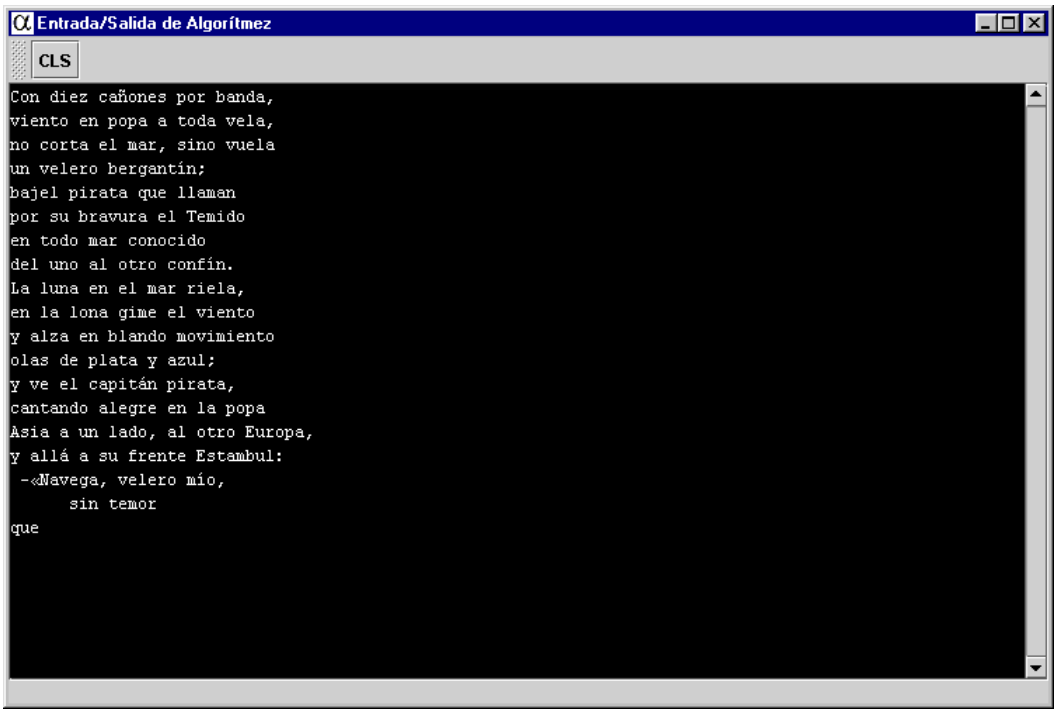


Figura 7.1. ventana de entrada/salida activa.

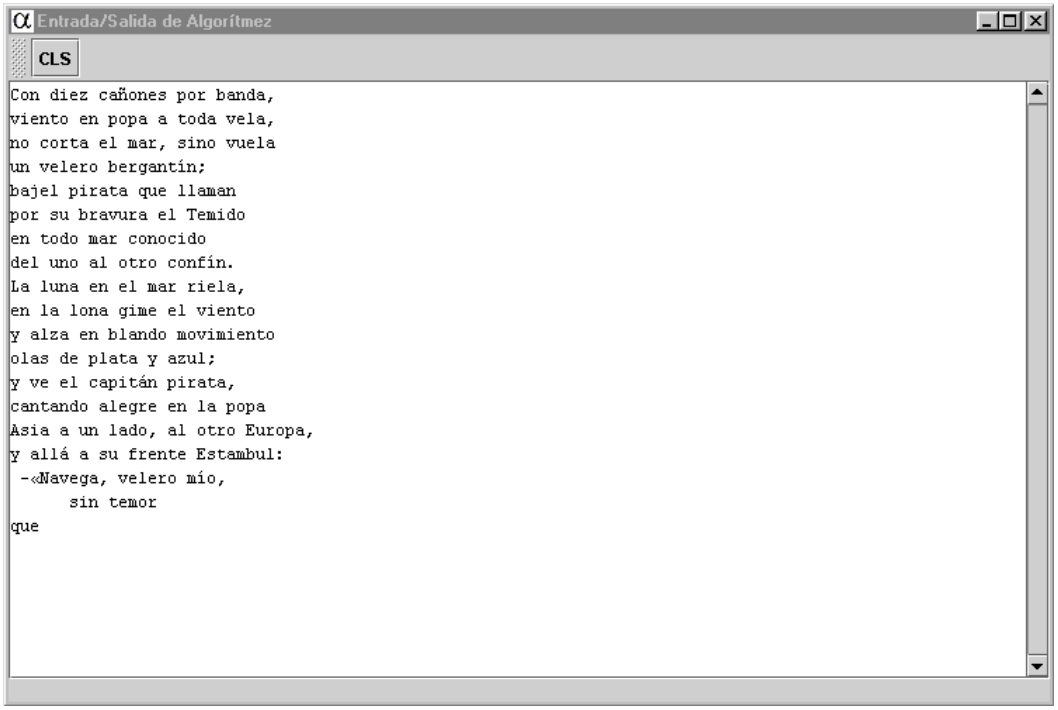


Figura 7.2. ventana de entrada/salida no activa.

7.1 Representación de los caracteres

La comunicación con el usuario a través de los periféricos se realiza mediante cadenas de caracteres. Para representar cada carácter de la cadena se utiliza el código ISO Latin1. Los caracteres que forman parte de una cadena se almacenan en posiciones consecutivas de memoria. El código ISO Latin1, como se verá a continuación, utiliza 8 bits para codificar un carácter; o lo que es lo mismo, media palabra de la memoria de Algorítmez.

Es importante indicar que cuando el usuario quiera introducir un número en Algorítmez, o cuando se quiera sacar un número por pantalla será necesario representar éste como una cadena de caracteres. Supóngase que se quiere sacar por pantalla el número 241. Si se transfiere desde el acumulador al puerto de datos de la pantalla el valor 241, tras cumplirse el ciclo de pantalla aparecerá en ésta el carácter "ñ" en lugar del número 241. Para lograr que aparezca el número, se debe transferir en operaciones consecutivas los caracteres "2", "4" y "1" al puerto de datos de la pantalla. Es decir, se deben introducir consecutivamente en el acumulador los números "50", "52" y "49", que son los que se corresponden en ISO Latin1 con "2", "4" y "1" respectivamente. Por tanto es labor del programa el traducir de cadenas de caracteres a números, o viceversa, cuando quiera sacarlos por pantalla, o introducirlos por teclado.

Se han implementado las siguientes secuecias:

Dec	Hex	Secuencia
0	00	Borrado de pantalla
1	01	Borrado de 1ª línea

Tabla 7.1. Tabla de secuencias no ASCII implementadas.

Además, del juego de caracteres de ISO Latin1, los que entiende Algorítmez son los mostrados en las tablas siguientes.

Dec	Hex	Secuencia de escape
9	09	TAB (horizontal tab)
13	0D	CR (carriage return)

Tabla 7.2. Tabla de secuencias ASCII implementadas.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20	SPACE	64	40	@	96	60	‘
33	21	¡	65	41	A	97	61	a
34	22	"	66	42	B	98	62	b
35	23	#	67	43	C	99	63	c
36	24	$	68	44	D	100	64	d
37	25	%	69	45	E	101	65	e
38	26	&	70	46	F	102	66	f
39	27	’	71	47	G	103	67	g
40	28	(	72	48	H	104	68	h
41	29	)	73	49	I	105	69	i
42	2A	*	74	4A	J	106	6A	j
43	2B	+	75	4B	K	107	6B	k
44	2C	,	76	4C	L	108	6C	l
45	2D	-	77	4D	M	109	6D	m
46	2E	.	78	4E	N	110	6E	n
47	2F	/	79	4F	O	111	6F	o
48	30	0	80	50	P	112	70	p
49	31	1	81	51	Q	113	71	q
50	32	2	82	52	R	114	72	r
51	33	3	83	53	S	115	73	s
52	34	4	84	54	T	116	74	t
53	35	5	85	55	U	117	75	u
54	36	6	86	56	V	118	76	v
55	37	7	87	57	W	119	77	w
56	38	8	88	58	X	120	78	x
57	39	9	89	59	Y	121	79	y
58	3A	:	90	5A	Z	122	7A	z
59	3B	;	91	5B	[	123	7B	{
60	3C	<	92	5C	\	124	7C	|
61	3D	=	93	5D	]	125	7D	}
62	3E	>	94	5E	^	126	7E	~
63	3F	?	95	5F	_	127	7F	DEL

Tabla 7.3. Tabla 1 de caracteres ISO Latin 1.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
160	A0	SPACE	192	C0	À	224	E0	à
161	A1	¡	193	C1	Á	225	E1	á
162	A2	¢	194	C2	Â	226	E2	â
163	A3	£	195	C3	Ã	227	E3	ã
164	A4	¤	196	C4	Ä	228	E4	ä
165	A5	¥	197	C5	Å	229	E5	Å
166	A6	¦	198	C6	Æ	230	E6	Æ
167	A7	§	199	C7	Ç	231	E7	Ç
168	A8	¨	200	C8	È	232	E8	È
169	A9	©	201	C9	É	233	E9	É
170	AA	ª	202	CA	Ê	234	EA	Ê
171	AB	«	203	CB	Ë	235	EB	Ë
172	AC	¬	204	CC	Ì	236	EC	Ì
173	AD	-	205	CD	Í	237	ED	Í
174	AE	®	206	CE	Î	238	EE	Î
175	AF	¯	207	CF	Ï	239	EF	Ï
176	B0	°	208	D0	Ð	240	F0	Ð
177	B1	±	209	D1	Ñ	241	F1	Ñ
178	B2	²	210	D2	Ò	242	F2	Ò
179	B3	³	211	DD	Ó	243	F3	Ó
180	B4	´	212	D4	Ô	244	F4	Ô
181	B5	µ	213	D5	Õ	245	F5	Õ
182	B6	¶	214	D6	Ö	246	F6	Ö
183	B7	•	215	D7	×	247	F7	÷
184	B8	¸	216	D8	Ø	248	F8	Ø
185	B9	¹	217	D9	Ù	249	F9	Ù
186	BA	º	218	DA	Ú	250	FA	Ú
187	BB	»	222	DB	Û	251	FB	Û
188	BC	¼	220	DC	Ü	252	FC	Ü
189	BD	½	221	DD	Ý	253	FD	Ý
190	BE	¾	222	DE	Þ	254	FE	Þ
191	BF	¿	223	DF	ß	255	FF	Ÿ

Tabla 7.4. Tabla 2 de caracteres de ISO Latin 1.

Capítulo 8

Mensajes de error del proceso de ensamblaje

En este capítulo se explican los mensajes de error con los que puede encontrarse el usuario a la hora de ensamblar un programa escrito en lenguaje ensamblador de Símplez o de Algorítmez. Es importante tener en cuenta que los comentarios, las líneas en blanco y los espacios al final de las líneas son ignorados por el ensamblador. Tan sólo cuentan para el cómputo de líneas y de caracteres por línea, pero pueden aparecer en cualquier punto del módulo sin que afecten al código.

8.1 Mensajes relativos a errores léxicos

`La cadena debe acabar en la misma línea que empezó. Falta el cierre de las comillas (").`

Cuando en una línea se define de forma incompleta una cadena de caracteres, al haberse encontrado un punto en el que las comillas (") se abren sin que se vea que sean cerradas más tarde. Las cadenas deben empezar y terminar en la misma línea.

`Las palabras no pueden tener más de 12 caracteres.`

Cuando se está reconociendo una palabra donde hasta el momento todos los caracteres son válidos, pero se llega a un punto en que el número de caracteres es mayor que doce.

`El número queda fuera del rango admisible por` *`ordenador`*`.`

Cuando el número leído queda fuera del rango admisible por el ordenador que se está simulando.

`Error de formato del número.`

Cuando no ha sido posible decodificar el número a partir de los caracteres leídos y siempre que no se haya podido facilitar algún error más específico.

`El carácter guión bajo (_) sólo es válido dentro de un símbolo y nunca como primer carácter.`

Cuando aparece el carácter *guión bajo* (_) fuera de un símbolo o intentando ser el primer carácter de un símbolo.

`El carácter comilla simple (') sólo es válido para indicar la base en que se ha escrito un número.`

Cuando aparece el carácter *comilla simple* (') fuera de la definición de la base de un número.

`carácter` `no es un carácter válido para el lenguaje ensamblador de` *`ordenador`*`.`

Cuando es leído un carácter que no pertenece al juego de caracteres admisible por el lenguaje ensamblador del ordenador que se está simulando. De todas formas, dentro de una cadena (caracteres entre comillas) es aceptado cualquier carácter.

8.2 Mensajes relativos a errores sintácticos y semánticos de cualquiera de los ordenadores

Leyendo la línea de la directiva 'MODULE'

`La primera línea de todo módulo fuente tiene que ser la correspondiente a la directiva 'MODULE'.`

Cuando la primera línea del módulo fuente, sin tener en cuenta comentarios y líneas en blanco, no es la correspondiente a la directiva '`MODULE`'.

`Después de la directiva 'MODULE' tiene que dejarse un espacio y a continuación debe ir el nombre del módulo.`

Cuando inmediatamente después de la directiva '`MODULE`' aparece un carácter diferente de un espacio en blanco o un tabulador.

`El nombre del módulo debe ser una cadena que empiece con una letra y esté formada por letras, dígitos y/o caracteres guión bajo (_).`

Cuando después de la directiva '`MODULE`' y de un espacio no se ha encontrado un nombre de módulo válido.

`El nombre del fichero no coincide con el nombre que acompaña a 'MODULE'.`

Cuando los doce primeros caracteres del nombre del fichero no coinciden con el nombre que acompaña a la directiva '`MODULE`'.

`Dentro del nombre del módulo sólo son válidos: letras, dígitos y el carácter guión bajo (_).`

Cuando después de la directiva '`MODULE`' se ha leído un espacio y una serie de caracteres válidos como nombre de módulo a los que sigue un carácter que no es letra, dígito ni el carácter guión bajo (_) y que tampoco es un espacio.

`Después de 'MODULE <nombre>' se esperaba fin de línea.`

Cuando después de la directiva '`MODULE`' se ha leído un espacio y un nombre de módulo válido, pero en lugar de terminar ahí la línea, aparecen un espacio seguido de más caracteres.

Leyendo líneas con la directiva '`FROM... IMPORT`'

`En la directiva 'FROM... IMPORT', después de 'FROM' tiene que dejarse un espacio y a continuación debe ir el nombre del módulo del que se importa.`

Cuando leyendo la cabecera inmediatamente después de '`FROM`' aparece un carácter diferente de un espacio en blanco o un tabulador.

`Se esperaba el nombre del módulo del que se importa. Un nombre de módulo debe ser una cadena que empiece con una letra y esté formada por letras, dígitos y/o caracteres guión bajo (_).`

Cuando leyendo la directiva '`FROM... IMPORT`', después de '`FROM`' y de un espacio no se ha incluido un nombre de módulo válido.

`Después del nombre del módulo del que se importa tiene que dejarse un espacio que debe ir seguido de 'IMPORT'.`

Cuando leyendo la directiva '`FROM... IMPORT`', después de '`FROM`' se ha leído un espacio y una serie de caracteres válidos como nombre de módulo a los que sigue un carácter que no es letra, dígito ni el carácter guión bajo (_) y que tampoco es un espacio.

`Se esperaba 'IMPORT'.`

Cuando leyendo la directiva '`FROM... IMPORT`', después de '`FROM`' se ha leído un espacio, un nombre de módulo válido y otro espacio al que no sigue '`IMPORT`'.

`En la directiva 'FROM... IMPORT', después de 'IMPORT' tiene que dejarse un espacio y a continuación debe ir la lista de símbolos a importar.`

Cuando leyendo la directiva '`FROM... IMPORT`', después de '`IMPORT`' aparece un carácter diferente de un espacio en blanco o un tabulador.

`Se esperaba un nombre de símbolo a importar. Un nombre de símbolo debe ser una cadena que empiece con una letra y esté formada por letras, dígitos y/o caracteres guión bajo (_).`

Cuando leyendo la directiva '`FROM... IMPORT`', después de '`IMPORT`' y de un espacio no se ha encontrado un nombre de símbolo válido.

`Se está tratando de importar dos veces el mismo símbolo.`

Cuando leyendo la directiva '`FROM... IMPORT`', se encuentra con que se está importando un símbolo previamente importado.

`Después del nombre de un símbolo externo tiene que, o bien terminar la línea, o bien aparecer una coma (,) seguida de más símbolos a importar.`

Cuando leyendo la directiva '`FROM... IMPORT`', cuando se está leyendo un símbolo externo, se han leído una serie de caracteres válidos a los que sigue un carácter que no es letra, dígito ni el carácter guión bajo (`_`) y que tampoco es un espacio ni una coma (`,`).

Y también cuando leyendo la directiva '`FROM... IMPORT`', después del espacio opcional posterior a un símbolo externo, ni acaba la línea ni aparece una coma (`,`) seguida de más símbolos a importar, sino que aparece otro carácter no esperado.

Leyendo líneas con la directiva '`EXPORT`'

`Después de la directiva 'EXPORT' tiene que dejarse un espacio y a continuación debe ir la lista de símbolos de acceso.`

Cuando leyendo la cabecera inmediatamente después de '`EXPORT`' aparece un carácter diferente de un espacio en blanco o un tabulador.

`Se esperaba un nombre de símbolo de acceso. Un nombre de símbolo debe ser una cadena que empiece con una letra y esté formada por letras, dígitos y/o caracteres guión bajo (_).`

Cuando después de la directiva '`EXPORT`' y de un espacio no se ha incluido un nombre de símbolo válido.

`Se está tratando de exportar dos veces el mismo símbolo.`

Cuando leyendo la directiva '`EXPORT`', se encuentra con que se está exportando un símbolo previamente exportado.

`Se está tratando de exportar un símbolo no definido en este módulo.`

Cuando leyendo la directiva '`EXPORT`', se encuentra con que se está tratando de exportar un símbolo que no ha sido definido en ningún punto del módulo.

`No puede exportarse un símbolo no reubicable.`

Cuando leyendo la directiva '`EXPORT`', se encuentra con que se está tratando de exportar un símbolo no reubicable.

`Después del nombre de un símbolo de acceso tiene que, o bien terminar la línea, o bien aparecer una coma (,) seguida de más símbolos a exportar.`

Cuando leyendo la directiva '`EXPORT`', después de un símbolo de acceso, ni acaba la línea ni aparece una coma (`,`) —que puede ir precedida o no de un espacio en blanco— seguida de más símbolos a exportar.

Y también cuando leyendo la directiva ‘EXPORT’, después del espacio opcional posterior a un símbolo de acceso, ni acaba la línea ni aparece una coma (,) seguida de más símbolos a exportar, sino que aparece otro carácter no esperado.

Leyendo líneas con la directiva ‘ORG’

`La directiva 'ORG' no debe llevar etiqueta.`
('ORG' no debe llevar etiqueta.)

Cuando después de una etiqueta seguida de un espacio se encuentra el código nemónico de la directiva ‘ORG’, que no debe llevar etiqueta.

`Después de la directiva 'ORG' tiene que dejarse un espacio y a continuación debe ir una dirección de memoria.`

Cuando inmediatamente después de ‘ORG’ aparece un carácter diferente de un espacio en blanco o un tabulador.

`Se esperaba una dirección de memoria.`

Cuando después de la directiva ‘ORG’ y de un espacio no se ha incluido un número.

`Dirección fuera de la memoria.`

Cuando el número que acompaña a la directiva ‘ORG’ no es una dirección válida de memoria.

`Después de 'ORG  <dirección>' se esperaba fin de línea.`

Cuando después de la directiva ‘ORG’ se ha leído un espacio y una dirección válida de memoria, pero en lugar de terminar ahí la línea, aparecen a continuación más caracteres.

Leyendo líneas del bloque de instrucciones, seudoinstrucciones y directivas ‘EQU’

`Etiqueta redefinida.`

Cuando la misma etiqueta está definida previamente en ese mismo módulo.

`La etiqueta tiene el mismo nombre que uno de los símbolos externos.`

Cuando la etiqueta tiene el mismo nombre que uno de los símbolos externos.

`Después de una etiqueta tiene que dejarse un espacio y a continuación debe ir el código nemónico de una instrucción, una seudoinstrucción, o una directiva.`

Cuando leyendo el nombre de una etiqueta se encuentra un carácter no válido. Entre las líneas de instrucciones, si una línea empieza por una cadena de caracteres que no sea ‘ORG’ o ‘END’, se supone que es una etiqueta; y que forman parte de la etiqueta todos los caracteres que le siguen mientras sean letras, dígitos o el carácter guión bajo (_).

Acabada la secuencia de caracteres válidos debería de encontrarse un espacio en blanco o un tabulador.

`Una etiqueta debe ser una cadena que empiece con una letra y esté formada por letras, dígitos y/o caracteres guión bajo (_).`

Cuando la línea no empieza ni por un espacio ni por una letra, por lo que se supone que se está queriendo definir una etiqueta, pero que se ha empezado con un carácter no válido.

`Se esperaba el código nemónico de una instrucción, una seudoinstrucción, o una directiva.`

Cuando la línea empieza por una etiqueta a la que sigue un espacio o cuando la línea empieza directamente con un espacio, pero después, o bien no hay una palabra, o bien no coincide con ninguno de los códigos nemónicos válidos.

`Se esperaba fin de línea. La línea ya se considera completa antes de llegar a este punto.`

Cuando después de lo que resulta ser una instrucción, seudoinstrucción, o directiva válida y completa, la línea, en lugar de terminar ahí, contiene todavía más caracteres.

`La instrucción quedaría fuera de la memoria. Este módulo no cabe en la memoria del ordenador.`

Cuando se comprueba que la dirección de memoria relativa dentro del módulo supera la máxima dirección de memoria del ordenador, por lo que el módulo no cabría de ninguna manera en la memoria.

`etiqueta` `tiene definición recursiva o contiene alguna etiqueta desconocida o que representa a un registro o a una cadena.`

Cuando después de pasar por todo el programa no es posible evaluar la expresión porque alguna tiene definiciones recursivas o contiene alguna etiqueta desconocida o que representa a un registro o a una cadena. Tampoco podrá evaluar la expresión si contiene algún símbolo externo, puesto que su valor no será conocido hasta el montaje.

`Si después de la seudoinstrucción se pone un punto, hay que poner a continuación 'B' o 'W'.`

Cuando inmediatamente después de una seudoinstrucción ‘`DATA`’ o ‘`RES`’ aparece un punto (`.`) al que no siguen una ‘`B`’ (que indica media palabra) o una ‘`W`’ (que indica palabra entera).

`Final prematuro del programa`

Cuando no hay más caracteres que leer en el programa fuente y no se ha encontrado la directiva ‘`END`’ completa.

Leyendo líneas con las seudoinstrucciones EQU

`La directiva 'EQU' debe llevar etiqueta.`

Cuando la línea empieza con un espacio y a continuación se encuentra el código nemónico de la directiva '`EQU`', que debe llevar etiqueta.

`Después de la directiva 'EQU' tiene que dejarse un espacio y a continuación debe ir la constante, la expresión o el símbolo asociado a la etiqueta.`

Cuando inmediatamente después de '`EQU`' aparece un carácter diferente de un espacio en blanco o un tabulador.

`Se esperaba la constante, expresión, símbolo o registro asociado a la etiqueta.`

Cuando después de '`EQU`' aparecen un espacio en blanco o un tabulador a los que sigue un carácter que no corresponde con los patrones de registro, cadena, número o expresión.

Leyendo líneas con las seudoinstrucciones RES y RES.B

`Después de la seudoinstrucción 'RES' (o de 'RES.B') tiene que dejarse un espacio y a continuación debe ir una constante numérica o simbólica.`

Cuando inmediatamente después de la seudoinstrucción '`RES`', o de '`RES.B`', aparece un carácter diferente de un espacio en blanco o un tabulador.

`símbolo` `tendría que haberse definido antes.`

Cuando el símbolo que acompaña a la seudoinstrucción '`RES`', o de '`RES.B`', (número de palabras o bytes a reservar) no ha sido definido en ningún punto del módulo.

`símbolo` `hace referencia a un registro en lugar de a una constante numérica o simbólica.`

Cuando el símbolo que acompaña a la seudoinstrucción '`RES`', o de '`RES.B`', (número de palabras o bytes a reservar) hace referencia a un registro en lugar de a una constante numérica o simbólica.

`Se esperaba la constante numérica o simbólica que indique el número de palabras o bytes a reservar.`

Cuando después de la seudoinstrucción '`RES`', o de '`RES.B`', aparecen un espacio en blanco o un tabulador a los que sigue un carácter que no corresponde con los patrones de una constante numérica o simbólica.

Leyendo líneas con las seudoinstrucciones DATA y DATA.B

Después de la seudoinstrucción 'DATA' (o de 'DATA.B') tiene que dejarse un espacio y a continuación deben ir una o varias constantes.

Cuando inmediatamente después de la seudoinstrucción 'DATA', o de 'DATA.B', aparece un carácter diferente de un espacio en blanco o un tabulador.

Número fuera de rango. No se puede almacenar en media palabra.

Cuando en la seudoinstrucción 'DATA.B' alguna de las constantes numéricas de la lista tiene un valor que no es posible almacenar en media palabra.

Número fuera de rango. No se puede almacenar en una palabra.

Cuando en la seudoinstrucción 'DATA' alguna de las constantes numéricas de la lista tiene un valor que no es posible almacenar en una palabra.

Una dirección de memoria necesita una palabra completa para su almacenamiento.

Cuando en la seudoinstrucción 'DATA.B' alguna de las constantes numéricas de la lista viene definida por un símbolo que se corresponde con una dirección de memoria; puesto que una dirección necesita una palabra entera para ser almacenada.

símbolo no se puede traducir a un valor numérico.

Cuando en la seudoinstrucción 'DATA' o 'DATA.B' alguna de las constantes de la lista viene definida por un símbolo que no se puede traducir a un valor numérico. Ocurrirá, en concreto, porque se corresponda con un registro.

símbolo no ha sido declarado en este módulo y tampoco es un símbolo externo.

Cuando en la seudoinstrucción 'DATA' o 'DATA.B' alguna de las constantes numéricas de la lista viene definida por un símbolo que no ha sido declarado en este módulo y tampoco está entre los símbolos externos.

símbolo no se corresponde con una constante numérica, ni con una constante simbólica que se pueda traducir a un valor numérico

Cuando en la seudoinstrucción 'DATA' o 'DATA.B', en la lista de constantes aparece un elemento que no es ni un número, ni una constante simbólica, ni una cadena de caracteres.

Reconociendo una expresión

`Se esperaba un valor, que podría venir dado por una etiqueta numérica o simbólica o por una expresión.`

Cuando se espera un valor (que podría venir dado por una etiqueta numérica o simbólica o por una expresión), y se encuentra otra cosa. Una expresión puede empezar por un número positivo o negativo, una etiqueta, o una apertura de paréntesis (`(`).

`En este punto de la expresión se esperaba un valor (que podría venir dado por una etiqueta numérica o simbólica) o una subexpresión (que empezaría con la apertura de un paréntesis).`

Cuando ya dentro de una expresión, y después de un signo de operación, no se encuentra un valor, que es lo que se esperaba. El valor puede venir dado por una etiqueta numérica o simbólica o por una subexpresión —que empezaría con la apertura de un paréntesis (`(`)—.

`Se están cerrando más paréntesis de los que hay abiertos.`

Cuando analizando una expresión se encuentra con que en algún punto se están cerrando más paréntesis (`)`)de los que hay abiertos.

`Expresión incompleta.`

Cuando analizando una expresión se encuentra con ésta acaba con un signo de operación o con una apertura de paréntesis.

`Hay paréntesis abiertos que no se llegan a cerrar.`

Cuando analizando una expresión se encuentra con ésta acaba con un número de paréntesis abiertos (`(`) mayor que el de los que se han cerrado (`)`).

Evaluando una expresión

`No se pueden sumar dos valores reubicables.`

Cuando se está tratando de sumar dos valores reubicables, cosa que no está permitida.

`No se le puede restar un valor reubicable a uno absoluto.`

Cuando se está tratando de restar un valor reubicable a uno absoluto, cosa que no está permitida.

`El resultado de la expresión queda fuera del rango admisible por` *`ordenador`*`.`

Cuando al terminar de evaluar la expresión, el resultado es un número que queda fuera del rango admisible por el ordenador que se está simulando.

Leyendo la línea con la directiva 'END'

`La directiva 'END' no debe llevar etiqueta.`

Cuando después de una etiqueta seguida de un espacio se encuentra el código nemónico de la directiva 'END', que no debe llevar etiqueta.

`Después de la directiva 'END' tiene que, o bien acabar la línea, o bien dejarse un espacio y que a continuación vaya una dirección de memoria en forma numérica o simbólica.`

Cuando inmediatamente después de 'END' no acaba la línea y aparece un carácter diferente de un espacio en blanco o un tabulador.

Y cuando después de 'END' hay un espacio en blanco o un tabulador a los que no sigue un número o un símbolo.

`Dirección fuera de la memoria.`

Cuando el número que acompaña a la directiva 'END' (primera dirección a ejecutar) no es una dirección válida de memoria.

`símbolo` `tendría que haberse definido antes.`

Cuando el símbolo que acompaña a la directiva 'END' (primera dirección a ejecutar) no ha sido definido en ningún punto del módulo.

`símbolo` `hace referencia a un registro.`

Cuando el símbolo que acompaña a la directiva 'END' (primera dirección a ejecutar) hace referencia a un registro en lugar de a una dirección válida de memoria.

`símbolo` `hace referencia a una cadena.`

Cuando el símbolo que acompaña a la directiva 'END' (primera dirección a ejecutar) hace referencia a una cadena en lugar de a una dirección válida de memoria.

`Dirección fuera de la memoria.`

Cuando el número que acompaña a la directiva 'END' (primera dirección a ejecutar) no es una dirección válida de memoria.

`Después de 'END <dirección>' se esperaba fin de línea.`

Cuando después de la directiva 'END' se ha leído un espacio y una dirección válida de memoria, pero en lugar de terminar ahí la línea, aparecen a continuación más caracteres.

`En las líneas posteriores a la directiva 'END' sólo pueden aparecer comentarios y líneas en blanco.`

Cuando después de haber reconocido la línea correspondiente a la directiva 'END' se encuentra alguna línea con caracteres que no forman parte de un comentario

8.3 Mensajes relativos a errores sintácticos y semánticos exclusivos de Símplez

`Se esperaba una etiqueta o una expresión válida que resolver.`

Cuando después de la barra (`/`) no se encuentra ni una etiqueta ni una expresión válida.

`El valor numérico del CD debe estar entre 0 y 511.`

Cuando el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre 0 y 511.

8.4 Mensajes relativos a errores sintácticos y semánticos exclusivos de Algorítmez

Evaluando Instrucciones

`Después del código nemónico de una instrucción del tipo 1 tiene que dejarse un espacio y a continuación debe ir un registro.`

Cuando inmediatamente después del código nemónico de una instrucción de tipo 1 aparece un carácter diferente de un espacio en blanco o un tabulador.

`En las instrucciones IN y OUT, después del registro de la memoria local tiene que escribirse una coma (,) seguida del puerto que se direcciona.`

Cuando leyendo una instrucción IN o OUT, después del registro de la memoria local, no se encuentra una coma (`,`), o un espacio seguido de una coma (`,`).

`Después del código nemónico de una instrucción del tipo 2 tiene que dejarse un espacio y a continuación debe ir un registro.`

Cuando inmediatamente después del código nemónico de una instrucción de tipo 2 aparece un carácter diferente de un espacio en blanco o un tabulador.

`En las instrucciones del tipo 2, después del registro de la memoria local tiene que escribirse una coma (,) seguida de una dirección.`

Cuando leyendo una instrucción del tipo 2, después del registro de la memoria local, no se encuentra una coma (`,`), o un espacio seguido de una coma (`,`).

`Después del código nemónico de las instrucciones CMP (CMP.W) y CMP.B tiene que dejarse un espacio y a continuación debe ir un registro.`

Cuando inmediatamente después del código nemónico de las instrucciones `CMP` (`CMP.W`) o `CMP.B` aparece un carácter diferente de un espacio en blanco o un tabulador.

`En las instrucciones CMP (CMP.W) y CMP.B, después del registro de la memoria local tiene que escribirse una coma (,) seguida de una dirección.`

Cuando leyendo las instrucciones `CMP` (`CMP.W`) o `CMP.B`, después del registro de la memoria local, no se encuentra una coma (`,`), o un espacio seguido de una coma (`,`).

`Después del código nemónico de las instrucciones LD (LD.W), LD.B, ST (ST.W) y ST.B tiene que dejarse un espacio; y a continuación debe ir un registro.`

Cuando inmediatamente después del código nemónico de las instrucciones `LD` (`LD.W`), `LD.B`, `ST` (`ST.W`) o `ST.B` aparece un carácter diferente de un espacio en blanco o un tabulador.

`En las instrucciones LD (LD.W), LD.B, ST (ST.W) y ST.B, después del registro tiene que escribirse una coma (,) seguida de una dirección.`

Cuando leyendo las instrucciones `LD` (`LD.W`), `LD.B`, `ST` (`ST.W`) o `ST.B`, después del registro, no se encuentra una coma (`,`), o un espacio seguido de una coma (`,`).

Analizando referencias a registros

`No es posible acceder con esta instrucción al registro de estado.`

Cuando se intenta acceder al registro de estado con una instrucción distinta de `LD`, (`LD.W`), `LD.B`, `ST` (`ST.W`) o `ST.B`.

`(Una palabra reservada no puede usarse como etiqueta.)`

`La etiqueta` *`etiqueta`* `no ha sido declarada en este módulo.`

Cuando se espera encontrar un registro y se lee una palabra que no se corresponde con ninguna etiqueta declarada en este módulo y tampoco es un símbolo externo.

`La etiqueta debería corresponderse con un registro.`

Cuando se espera encontrar un registro y se lee una palabra que es una etiqueta declarada en este módulo pero que no se corresponde con un registro.

`El número para hacer referencia al registro debe estar entre 0 y 15.`

Cuando se espera encontrar un registro y después del punto (`.`) se lee un número que no está entre 0 y 15.

`Para hacer referencia a un registro, después del punto (.) debe especificarse su número si pertenece a la memoria local o la letra E si es el registro de estado.`

Cuando se espera encontrar un registro y después del punto (`.`) no se lee ni un número ni la letra `E` del registro de estado.

`En este punto de la instrucción debería especificarse un registro.`

Cuando se espera encontrar un registro no se encuentra ni una palabra —que podría corresponderse con una etiqueta que equivaliese a un registro— ni un punto (`.`) —que sería el comienzo de la especificación normal de un registro—.

Analizando referencias a puertos

`Se esperaba encontrar un puerto. Los puertos se direccionan a través de un registro que se escribe entre corchetes: [registro].`

Cuando se espera encontrar un puerto y no se encuentra una apertura de corchete (`[`).

`Después del registro que direcciona al puerto se debe cerrar el corchete (]).`

Cuando se está leyendo un puerto y después del registro que lo direcciona no se encuentra el cierre de corchete (`]`).

Analizando direcciones

`MD inmediato: Se esperaba un valor imediato, pero no se ha encontrado ni un número, ni una etiqueta que evaluar, ni una expresión válida que resolver.`

Cuando en el direccionamiento inmediato, después de la almohadilla (**#**) no se puede reconocer un valor inmediato al no ser un ni un número, ni una etiqueta ni una expresión válida.

`El valor numérico del CD para direccionamiento inmediato referido a byte debe estar entre H'00 y H'FF.`

Cuando el valor numérico del CD para direccionamiento inmediato referido a byte no está entre H’00 y H’FF; o lo que es lo mismo, entre 0 y 255, considerando un byte sin signo.

`El valor numérico del CD para direccionamiento inmediato referido a palabra debe estar entre H'0000 y H'FFFF.`

Cuando el valor numérico del CD para direccionamiento inmediato referido a palabra no está entre H’0000 y H’FFFF; o lo que es lo mismo, entre 0 y 65535, considerando una palabra (dos bytes) sin signo.

`MD directo o MD indexado: Se esperaba una etiqueta o una expresión válida que resolver.`

Cuando en los modos de direccionamiento directo e indexado, después de la barra (**/**) no se encuentra ni una etiqueta ni una expresión válida.

`En el direccionamiento indexado, después del registro de índice, se debe cerrar el corchete (]).`

Cuando en el modo de direccionamiento indexado, inmediatamente después del registro de índice no se encuentra el cierre de corchete (]).

`no se permiten referencias a otros módulos en la etiqueta del direccionamiento indexado.`

Cuando en el modo de direccionamiento indexado, la etiqueta se corresponde con un símbolo externo (importado de otro módulo).

`El valor numérico del CD para direccionamiento indexado debe estar entre H'00 y H'FF.`

Cuando en el modo de direccionamiento indexado, el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre H'00 y H'FF; o lo que es lo mismo, entre –128 y 127 considerando un byte con signo.

`El valor numérico del CD para direccionamiento directo debe estar entre 0 y 65535.`

Cuando en el modo de direccionamiento directo, el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre 0 y 65535.

`MD indexado indirecto: Se esperaba una etiqueta o una expresión válida que resolver.`

Cuando en el modo de direccionamiento indexado indirecto, después de la barra (/) no se encuentra ni una etiqueta ni una expresión válida.

`MD indexado indirecto: Se esperaba encontrar un registro de índice, que se reprenta entre corchetes: [registro].`

Cuando en el modo de direccionamiento indexado indirecto, después de la etiqueta o de la expresión no se encuentra una apertura de corchete ([).

`MD indexado indirecto: Después del registro de índice se debe cerrar el corchete (]).`

Cuando en el modo de direccionamiento indexado indirecto, inmediatamente después del registro de índice se encuentra un carácter distinto del cierre de corchete (]).

`MD indexado indirecto: Después del registro de índice se deben cerrar los dos corchetes (]).`

Cuando en el modo de direccionamiento indexado indirecto, inmediatamente después del cierre de corchete posterior al registro de índice, se encuentra un carácter distinto del segundo cierre de corchete (]), que era lo esperado.

`El valor numérico del CD para direccionamiento indexado indirecto debe estar entre H'00 y H'FF.`

Cuando en el modo de direccionamiento indexado indirecto, el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre H'00 y H'FF; o lo que es lo mismo, entre –128 y 127 considerando un byte con signo.

MD autoincremento: Después del registro de autoincremento deben aparecer dos signos más (+).

Cuando en el modo de direccionamiento por autoincremento, inmediatamente después del registro de autoincremento se encuentra un carácter distinto del signo más (+).

Y cuando también en el modo de direccionamiento por autoincremento, inmediatamente después del signo más (+) posterior al registro de autoincremento se encuentra un carácter distinto de un segundo signo más (+).

MD autoincremento: Después de los dos signos más se debe cerrar el corchete (]).

Cuando en el modo de direccionamiento por autoincremento, inmediatamente después de los dos signos más (+) posteriores al registro de autoincremento se encuentra un carácter distinto de un cierre de corchete (]).

MD autoincremento indirecto: Después del registro de autoincremento deben aparecer dos signos más (+).

Cuando en el modo de direccionamiento por autoincremento indirecto, inmediatamente después del registro de autoincremento se encuentra un carácter distinto del signo más (+).

Y cuando también en el modo de direccionamiento por autoincremento indirecto, inmediatamente después del signo más (+) posterior al registro de autoincremento se encuentra un carácter distinto de un segundo signo más (+).

MD autoincremento indirecto: Después de los dos signos más se deben cerrar los dos corchetes (]).

Cuando en el modo de direccionamiento por autoincremento indirecto, inmediatamente después de los dos signos más (+) posteriores al registro de autoincremento se encuentra un carácter distinto de un cierre de corchete (]).

Y cuando también en el modo de direccionamiento por autoincremento indirecto, inmediatamente después del primer cierre de corchete (]) posterior a los dos signos más (+) se encuentra un carácter distinto de un segundo cierre de corchete (]).

MD relativo a programa e indirecto: Se esperaba una etiqueta o una expresión válida que resolver.

Cuando en el modo de direccionamiento relativo a programa e indirecto, inmediatamente después de la apertura de corchete ([) no se encuentra ni una etiqueta ni una expresión válida.

MD relativo a programa e indirecto: Después la etiqueta o la expresión se debe cerrar el corchete (]).

Cuando en el modo de direccionamiento relativo a programa e indirecto, inmediatamente después del registro de índice se encuentra un carácter distinto del cierre de corchete (]).

`El valor numérico del CD para direccionamiento relativo a programa e indirecto debe estar entre H'00 y H'FF.`

Cuando en el modo de direccionamiento relativo a programa e indirecto, el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre H'00 y H'FF; o lo que es lo mismo, entre –128 y 127 considerando un byte con signo.

`MD relativo a programa: Se esperaba una etiqueta o una expresión válida que resolver.`

Cuando en el modo de direccionamiento relativo a programa, inmediatamente después de la apertura de corchete ([ ) no se encuentra ni una etiqueta ni una expresión válida.

`El valor numérico del CD para direccionamiento relativo a programa debe estar entre H'00 y H'FF.`

Cuando en el modo de direccionamiento relativo a programa, el valor la etiqueta, o de la expresión que apareciese en su lugar, tienen un valor no comprendido entre H'00 y H'FF; o lo que es lo mismo, entre –128 y 127 considerando un byte con signo.