Ingeniería Técnica en Informática de Gestión, Curso 2001-2002
LABORATORIO DE ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES

Sesión 8

Introducción al simulador SPIM

En esta práctica se introduce el simulador SPIM utilizado para ejecutar los programas desarrollados en lenguaje ensamblador para los computadores basados en el procesador MIPS R2000. A continuación, se describe un conjunto de conceptos básicos para la programación en lenguaje ensamblador que se irá ampliando a medida que se avance en los capítulos de este manual. Así pues, este guión de práctica se divide en dos apartados que contienen:

- La descripción del simulador SPIM en sus dos versiones para Linux y para Windows.
- La descripción de los recursos de programación básicos que permite la programación en ensamblador.

Descripción del Simulador XSPIM

El SPIM (MIPS al revés) es un simulador que ejecuta programas en lenguaje ensamblador de los computadores basados en los procesadores MIPS R2000/R3000. La arquitectura de este tipo de procesadores es RISC, por lo tanto simple y regular, y en consecuencia fácil de aprender y entender.

La pregunta obvia en estos casos es por qué se va a utilizar un simulador y no una máquina real. Las razones son diversas: entre ellas cabe destacar la facilidad de poder trabajar con una versión simplificada y estable del procesador real. Los procesadores actuales ejecutan varias instrucciones al mismo tiempo y en muchos casos de forma desordenada, esto hace que sean más difíciles de comprender y programar.

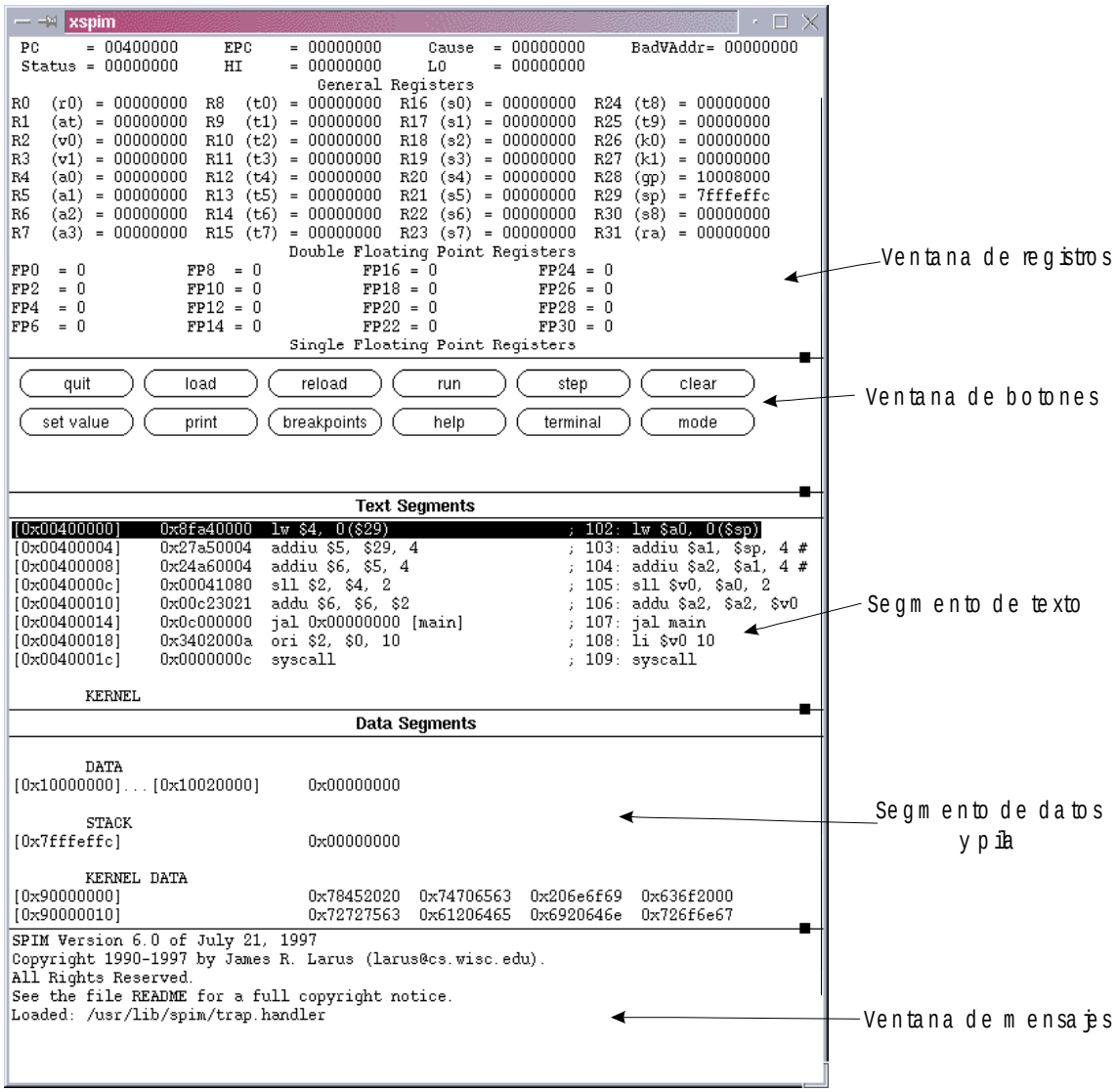
El simulador a utilizar en prácticas es una versión "X" del SPIM, es decir, una versión gráfica con ventanas, denominada XSPIM.

La instalación del XSPIM es sencilla:

Windows: Ejecuta el programa "spimwin.exe". Se realizará la instalación y sólo habrá que ejecutar el icono "PCSpim for Windows" para arrancar el programa.

Linux: Ejecuta "rpm –i spim.rpm". Una vez realizada la instalación, el programa se ejecuta mediante "xspim".

A continuación se pasa a describir de forma más detallada el simulador en cada una de estas plataformas.



Versión para Linux

El XSPIM, en la versión de Linux, posee una interfaz gráfica, dividida en cinco ventanas:

- La parte superior se llama *register display* (ventana de registros). Muestra los valores de todos los registros de la CPU y la FPU (unidad de coma flotante) del MIPS.

- La siguiente parte llamada *control buttoms* (ventana de botones) contiene los botones de control que permiten gestionar el funcionamiento del simulador,

- El siguiente apartado llamado *text segments* (segmentos de texto), muestra las instrucciones del programa de usuario y del núcleo del sistema (*Kernel*) que se carga automáticamente cuando *xspim* empieza su ejecución.

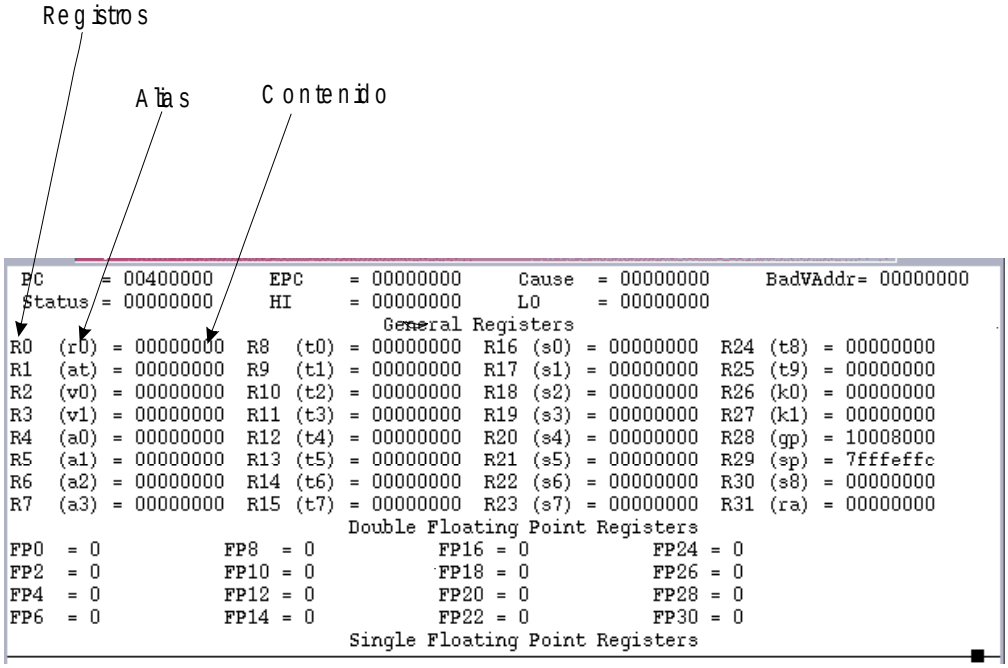
- La siguiente parte, llamada *data segments* (segmento de datos y pila) muestra los datos de la memoria y de la pila del programa de usuario cargado en el simulador.

- El apartado inferior es el de mensajes de XSPIM (ventana de mensajes), que *xspim* usa para escribir todo tipo de información generada durante la ejecución del mismo.

Cada una de estas ventanas contiene la siguiente información:

Ventana de registros.

En este panel se muestran el nombre y el contenido de los registros enteros de la CPU y de la FPU (unidad de coma flotante). Estos son: los registros **R0** a **R31**, con sus correspondientes alias entre paréntesis, los registros de coma flotante, **FP0** a **FP31**, los registros de control de la CPU (**BadVAddr**, **Cause**, **Status**, **EPC**) y los registros especiales para la multiplicación y división entera, **HI** y **LO**. La longitud de cada uno de estos registros es de 32 bits (4 bytes).

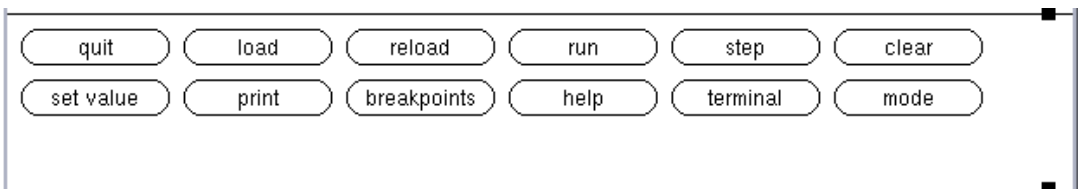


Las características de la información contenida en esta ventana son las siguientes:

- El contenido de los registros siempre se mostrará en hexadecimal.
- Cuando se ejecuta *xspim*, o se inicializen los registros, el contenido de estos será cero, excepto el del registro que apunta a la pila (R29) que contiene el valor 0x7fffeffc.
- Esta ventana se actualiza siempre que el simulador detiene la ejecución de un programa con los resultados obtenidos durante la misma.

Ventana de botones de control

En este panel se pueden observar los botones que permiten controlar el funcionamiento del simulador. Son los siguientes:



⇒ **quit**: Termina la sesión del simulador.

⇒ **load**: Lee un fichero en lenguaje ensamblador, lo ensambla y lo carga en la memoria. Cuando se pulsa este botón se pide el nombre del fichero a cargar.

⇒ **run:** Ejecuta el programa ensamblado y cargado en memoria. Esta opción pide como parámetro la dirección de memoria a partir de la cual se quiere que empiece la ejecución del programa simulado. Por defecto esta ejecución empieza en la dirección 0x00400000. A partir de esta dirección se encuentran almacenadas las instrucciones que se cargan automáticamente cuando se ejecuta *xspim*, y que constituyen el código de inicio que sirve para invocar el programa de usuario que se pretende ejecutar.

⇒ **step:** Esta opción es útil cuando se quiere depurar programas, puesto que permite realizar una ejecución de las instrucciones una a una (ejecución paso a paso), y comprobar el resultado de la ejecución de cada una de las instrucciones. Esta opción pide como parámetro el número de instrucciones que ejecutará el simulador de forma continuada. Si éste es 1 la ejecución se realizará instrucción a instrucción.

⇒ **clear:** Reinicializa el contenido de los registros y/o la memoria, poniéndolo todo al valor 0, excepto el registro sp (R29).

⇒ **set value:** Fuerza a un valor el contenido de un registro o posición de memoria.

⇒ **print:** Imprime el valor de registros o memoria.

⇒ **breackpoint:** Introduce o borra puntos de ruptura (parada) en la ejecución de un programa. Esta opción permite detener la ejecución de un programa justo antes de ejecutar una instrucción particular. Habrá que pasarle como parámetro la dirección de la instrucción donde se quiere detener la ejecución. Se pueden añadir tantos puntos de ruptura como se necesiten.

⇒ **help:** Imprime un mensaje de ayuda.

⇒ **terminal:** Visualiza o esconde la ventana de consola (llamada terminal). Si el programa de usuario lee o escribe del terminal, *xspim* crea otra ventana, todo los caracteres que escriba el programa aparecen en ella y cualquier cosa que se quiera introducir al programa se debe teclear en esta ventana.

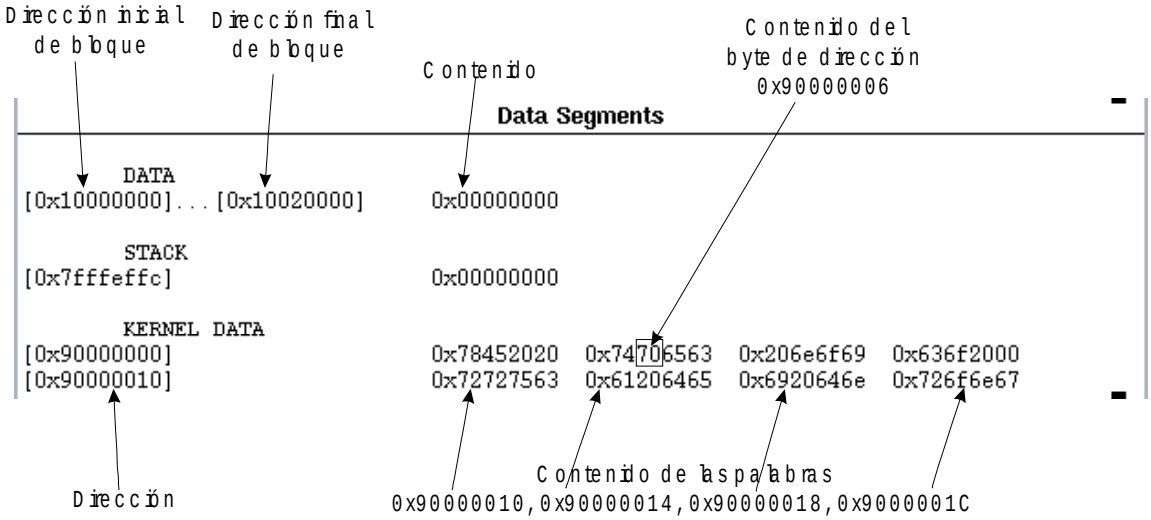
⇒ **mode:** Modifica los modos de funcionamiento del XSPIM. Estos son: quiet y bare. El primero hace que el simulador no escriba un mensaje en la ventana de mensajes cuando se produzca una excepción. El segundo simula una máquina MIPS pura sin pseudoinstrucciones o modos de direccionamiento adicionales ofrecidos por el ensamblador. Ambas opciones deben estar desactivadas para simular nuestras ejecuciones.

Ventana del segmento de datos y pila.

En esta ventana se muestran las direcciones y datos almacenados en las zonas de memoria de datos de usuario (a partir de la dirección 0x10000000 hasta la dirección 0x10020000), del núcleo del simulador (a partir de la dirección 0x90000000) y de la pila (referenciada mediante el registro sp, y creciendo hacia direcciones decrecientes de memoria a partir de la dirección 0x7fffeffc).

Cuando un bloque de memoria contiene la misma información, se muestra la dirección inicial (columna de la izquierda), final (siguiente columna) y el contenido del bloque (columna de la

derecha). En otro caso, se muestra en una misma fila el contenido de 4 palabras de memoria, indicando en la columna de más a la izquierda la dirección de la primera palabra y en las cuatro columnas siguientes el contenido de cada una de las cuatro palabras. La longitud de una palabra de memoria es de 4 bytes.



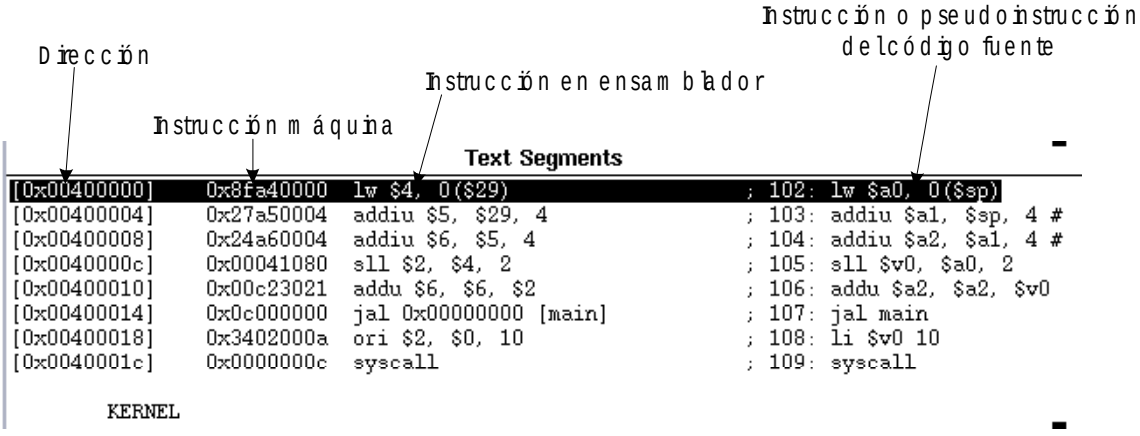
Las características de la información contenida en esta ventana son las siguientes:

- El contenido de las posiciones de memoria se muestra en hexadecimal.
- Cuando se ejecuta el simulador sin cargar ningún programa o al pulsar el botón *clear* con la opción *register & memory* el contenido del segmento de datos de usuario y de la pila se inicializan a cero.

Ventana del segmento de texto

Se muestran cada una de las direcciones, el código máquina, las instrucciones ensambladas y el código fuente del programa del usuario (a partir de la dirección 0x00400000) y del núcleo del simulador (a partir de la dirección 0x80000000).

A partir de la dirección 0x004000000, cuando se ejecuta el *xspim* se cargan una serie de instrucciones (código de inicio, start-up), que ayudan a la ejecución y depuración del código del usuario.



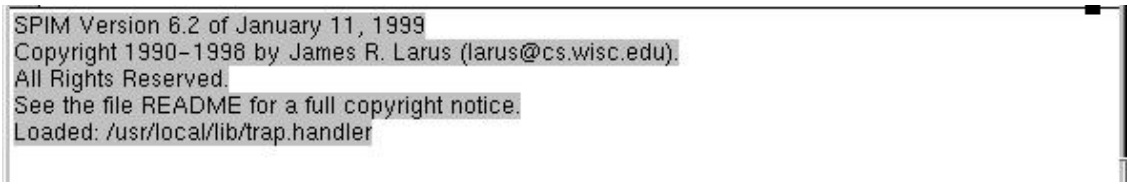
En esta ventana se muestra la información organizada en columnas, la de más a la derecha indica la instrucción o pseudoinstrucción del código fuente, la siguiente muestra la instrucción o instrucciones

en ensamblador por las que el simulador ha traducido la instrucción de la columna anterior, a continuación se muestra la instrucción máquina, y finalmente la dirección de memoria donde está almacenada esta instrucción máquina.

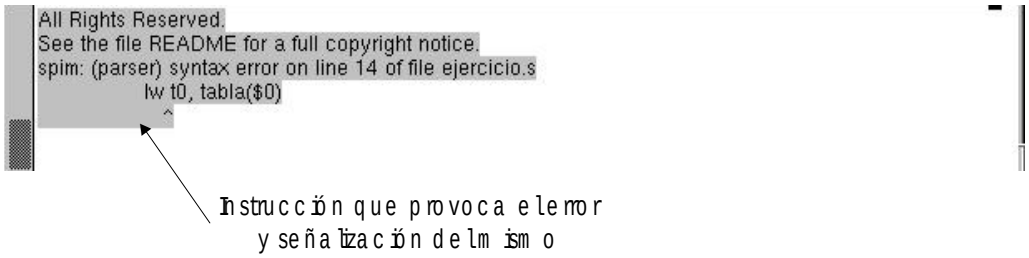
Mensajes del simulador

En este panel se observan los diversos mensajes que comunica el simulador, que nos tendrán informados del resultado y evolución de las acciones que éste lleva a cabo. En esta ventana aparecerán los mensajes de error que se generan. Estos pueden darse durante el ensamblado y carga del programa o durante la ejecución de éste en el simulador.

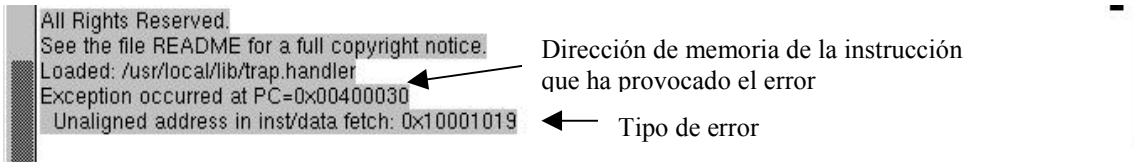
Si el programa se carga en memoria sin ningún tipo de error, en esta ventana se muestra el siguiente mensaje:



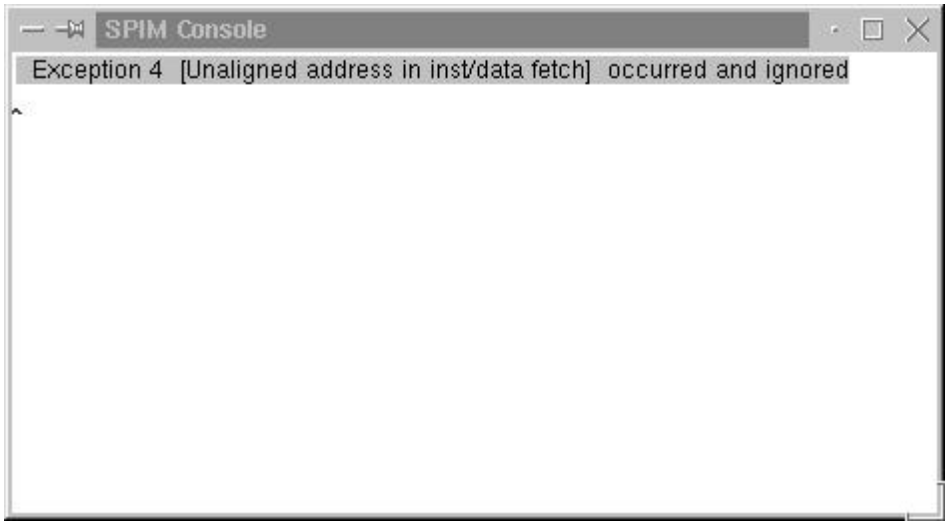
En cambio si durante la traducción del programa ensamblador a lenguaje máquina se encuentra algún tipo de error, en esta ventana se muestra la instrucción que ha provocado el error y una señalización del mismo. La aparición de este error indica que el resto del programa no se ha traducido a lenguaje máquina y por lo tanto tampoco se ha cargado en memoria.



Si el error se produce durante la ejecución del programa, en esta ventana se muestra un mensaje indicando el error producido,



y se abre otra ventana indicando el tipo de excepción que genera este error.



Carga y ejecución de programas

Los ficheros de entrada a *xspim* son de tipo texto ASCII, que incluyen las instrucciones en ensamblador del programa que se quiere simular su ejecución.

Para cargar un programa se selecciona en la ventana de botones el botón **load**, con lo que aparecerá un cuadro de diálogo donde se especifica el nombre del fichero que contiene el código del programa a ejecutar.

Al cargar cualquier programa de usuario en memoria, éste se puede cargar solo o junto con el manejador de excepciones. En el primer caso (-notrap) el programa de usuario se almacenará a partir de la dirección 0x00400000.

En el segundo caso, opción por defecto (-trap), antes de las instrucciones del programa de usuario, se carga un conjunto de instrucciones, llamado código de inicio (start-up), que permite lanzar a ejecución el programa de usuario y gestionar las excepciones asociadas a los errores producidos durante la ejecución del código. En este caso, al cargar el programa en memoria, el código del programa de usuario se cargará a partir de la dirección 0x00400020 y el código de inicio se cargará a partir de la dirección 0x00400000.

Para ejecutar el programa de forma continuada, hasta su finalización, se selecciona el botón **run** en la ventana de botones, lo que hará que *xspim* empiece a simular la ejecución del programa cargado. Previamente pedirá que se le indique la dirección de comienzo del programa (en hexadecimal). En nuestro caso será normalmente la dirección 0x00400000 (donde comienza el segmento de texto con el código de inicio). Si el código de usuario no tiene definida la etiqueta main, el código de inicio no lanza correctamente su ejecución y se muestra un mensaje que alerta de esta situación.

Si el programa incluye operaciones de lectura o escritura desde el terminal, *xspim* abre una ventana independiente, llamada *terminal*, a través de la cual se realiza la entrada/salida (se simula un terminal de la máquina MIPS).

Depuración de programas

Si un programa no hace lo que se esperaba, hay algunas herramientas del simulador que ayudarán a depurar el programa. Con la opción **Step** (en la ventana de botones) es posible ejecutar las instrucciones del programa una a una (paso a paso). Esta opción permite seleccionar cuantas instrucciones se desean ejecutar de forma continuada antes de que el simulador detenga la ejecución

del programa y actualice las ventanas de registros y datos con los resultados obtenidos durante la misma. Esto permite verificar el contenido de los registros, la pila, los datos, etc., en cada paso.

El simulador *xspim* también permite ejecutar todas las instrucciones de un programa hasta llegar a un determinado punto, denominado *breakpoint* (punto de ruptura), a partir del cual se puede recuperar el control del programa y, por ejemplo, continuar paso a paso. Para ello, se selecciona la opción **Breakpoints** (en la ventana de botones). Una vez seleccionada esa opción, *xspim* muestra una ventana en la que pide la(s) dirección(es) en la(s) que se quiere que el programa se detenga, para recuperar el control sobre el mismo. Se debe mirar cuál es la dirección en que interesa parar el programa, en la ventana del segmento de texto, e introducirla (en hexadecimal) en la ventana, pulsando a continuación la tecla **Add**, para añadir dicho breakpoint. Se pueden introducir tantos puntos de ruptura como se desee.

Una vez encontrado el error y corregido, se vuelve a cargar el programa con la opción **Reload**. Con **Clear Registers** se pone el contenido de los registros a cero (excepto sp), mientras que **Set Value** permite cambiar el valor actual de un registro o de una posición de memoria por un valor arbitrario. Todas estas opciones están en la ventana de botones.

A lo largo de cada una de las partes que constituyen estas prácticas se irá ampliando y aclarando el funcionamiento del simulador y aprendiendo cómo se programa en el ensamblador del R2000.

Aunque las prácticas se van a desarrollar utilizando la versión del Linux de este simulador, en el apartado siguiente se describe brevemente la versión de Windows, que el alumno puede instalar también fácilmente.

Opciones de XSPIM en la línea de comandos

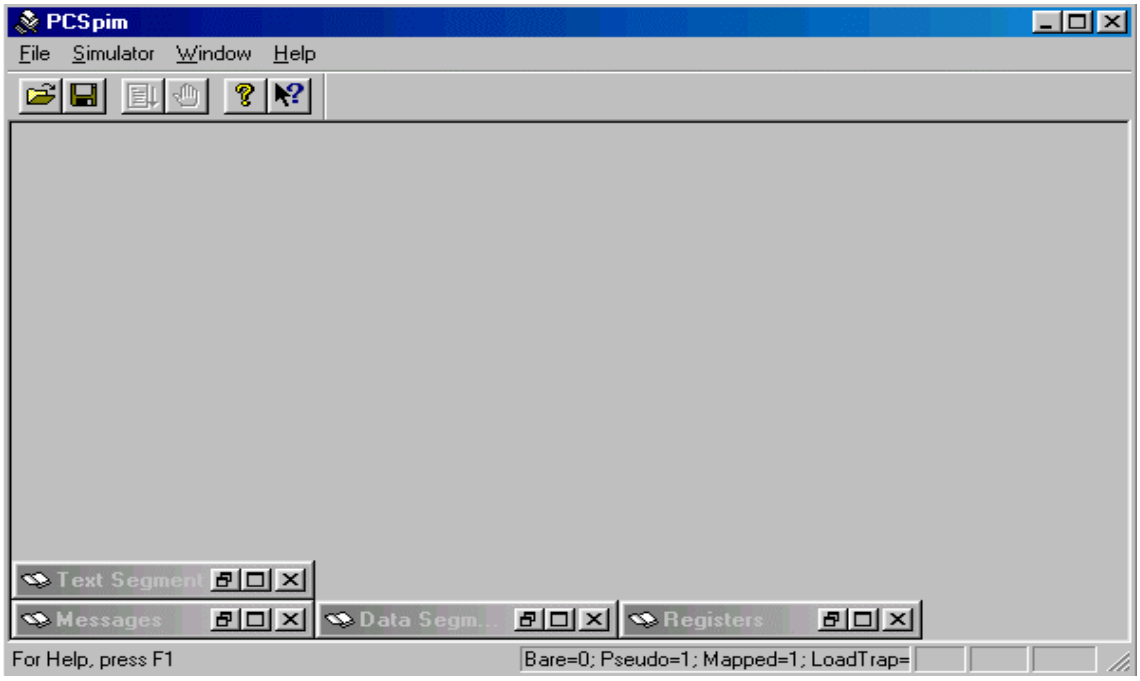
El simulador XSPIM acepta las siguientes opciones en la línea de comandos:

- -bare: Simula una máquina pura, sin pseudoinstrucciones o modos de direccionamiento adicionales ofrecidos por el programa ensamblador. La opción opuesta es -asm, que simula la máquina virtual MIPS que ofrece el programa ensamblador.
- -notrap: No carga la rutina de atención a las excepciones ni el código de inicio. Cuando se produce una excepción, XSPIM salta a la posición 0x80000080, que debe contener código para servir la excepción. La opción opuesta es -trap que si que carga la rutina de atención a las excepciones y el código de inicio.
- -nomapped_io: Deshabilita la utilidad de E/S ubicada en memoria (opción por defecto). En este caso el manejo de las operaciones de E/S serán gestionadas por el sistema a través de la llamada *syscall*. La opción opuesta es -mapped_io, que deberá estar activada cuando la gestión de la E/S la realice al usuario a través de los puertos asociados a cada dispositivo.

Versión para windows

En este apartado se presenta el uso del simulador para Windows, como variante de los programas *spim* y *xspim* de Unix. Para instalar la versión bajo Windows del simulador de MIPS, se ejecuta el fichero de instalación *spimwin.exe* que se puede encontrar en la dirección web de la asignatura.

Al ejecutar *PCSpim* aparece la interfaz que se muestra a continuación. A diferencia de la versión de Linux, no presenta desplegadas cada una de las ventanas que la componen, segmento de texto, segmento de datos, ventana de registros y ventana de mensajes. Para ver cada una de ellas hay que abrirlas explícitamente.



Esta ventana se divide en cuatro partes:

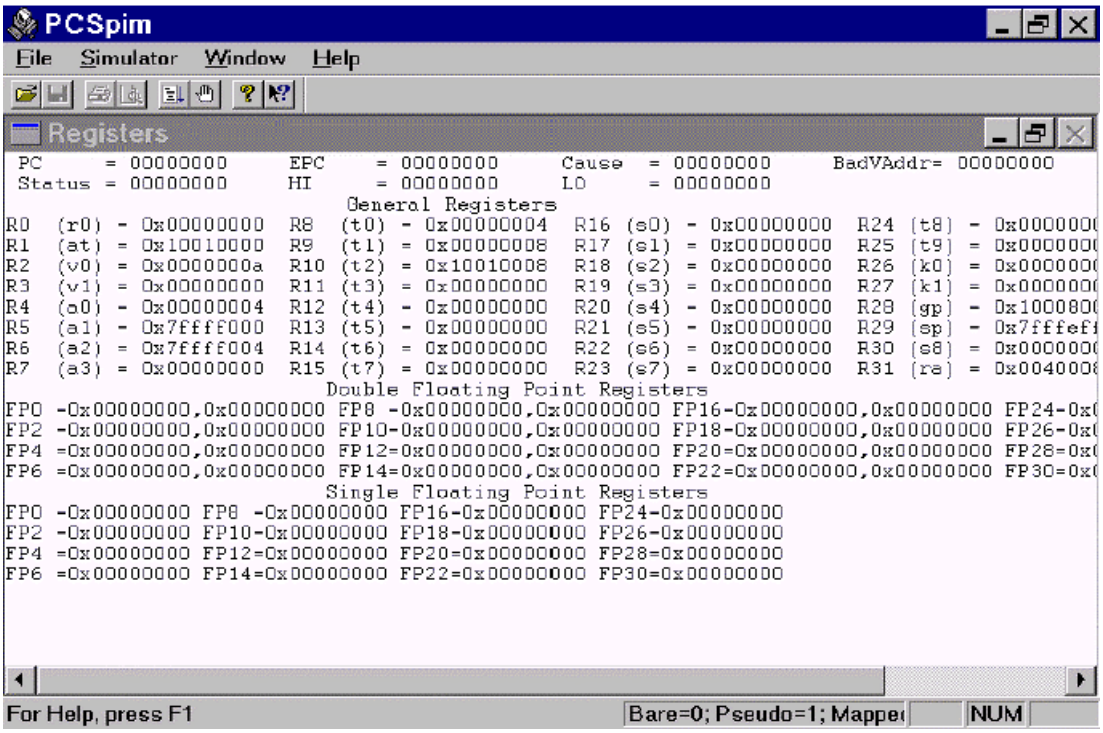
La parte superior es la **barra de menús** que permite:

- acceder a las operaciones con ficheros (*menú File*),
- especificar las opciones del simulador (*menú Simulator*),
- seleccionar la forma de visualización de las ventanas incluidas en la ventana principal (*menú Window*), y
- obtener información de ayuda (*menú help*)

2. Debajo de la barra de menús se encuentra la **barra de herramientas** que incluye en forma de botones algunas de las acciones más comunes en PCSpim.

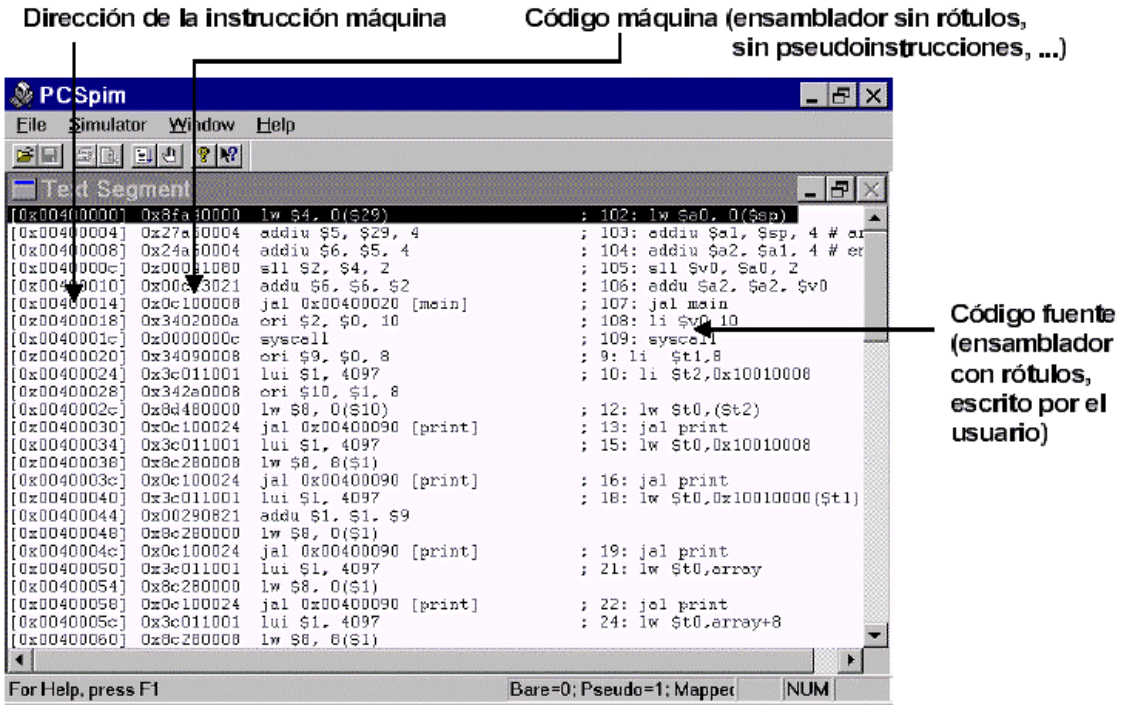
3. La parte central de la ventana de la aplicación sirve para visualizar las cuatro ventanas: **Registros** (*Registers*), **Segmento de Texto** (*Text Segment*), **Segmento de datos** (*Data Segment*) y **Mensajes** (*Messages*). Estas cuatro ventanas son similares a las que se han descrito en el apartado anterior (versión de Linux del simulador). La única diferencia con la versión Linux consiste en que en la de Windows las ventanas pueden estar abiertas o cerradas de forma independiente. A continuación se muestran cada una de ellas:

    - **Ventana de Registros** (*Registers*)

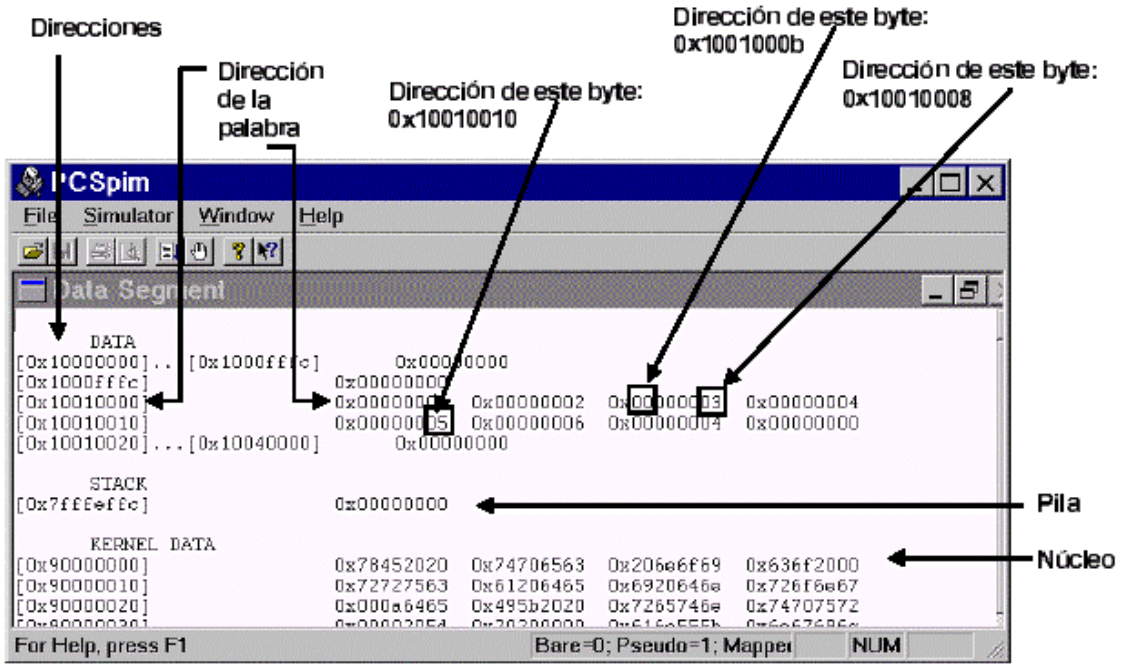


Por defecto la información contenida en los registros se visualiza en decimal. Mediante las opciones del simulador se puede cambiar a hexadecimal.

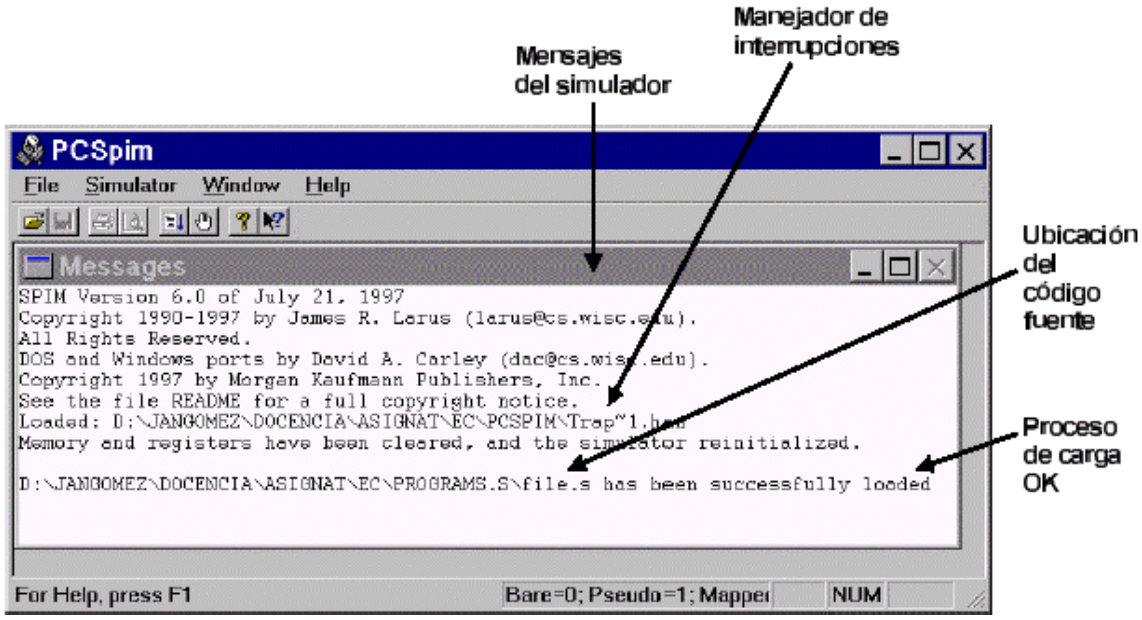
- **Segmento de texto** (*Text Segment*)



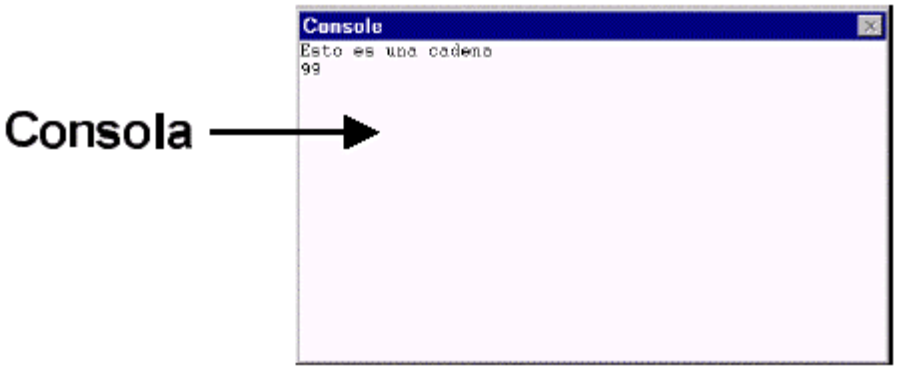
- **Segmento de datos** (*Data Segment*)



- **Mensajes (*Messages*)**



Existe una quinta ventana llamada Consola (terminal en la versión linux), independiente, a la que se accede con la opción **Window → Console**, y que sirve para realizar la entrada/salida del programa simulado.



Carga y ejecución de programas

Los ficheros de entrada a *PCSpim* también son de tipo texto ASCII, e incluyen las instrucciones en ensamblador del programa que se desea ejecutar.

Para cargar un programa se selecciona **File → Go** (o el botón **Open** de la barra de herramientas, con el icono de la carpeta abriéndose) con lo que aparecerá un cuadro de diálogo donde se puede seleccionar el fichero que se quiere abrir. Si en el código de usuario no aparece la etiqueta main y el simulador tiene activa la opción "Load trap file", el programa no será cargado y aparecerá un mensaje de error.

Para ejecutar el programa se selecciona, **Simulator → Go** (o el botón **Go** de la barra de herramientas, con el icono de un programa con una flecha que indica ejecución), hará que *PCSpim* comience a simularlo. Previamente pedirá que se le indique la dirección de comienzo del programa (en hexadecimal). En nuestro caso este valor será normalmente 0x00400000 (donde comienza nuestro segmento de texto). Si se desea detener la ejecución del programa se selecciona **Simulator → Break** (Ctrl-C). Para continuar con la ejecución, **Simulator → Continue**.

Si el programa incluye operaciones de lectura o escritura desde el terminal, *PCSpim* despliega una ventana independiente llamada *Console*, a través de la cual se realiza la entrada/salida (se simula un terminal de la máquina MIPS).

Depuración de programas

Si un programa no hace lo que se esperaba, hay algunas herramientas del simulador que ayudarán a depurar el programa. Con la opción **Simulator → Single Step** (o bien pulsando la tecla F10) es posible ejecutar las instrucciones del programa una a una (paso a paso). Esto permite verificar el contenido de los registros, la pila, los datos, etc., tras la ejecución de cada instrucción. Utilizando **Simulator → Multiple Step** se consigue ejecutar el programa un número determinado de instrucciones.

El programa *PCSpim* también permite ejecutar todas las instrucciones de un programa hasta llegar a un determinado punto, denominado *breakpoint* (punto de ruptura), a partir del cual se puede recuperar el control del programa y, por ejemplo, continuar paso a paso. Para ello, se selecciona **Simulator → Breakpoints** (o el botón **Breakpoints** de la barra de herramientas, con el icono de una mano indicando detención). Una vez seleccionada esa opción, PCSpim muestra una ventana en la que pide la(s) dirección(es) en la(s) que se quiere que el programa se detenga, para recuperar el control sobre el mismo. Se debe mirar cuál es la dirección en que interesa parar el programa, en la ventana del segmento de texto, e introducirla (en hexadecimal) en la ventana, pulsando a

continuación la tecla **Add**, para añadir dicho breakpoint. Se pueden introducir tantos puntos de ruptura como se desee.

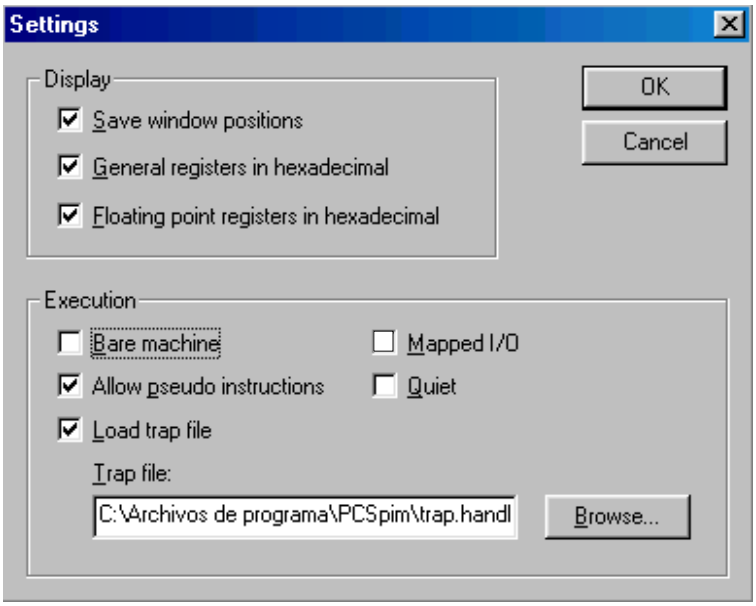
Una vez encontrado el error y corregido, se vuelve a cargar el programa con **Simulator → Reload** <nombre_fichero>. Con **Simulator → Clear Registers** se pone el contenido de los registros a cero (excepto sp), mientras que **Simulator → Set Value** permite cambiar el valor actual de un registro o de una posición de memoria por un valor arbitrario.

Otras opciones útiles disponibles en el menú principal son las contenidas en el menú de ventana (*Window*). Con ellas se pueden mostrar y ocultar las barras de herramientas y de estado, así como las distintas ventanas del simulador, organizar visualmente las mismas y limpiar la consola de entrada/salida.

El *PCSpim* también proporciona ayuda en línea (Opción **Help → Help Topics**), que muestra un documento con ayuda sobre el programa, y el icono de la barra de herramientas con una flecha y una interrogación, que sirve para pedir ayuda sobre una opción concreta de los menús.

Opciones del simulador

Al elegir la opción **Simulator → Setting** se muestran las diversas opciones que ofrece el simulador. El *PCSpim* utiliza estas opciones para determinar cómo cargar y ejecutar los programas. Una vez escogida esta opción aparece la siguiente ventana



La parte superior de la ventana (*Display*) permite:

- Salvar la posición de las ventanas al salir, y recuperarla al ejecutar de nuevo PCSpim (opción ***Save window positions*** activa)

- Fijar el formato de los registros en hexadecimal (opciones ***General registers/Floating point registers in hexadecimal*** activas). En caso de no estar seleccionada esta opción, el contenido se muestra en decimal (con signo, es decir, interpretado en complemento a 2).

La parte inferior (*Execution*) permite fijar distintos modos de funcionamiento del simulador. Para que la simulación de nuestros programas sea correcta, estas opciones deben estar activadas o

desactivadas como se muestra en la figura. La opción **Bare machine** simula el ensamblador sin pseudoinstrucciones o modos de direccionamiento suministrados por el simulador. La opción **Allow pseudo instrucctions** determina si se admiten las pseudoinstrucciones. **Load trap file** indica si se carga el manejador de intrerrupciones (fichero *trap.handler*). En tal caso, cuando se produce una excepción, *PCSpim* salta a la dirección 0x80000080, que contiene el código necesario para tratar la excepción. La opción **Mapped I/O** permite seleccionar si se activa la entrada/salida mapeada en memoria. Los programas que utilizan llamadas al sistema (syscall), para leer o escribir en el terminal, deben desactivar esta opción. Aquellos otros programas que vayan a hacer entrada/salida mediante mapeado en memoria deben tenerla activada. Finalmente la opción **Quiet** permite seleccionar que *PCSpim* no imprima mensaje alguno cuando se producen las excepciones.

1.1. Sintaxis del lenguaje ensamblador del MIPS R2000

El *lenguaje ensamblador* es la representación simbólica de la codificación binaria de un computador, *el lenguaje máquina*. El lenguaje máquina está constituido por *instrucciones máquina* que indican al computador lo que tiene que hacer, es decir, son las órdenes que el computador es capaz de comprender. Cada instrucción máquina está constituida por un conjunto ordenado de unos y ceros, organizados en diferentes campos. Cada uno de estos campos contiene información que se complementa para indicar al procesador la acción a realizar.

El lenguaje ensamblador ofrece una representación más próxima al programador y simplifica la lectura y escritura de los programas. Las principales herramientas que proporciona la programación en ensamblador son: la utilización de nombres simbólicos para operaciones y posiciones de memoria, y unos determinados recursos de programación que permiten aumentar la claridad (legibilidad) de los programas. Entre estos recursos cabe destacar: la utilización de directivas, pseudoinstrucciones y macros que permiten al programador ampliar el lenguaje ensamblador básico definiendo nuevas operaciones.

Una herramienta denominada *programa ensamblador* traduce un programa fuente escrito de forma simbólica, es decir, en lenguaje ensamblador, a instrucciones máquina. El programa ensamblador lee un fichero fuente escrito en lenguaje ensamblador y genera un fichero objeto. Este fichero contiene instrucciones en lenguaje máquina e información que ayudará a combinar varios ficheros objeto para crear un fichero ejecutable (puede ser interpretado por el procesador).

Otra herramienta denominada *montador*, combina el conjunto de ficheros objeto para crear un único fichero que puede ser ejecutado por el computador.

En el simulador SPIM, para cargar un programa en memoria se utiliza la opción *load*, que lleva a cabo todos los pasos necesarios, desde la traducción del programa ensamblador a lenguaje máquina hasta la carga del mismo en memoria. En caso de encontrar errores sintácticos de ensamblado aparece un mensaje de error en la ventana de mensajes y, a partir de esa instrucción, no se traduce ni se carga en memoria ninguna otra.

La sintaxis del lenguaje ensamblador se descubrirá poco a poco a lo largo de los diferentes capítulos que componen este manual, pero es interesante introducir ya algunos conceptos básicos. Los primeros que se verán están referidos a los recursos de programación que permite utilizar la programación en ensamblador y que facilitan la programación:

- ***Comentarios***. Estos son muy importantes en los lenguajes de bajo nivel ya que ayudan a seguir el desarrollo del programa y, por tanto, se usan con profusión. Comienzan con un carácter de almohadilla "#" y desde este carácter hasta el final de la línea es ignorado por el ensamblador.

- ***Identificadores.*** Son secuencias de caracteres alfanuméricos, guiones bajos (_) y puntos (.), que no comienzan con un número. Los códigos de operación son palabras reservadas que no pueden ser utilizadas como identificadores.

- ***Etiquetas.*** Son identificadores que se sitúan al principio de una línea y seguidos de dos puntos. Sirven para hacer referencia a la posición o dirección de memoria del elemento definido en ésta. A lo largo del programa se puede hacer referencia a ellas en los modos de direccionamiento de las instrucciones.

- ***Pseudoinstrucciones.*** Son instrucciones que no tienen traducción directa al lenguaje máquina que entiende el procesador, pero el ensamblador las interpreta y las convierte en una o más instrucciones máquina reales. Permiten una programación más clara y comprensible. A lo largo del desarrollo de las prácticas se irán introduciendo diferentes pseudoinstrucciones que permite utilizar este ensamblador.

- ***Directivas***. Tampoco son instrucciones que tienen traducción directa al lenguaje máquina que entiende el procesador, pero el ensamblador las interpreta y le informan a éste de cómo tiene que traducir el programa. Son identificadores reservados, que el ensamblador reconoce y que van precedidos por un punto. A lo largo del desarrollo de las prácticas se irán introduciendo las distintas directivas que permite utilizar este ensamblador.

Por otro lado, los números se escriben, por defecto, en base 10. Si van precedidos de 0x, se interpretan en hexadecimal. Las cadenas de caracteres se encierran entre comillas dobles ("). Los caracteres especiales en las cadenas siguen la convención del lenguaje de programación C:

- Salto de línea: \n

- Tabulador: \t

- Comilla: \"

Problemas propuestos

1. Dado el siguiente ejemplo de programa ensamblador:

```
.data
dato:      .byte 3          # inicialización de una posición
                            # de memoria a 3
.text
.globl main                 # debe ser global
main:      lw $t0,dato($0)
```

Indica las etiquetas, directivas y comentarios que aparecen en el mismo.

2. Dado el siguiente ejemplo de programa ensamblador:

```
.data
dato:      .byte 3          # inicialización de una posición
                            # de memoria a 3
datow:     .word -23        # inicializacion de una palabra a -23

.text
.globl main                 # debe ser global
main:      lw $t0,dato($0)
           lw $t1,datow($zero)
```

Indica las etiquetas, directivas y comentarios que aparecen en el mismo. Escribe el programa y cárgalo en el simulador. Ayudado del anexo con las instrucciones y formato de que dispones en la página web de la asignatura obtén la codificación de las instrucciones ejecutables del programa anterior. Carga el programa en el simulador. Compara la codificación que tu ha obtenido con la que ha generado el ensamblador. Ejecuta el programa e indica el valor final de cada uno de los registros.