Getting Started with SPIM

The rest of this appendix contains a complete and rather detailed description of SPIM. Many details should never concern you; however, the sheer volume of information can obscure the fact that SPIM is a simple, easy-to-use program. This section contains a quick tutorial on SPIM that should enable you to load, debug, and run simple MIPS programs.

SPIM comes in multiple versions. One version, called spim, is a command-line-driven program and requires only an alphanumeric terminal to display it. It operates like most programs of this type:you type a line of text, hit the enter key, and spim executes your command.

A fancier version, called xspim, runs in the X-windows environment of the Unix system and therefore requires a bit-mapped display to run it. xspim, however, is a much easier program to learn and use because its commands are always visible on the screen and because it continually displays the machine's registers. Another version, PCSpim, is compatible with Windows 3.1, Windows 95, and Windows NT. The Unix, Windows, and DOS versions of SPIM are available through *www.mkp.com/cod2e.htm*. This section of the document describes PCSpim, the Windows version of SPIM under Windows 95.

Installation and Graphic Interface Description

To install the Windows version of SPIM, you can download the installation file, *spimwin.exe*, through *www.mkp.com/cod2e.htm*. Execute the installation file and follow the installation procedure. In Windows 95, you can simply activate the icon associated with the file just like any Windows program, or you can select *Start->Run* and type in the directory path and filename. The installation program will execute and inform you when the installation process is complete. When the installation is complete, a group folder with executable file, help files, and uninstaller program is created.

To start PCSpim for Windows, you simply activate the icon labeled *PCSpim for Windows* like any other Windows program. For example, in Windows 95, you can use select *Start->Programs->PCSpim for Windows ->PCSpim for Windows* from the Windows 95 task bar. In Windows 3.1, you can select the application from the File Manager.

When PCSpim starts up, it brings up a large window on your screen (see FigureA.1). The application window is divided into four parts:

- The top section is the menu bar. The menu bar allows you to select *File* operations, set *Simulator* settings, select *Windows* views, and obtain online *Help* information.
- The next section below the menu bar is the toolbar. The toolbar provides quick mouse access to many tools used in PCSpim for Windows.
- The large section in the middle of the application window is the window display section. There are four display windows: Registers, Text Segment, Data Segment, and Messages. To change the view of these four windows, you can select a tiled view from the menu bar: *Windows->Tile*. All of the display windows will be empty when you first execute the program. The following list describes each display window.
    - The Register window display shows the values of all registers in the MIPS CPU and FPU.
    - The Text Segment window display shows instructions both from your program and the system code that is loaded automatically when PCSpim is running.
    - The Data Segment window display shows the data loaded into your program's memory and the data of the program's stack.
    - The Messages window display is the where PCSpim uses to write messages. This is where error messages appear.

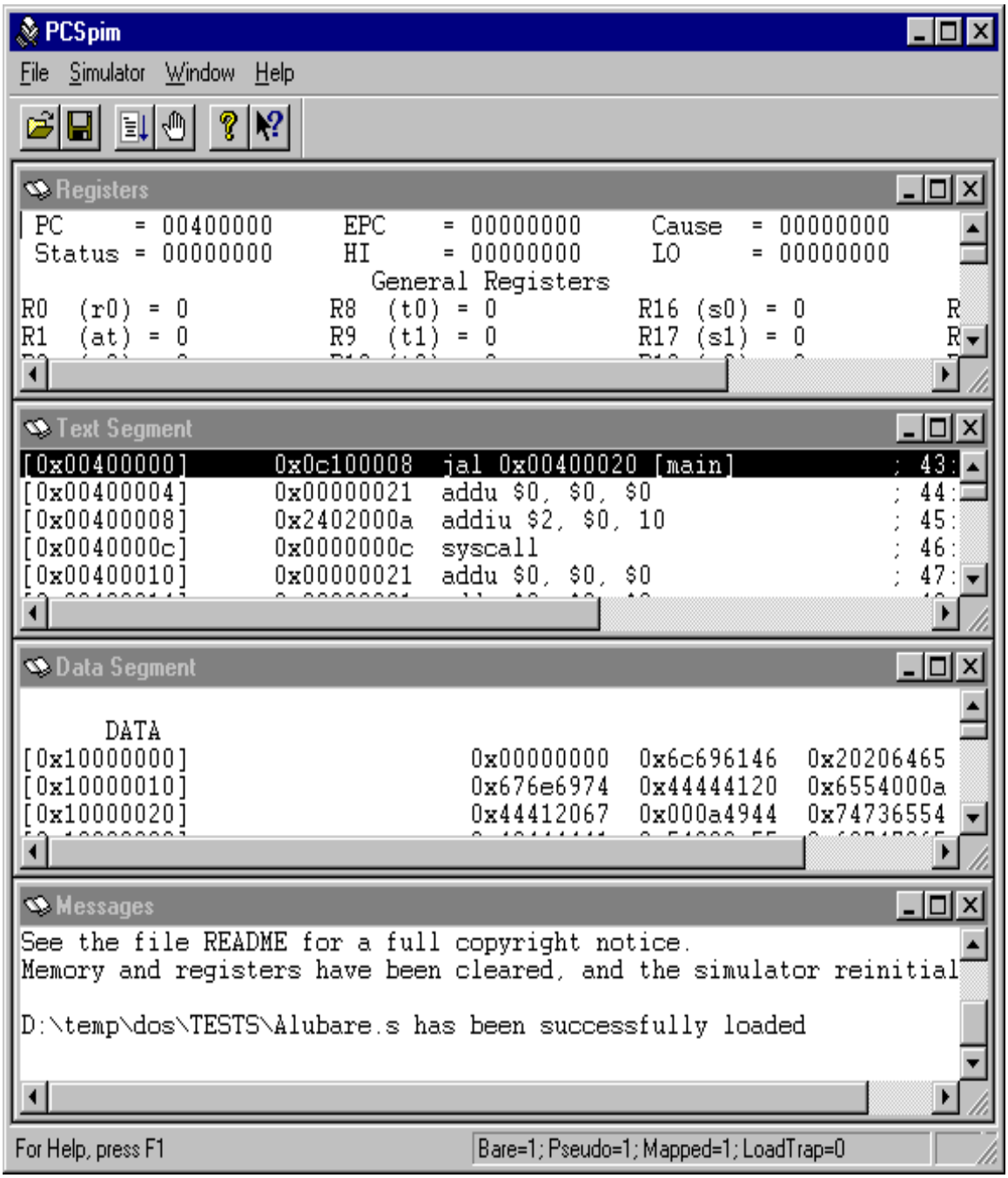

Figure A.1. PCSpim's window interface

- The Status bar section is at the bottom of the application window. The status bar provides information and the current settings of the simulator.

SPIM Command-Line Options

The Windows version of SPIM accepts the following command-line options:

-bare Simulate a bare MIPS machine without pseudoinstructions or the additional addressing modes provided by the assembler. Implies quiet.

-asm	Simulate the virtual MIPS machine provided by the assembler. This is the default.
-pseudo	Allow the input assembly code to contain pseudoinstructions. This is the default.
-nopseudo	Do not allow pseudoinstructions in the input assembly code.
-notrap	Do not load the standard exception handler and start-up code. This exception handler handles exceptions. When an exception occurs, SPIM jumps to location 80000080_{hex}, which must contain code to service the exception. In addition, this file contains start-up code that invokes the routine main. Without the start-up routine, SPIM begins execution at the instruction labeled __start.
-trap	Load the standard exception handler and start-up code. This is the default.
-noquiet	Print a message when an exception occurs. This is the default.
-quiet	Do not print a message at exceptions.
-nomapped_io	Disable the memory-mapped I/O facility. This is the default.
-mapped_io	Enable the memory-mapped I/O facility. Programs that use SPIM syscalls to read from the terminal *cannot* also use memory-mapped I/O.
-file	Load and execute the assembly code in the file.
-execute	Load and execute the code in the MIPS executable file *a.out*. This command is only available when SPIM runs on a system containing a MIPS processor.
-s <seg> size	Sets the initial size of memory segment *seg* to be *size* bytes. The memory segments are named:text, data, stack, ktext, and kdata. The text segment contains instructions from a program. The data segment holds the program's data. The stack segment holds its runtime stack. In addition to running a program, SPIM also executes system code that handles interrupts and exceptions. This code resides in a separate part of the address space called the *kernel*. The ktext segment holds this code's instructions, and kdata holds its data. There is no kstack segment since the system code uses the same stack as the program. For example, the pair of arguments -sdata 2000000 starts the user data segment at 2,000,000 bytes.
-l <seg> size	Sets the limit on how large memory segment *seg* can grow to be *size* bytes. The memory segments that can grow are data, stack, and kdata.

Loading and Running a Program

Let's see how to load and run a program. The first thing to do is to select the open file icon from the toolbar. Alternatively, you can select from the menu bar: *File->Open*. A file open dialog box will appear for you to select the appropriate assembly file. Select the appropriate assembly file and click on the button labeled *Open* in the dialog box. If simulator settings are not correct for the file, and it fails to load, PCSpim will provide you an opportunity to change simulator settings and automatically reload the file.

If you change your mind, click on the button labeled *Cancel*, and PCSpim removes the dialog box. When you load an assembly file, PCSpim removes dialog box, then loads your program and redraws the screen to display its instructions and data. If you have not done so, change the view of the four display windows to a tiled format by selecting from the menu bar: *Windows->Tile*. You should be able to see the program in the Text segment window display.

Each instruction in the Text segment window display is shown on a line that looks like

```
[0x00400000]  0x8fa40000      lw  $4,  0 ($29);     89: lw $a0, 0($sp)
```

The first number on the line, in square brackets, is the hexadecimal memory address of the instruction. The second number is the instruction's numerical encoding, again displayed as a hexadecimal number. The third item is the instruction's mnemonic description. Everything following the semicolon is the actual line from your assembly file that produced the instruction. The number 89 is the line number in that file. Sometimes nothing is on the line after the semicolon. This means that the instruction was produced by SPIM as part of translating a pseudoinstruction.

To run your program, click on the *Go* button in the toolbar. Alternatively, you can select *Simulator->Go* from the menu bar. Your program will begin execution. If you want to stop the execution of your program, select *Simulator->Break* from the menu bar. Alternatively, you can type *Control-C* when PCSpim application window is in focus. A dialog box will appear and ask if you want to continue execution. Select *No* to break the execution. Before doing anything, you can look at memory and registers contents in the Register display window to find out what your program was doing. When you understand what happened, you can either continue the program by selecting *Simulator->Continue* or stop your program by selecting *Simulator->Break* from the menu bar.

If your program reads or writes from the terminal, PCSpim pops up another window called the console. All characters that your program writes appear on the console, and everything that you type as input to your program should be typed in this window.

Suppose your program does not do what you expect. What can you do? SPIM has two features that help debug your program. The first, and perhaps the most useful, is single-stepping, which allows you to run your program an instruction at a time. Select *Simulator->Single_Step* to execute only one instruction. Alternatively, you can press the F10 function key to single step. Each time you step through a program, PCSpim will execute the next instruction in your program, updates the display, and returns control to you. You can also choose the number of instructions in your program to step by selecting *Simulator->Multiple_Step* instead of single stepping through your program. A dialog box will appear and ask you the number of instructions to step.

What do you do if your program runs for a long time before the bug arises? You could single-step until you get to the bug, but that can take a long time, and it is easy to get so bored and inattentive that you step past the problem. A better alternative is to use a breakpoint, which tells PCSpim to stop your program immediately before it executes a particular instruction. Select *Simulator->Breakpoints* from the menu bar. The PCSpim program pops up a dialog box window with two boxes. The top box is for you to enter breakpoint address and the second box is a list of active breakpoints. Type in the first box the address of the instruction at which you want to stop. Or, if the instruction has a global label, you can just type the name of the label. Labeled breakpoints are a particularly convenient way to stop at the first instruction of a procedure. To actually set the breakpoint, and click on the button labeled *Add*. When you are done adding breakpoints, click on the button labeled *Close*. You can then run your program.

When the simulator is about to execute the breakpointed instruction, PCSpim pops up a dialog box with the instruction's address and asks if you want to continue the execution. The *Yes* button continues running your program and the *No* button stops your program. If you want to delete a breakpoint, you can select *Simulator->Breakpoints* from the menu bar, click on the address in the dialog box, and click on the button labeled *Remove*.

Single-stepping and setting breakpoints will probably help you find a bug in your program quickly. How do you fix it? Go back to the editor that you used to create your program and change your source file. After you have made the changes to your source file, simply reload it into PCSpim for Windows by choosing*Simulator->Reload<filename>* from the menu bar. This causes PCSpim to clear its memory and registers and return the processor to the state it was in when PCSpim first started. Once the simulator has reinitialized itself, it will reload your recently modified file.

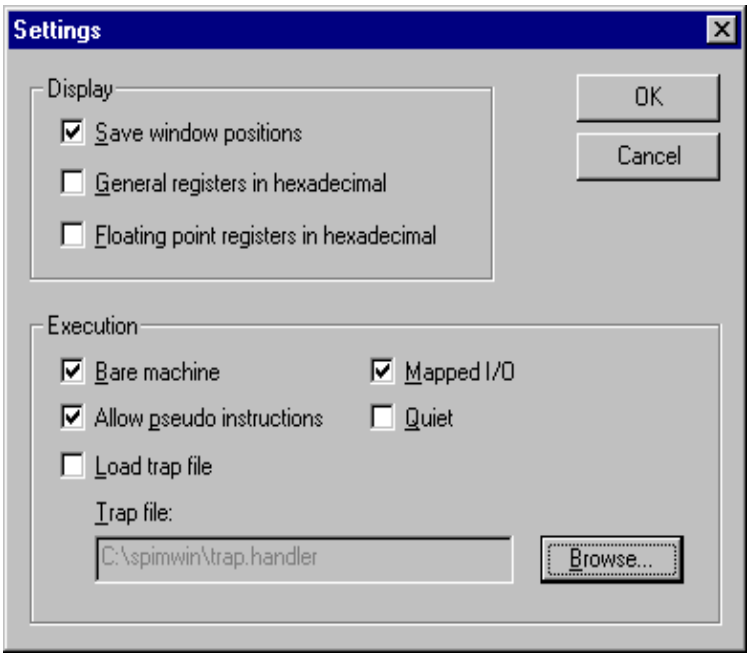

Figure A.2. PCSpim simulator setting dialog box

PCSpim performs other functions that are occasionally useful. When you are more comfortable with PCSpim, you should look at the description in the online help to see what they do and how they can save you time and effort. You can view the online help available with the simulator by selecting *Help->Help_topics* from the menu bar.

Simulator Setting

PCSpim has a graphical interface to view the current setting of the simulator (see FigureA.2). When you start PCSpim, you do not have to enter with any command line parameters. However, you should check your simulator settings either on PCSpim's status bar or the simulator setting dialog box before you load your program. To view or change PCSpim settings in the simulator setting dialog box, select *Simulator->Settings* from the menu bar.

It is very important to set the simulator in the correct setting for your program. PCSpim determines how to load and how your program executes from these settings, so an incorrect setting may cause errors when you run your program. If the simulator setting is incorrect and the program is unable to load correctly, PCSpim allows you to change the simulator settings and reload your program. If you want to change PCSpim settings after you load your program, you should reload your program by selecting *Simulator->Reload* from the menu bar.

The following paragraphs describes the operation of each of the settings in the simulator setting dialog box shown in Figure A.2. Most of the functions are similar to SPIM, its counterparts in the terminal interface version without the graphical interface.

- **Display**

  You can select to view the register contents in decimal or hexidecimal notation. If the check boxes for general registers or floating point registers are selected, a check mark will appear and the register contents will be displayed in hexidecimal notation.

- **Save window positions**

When selected, PCSpim for Windows will record the position of its windows when you exit, and restore them to the same location the next time you run PCSpim.

- **Bare machine**

  When selected, you can simulate a bare MIPS machine without pseudoinstructions or the additional addressing modes provided by the assembler.

- **Allow pseudo instructions**

  If this setting is selected, pseudoinstructions are allowed in your program; otherwise, if the setting is not selected, they are not allowed.

- **Load trap file**

  If this setting is selected, the standard exception handler and start-up code is loaded. When an exception occurs, SPIM jumps to location 80000080hex, which must contain code to service the exception. In addition, the trap handler contains start-up code that invokes the routine main. Without the start-up routine, SPIM begins execution at the instruction labeled __start. The default trap file comes with PCSpim, but you can choose another using Browse button.

- **Mapped I/O**

  If this setting is selected, the memory-mapped I/O facility is enabled. Programs that use SPIM syscalls to read from the terminal cannot also use memory-mapped I/O.

- **Quiet**

  When this setting is enabled, PCSpim does not print a message at exceptions; otherwise, a message is printed when an exception occurs.