Graphical Interface of Simulator

For previous assignments, you have used a terminal interface to the simulator. We called that version `simp`. For simulating a MAL program, you will want to use more sophisticated interface to the simulator. It has the ability to show the contents of registers and memory. We call this version `sim`.

To run a MAL program using `sim`, type

```
/p/course/cs354-common/public/bin/sim filename
```

where `filename` is the file containing your MAL source code.

Window Interface

The Windows version of SPIM/SAL looks different, but operates in the same manner as the non-window version. The UNIX Windows interface has five panes as shown in Figure 1. The top pane displays the contents of the registers. It is continually updated, except while a program is running.

The next pane contains the buttons that control the simulator.

The next two panes display the memory contents. The top one shows instructions from the user and kernel text segments. The lower of these two panes displays the data and stack segments. Both panes are updated as a program executes.

The bottom pane is used to display messages from the simulator. It does not display output from an executing program. When a program reads or writes, its I/O appears in a separate window, called the Console, which pops up when needed.

The buttons that control the simulator have the following functions:

quit

Exit from the simulator. Confirmation that the simulator should be exited is requested.

load

Read a source or executable file into memory. As a side effect, the PC is set to the starting address of the text segment.

run

Begin simulating instructions, starting with the instruction currently specified by the PC.

step

Simulate the single instruction currently specified by the PC.

clear

Clear the contents of registers or memory. The user is given the choice of clearing only the registers or clearing both registers and the user portion (stack and text sections) of memory.

set value

Set the value of a register or memory location. The user specifies which register or memory location is to receive a value, and the user also specifies the value to be set. A register may be specified in one of several ways. The register's number may be given, or the register number may be given preceded by a $, the letter 'r', or the letter 'R'. Each register also has an alias which may be used. The following examples of a register specification all specify the same register.

3
$3
r3
R3
v1
$v1

A memory location is specified by a hexadecimal address. The prefix `0x` on a value forms the syntax for a hexadecimal value. The cursor must be located within the box inorder to enter this information.

print

Print the value of one or more memory locations. The user specifies starting and ending hexadecimal addresses to be displayed in the window used for messages from the simulator. The prefix `0x` on a value forms the syntax for a hexadecimal value. If no value is entered as an ending address, then just the contents of the starting address are displayed.

breakpoint

Set a breakpoint, delete a breakpoint, or list all breakpoints.

help

Print a short description of each of the commands.

halt

Stop the simulator from executing code.

terminal

Show or hide the console or terminal2 windows.

mode

Set SPIM/SAL operating modes to bare, quiet, or memio.

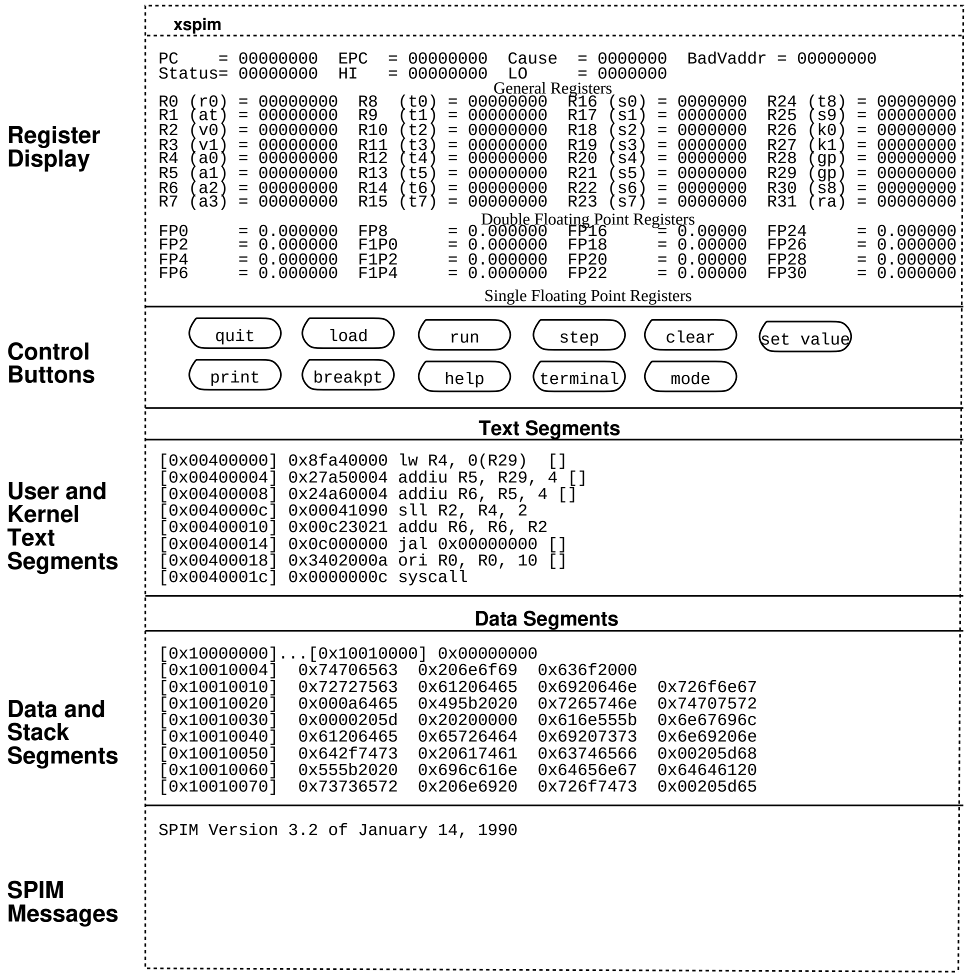

Figure 1 The UNIX window interface.