• **Introduction**

**SPIM** (MIPS spelled backwards) is a simulator for the MIPS processor, written by James Larus at the University of Wisconsin. It comes in three versions:

**spim** - a Unix text-based version for running on any generic Unix processor. This version provides the functionality of the other versions, but does not have a graphical interface. However, it can be used on ordinary terminals, and does not require any specific hardware.

**xspim** - an X-Windows based version that provides an intuitive user interface on systems equipped with X-terms. It is the version that is discussed specifically in this handout.

**PCspim** - A Microsoft Windows - based version that runs on PC’s. It provides a Windows interface that is very similar to **xspim**.

**spim** and **xspim** are installed on the CS department’s HP-UX machines. The PC version can be downloaded from the location specified at the textbook’s website - a link to this website is on the class website. A manual, “SPIM S20: A MIPS R2000 Simulator,” by James Larus, is available from the same ftp site. A more complete treatment of the MIPS architecture and the SPIM simulator is given in Appendix A of the textbook.

The purpose of this handout is to describe how to use **xspim** on the CS Department computers, and to provide a quick tutorial on how to do a few common operations. Much of the information also applies to **spim** and **PCspim**.

• **Setting up XSPIM - First time setup**

Before you can use **xspim**, you need to copy the trap handler file to your directory:

• Login to one of the CS department HP machines.

• Copy the file named **trap.handler** to your directory:

    ```bash
    cp /var/spim/trap/trap.handler .
    ```

• **Starting XSPIM**

Type **xspim**. A window will pop up, similar to that shown in Figure 1. This window contains five separate panes, showing (from top to bottom) ① the contents of the CPU registers, ② control buttons, ③ the **text** segment, ④ the **data** (and stack) segment, and ⑤ messages.
Figure 1: Picture of the initial xspim window.
When **xspim** is initially started up, a small amount of MIPS code, called the *trap handler*, is automatically loaded (from the file **trap.handler**). This code, which appears in the text window at locations 0x00400000 - 0x0040001c, initializes the simulator and calls your **main** program when executed.

### SPIM Memory Configuration

SPIM allocates memory locations for the **text** and **data** segments according to standard MIPS assembler conventions. The memory layout is shown in Figure 2. The **text** (code) segment starts at location 0x00400000, **data** starts at 0x10000000 and extends upward, and the system stack grows downward from the top of memory, 0xffffffff.

### Loading a program into xspim

SPIM accepts MIPS assembler source code, usually in a file with a name of the form **filename.s**. SPIM has a built-in assembler, so your program will be assembled (translated) as it is loaded. To load a file into SPIM, click the **load** command button, and type the file name at the resulting prompt. The file you specify will be loaded into the memory of the simulator. Figure 3 shows the resulting display after the small program **hello.s** is loaded into the simulator. Since the trap handler already occupies the first few text and data locations, in this example the **main** symbol is loaded at location 0x00400020 and the program data starts at 0x10010000. As the file is loaded, any errors that the simulator encounters in reading and translating the source file are shown in the **messages** pane.

### Stepping and Executing the Program

The program can now be executed. To run the program straight through, click the **run** command button. Most likely, the first time you run a program, you will want to observe the

![Figure 2: Standard layout of MIPS memory](image)

---

**SPIM Simulator - 3**
Figure 3: xspim window immediately after the hello.s program has been loaded. The first instruction in the program (specified by the symbol main) is at address 0x00400020. Note that in the Data Segment, the ASCII characters for the string "Hello, World" have been loaded starting at location 0x10010000 (48 hex = 'H', 65 hex = 'e', 6c hex = 'l', etc.)
operation of the program – click on the \texttt{step} button. Figure 4 shows the resulting display. One instruction will be executed each time the \texttt{step} button is clicked. The register contents are updated after each instruction is executed, and the instruction that will be executed next is highlighted, so that you can verify correct program operation.

If any output is produced by the program, it is displayed in a separate window that pops up when the first output is produced. Figure 5 shows the output window and the results produced by the complete execution of \texttt{hello.s}. The output window can be hidden or raised manually by clicking the \texttt{terminal} button.

• **SPIM Services**

SPIM provides a few primitive services to the programmer; these services would normally be provided for the programmer of a “real” MIPS machine by the operating system. These services are called by loading a function code into \$2, loading any arguments needed into \$4, and then executing the \texttt{syscall} instruction. The following table lists the services, call codes, arguments and results of the SPIM services.

<table>
<thead>
<tr>
<th>Routine</th>
<th>Code in $2</th>
<th>Arguments</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>print_int</td>
<td>1</td>
<td>$4 = integer</td>
<td></td>
</tr>
<tr>
<td>print_float</td>
<td>2</td>
<td>$f12 = float</td>
<td></td>
</tr>
<tr>
<td>print_double</td>
<td>3</td>
<td>$f12 = double</td>
<td></td>
</tr>
<tr>
<td>print_str</td>
<td>4</td>
<td>$4 = addr of string</td>
<td></td>
</tr>
<tr>
<td>read_int</td>
<td>5</td>
<td></td>
<td>integer in $2</td>
</tr>
<tr>
<td>read_float</td>
<td>6</td>
<td></td>
<td>float in $f0</td>
</tr>
<tr>
<td>read_double</td>
<td>7</td>
<td></td>
<td>double in $f0</td>
</tr>
<tr>
<td>read_str</td>
<td>8</td>
<td>$4 = addr of buffer (memory)</td>
<td>str in buffer</td>
</tr>
<tr>
<td>malloc</td>
<td>9</td>
<td>$5 = max length of str + 1</td>
<td>address in $2</td>
</tr>
<tr>
<td>exit</td>
<td>10</td>
<td>$4 = # of bytes desired</td>
<td></td>
</tr>
</tbody>
</table>

• **Other Command Buttons**

\texttt{clear} - Reinitializes registers or memory.
\texttt{set value} - allows you to specify the value in a register or memory location.
\texttt{print} - prints (in the message pane) the value of a register or memory location.
\texttt{breakpoint} - allows you to set, clear or list \textit{breakpoints}. If a breakpoint is set at a particular memory location, the simulator will execute the program until it reaches that memory location.
\texttt{mode} - sets the SPIM operating mode (we shouldn’t need to use this button).
\texttt{quit} - exits the simulator.
**Figure 4: xspim window, showing the single step button activated. The simulator is just about to execute the first instruction (highlighted) in the hello.s program**

**SPIM Simulator - 6**
Figure 5: xspim window immediately after executing hello.s. Note the values in the registers that were used by the program. Also, note the terminal window that has popped-up with the output results.