

**CPSC 321:501-503 - Computer Architecture  
Texas A&M University  
Department of Computer Science**

**Fall 2006**

**Lab 1 (10 pts) - Introduction to SPIM Simulator for the MIPS Assembly Language on  
the UNIX and PC Environments  
Complete by yourself**

---

**Release date: 4 September 2006**  
**Due date: One week following lab**

---

### **Objective**

This laboratory assignment will help you familiarize yourself with the UNIX and PC environments (briefly), the **spim** simulator and other utilities, such as text editors. You will be using SPIM, the MIPS simulator by James R. Larus to code and run your first MIPS assembly language program. In addition to the SPIM documentation under the [Useful Links](#) section of the course Web site, your textbook [1] contains a discussion of SPIM in Appendix A (on the CDROM). You will also learn how to interact with UNIX processes running on a UNIX host from a PC console using the X11 windowing environment.

### **Prelab Requirements**

*Before* entering the lab, make sure that you have your own UNIX and PC accounts and that you can log into them.

### **Introduction to SPIM**

SPIM is a software simulator that loads and executes assembly language programs for the MIPS R2000/R3000 RISC computers. Assembly is a low-level language with instructions that correspond very closely to the machine code that a processor executes. SPIM can read and immediately run files containing MIPS assembly language statements. SPIM is a self-contained system for running these programs and contains a debugger and interface to the operating system. The installed version of SPIM is 6.5.

SPIM was written by James R. Larus, then at the Computer Sciences Department of University of Wisconsin, Madison ([larus@cs.wisc.edu](mailto:larus@cs.wisc.edu)). SPIM is very portable, which allows students to generate code for a simple, clean, orthogonal computer. SPIM currently runs on a wide variety of UNIX and Windows systems. SPIM is copyrighted by James Larus and can be freely used for non-commercial purposes. You can get source and pre-compiled files from <http://www.cs.wisc.edu/~larus/spim.html>.

**Note:** If you have a personal computer, it is recommended that you download and install SPIM (ver. 6.5) on your own machine. The Computer Science Labs maintain versions for both UNIX and PC platforms.

SPIM implements almost the entire MIPS assembler-extended instruction set for the R2000/R3000 (with the exception of some complex floating-point comparison instructions and direct manipulation of the memory system page tables). The MIPS architecture has evolved considerably since then (in particular the 64-bit extensions), meaning that SPIM will not run programs compiled for recent MIPS processors.

SPIM implements both a simple, terminal-style (command line) interface and a visual windowing interface. On UNIX, the **spim** program provides a terminal interface and the **xspim** program provides the X11 window interface. On PCs, the **spim** program provides a text interface and **PCSpim** provides a Windows interface. [1] Appendix A or [2] are the best introduction to the software. [3] is a detailed language reference.

## MIPS and Laboratory Assignments

For the assembly code lab assignments, you will use two programs: SPIM and a text editor. We recommend that you learn to use a real programmer's text editor. That is, an editor that is designed specifically for writing computer programs, not just an editor designed for arbitrary text, such as Pico or Microsoft Notepad. We strongly recommend that you use an editor such as EMACS (**xemacs**, **emacs**). The department maintains EMACS on both UNIX and PCs.

### SPIM on PC

There are two versions of the SPIM for the PC. **spim** (a command line version) and **PCSpim** (a graphical interface one). Look under the **Start->Programs->Programming Tools** menu to find these programs. (You can find **emacs** under **Start->Programs->Publishing**). Launch **PCSpim for Windows**.

### SPIM on UNIX

Under UNIX there are also two versions of SPIM: **spim**, a command line version, that runs in a text window, and **xspim**, one with a graphical interface.

Both **spim** and **xspim** can be executed from the shell command line. Make sure that the file called **trap.handler** is in the same directory from within which you launch **xspim** or **spim**.

### Building SPIM on UNIX Systems

We maintain copies of the SPIM executables for UNIX (**xspim**, **spim**) and their source code on the [Useful Links](#) section of the class web page. If you desire, you can create your own local version. Login to your CS UNIX account, and create a subdirectory dedicated to `cpsc321`. Then download SPIM from <http://www.cs.wisc.edu/~larus/spim.html> and save the file to the above subdirectory. You need to decompress and “untar” the archive file you downloaded. A quick way is to pipe the file through the command sequence

```
> gunzip -c spim.tar.gz | tar xvf - +
```

and then build and install SPIM in your account following the instructions in the `README` file found in this distribution. You may ask the TA for help with this. Alternatively, use the local copy of SPIM on our Web page.

We recommend that you learn and use **xemacs** or **emacs** under UNIX. This is family of full-featured editors. Under UNIX, to invoke XEmacs, simply type **xemacs** at the shell prompt.

## A MIPS Assembly Language Program Example

In this class we will be studying the MIPS instruction set and its architecture. An example MIPS assembly language program is shown below.

```
## Your first MIPS assembly program
## Notice the stylized format of the code:           3 columns:
```

```

## (1) Optional labels,
## (2) Machine instructions, assembler directives and their operands,
## (3) Optional comments: everything to the right of a '#' until end of line is
##     ignored.

        .data                # "data section" global, static modifiable data

SID:    .word    100
spc1:   .asciiz  " "
nl:     .asciiz  "\n"
tb:     .asciiz  "\t"
msg1:   .asciiz  "Hello, World\n"
msg2:   .asciiz  "My name is: XXXXXXXXXXXX\n"
msg3:   .asciiz  "\nMy name is still XXXXXXXXXXXX !\n"

        .text                # "text section" code and read-only data

        .globl  main         # declare `main' as a global symbol
main:   la      $a0,  msg1
        li      $v0,  4      # "print string" system call
        syscall
        la      $a0,  msg2
        li      $v0,  4
        syscall
        lw      $a0,  SID

        la      $a1,  spc1

Loop:   beq     $0,    $a0,  Exit
        add     $a0,  $a0,  -1
        li      $v0,  1      # "print int" system call
        syscall
        move    $t0,  $a0
        move    $a0,  $a1
        li      $v0,  4
        syscall
        move    $a0,  $t0
        j      Loop

Exit:   la      $a0,  msg3
        li      $v0,  4
        syscall

## Exit from program
        li      $v0,  10     # "Exit" system call
        syscall

```

## Requirements for Lab 1

1. Type the MIPS code shown above into a text editor. Replace the **XXXXXXXXXX** with your name and then save the file under the name **LAB1.s**. Once you have saved the file, run this MIPS assembly program using both the **PCSpim** and **xspim** simulators. The MIPS assembly code is listed below. We will explain the meaning of the instructions in class. Load your source file into **PCSpim** using **File->Open**. Then run the program using **Simulator->Go**. If you typed the sample program in correctly, the message "... **successfully loaded**" will appear in the "**Messages**" window. **Note:** the first time you run the program you may get an error message that file "**trap.handler was not found.**" From **Simulator->Settings**, change the path for the

trap file from the default setting to: **C:\ProgramFiles\PCSpim\trap.handler** and the problem should go away. If your program runs correctly, a number of messages should appear in the “**Console**” window.

2. Run **LAB1.s** under UNIX as well.
3. Write a short paragraph explaining what this program does. You do not have to understand the assembly details at this point. However, you need to be able to tell what the code does in rough terms. Save this in a file called **LAB1.rep**. Discuss which environment is better overall for you to use and for which reasons.
4. Package the two files into one archive file called **LAB1.tar** (or **LAB1.zip**) in the UNIX (PC) environment. Do not forget to include the student information according to the assignment submission guidelines shown in the class web page.
5. Turn in your work using the **turnin** command.

## Running and Interacting with UNIX from Remote Hosts

This section is optional, but provides useful information for interacting with UNIX processes from other hosts. One very interesting feature of UNIX is the capability to execute and interact seamlessly with commands at remote UNIX hosts. UNIX uses the X11 windowing server to make remote command interaction seamless. The basic mode of operation is that a remote UNIX host executes any UNIX command but the user interacts with it from another UNIX or PC host. This is accomplished by having the host that a user is using run an X11 server. The remote command can open an X11 (client) window on the host with the X11 server and let the user interact with it directly as if the user was sitting on the host that executes the command.

### Interacting with UNIX Applications from a PC

To run and UNIX applications from within a PC, perform the following steps:

1. Locate and launch the X11 server on the PC (it is called **X-Win32**).
2. Use the **ssh** command from the PC to login to a UNIX host (say to **unix.cs.tamu.edu**).
3. After you login, type the command **who** at the shell prompt. You should see a listing of users and at the far right end you should see the host names from which these users have logged on, inside parentheses. Assume that you are logged from a PC named **myPC**.
4. Then issue the command **setenv DISPLAY myPC:0**.
5. To run a UNIX command, type **xterm** to launch an X11 terminal window, which will be displayed on your PC. You can then type a UNIX command at the prompt. Instead of a terminal window, you can just type **xspim** to launch SPIM on UNIX and interact with it from your PC.

### Interacting with UNIX Applications from other UNIX Hosts

To run and UNIX applications from within another UNIX host follow these steps:

1. Log on to a UNIX host and launch the X11 server (all SUN Solaris workstations at CS run the CDE graphical environment).
2. Use the **ssh** command to login to the remote UNIX host, say **unix.cs.tamu.edu**.
3. After logging in, type the command **who** at the shell prompt. You should see a listing of users and at the far right end you should see the hosts names from which these users have logged on, inside parentheses. Assume that you are logged from a SUN workstation called **interactive**.
4. Issue the command **setenv DISPLAY UNIX:0**.
5. To run a UNIX command, type **xterm** to launch an X11 terminal window, which will be displayed on your UNIX host (**interactive**). You can then type a UNIX command at the prompt. Instead of a terminal window, you can just type **xspim** to launch SPIM on the remote UNIX and interact

with it from your local UNIX host.

## References

- [1] David. A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan-Kaufmann Publishers Inc., third edition, 2005, ISBN 1-55860-604-1.
- [2] James R. Larus, □SPIM S20: A MIPS R2000 Simulator,□ Unpublished Documentation for SPIM, Computer Sciences Department, University of Wisconsin–Madison, <http://www.cs.wisc.edu/~larus/spim.html>. Latest version is 6.4.
- [3] Robert Britton, □MIPS Assembly Language Programming,□ Prentice Hall, 2003.

Copyright © 2001-2003, Michael E. Thomadakis. This document may be copied and used for non-commercial purposes, as long as this copyright notice remains on it. This document was converted to HTML and updated in 2006 by Hank Walker.