Texas A&M University
College of Engineering
Computer Science Department
CPSC 321:501–506 Computer Architecture
Fall Semester 2004
Lab1

Introduction to SPIM Simulator for the MIPS Assembly Language on the UNIX and PC Environments

Due Date: One week after your lab session – Complete by yourself.

1 Objective

This laboratory assignment will help you familiarize yourself with the UNIX and PC environments (briefly), the `spim` simulator and other utilities, such as, programmers' editors. You will be using SPIM, the MIPS simulator by James R. Larus to code and run your first MIPS assembly language program. Bring your [1] textbook to lab with you, to use the SPIM documentation in Appendix A (A-38). You will also learn how to interact with UNIX processes running on a UNIX host from a PC console using the X11 windowing environment.

2 Prelab Requirements

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

3 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 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`). 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, DOS 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 URL:
`http://www.cs.wisc.edu/~larus/spim.html`

Note: It is recommended that you download a version (ver. 6.5) of SPIM for the platform of your choice and install it 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 or SGI processors.

SPIM implements both a simple, terminal-style (command line) interface and a visual windowing interface. On Unix, the `spim` program provides the terminal interface and the `xspim` program provides the X11 window interface. On PCs, the `spim` program provides a DOS (text) interface and `PCSpim` provides a Windows interface. [1, Appendix A] (Hennessy & Patterson, *Computer Organization and Design: The Hardware/Software Interface*) or [2] are the best introduction to the software.

4 MIPS and Laboratory Assignments

For the first part of the course you will use two programs: SPIM and a decent 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 editing arbitrary text such as Pico or Microsoft Notepad. We strongly recommend to become familiar with any of the EMACS (`xemacs`, `emacs`) or VI (`vi`, `vim`) families of text editors. There are versions of both editors on most platforms. The department maintains both EMACS and VI editors on UNIX and PC workstations.

4.1 PC Environment

4.1.1 Locating Software in Windows-2K/XP

On department PC machines, a good choice is `VIM` (an implementation of the `vi` editor, which was developed by Bill Joy for the Unix operating system), or `xemacs/emacs` (developed by Richard Stallman, founder of the GNU project). You are responsible for selecting and learning the editor of your choice.

4.1.2 SPIM on PC

There are two versions of the SPIM for the PC. `spim` (a command line version) and `PCSpim` (a graphical interface one). To launch `PCSpim` (or `vim` or `Emacs`), find the "`Programming Tools`" menu under the "`Programs`" menu, which is on the "`Start`" button. Use this key sequence (which from now on will be written as: `Start->Programs->Programming Tools`) and launch "`PCSpim` for Windows."

4.2 UNIX Environment

4.2.1 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` readily execute 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`. More documentation about the SPIM simulator is found in Appendix A of our main textbook by Patterson and Hennessy. You may develop your MIPS programs on UNIX, but you are not required to. The UNIX environment is more powerful and flexible in developing code, especially remotely.

4.2.2 Local Copies of SPIM for UNIX

We maintain copies of the SPIM executables for UNIX (`xspim`, `spim`) and their source code on the class web page. Download the executables or the source code from the appropriate link.

4.2.3 Building SPIM on UNIX and UNIX-Like Systems

Login to your CS UNIX account, and create a subdirectory dedicated to `cpsc321`. Then download SPIM from URL `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

```
    > 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, download the source code from the web page of our course, as explained above.

4.2.4 Editing in the UNIX Environment

We recommend that you learn and use `xemacs` or `emacs` under UNIX. This is family of editors which is highly customizable with respect to the type of file that is being edited. Under UNIX, to invoke XEmacs, simply type `xemacs` at the shell prompt.

5 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: XXXXXXXXX\n"
msg3:   .asciiz "\nMy name is still XXXXXXXXX !\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
```

6 Requirements for Lab #1

1. Type the MIPS code shown above into a text editor. Replace the `XXXXXXXXX` by 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 web page of the class.

5. Turnin your work using the `turnin` UNIX command.

7 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.

7.1 Interacting with UNIX Applications from a PC

To run and UNIX applications from within a PC follow the steps below.

1. Locate and launch the X11 server on the PC (it is called `Xwin-32`).
2. Use the `ssh` command from the PC to login to a UNIX host (say to `unix.cs.tamu.edu`).
3. After log on, 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 PC named "`myPC`".
4. Then issue the command `setenv DISPLAY myPC:0`.
5. Run any UNIX command as follows.
   - Type `xterm` to launch an X11 terminal which will be displayed on your PC (`myPC`). An X11 window will appear on your PC which you may use and type any UNIX command.
   - Or, type `xspim` to launch SPIM on UNIX and interact with it from your PC.

7.2 Interacting with UNIX Applications from other UNIX Hosts

To run and UNIX applications from within another UNIX host follow the steps below.

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 log on, 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 interactive:0`.
5. Run any UNIX command as follows.
   - Type `xterm` to launch an X11 terminal which will be displayed on your UNIX host (`interactive`). An X11 window will appear on your workstation and the you may type any UNIX command there.

- Type `xspim` to launch SPIM on the remote UNIX host and interact with it from your local UNIX host.

[1]

References

[1] David. A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Morgan-Kaufmann Publishers Inc., second edition, 1998, ISBN 1-55860-428-6. Publisher's URL: `http://www.mkp.com/books_catalog/catalog.asp?ISBN=1-55860-428-6`.

[2] James R. Larus, "SPIM S20: A MIPS R2000 Simulator," Unpublished Documentation for SPIM, Computer Sciences Department, University of Wisconsin–Madison, URL: `http://www.cs.wisc.edu/~larus/spim.html`. Latest version is 6.4.

[3] Robert Britton, "MIPS Assembly Language Programming," Prentice Hall, 2003.

[1]Copyright ©2001–2003 by Michael E. Thomadakis. This document may be copied and used for non-commercial purposes, as long as this copyright notice remains on it.