CS220: Introduction to SPIM

SPIM is a MIPS processor simulator. It can be used to run and test MIPS assembly language programs. The simulator is installed on several lab machines. The list of machines can be found at the end of this document. You can ssh to any of these machines and invoke SPIM by typing `spim` on the command line. I have posted a document that lists the spim commands. The most useful ones are `load <filename with full path>` and `run`. The first command loads the assembly language program in file `filename` into the simulated MIPS processor's memory. The second one runs the program. Before you load and run another new program, you should clear the processor's memory and states by invoking the command `reinitialize`. You can exit the simulator by typing `exit`. Here is a small example. Suppose the following MIPS assembly language program is stored in file `mips_programs/helloworld.s`.

```
        .data
msg:    .asciiz "Hello World"

        .text
        .globl main
main:   li $v0, 4       # syscall 4 (print_string)
        la $a0, msg     # argument: string (la is a pseudo-instruction: load address)
                        # loads the address of the label msg into $a0
        syscall         # print the string

        jr $ra          # retrun to caller
```

In this program, `.data`, `.asciiz`, `.text`, `.globl` are called assembler directives. You can read about these directives in Section A.10 of Appendix A of Patterson and Hennessy. Note that anything following a # mark is a comment. One new instruction in this program is `syscall`. It stands for system call. System calls are required for communicating with resources/hardware outside the CPU. The `syscall` instruction takes a syscall number in register $v0 and four arguments in $a0, $a1, $a2, $a3. Return value of system calls are put in $v0. In this example, a system call is used to print a string on the display. The supported system calls and their numbers are listed in Figure A.9.1 of Section A.9 of Appendix A of Patterson and Hennessy. You will mostly need the print system calls for printing the outputs of a program and read system calls for reading inputs to the program. In this example, syscall number is 4 (which is the number for print_string). The argument of the system call is the pointer to the string to be printed i.e., address of the label `msg`. The syscall number and the arguments must be set up before invoking the `syscall` instruction. Note that since `main` is a function, you must have `jr $ra` at the end of `main`. Whoever calls `main` will set up $ra correctly. You don't have to worry about it.

To run this program on SPIM, you have to start `spim` and do the following.

```
$ spim
Loaded: /usr/share/spim/exceptions.s
(spim) load "mips_programs/helloworld.s"
(spim) run
Hello World(spim) exit
$
```

You can do some simple changes to this program. For example, try changing the string to be displayed e.g., `msg:  .asciiz "Hello World\n"`. Try to print an integer by changing the syscall number to the number of print_int and changing `la $a0, msg` to `li $a0, N` where replace `N` by an integer value. Once you are familiar with the environment, try more involved programs like recursive codes, etc.. You may attempt exercises A.2, A.6, A.7, A.8, A.9, and A.10 from the end of Appendix A of Patterson and Hennessy. You can also attempt the worked out examples from Chapter 2 and Appendix A. Note that for reading inputs, you will have to use the read_int, read_float, read_double, read_char, and read_string system calls from Figure A.9.1 of Appendix A. The values read by these system calls can be found in $v0 (for read_int and read_char) or $f0 (for read_float) or the pair ($f0, $f1) (for read_double). Here is a simple program that reads an integer from keyboard, adds 42 to the read integer, and prints the result.

```
        .text
        .globl main
main:   li $v0, 5        # syscall 5 (read_int)
        syscall
        addi $a0, $v0, 42 # print_int argument
        li $v0, 1        # syscall 1 (print_int)
        syscall          # print the string

        jr $ra           # retrun to caller
```

The list of machines where SPIM is installed is given below. You will have to ssh into one of these machines and then run `spim`.

```
172.27.19.63
172.27.19.64
172.27.19.65
172.27.19.66
172.27.19.67
172.27.19.68
172.27.19.69
172.27.19.70
172.27.19.71
172.27.19.73
172.27.19.75
172.27.19.77
172.27.19.78
172.27.19.79
172.27.19.80
172.27.19.82
172.27.19.83
172.27.19.85
172.27.19.87
172.27.19.88
172.27.19.89
172.27.19.90
172.27.19.92
172.27.19.96
172.27.19.98
172.27.19.101
172.27.19.103
172.27.19.104
172.27.19.106
```

```
172.27.19.107
172.27.19.108
172.27.19.109
172.27.19.111
172.27.19.114
172.27.19.115
172.27.19.116
172.27.19.118
172.27.19.119
172.27.19.121
172.27.19.122
172.27.19.123
172.27.19.124
```