Laboratory 1: Postlab

Date Sep 24/ 2014 Section 01

Name Ren Chen

Learn More About MIPS

In this exercise you will be using the floating-point registers of MIPS.

Background

For practical reasons, the original definition of the R2000 architecture defined a MIPS processor as composed of

- integer unit (the actual CPU)
- coprocessors

The idea was that the technology just did not allow to integrate everything on a single silicon die. Therefore coprocessors could be separate integrated circuits, or could just be software emulators (i.e. for floating point) if the price was a serious concern. Defining coprocessors neatly separates the architectural definition from the implementation constraints or details. Keep in mind that the same architecture may have several implementations, each using possibly different technologies and having different performance.

SPIM simulates two coprocessors

- coprocessor 0: handles interrupts, exceptions and the virtual memory system
- coprocessor 1: floating point unit (FPU)

The FPU performs operations on

- single precison floating point numbers (32 bit representation); a declaration like `float a=1.5;` in C would reserve space for a variable called a which is single precision floating point, and is initialized to 1.5
- double precision floating point numbers (64 bit representation); a declaration like `float a=1.5;` in C would reserve space for a variable called a which is double precision floating point, and is initialized to 1.5

The coprocessor has 32 registers, numbered from 0 to 31 (their names are **`$f0`** to **`$f31`**). Each register is 32 bit wide. To accomodate doubles registers are grouped together (0 with 1, 2 with 3, ..., 30 with 31). To simplify things, floating point operations use only even numbered registers.

© 1996, Virgil Bistriceanu

Step 1

Create a program (use *lab1.1.asm* as a model) that reads a float (i.e. single precision number) from the keyboard and then outputs it.

You will need to look at the instruction set to find out what instruction to use for moving a float from one floating point register to another (`addu $f12, $f0, $0` will not work).

Save the program as *lab1.3.asm*. Run the program and fill out the 'Single precision' section of the following table (the content of registers after program finished). The input you type at the keyboard when prompted will be the last four digits of your Social Security Number, followed by a period (.), followed by the current year (four digits).

Step 2

Create a program that reads a double (i.e. double precision number) from the keyboard and then outputs it. Save the program as *lab1.4.asm*. Run the program and fill out the 'Double precision' section of the following table (the content of registers after program finished). The input you type at the keyboard when prompted will be the same as at Step 1.

Register	Single precision	Double precision
$f0	3f666666	ffffffffcccccccd
$f2	0	3fecccc
$f4	0	0
$f6	0	0
$f8	0	0
$f10	0	0
$f12	3f666666	ffffffffcccccccd
$f14	0	0
$f16	0	0
$f18	0	0
$f20	0	0
$f22	0	0
$f24	0	0
$f26	0	0
$f28	0	0
$f30	0	0

© 1996, Virgil Bistriceanu

Step 3

Return to your lab instructor copies of *lab1.3.asm* and *lab1.4.asm* together with this postlab description. Ask your lab instructor whether copies of programs must be on paper (hardcopies), e-mail or both.

© 1996, Virgil Bistriceanu