Laboratory 1: Prelab

Date 10/09/2014 Section 01

Name Ren Chen

Step 1

Using a text editor, enter the program P.1. The sharp sign (#) starts a comment, a name followed by a colon (:) is a label, and names that start with a period (.) are assembler directives.

P.1:

```
            .data 0x10000000
msg1:       .asciiz "Please enter an integer number: "

            .text
            .globl main

# Inside main there are some calls (syscall) which will change the
# value in $31 ($ra) which initially contains the return address
# from main. This needs to be saved.

main:       addu $s0, $ra, $0       # save $31 in $16
            li $v0, 4               # system call for print_str
            la $a0, msg1            # address of string to print
            syscall

# now get an integer from the user
            li $v0, 5               # system call for read_int
            syscall                 # the integer placed in $v0

# do some computation here with the number
            addu $t0, $v0, $0       # move the number in $t0
            sll $t0, $t0, 2         # last digit of your SSN instead of 2

# print the result
            li $v0, 1               # system call for print_int
            addu $a0, $t0, $0       # move number to print in $a0
            syscall

# restore now the return address in $ra and return from main
            addu $ra, $0, $s0       # return address back in $31
            jr $ra                  # return from main
```

Before you continue make sure you have entered the last digit of your SSN instead of 2 in the instruction `sll $t0, $t0, 2`

Let's note here, before we move on, that we save the content of register **`$ra`** into another register. The register **`$ra`** (used for the call/return mechanism) has to be saved when you enter main, only if you either call system routines (using syscall) or if you call your own routines (there will be a laboratory dedicated to this topic). Saving **`$ra`** in **`$s0`** (or in any other register for that matter) only works if

- there is only one call level (in other words there are no recursive calls of the routine)
- the routines you are calling do not modify the register you use for saving

Step 2

Save the file under the name *lab1.1.asm*. Save the file in the same directory where the simulator itself is. Otherwise you will have to change the search path such that the system will be able to execute the simulator no matter what the current working directory is.

Here we use the '.asm' extension for the file name as to differentiate between hand written code and compiler generated assembly code. A compiler would use the '.s' extension for the file containing the assembly code.

Step 3

Start the SPIM simulator by typing `spim` at the prompt. You will see a copyright message, followed by a message indicating that the trap handler has been loaded.

In case you get an error message that says something like "spim: command not found", then you must make sure the directory where spim is located is in your search path.

Step 4

At the `(spim)` prompt type

```
load "lab1.1.asm"
```

If you have any error messages go back to Step 1 and make sure you have not made any mistakes when typing the program. If there is no error message, then your program has been translated and you can run it.

Step 5

At the `(spim)` prompt type

```
run
```

to have the program execute. You will be prompted for an integer number; after you enter it, the program will print a result and exit. You know the program has finished to execute since the simulator returns to the `(spim)` prompt.

You can run the program again either by typing `run` at the prompt or by simply pressing the Enter key (which re-executes the last command).

Step 6

You now try to figure out what program *lab1.1.asm* does. Run it several times with various input data. Use

both positive and negative integers. Fill out the following table:

Test cases for *lab1.1.asm*

Input number	Output number
-5	-20
-4	-16
-3	-12
-2	-8
-1	-4
1	4
2	8
3	12
4	16
5	20

Step 7

What is the formula that describes the relation between the output and the input?

```
output = input * 4
```