CSE 2304 Lab 1

2/7/2016

Due: 11:59PM, Monday 2/15/2016

The purpose of this lab assignment is to get you up and running with the simulation tool (PCSPIM- windows version of SPIM) we will be using throughout this semester to write MIPS assembly codes. This lab will give you a broad idea on how to write, execute, and debug MIPS codes in SPIM.

The version of MIPS simulator for windows (PCSPIM) is available in the computers in ITE 134. You should be able to use those computers during the regular hours the lab is open.

I recommend each of you to download a copy of SPIM for your platform from this website http://spimsimulator.sourceforge.net/ and install it in your home/personal computer, so that you can work on your assignments anytime, and anywhere you desire. You might also want to download the SPIM documentation manual from the same website. Furthermore, those of you who are interested can download and install Mipster from http://www.downcastsystems.com/mipster. Mipster has some nice features including Syntax highlighting, intellisense Tooltip, *etc*.

Part 1 – In-class: The "Hello World!" program has been for years the traditional way to initiate students to new programming languages. So our first program will be a simple "Hello World" program. This program will simply display the words "Hello World!" to the console. Learning how to display strings and numbers to the console early in the semester is very important, since this will be widely used in subsequent labs to display results of your calculations.

Steps to edit the MIPS code and run the program.

1- Use any of your favorite text editor, and type the following code:

```
#CSE 2304 Lab Assignment 1
#Part1
# This is a simple Hello World Program.

.text          # The text section of our program
main:          # the main label should be the first label where our program instructions begin.
la $a0, str    # put the address of the string to display in register $a0
li $v0, 4      # Move 4 to register $v0. This is the system service to display string messages
syscall        # System call to output our string str.

# Now we need to exit our program
li $v0, 10     # Move 10 to register $v0. This is the system service to exit the program.
syscall        # exit (Pass control back to the operating system.)
```

```
# data section of our program.
.data           # .data directive indicates the beginning of the data section of our program.
str: .asciiz "Hello World! \n";      # The '\n' indicates to move the cursor to a new line.
```

2- Save your code as an asm file (e.g: HelloWorld.asm)
3- Open your SPIM simulator. In our lab we use PCSPIM.
4- Load your HelloWorld.asm file. Loading the file is simply opening it.
5- Click on Simulator menu, then click on Go. Or just press F5 to run the program. Choose the starting address of the program from memory, in this case leave the default memory 0x00400000, then click ok.
6- You should see the words "Hello world!" displayed in the console.

Part 2-In-Class: We have seen how to display strings to the console. Now we will learn how to perform basic arithmetic operations and display integer results to the console. In addition, we will learn how to debug a program. The output of this program is: "The last time I checked 4 + 5 was: 9, and 4 – 5 was: -1. Is this still true?"

1- In a text editor, type the following program and save it as an asm file (e.g problem2.asm)

```
#CSE 2304 Lab Assignment 1
#Part 2

.text
main:              # the main label.

la $a0, str1       # put the address of the string to display in register $a0
li $v0, 4          # Move 4 to register $v0. This is the system service to display string messages
syscall            # System call to output our string str.

li $t0, 4          # Move 4 to $t0;
add $t1, $t0, 5    # Perform operation $t1 = $t0 + 5 -> $t1 = 4 + 5
move $a0, $t1      # put the the number to be displayed in register $a0
li $v0, 1          # Move 1 to register $v0. This is the system service to display integers.
syscall            # System call to output our string str.

la $a0, str2       # put the address of the string to display in register $a0
li $v0, 4
syscall            # System call to output our string str.

sub $t1, $t0, 5    # Perform operation $t1 = $t0 - 5 -> $t1 = 4 - 5
move $a0, $t1      # put the the number to be displayed in register $a0
li $v0, 1          # Move 1 to register $v0. This is the system service to display integers.
syscall            # System call to output our string str.


la $a0, str3       # put the address of the string to display in register $a0
```

```
li $v0, 4
syscall            # System call to output our string str.

# Now we need to exit our program
li $v0, 10         # Move 10 to register $v0. This is the system service to exit the program.
syscall            # exit (Pass control back to the operating system.)

# data section of our program.
.data              # .data directive indicates the beginning of the data section of our program.
str1: .asciiz "The last time I checked 4 + 5 was: ";        # First string
str2: .asciiz ", and 4 - 5 was: ";                          # Second string
str3: .asciiz ". Is this still true? \n";                   # Third string
```

2- Load the program to PCSPIM.
3- Now, instead of just running the whole program with F5, you will run the program in single steps. Press F10 (or Simulator->Single Step) to run the next instruction until you reach the end of the program. Note that the changes in registers $v0, $a0, $t0 and $t1 as you step through your program in the registers window (First window on top in PCSPIM)
4- Also you can put a breakpoint to lines of instructions of interest. To do that locate the address of the instruction(s) of interest, press Ctrl+B (or Simulator>Breakpoints), then enter the address(es) of instructions where you want to break.

Part 3 –Work alone: Now let's give you some stuff to start warming. You will write a program that will perform the following operations and display the results to the console. The program will read values of X, Y, and Z as integers from the console and save them in temporary registers. Use system service 5 to read integer inputs from the console. Once you have values for X, Y and Z, stores in registers, perform the operations below and display the results to the console.

<u>20 points each.</u>

1- $X + Y + Z$
2- $X - (Y + Z)$
3- $X + Y \times Z$
4- $X - \frac{Y}{Z}$
5- $(X^3 - Y)/Z$

Deliverables: You are expected to complete **Part 3** alone and submit both a softcopy of your MIPS source code and a one-page report discussing your design and implementation, any difficulties you have encountered to complete this lab, and what you think you have learned. In your MIPS code, write your name, class and section number in the first lines as comments. All of your source code files should be saved either as a .s file or .asm file.

All files should be zipped into one. The zipped file should be named in the format: Firstname_Lastname_Lab#.zip.

Note: You should be ready to demo your program in lab. No late submission will be accepted.

Useful notes.

General Functions	**Store Output String** In the .data section promptx: .asciiz "Enter X"
	Move a value from one register to another move $t1, $t0 #move value to register to save for later use
Input/Output	**Output a String** Use System Service 4 li $v0, 4 #Loads System service to print string la $a0, promptx #Loads string to be outputted syscall
	Output an Integer Use System Service 1 li $v0, 1 #Loads System service to print integer move $a0, $t0 #Move value of $t0 into argument register $a0 syscall
	Read in an Integer Use System Service 5 li $v0, 5 #Loads System service to read integer syscall #Integer inputted is saved in register $v0 move $t0, $v0 #move value to register to save for later use
Arithmetic	**Addition** Ex: Add $t2, $t0, $t1 #$t2 = $t0 + $t1
	Subtraction Ex: Sub $t2, $t0, $t1 #$t2 = $t0 – $t1
	Division Ex: div $t2, $t0, $t1 #$t2 = $t0 divided by $t1
	Multiplication Ex: mul $t2, $t0, $t1 #$t2 = $t0 times $t1