Lab 1: MARS and Your First MIPS Program

Due date: By the end of Thursday, 9/5/2019.

This Lab is worth a maximum of 10 points.

Objectives

In this lab, you will learn how to use MARS, a MIPS simulator, and experiment with arithmetic instructions and system calls. The goals include:

- Using MARS editor/assembler/simulator environment.
- Writing simple MIPS code. You need to understand MIPS assembly language format, assembler directives like .text and .globl, MIPS registers, and some MIPS instructions.
- Learning new functions/features from documents.

Run first MIPS program in Mars

Follow the steps listed below to run the provided example in Mars. If you use your own computer, you can place the files in a directory you prefer. If you use computer in the learning center, **save everything on your P drive**. You may lose your files if they are not on P drive.

1. If you plan to use the computers in the learning center, log in with your NetID and the associated password. If you cannot, see your TA immediately.
2. Download MARS if it is not already installed. MARS is a Java application as a single jar file. Save it in a directory on your P drive. You should be able to run MARS without the admin privilege.

   The download link is http://courses.missouristate.edu/KenVollmar/mars/ or you can google "mars mips simulator".
3. Download the example program lab1.s and place it in a directory, for example, 'lab1'.
4. Start MARS. On Windows, double click the jar file.
5. Open the example program. Use the File/Open menu to open 'lab1.s'. There is a nice built-in editor for editing source code directly in MARS.
6. In order to run the code, you need to assemble the instructions, i.e., convert the instructions to machine code. This can be done by using menu Run/Assemble. You can also use the keyboard shortcut F3. If there are errors in the code, MARS will report the errors and will not generate the binary code. You can go back to the editor to fix

problems. If the program is successfully assembled, you will see a new tab called "Execute".

7. After successfully assembling the instructions, you can run the code by selecting Run/Go, or pressing F5.
8. If you like to run the program again, reset the simulator first (Run/Reset or F12). Then select Run/Go.

Congratulations! You already run a MIPS program twice.

Get familiar with MARS

MARS provides many functions/features to help you debug your program. Knowing the features (set breakpoints, step/backstep, check/change register values, examine memory contents, etc.). will help you a lot when you work on later labs. Let us experiment with a few here.

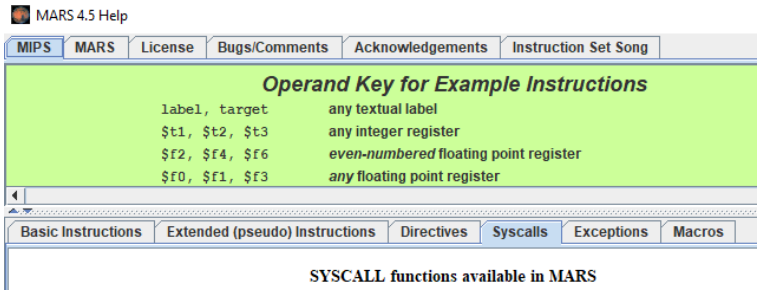
The example code only evaluates a simple expression and then exits. Instead of letting the program run to the end, you can run instructions one by one. After the simulator is reset, select Run/Step. The instruction that is going to be executed next is highlighted. There is a window that shows the values of all registers. You can observer how data in registers are updated. The simulator may show the values in hexadecimal. You can change them to decimal (uncheck Settings/Values displayed in hexadecimal). The register that has been modified is highlighted.

You can also set a breakpoint at an instruction by checking the checkbox on the same line of the instruction. You may need to select Run/Toggle all breakpoints first. The simulator will stop before executing the instructions at the breakpoints. You can examine the values in registers/memory, change the values if you like, and decide what to do next. For example, you can stop the program, step to the next instruction, or run to the end or the next breakpoint.

Now, step through the example program and observe how registers are updated.

Negate an integer

Read and understand the example code. The help section of MARS (the Help menu or F1) provides detailed information about the instructions supported, directives, and syscalls. For example, the MIPS/Syscalls tab lists all the system calls supported in MARS and instructions on how to use them. Read the section about the syscalls (MIPS/Syscalls), as shown in the following figure, and learn how to use the system calls for "print integer" and "read integer".



Now, it is time to do something magic. Add instructions in lab1.s to perform the following tasks.

1. Read an integer x with a syscall.
2. Compute –x, the negation of x.
3. Print –x with a syscall.

After the program starts, it waits for you to enter an integer. Then, it computes and prints the negation of the number.

For example, if you enter 10, the program prints -10. You see the following text in the "Run I/O" tab, next to the "Mars Messages" tab.

```
10
-10
-- program is finished running --
```

Deliverables

Submit revised lab1.s in HuskyCT.