Name:

Laboratory 1
Introduction to MARS and MIPS

CS 10

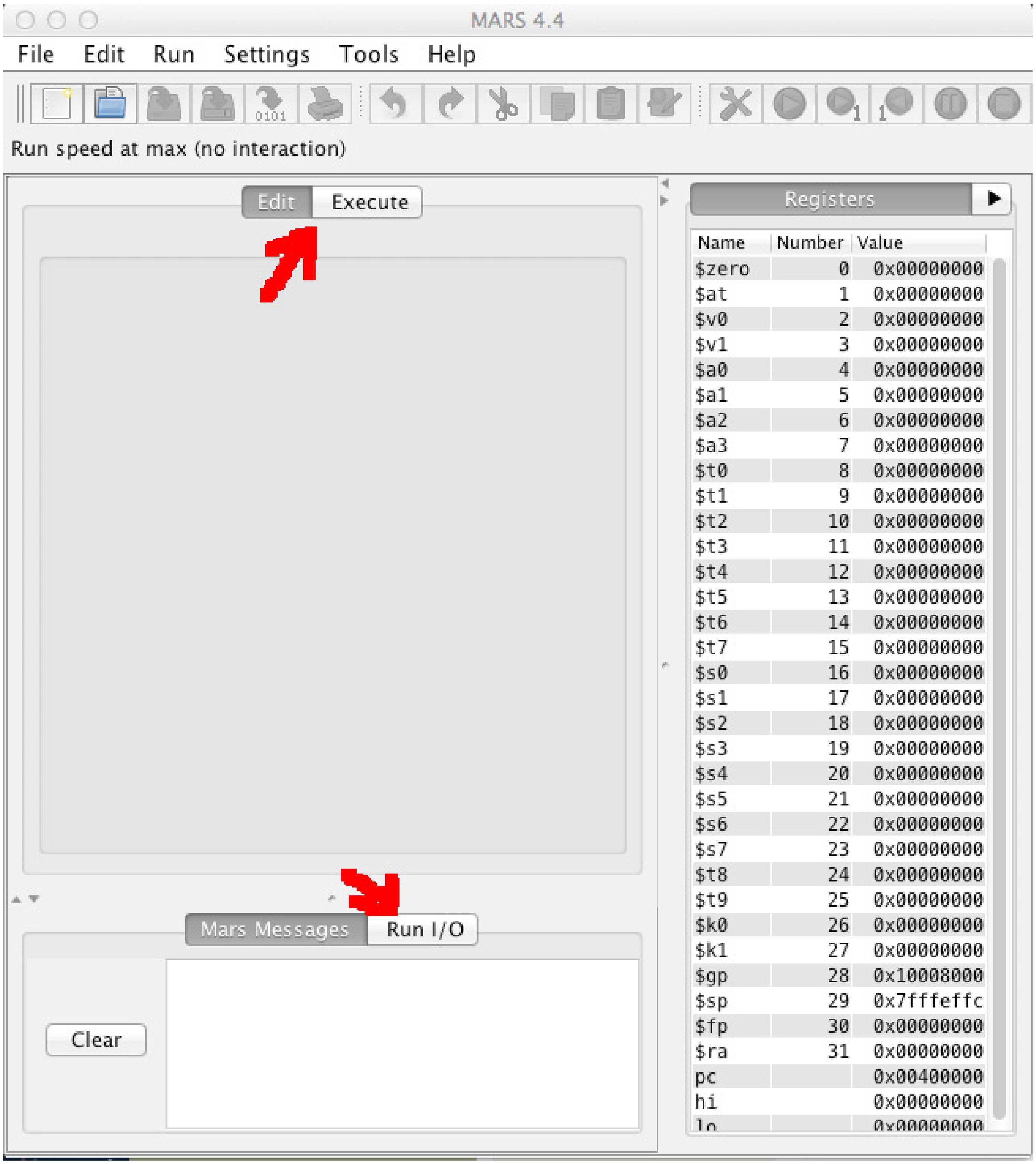
NOTE: shaded sections of the exercises are questions which you must answer as part of your lab report.

In this lab, you will be introduced to the *MARS* programming environment in which you will develop *MIPS* assembly language programs. It can be downloaded from:

http://courses.missouristate.edu/KenVollmar/MARS/

Basic MARS Use

1. Download and launch MARS on your computer. You will see something like this screenshot:



Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Notice the **Edit/Execute** tabs and the **Mars Messages/Run I/O** tabs. These are used to switch between the panels used for those purposes.

Also notice the **Registers** panel along the right-hand side of the screen. All 32 MIPS registers are labeled and numbered, in addition to 3 more listed at the bottom (the **pc** , or program counter, and **hi** and **lo** registers, which you will learn about later).

Which registers have a non-zero value at this point?

Explain what the value of the **pc** indicates:

Exercise 1:

2. In MARS, Under **File** select **New** and **Copy** the **lab1program.doc** text and paste it (using the paste icon in the MARS command bar) into the **Edit** panel.

3. Use **File…Save As** to save as **lab1.asm**, a MIPS program that adds two numbers.

NOTE: *(All icons have menubar equivalents; the remainder of these steps will use the icon whenever possible.)*

4. Assemble the program using the icon (also available from the **Run** menu).

Examine the **Mars Messages** panel, and notice that the message indicates the assembly was successful (hopefully).

Also notice that the tab automatically changes from **Edit** to **Execute**, and that the **Text Segment** and **Data Segment** panels are now displayed (similar to the earlier screenshot).

What does the **0x** notation mean which precedes the 8-digit numbers you see displayed in these panels?

5. The **Text Segment** contains the code from the **.text** section of the program (the program instructions). Explain what you think each column in this panel is used for:

Bkpt:

Address:

Code:

Basic:

Source:

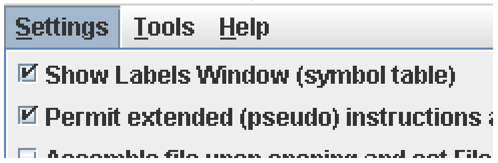
6. What is the starting address of the program?

7. The **Data Segment** contains the code from the **.data** section of the program (the variables and constants defined in the program). What is the starting address of the **Data Segment**?

8. Each row in the **Data Segment** lists the contents of 8 words in memory, each of which contains 32 bits, or 4 bytes, of data. Notice that the first 7 words in the **Data Segment** contain non-zero values. Why are these non-zero for this program?

9. Use the Settings menu to configure the MARS displays. The settings will be retained for the next MARS session.

- The Labels display contains the addresses of the assembly code statements with a label, but the default is to *not* show this display. Select the checkbox from the Settings menu.



☑ Show Labels Window (symbol table)
☑ Permit extended (pseudo) instructions

- Select the checkbox to allow pseudo-instructions (programmer-friendly instruction substitutions and shorthand).
- Select the startup display format of addresses and values to be hexadecimal.

10. Use the slider bar to change the run speed to 1 instruction per second.

This allows us to "watch the action" instead of the assembly program finishing directly.

11. There are a number of ways to execute the program:

- The icon runs the program to completion. Using this icon, you should observe the yellow highlight showing the program's progress in the **Text Segment**, and green highlight showing the registers being modified in the **Registers** panel. When there are changes to the **Data Segment**, they are also highlighted.
- The icon resets the program and simulator to initial values. Memory contents are those specified within the program, and register contents are generally zero.
- The icon is "single-step." Its complement is , "single-step backwards" (undoes each operation).

12. Run the program to completion, using the very slow 1 second per instruction speed. You will need to enter values twice in the **Run I/O** panel, to be used for the addition. Upon completion, the following will be displayed:

```
Enter a value:2
Enter a value:3
The sum is:5
program is finished running –
```

13. Reset and run the program one instruction at a time, using the single-step . Examine the registers after each instruction to verify that you understand any new values.

14. Set a breakpoint at address 0x400030 by clicking on the checkbox at the left of the instruction.

- Reset and run the program again, the program stops at the breakpoint, before executing the instruction.
- Examine the value of **$t3** at this point. What is it? __________
- Perform a single step is to execute the **add** instruction.
- Examine the value of **$t3** again. What is it now? __________
- Click to continue from the breakpoint. Note that you could modify register or memory values directly at a breakpoint before continuing, if it was necessary to do so for testing purposes.

15. Open the Help for information on MIPS instructions, pseudoinstructions, directives, and syscalls.

Exercise 2: Now that you have seen the basic operation of MARS, try and write your own program! Modify the add program so that it prompts you for your name and age, and outputs a message that greets you and tells you how old you will be in 4 years.

```
# to read in a string, do the following.  The string will be stored in memory at location "answer"
        li $v0,8            # system code for read string
        la $a0,answer        #put address of answer string in $a0
        lw $a1,alength     #put length of string in $a1
        syscall

#you also need the following definitions in your .data section for this to work:
answer:  .space    51  #will hold up to 50 characters, so the name must be 50 characters or less
alength: .word     50
```

When you run your program, your console should look like the following:

```
What is your name?  Harry Potter
What is your age?  11
Hello, Harry Potter
You will be 15 years old in four years
```

Format and comment your program appropriately in MARS.

Exercise 3: Write a MIPS program that plays the first dozen or so notes of your new hit single (or if you don't have your own hit single, then your favorite!) Experiment with system calls 31, 32 and 33. Put the notes to your song in a mips "array"