Audrey Zhu
5359476

Lab 05 - B: MIPS Functions

Due February 14, 2022 (Monday) at 5PM

Submission Instructions:

1. You are going to turn in a PDF using Gradescope (please do not change the formatting of the PDF, just add your answers).

2. You may collaborate on this homework with at most one person, an optional "homework buddy". You must turn in your own version of this lab, but you may discuss answers and approaches in detail.

Tasks:

Question 1:

Consider that the machine has the following state of memory for questions (a) and (b) below. Please note that each row corresponds to one byte of memory.

Memory Address	**Memory Contents**
0xBBBB0000	0x44(D in ASCII)
0xBBBB0001	0x4F(O in ASCII)
0xBBBB0002	0x47(G in ASCII)
0xBBBB0003	0x53(S in ASCII)
0xBBBB0004	0x00
0xBBBB0005	0x00
0xBBBB0006	0x00
0xBBBB0007	0x00

a) Consider that register **$t0** holds the address **0xBBBB0000** and **$t1** holds **0x168D09FC**. How will the following memory locations look like after executing: **sw $t1, 4($t0)**. Show in the table using **HEX** values.

Memory Address	**Memory Contents**
0xBBBB0004	0x16
0xBBBB0005	0x8D
0xBBBB0006	0x09
0xBBBB0007	0xFC

b) Assume that the above table is the view of the memory for the following section of code:

```
.data
str: .ascii “DOGS”
```

What will be the contents of **$t1** (in **8-digit HEX**) after the following instruction is executed?

```
.text
main:
        la $t0, str
        lw $t1, 0($t0)
```

Also, add an explanation to your answer.

0x444F4753

la $t0, str stores the value of str (from data) into the register $t0. The “DOGS” is equivalent to 0x44, 0x4F, 0x47, and 0x53 (each hex corresponding to the ascii value of 1 letter). The first 4 bytes of $t0 would have the value 0x444F4753. **lw $t1, 0($t0)** would put that into the register $t1, so the contents of $t1 would also be 0x444F4753.

Question 2:

Give a code snippet to copy a value from register $t0 to $t1 using the **stack** (i.e. do not just use "move" instruction). Assume that $t0 already holds the value which you want to copy. Now, PUSH the contents of $t0 onto the stack and POP the same value from stack into register $t1. Make sure that the state of the stack is the same, before and after the execution of your code (Only a code snippet is needed. No need to print the value).

```
# push content of $t0 into stack
addiu $sp, $sp, -4
sw $t0, 0($sp)

# pop content of stack into $t1
lw $t1, 0($sp)
addiu $sp, $sp, 4
```

Question 3:

Consider the following instructions and answer the question:

(i) jal label - The "jump and link" instruction makes the jump to address <label>, but also does something else. What does jal do with the address of the instruction that follows it in the program?

It stores the address of the next instruction into the register $ra. You use jr $ra to return to that address.

(ii) jr $ra - The “Jump Register” instruction jumps to the address stored in a particular register, in this case to $ra. A jr to $ra is used to return from a function call. Why can’t we simply put a <label> right after the function call and then use the jump instruction j <label> to return from a function?

This would not work if the function is used in multiple places. If it always jumps back to a label, it won’t always go back to where it was called, which makes the function not reusable.