CSCI 4350: Computer Architecture
Project 1: Counting Numbers in an Array
Due: Mar/13 (Mon) 11:59pm (7 points)

1. Overview

In this project, you will implement a simple MIPS assembly program. In this lab, you will learn about how to implement and call functions using the MIPS assembly language, which will be used throughout the semester. To complete this project, you will use the MIPS simulator called Mars 4.5, which is running on JVM. You can download the Mars jar file and the JVM from Canvas. You can find detail information about Mars 4.5 from the link:

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

2. Project Details

In this project, you will count the numbers in the array. You will implement the three functions below.

1. `countNumbers (int input[], int cnt)`: This function will count each number in the array. The first parameter (`input`) is the input array (base) address. The second parameter (`cnt`) indicates the length of the array. You can assume that `cnt` value is between 1 ~ 50, i.e., the minimum array length is 1 and the maximum array length is 50. In addition, You can assume the numbers in the array are between 0 and 49. You will need to implement a loop to count numbers by calling `increaseCnt()` for each number. You must print an error message and terminate the program if `increaseCnt()` returns 0.
2. `increaseCnt (int output[], int num)`: This function will increase the `output[num]` value by 1, i.e., `output[num] += 1`. The first parameter (`output`) is the array address of output and the second parameter (`num`) indicates the index of the array to be increased. This function will check if `num` is in valid range (0 ~ 49). This function returns 0 if the `num` is not in the valid range, else returns 1 after increasing `output[num]` by 1.
3. `printResult (int output[], int cnt)`: This function will print the results, i.e., count for each number in the input array. You must skip the number not shown in the input array. For example, for the `arr[]` = {10, 2, 1, 1, 3, 10 , 49}, this function will print

   Result:

   1: 2

   2: 1

   3: 1

   10: 2

   49: 1

The `main` function will call `countNumbers()` and `printResult()` to count and show numbers in the array.

3. Implementation Details

You will need to use Mars 4.5 system calls to print messages (and numbers for debugging). You can use MIPS example code available in Canvas. You will need to use the MIPS instructions below (but not limited to) to complete this project.

Instruction	Example	Description
`li`	`li $t0, 1`	An integer value is loaded into a register ($t0) with 1
`la`	`la $t0, sym`	An address is loaded into $t0 with the address 'sym'
`lw`	`lw $t0, offset($t1)`	A word is loaded into $t0 from the specified address (offset + $t1)
`sw`	`sw $t0, offset($t1)`	The contents of $t0 is stored at the specified address (offset + $t1)
`add`	`add $t0, $t1, $t2`	Adds $t1 and $t2 and stores the result in $t0
`addi`	`addi $t0, $t1, 1`	Adds $t1 and a sign-extended immediate value (1) and stores the result in $t0
`sll`	`sll $t0, $t1, 4`	Shifts $t1 value left by the shift amount (4) and places the result in $t0. Zeroes are shifted in
`mul`	`mul $t0, $t1, $t2`	Multiply $t1 and $t2 and stores the result in $t0
`jal`	`jal target`	Jumps to the calculated address and stores the return address in $ra ($31)
`j`	`j target`	Jumps to the calculated address
`beq`	`beq $t0, $t1, target`	Branches to target if $t0 and $t1 are equal
`blt`	`blt $t0, $t1, target`	Branches to target if $t0 is less than $t1
`slt`	`slt $t0, $t1, $t2`	If $t1 is less than $t2, $t0 is set to 1, 0 otherwise

For function calls**, you must use 1) a stack to store and load values of registers and 2) a pair of JAL and JR instructions. In addition, the $a0~ $a3 and $v0 registers must be correctly used for parameters and a return value.** You will lose points if you did not implement your program as requested above, even if your program works well. You will implement the program using C (or Java) first to understand how the function works clearly. You can look at any other MIPS code or references from the Internet or books. You can discuss this project with your classmates to get some hints. **But you are not allowed to share the solution with your classmates.**

DO not post and search this project on web sites, e.g., chegg.com and coursehero.com!!! You must complete this project by yourself. I will use a tool to detect cheating.

3. Testcase

A skeleton code (**prj1.asm**) will be provided. The code includes a `main` function and some other code for initialization. The arrays for input and output are hard coded in .data section in the code,

```
input:      .word 0, 0, 0, 0, 0, 0, 0, 0, 0, … 0 (up to 50 numbers)
output:     .word 0, 0, 0, 0, 0, 0, 0, 0, 0, … 0 (up to 50 numbers)
```

You will need to change the `input` array values (and its size) to test your code. **Please write down your name and student ID on C (Java) and MIPS code.**

4. Deliverables

a. Document describing how your assembly code works. Not to exceed 1 page.
b. C (or java) and MIPS code