CSC B58 - Lab 9

System Calls and Arrays

1 Learning Objectives

The purpose of this lab is to learn how to further explore assembly language program, learn about arrays, data segment and system calls. This lab is worth 4% of your final grade.

Preparation Before the Lab

You are required to complete all Parts of the lab by writing and testing your assembly code in MARS and writing your project proposal document. During your allotted lab time, you should demonstrate it to your TA.

2 Introduction

Last week, we built branches and loops in MIPS assembly using labels and branches. This week, we will write some programs with arrays and practice using syscalls. You may also find these techniques useful for your project.

Document your code with comments, and write your name at the top of your submissions! **(PRELAB)**

3 Part I - Input and Output Using Syscalls

Syscalls can be used to get input from and output to the user (see lecture). The following program writes ”Hello!”, reads a number from the user, and outputs the number plus 1.

```
.data
str:    .asciiz   "Hello!\n"

.text
.globl main
main:
        # Print "Hello!" (address of string to print should be in $a0)
        li $v0, 4
        la $a0, str
        syscall

        # Read a number (result will be in $v0)
        li $v0, 5
        syscall

        addi $t0, $v0, 1   # $t0 = $v0 + 1

        # Print result
```

```
        li $v0, 1
        move $a0, $t0       # number to print should be in $a0
        syscall

        # End program
        li $v0, 10
        syscall
```

Write a simple assembly that does the following:

1. Print "Enter 3 numbers: a, b and c". Don't forget to add a newline \n at the end.
2. Read a number from the user (let's call this number a).
3. Read another number (let's call this number b).
4. Read a third number (let's call this number c).
5. Compute the number of of solutions to the equation $ax^2 + bx + c$. See previous lab for how to do that. You can use the same code if you want.
6. Write the following output "There are **N** solutions for ax^2+bx+c" where **N** is replaced by the number of solutions. **Hint:** you will need to combine multiple syscalls to do this! Think about when you should or shouldn't include \n.

Save your code as `lab09a.asm` and submit to Quercus. Be prepared to explain your code. **(PRELAB)**

4 Part II - Arrays

Arrays are declared in the `.data` section of the assembly code. For example, the following code fragment declares two arrays of 7 integers. The first one is initialize to some numbers, and the second one has 7 integers all initialized to 0.

```
.data
A:      .word    5, 8, -3, 4, -7, 2, 33
B:      .word    0:7
```

To access the elements of an array, we need to perform memory access instructions such as `lw` and `sw`. To access an element we must first compute its memory address. The basic formula to compute the memory address of an element is `base + offset`, where `base` is the address of the first element of the array (the label `A`), and `offset` is the index of the element multiplied by the size (in bytes) of each element of the array. Index of arrays is zero-based: it is always between 0 and array length minus 1. For example, the value of A[4] is -7, and the memory address for A[4] is `A + 4*4`. See the lectures for more details.

Write a program code that iterates through A, multiplies each element by a number that depends on its sign, then stores the result in the equivalent element of B. Positive elements of A are multiplied by 10, while negative elements of A are multiplied by 5.
In Python this would be something like: `B[i] = A[i] * (10 if A[i] > 0 else 5)`.

Save your code as `lab09b.asm` and submit to Quercus. Be prepared to explain your code. **(PRELAB)**

5 Part III - Project Preparation

In this part of the lab, you have to complete three tasks as listed below:

1. Animate your avatar to move in one direction
2. Generate an object at a random location on the screen
3. Describe your project proposal

Firstly, you are tasked with animating the avatar you designed in Lab 4 to move across the screen in the left/right/up/down direction. For example, if your avatar is located at the bottom and center location of the Bitmap Display screen (x,y), you should erase and re-draw your avatar at (x,y+4) and then again erase and redraw it at (x,y+8) position and so on until it reaches the topmost y-coordinate location. You must write your program such that the avatar stops moving when it reaches the top of the screen. You can choose to move your avatar in any one of the four directions. If your avatar is taking up more than 25% of the screen in x or y direction, you have to shrink it in size. For example if your avatar is taking up 30% of the x-coordinates (20 pixels out of a maximum 64), then you have to shrink your avatar to take up less than 25%.

For the second task, you should use the random number generator in MARS to generate the position of an obstacle/meteor/enemy unit and make it appear on the Bitmap Display screen. This obstacle can be a small square box of the color of your choice or you can make it something more elaborate. The obstacle should appear at different locations on the screen every time you run the code (since that's how random number generation works). This object should appear once your avatar has stopped moving and must not overlap the avatar's position on the screen. A sample code for using the random number generator is given below:

```
# Random Number Generator
        li $v0, 42          # Service 42, random int range
        li $a0, 0           # Select random generator 0
        li $a1, 27          # Select upper bound of random number
        syscall             # Generate random int (returns in $a0)
```

Save your code as `lab09c.asm` and submit to Quercus. Be prepared to explain your code. **(PRELAB)**

For the last task, you have to describe your project proposal via a document that contains the following:

1. Team or Individual project. If Team: Add a table with a column for each member and an itemized list of what each team member is responsible for. If a feature will be jointly developed, write out what percentage of it will be done by each member.
2. Description of the your game: Player abilities (fire, speed-up etc.), enemy/obstacle abilities, text display, object collisions, planned features for Milestone 3, etc.
3. Proposed methodology: How you plan to implement all the things mentioned in your description above. Example: I will use the random number generator shown earlier to generate obstacles. Then I will use the avatar animation code to make the obstacle/enemy move on the screen. I will detect collisions by using ... I will create multiple enemies/obstacles on the screen at the same time by doing ... I will implement my Feature 1 for Milestone 3 by creating a function that does ...

Save your document as `projectProposal.pdf` and submit to Quercus. Be prepared to explain it to your TA and spend most of your Lab 9 time here. **(PRELAB)**

Write down any feedback or suggestions your TA gives you to help improve your chance of success for the project. **(IN-LAB)**