ECE232 Lab1

MIPS Assembly Language Coding and QtSpim

(Due Thursday, Feb 12, 11:00 PM EDT via Moodle)

1 Objectives

- Understand high-level language statements from the low-level assembly perspective.
- Understand procedure call mechanisms and resizing of the stack during procedure calls.
- Gain familiarity with the structure of a stack frame.
- Gain familiarity with converting a C language to assembly code.
- Gain experience with the QtSpim simulator.

2 The Problem: Sort your name alphabetically

In this lab assignment, you will store your name in computer memory as an array of characters and write a program in MIPS to sort your name alphabetically. For example, if your name is "mike", the output of the program would be "eikm". Note that we would use all lowercase characters for simplicity.

In earlier courses, you learned how to define and use groups of characters stored in an array. For example, in C language an character array is defined to store values as follows.

```
char array[]="abcd";
```

In this assignment you will learn how to represent a character array in memory using assembler representation and use a series of MIPS instructions to perform the following tasks:

- Store your name as a character array in the computer memory
- Use Insertion sort to sort your name alphabetically. For example, if your name is "Mike Adam", you should get a char array as:

  ```
  char name[]="mikeadam";
  ```

  After the insertion sort, the result array will be:

  ```
  char name[]="aadeikmm";
  ```

Note that it's possble to have the same character in the string multple times.

- Debug and simulate your program using QtSpim

For reference, the C code for the assignment is given in Section 5. The code is very similar to Java so even if you don't know all the details of C you should be able to figure out what is happening. Please use *this* C code when performing the hand conversion to assembly code. You will need to get your code working using QtSpim and take several screen shots of its operation. Prof. Tessier will also run your submitted assembly code after you submit it.

3 Implementation Using MIPS Assembly

In this lab, you are asked to do conversion from the high-level language description to the low-level assembly implementation. Please use the template provided in Section 6 for your assembly code. Part of the *main* function is provided; you need to complete the remainder of the *main* function and the *swap* function. Note that the stack pointer is initialized to 0x80000000 at the beginning of the program.

Several tips:

- Refer to the "Resources" page on the ECE 232 course website for information on converting C statements to assembly code and for information on taking screen shots.
- Debug your assembly code using QtSpim. You can set breakpoints in QtSpim! Single stepping is also highly recommended.
- Use the "load address" pseudoinstruction to get the base address of the array as shown below. (You may use registers other than $t0 as well).

```
la $t0, name  # This pseudoinstruction loads the
                  # MIPs address of the label "name" into $t0
```

- Use lbu and sb instructions to load and store characters in the array.

4 Materials for Submission

For this assignment you must turn a written report (in pdf or Word format) and a separate file including your MIPS assembly code (e.g. project1.s). **Each line of code should be commented to indicate what is happening in terms of the algorithm.** Don't simply say "Move s0 into t0", for example. You can use the assembly code template at the end of this assignment to get started.

Your final report should include a one paragraph discussion of any problems that you encountered. Additionally, please include the following screenshots taken after your program is working:

1. The data memory after loading your name into the array.
2. The data memory after sorting your name.

Finally, include answers to the following questions in your report. Please include both the questions and the answers:

- What is the structure of an activation record for a function? Give an example of what is stored in the stack for an activation record.
- How many assembly instructions are there in your program?

5 C Code

```
#define ARRAYSIZE 8 //Define this array size based on your name size
char name[]="mikeadam";

void swap (int index1, int index2)
{
  char swapvalue;
  swapvalue = name[index1];
  name[index1] = name[index2];
  name[index2] = swapvalue;
}

int main()
{
  int i, j;

  for (i=0; i < ARRAYSIZE - 1; i++) {
      j = i;
      while(j >= 0 && name[j] > name[j+1]) {
        swap(j, j+1);
        j--;
    }
  }
  return 0;
}
```

6 Assembly Code Template

```
.globl main
.data
name:
     .ascii "mikeadam"

.text
    lui $sp 0x8000      # initialize the stack pointer
main:
    addiu $sp,$sp,-16   # stack grows by 16 bytes
    sw $ra,12($sp)      # save return address into stack
    sw $s0,8($sp)       # store $s0 so it can be used for i
    sw $s1,4($sp)       # store $s1 so it can be used for j
     ...                     #include any number of instructions you need
     ...                     #perform the first loop
     ...                     #perform the second loop
     ...                     #pass the argument to the swap function
     ...                     #call sub function swap
     ...                     #restore $s0 from stack
     ...                     #restore $s1 from stack
     ...                     #restore return address
     ...                     #shrink stack
     ...                     #return

swap:
      # your own code  II
```