# Project: Recursion in MIPS

## Project: Recursion in MIPS

### Objectives

The purpose of this assignment is to help you gain more practice programming in MIPS, and to help you better understand how functions in higher-level languages are translated into MIPS procedures. A secondary objective is to get you "thinking like a compiler": how does a compiler decide how to translate a high-level program into assembly?

This is a pair programming assignment. Please work with your assigned partner (the same partner as for the last assignment). You only need to hand in one copy of the code, although **both of your names should appear at the top of each file you submit.**

### Your task

As part of last week's project, you wrote a procedure to calculate the length of an array using the "sentinel" model of array storage (i.e., the end of the array is marked by a specific value, in this case -1.) In this week's assignment, you will be writing a MIPS procedure to **recursively** calculate the length of an array.

The recursive procedure should be labeled LENR. The input argument should be the starting address of the array, and the procedure should return the array's length. Your procedure should be able to deal with special cases, such as an empty array. You may assume that -1 will not appear anywhere in the array's data, other than in the last position. The sentinel should **not** be counted as part of the array's values.

The code to call the procedure should load array values into memory using a combination of li and sw. (See the hint from the last project.) The returned length should, at the conclusion of the program, be stored in register $s0. The program should store the starting address of the array in register $s2, and I should still be able to read the starting address out of this register at the end of the program (i.e. don't destroy it!).

Your code should be commented, and the comments should clearly indicate the procedure and the calling code.

Finally, you should turn in a **plain text file** with a 1 paragraph description of your program design. How did you decide to allocate

registers among the procedures and the main code? How much did you utilize the stack, and for what purposes? How efficient would you say your code is, and in what ways, if any, could you improve its efficiency?

## Helpful hints

I highly recommend that you write out the procedure in your favorite high-level programming language first. Doing so will make it easier to see all of the pieces of the program and will ultimately make it easier to write the MIPS versions of the procedure.

For this assignment, an empty array should consist solely of a -1 stored at the starting address.

One of the really great features of the MARS emulator we've been using is that it autocompletes whatever MIPS instruction you type in. However, as you may have noticed, sometimes the autocompletion menu brings up some possibilities we have not covered in class. In this class, we are only studying a subset of the MIPS instruction set. When you write MIPS code in this class, I want you to stick to this subset of the instruction set. So, when in doubt, check the green instruction card in the front of your textbook to see if that instruction you're about to use is "legal" for this class.

## Evaluation and grading checklist

This project is worth 15 points:

- **Code to set up and populate the array, call the procedure, etc: 4 points.** Make sure you've followed the guidelines in terms of register usage, retaining register values, etc, and that you've used the proper MIPS instructions to both call and return from procedures. Make sure your array ends with a -1 in the last slot. Also, make sure that your code terminates and is commented!
- **Recursive procedure: 8 points.** You should first and foremost ensure that your procedure correctly calculates the length of an array, including special cases. You should ensure that you're using the proper registers for input arguments and return values, and that you're only using the input arguments and return values specified in the problem. Your procedure should follow proper conventions with respect to register usage and preservation. The procedure should recurse and terminate correctly. It should be labeled LENR.
- **Writeup: 3 points**. Your writeup should be 1-2 paragraphs in **plain text** and should answer all of the questions posed above, in addition to providing a brief overview to your code.

## Submission instructions

Please put all of your code in a **single** file named length.asm and your writeup in a file named length.txt. Submit both of these on

Moodle. Please make sure that your name and your partner's name appear at the top of your code and your writeup!

## Grading summary

| | |
|---|---|
| Participants | 33 |
| Submitted | 17 |
| Needs grading | 0 |
| Due date | Wednesday, 26 September 2012, 12:30 PM |
| Time remaining | Assignment is due |

View/grade all submissions