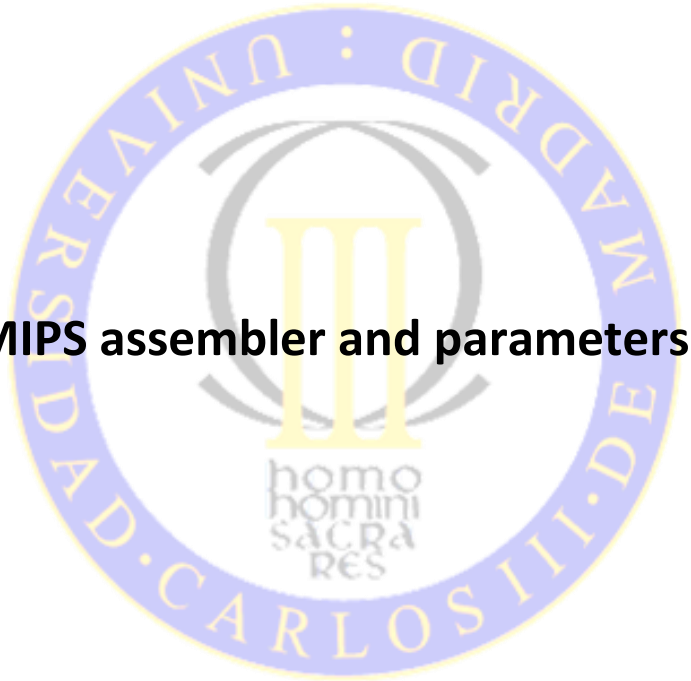
Career: Engineering in Computer Science December 21, 2012
Teacher: Juan Manuel Tirado Martín
Subject: Computer Structure



MIPS assembler and parameters

Adrián María Mora Carreto NIA: 100291775
Ramses Joel Salas Machado NIA: 100292149

Index

Description of the implemented methods

Exercise 2

In this exercise, we have declared a sentence in the .data section where “%d” or “%i” have to be changed by some numbers also declared in this section. We print this string character by character while we evaluate if a “%” is the next character and, depending on which character is read, we have created different paths in order to print integer, float numbers or continue printing the string. In order to compare the characters, we use the ASCII code of each character.

Exercise 3

In this exercise we have declare an array in the .data section, its dimension, an error message and two strings to separate the data when printing. The program loads the array and goes through its main diagonal checking and changing its values from the addition of its neighbors. In order to get the values from the neighbors, the function *neighbors* have 8 cases (the 8 numbers around the chosen element) in which we have to check if there is a number in a specific position (we have developed some if’s in order to check if the position is valid or not.), and add it (obtaining it with the function *get*) into the sum.

Exercise 4

In order to do this practice we have divided the exercises as following: each one made the second exercise and then we both merge then in a better one taking the best things of each version. Adrián implemented the exercise 3, and then Ramses reused this code for the exercise 5. Both of us were involved in the developing of the practice, checking the work of each other in order to verify that the program works and that we understood what we were doing. During this process we were asking each other some doubts using mails, Facebook, Skype and calls. We also have a shared folder in Dropbox in order to share the code that each person has written.

After we check that every code is fully-functional, together we start to write this document and this exercise, where we explain what the proposed programs do and in which way we have done them.

Exercise 5

In this exercise we have declare an array (which represents the board of the game of life) in the .data section, its dimension, an error message and two strings to separate the data when printing. Basically, in this exercise we programed the game of life (see rules Wikipedia - Game of life). With the function *malloc* we allocated new space for calculate each iteration of the game based on the content of the array defined in the .data section. In order to implement an iteration we clean the new array and reuse the code implemented in exercise 3 to check and change the values of this auxiliary array, each time we finish an iteration we copy this content of the new array to the one previously defined with function *copy_array* and we repeat the process until we reach the requested iteration.

Done tests and results

In each exercise we have made at least 3 tests in order to ensure the correct operation of them, changing its values and size of arrays (when an array is presented). The most difficult exercise for us was exercise 3 and 5, in which we have to get and set some elements in the array and check the position of the elements in the array. Also, the Game of Life has changed our perspective of arrays in order to have a better way to deal with them.

Conclusion

This practice has been very useful in order to have a better understanding about MIPS and programs/functions that uses parameters: how to deal with the stack and the registers, “tricks” in order to remember the position of the values and algorithms to check/print/get/set elements in an array (that we have to deal in a 2 dimensional way and not in a 3 dimensional).