Computer Structure Laboratory
Course 2018-2019

Task 3:
Assembler Programming

Tutorial for using MIPS processor simulator and an example of an assembler program

During this tutorial, we will explain the operation of MIPS processor simulator, which can be downloaded from Moodle (material from unit 3). The proposed example will carry out a program that performs the same function as the following C code:

```
int a = 20;
int b = 10;
int c;

int main()
{

    if(a < b)
       c = b;
    else
       c = a;

   while(1);
}
```

Exercise 1. How to call a function and to pass parameters

In this exercise, you must write an assembler program that has a functionality equivalent to the proposed C program and verify its correct operation. To do this, you must understand how to code a function call in assembler program and its parameters passing.

Do the parameters passing, and the return using the stack.

An equivalent code in C for this exercise is attached:

```
int X = 10;
int Y = 4;
int R;

int calculaSumaMult (int a, int b)
{
    return (a+b)*2;
}

int main()
{
        R = calculaSumaMult(X,Y);

        while(1);
}
```

You should consider that the main function of the C program includes multiple operations:

• Reading of X variable from memory
• Reading of Y variable from memory
• Writing in stack of the two read variables
• Function call
• Recovery of the return stored in stack
• Writing the return on the R variable from memory

At the same time, the main instruction of the function *CalculaSumMult* includes the following operations:

• Recovery of parameters by reading from the stack
• Sum of parameters
• Multiplication by 2 of previous sum
• Writing the result in the stack
• Return to the main

Objective

To learn to make calls to functions and to manage passing and returns of parameters. To carry out this exercise, you must configure the memory as **<< Compact with the start address of the text segment in x0 >>.** To do this going to "Settings -> Memory Configuration ..." and select "Compact, Text at Address 0". Repeat these steps for all the exercises in this practice.

Exercise 2. C code compilation

In this exercise, you must write an assembler program that has a functionality equivalent to the proposed C program and verify its correct operation. To do this, you must understand how to code a for loop in assembler and how to perform readings and writes of a vector.

```
int N=10;
int A[N]={2,2,4,6,5,6,7,8,9,10};
int B[N]={-1,-5,4,10,1,-2,5,10,-10,0};
int C[N];

int main()
{
     int i;
     for(i=0; i<N; i++)
          C[i]=A[i]+B[i]*4;
     while(1);
}
```

Objective

To understand the proposed problem, propose and implement a solution to this problem and verify its correct functionality. To carry out this exercise, you must configure the memory as **<< Compact with the start address of the text segment in x0 >>**. To do this going to "Settings -> Memory Configuration ..." and select "Compact, Text at Address 0". Repeat these steps for all the exercises in this practice.

Note: Use the labels A, B, C and N for these four elements in the data memory.

Other exercises

- What changes would be necessary for exercise 1 to carry out passing and return parameters with registers $a0- $a3 and $v0- $v1 respectively?
- What changes would be necessary for exercise 2, both in C and assembler code, to implement the same functionality using a while loop?
- How would you access the third position of a vector whose base address is referenced by the "vector" label?