ASSEMBLER PROGRAMMING IN RASPBIAN

Objectives

- Become familiar with Assembler instructions to manipulate registers at low level programming
- Become familiar with development tools for building executable images from Assembler coding

Part I: Code assembly, compiling and testing

The following activities provide testing ASM codes for you to assemble, compile and run on the Raspberry Pi. Try out each of the sample programs for each activity on your assigned RPi and take screenshots of your outputs. All ASM codes are available on GitHub on the following link: https://github.com/matias-vazquez/SistemasEmbebidos/tree/main/Lab04.

To assemble the program `test.s`:

```
as -o test.o test.s
```

Then, to compile it:

```
gcc -o test test.o
```

And finally, to run it:

```
./test
```

Activities

- Activity 1. Program that returns an Exit code: `first.s`
    - For this case, run the program as follows to see the Exit code in the Terminal:

```
./first | echo $?
```

- Activity 2. Hello, World!: `hello.s`
- Activity 3. Arithmetic with integer variables: `sum1.s`
- Activity 4. Arithmetic with integer variables, version 2: `sum2.s`

- Activity 5. Passing parameters by value: `sum3.s`
- Activity 6. Passing parameters by reference: `sum4.s`
- Activity 7. Using the C-function *scanf()* for User input: `scanfExample.s`
- Activity 8. Pausing the program using the *sleep()/usleep()* C functions: `delayExample.s`
- Activity 9. Recursive Towers of Hanoi: `hanoi.s` and `hanoi2.s`
- Activity 10. Blinking LED: `blink.s`

Part II: Firmware development

1. Write a program that writes the values `0xAAAAAAAA`, `0xBBBBBBBB` and `0xCCCCCCCC` to the memory location assigned to the variable `y` using three different addressing modes (three different instructions). Using the development tools like *gcc* and *gdb*.
    a. Determine the address assigned to the memory location and report this value.
    b. Display the register content and memory location before and after executing each instruction and report these values. Justify the results according to the addressing mode.
2. Write a program to ask the user for two numbers, add the numbers and display the result. The program should prompt as follows:
    ```
    a. Give me the first operand:
    b. Give me the second operand:
    c. The result of # + # is: #
    ```
3. Modify the previous code to request the operation to be performed
    ```
    a. Give me the first operand:
    b. Give me the operation to be performed (+, -, *, /):
    c. Give me the second operand:
    d. The result of # + # is: #
    ```
4. Modify the previous code to include **three** operands
5. Write a program to compute the following equation 6 x 2 + 9x + 2. Read the value of x using *scanf* and print the result using *printf*. Use the `MUL` instruction to compute multiplications found in this program.

Report

Turn in a technical report including the following:

- Links to GitHub of Activities of part I and II
- Links to YouTube with demo videos of your working code
- Screenshots of the Terminal showing the output of your programs
- Conclusions