Lab 1

Warm-up : Python and the target machine : LEIA

Objective

- Start with Python.
- Be familiar with the LEIA [1] instruction set.
- Understand how it executes on the LEIA processor with the help of a simulator.
- Write simple programs, assemble, execute.

Todo in this lab:
- Play and learn Python!
- Play and learn the LEIA ISA.
- Finish at home, nothing will be evaluated in this lab.

1.1 Quick intro to Python - 1h max

This part is strongly inspired by the Project 1 of ENSL (L3).

Please use a correct text editor ! We don't really care if it is SublimeText, Emacs, Atome or Vim, but please use a text editor made for programming.

`https://www.python.org/` Official tutorial: `https://docs.python.org/3/tutorial/` An amazing interactive one `http://www.learnpython.org/en/Welcome`

Inside the interpreter

And now, let's get to the heart of the matter.

EXERCISE #1 ▶ **Launch!**
Launch the Python interpreter (`python3`, in the terminal). Which version is it ? Use a version of Python not older than 3.5. Quit the interpreter with CTRL-D or quit().

EXERCISE #2 ▶
Launch the interpreter in interactive mode and **use it as a calculator** to solve these equations:

$$2+2=x$$

$$11=3k+r$$

where k and r positive or null integers

$$27^{98} \bmod 97 = y$$

EXERCISE #3 ▶ **Strings**
Try the following code:

```
x = 'na'
'Ba' + 2 * x
```

Then write "j'aime les bons bonbons" with the same technique.

[1] LEIA stands for "Literally Everything Is Awful"

Lists

EXERCISE #4 ▶ **Lists**
Create a list `li` of integers containing various éléments. Replace one of the elements with a new value. At last, use + or += to add elements at the end of the list.

EXERCISE #5 ▶ **Sorts**
Sort a list using function `sorted`. What is the complixity in the worst case? In the best case? Use function `len()`; same questions.

Print

EXERCISE #6 ▶ **Formatting**
Give 3 different ways of building the following character string:
`"2.21 Gigawatts !! 2.21 Gigawatts !! My godness !"` using one variable `x = 2.21`, and another variable that uses `str()`, then the operator `%`, then the method `.format()`.

Tiny programs

Now, write your programs in `.py` files (with an editor), starting with:

```
# -*- coding: utf-8 -*-
```

to avoid encoding issues.

EXERCISE #7 ▶ **Hello**
Edit a file named `hello.py` with the following content:

```
# -*- coding: utf-8 -*-
print("Hello World")
```

Save, execute with: `python3 hello.py`.

EXERCISE #8 ▶ **If then else**
Write a program that initializes an int value to a number given by the user (use `input()`) and prints a different message according to its parity (odd/even).

EXERCISE #9 ▶ **While**
Write a program that declares two integer values `a` and `b`, then computes and prints their pgcd.

EXERCISE #10 ▶ **Imperative For**
Using the construction `for i in ...`, write a program that sums all even *i* from 2 to 42 (inclusive).

EXERCISE #11 ▶ **For expression / Lists**

- Write a program that declares and initialises a list, and computes the sum of all its elements.
- Write a 1-line code that, from a list `l`, returns a list whose elements are the squares of the elements in `l`.
- Write a 1-line code that, from a list `l`, returns a liste containing the even elements of `l.l`.

EXERCISE #12 ▶ **Dicts**

1. What are the types of `{}`, `{'a'}`, `{'a', 'b'}` and `{'a': 'b'}`?
2. What is the following code doing (where `t` is a dictionary):

   ```
   while id in t:
           id = t[id]
   print(id)
   ```

   What is the problem?

3. Write a code doing the same operation but without the same drawback (*i.e.*: if needed, it doesn't print anything)

EXERCISE #13 ▶ **Functions**

1. Declare a function `fact` that computes the factorial of a number.
2. What returns `help(fact)`? If it is not done, document your function.

1.2 The LEIA processor, instruction set, simulator

EXERCISE #14 ▶ **Lab preparation**
Clone the github repository for this year's labs:

```
git clone https://github.com/lauregonnord/mif08-labs.git
```

Then:

- Follow the instructions of `leia/README.md` to compile and install the LEIA assembler and simulator. Some more documentation can be found in the LEIA ISA on the course webpage :

  http://laure.gonnord.org/pro/teaching/compilM1.html

- The documentation for the simulator can be found in `leia/simulateur/README.md`.
- The files you need for this lab are in `TP01`.

The assembly language for this year is a toy language called LEIA. We already played a bit with it in the exercise session.

EXERCISE #15 ▶ **Hand assembling, simulation of the hex code**
Assemble by hand the instructions :

```
begin:
2 and r0 r0 0
snif r0 gt 2
jump begin
```

You will need the set of instructions of the LEIA machine and their associated opcode. All the info is in the ISA documentation (and in the simulator Readme file for graphical instructions). Save your (hex) encoding in a file `dummy.obj`, and launch the LEIA simulator on it:

```
$./<path/to/simulateur/>LEIA dummy.obj
```

Carefully follow each step of the execution.

From now on, we are going to write programs using an easier approach. We are going to write instructions using the LEIA assembly.

EXERCISE #16 ▶ **Execution and modification**

1. First test assembling and "terminal simulation step by step with" on the file `tp1-simple.s`
   ```
   $python3 <path/to/leia/assembleur/>asm.py tp1-simple.s
   assembling tp1-simple.s
   $</path/to/leia/simulateur/>LEIA -s tp1-simple.obj
   ```

Listing 1.1: tp1-simple.s

```
        .set  r2 data
        rmem r1 [r2]
        jump 0
data:
        .word   7
```

2. Guess the purpose of the following files: `tp1-3a.s` et `tp1-3b.s`. Check with the simulator. What is the difference between the primitives putchar and printstr, that are provided by the operating system?

Listing 1.2: tp1-3a.s

```
    call  clearscr
    .let  r4  1
    .let  r0  0x0000
    .let  r1  10
    .let  r2  95
    .set  r3 HELLO
    call  putstr        ; putstr code is in lib.s
    refresh
    .let  r1  10
    .let  r2  85
    .set  r3 COURSE
    call  putstr        ; putstr code is in lib.s
    refresh

    jump 0

HELLO:
    .string  "Hello"
COURSE:
    .string  "CAP ENSL 2017-18!"

#include lib.s
```

Listing 1.3: tp1-3b.s

```
    ;;  graphical "reserved" registers: r1,2,3,4
    ;;  r12 for the star
    call  clearscr
    .let  r4  1
    .let  r0  0x0000
    .let  r2  95
    .let  r1  10
    .set  r10 star          ; takes the @ - not affected
    .set  r14 N
    rmem r6 [r14]           ; loop counter init=N
loopi:
    rmem r3 [r10]
    copy r13 r6
    copy r12 r2
    copy r11 r1
    call  putchar           ; store the context before call
    refresh
    copy r1 r11
    copy r2 r12
    copy r6 r13
```

```
        add r1 r1 15
22      sub r6 r6 1
        snif r6 eq 0
        jump loopi
        jump 0

27  star:
        .word 42            ; ascii for '*'
    N:
        .word 4

32
    #include lib.s
```

EXERCISE #17 ▶ **Algo in** LEIA **assembly**
Write a program in LEIA assembly that computes the min and max of two integers, and stores the result in a precise location of the memory that has the label `min`. Try with different values.

EXERCISE #18 ▶ **Algo in** LEIA **assembly - Bonus**
Write and execute the following programs in assembly :

- Count the number of non-nul bits of a given integer.
- Draw squares and triangles of stars (character '*') of size n, n being stored somewhere in memory.