Reading material

- Browse the chapter A first look at Assembly language from the book "Dandamudi, Sivarama P.: Guide to Assembly Language Programming in Linux"
- The IA-32 instruction set.
- Extra Reading: Assembly tutorial.
- Compiling assembly files using make file - reminder

Introduction to Assembly language

This lab may be done either solo or in pairs.

For this exercise, download and unzip ***lab4.zip***

Objectives

- Understand assembly code.
- Write a simple program in assembly.

Task 0: *Assembly Basics*

The folder `task0` contains a C program that receives two characters, determines which one is bigger in alphabetic terms, and prints the result to the standard output.

For Example:

```
$> task0 f q
The 2nd argument is bigger
```

Build your program using the provided *makefile* and make sure that you get the same result as in the example.

The file `my_cmp.asm` contains an assembly-based program where parts of the `my_cmp` function is missing.

1. Read the assembly source code and detect the missing code snippet.
2. Complete the missing part. Consult the assembly code from Class 3 for a conditional and unconditional jump.
3. Create a makefile to build your program.
4. Run your program with the assembly implementation and make sure that it works and outputs the right results.

Task 1: Reverse Engineering

1. Read `funcA.asm` and answer the following questions (write down your answers in the file *answers.txt* in the *task1* folder):
   1.1. How many parameters does this function have?
   1.2. What does this function return (what does the function do)?
   1.3. How many local variables are used by this function?
   1.4. Is there a function in the C standard library which is the counterpart of this function? If so, which one?
2. Write a simple C program that uses this function. Create a a makefile that compiles the entire program (including the assembly part).

Task 2: Calling Assembly Code from C

The file `scmp.c` contains a C program that makes a lexicographical compare between two strings and prints the difference of the ascii value of the first different character between them (or 0 if the strings are the same). This program does not work since the function `cmpstr` used by it is not implemented.

Examples:

```
$> ./scmp2 ab ab
"ab" > "aa"
The difference is 1

$> ./scmp2 ab aa
"ab" == "ab"
The difference is 0

$> ./scmp2 ab aba
"ab" < "aba"
The difference is -97
```

1. Write a C implementation for the missing function in a file called *cmpstr.c* (without using any C library functions).
2. Extend the *makefile* to build *cmpstr1.o* from *cmpstr.c* and *scmp2* from *scmp.o* and *cmpstr2.o*.
3. Implement the same function in Assembly language in a file called *cmpstr.s* (both s and asm are considered standard assembly file extensions). Consult the assembly code from task 1 for the file structure.
4. Extend the *makefile* to build *cmpstr2.o* from *cmpstr.s* and *scmp2* from *scmp.o* and *cmpstr2.o*.

Task 3: A Pure Assembly Program

In this task, you are expected to write a **pure** Assembly program that counts the number of times a certain character appears within a given string.

- Use the following code at the beginning of the file and use STR and X within your program for the string and the character respectively.

```
%define X 'A'
%define STR string1

section .rodata
    print_format db '"%c" appears in "%s" %d times', 10, 0
    string1 db 'ABBA', 0
    string2 db 'BBA', 0
    string3 db 'BB', 0
    string4 db '', 0
```

- Programmatically count the number of times X appears in the string.
- Prints the string and the number of times X appears in it using **printf**.
- Test different characters and different string.

Example:

```
$> ./task3
"A" appears in "ABBA" 2 times
```

Notes:

- See example of how to use **printf** from Assembly code in the file *printf.s*.
- After creating the object file ***task3.o*** with **nasm**, use the following line to run the linker and create an executable:
  `ld -melf_i386 -o task3 task3.o -lc -I/lib/ld-linux.so.2`

Submission

- Submit a zip file of your work at the end of the lab session in Moodle.

Good luck!