Assignment 5

Objectives:

- x86-64 function calls and stack
- Investigating the size of some x86-64 assembly instructions

---

Submission:

- Submit your document called **Assignment_5.pdf** om CourSys.
  - Add your full name and student number at the top of the first page of your document.
- **If you write your answers by hand (as opposed to using a computer application to write them),** when putting your assignment document together, do not take photos of your assignment sheets! Scan them instead! Better quality -> easier to read -> easier to mark! ☺

Due:

- Friday Oct. 29 at 4pm on CourSys
- Late assignments will receive a grade of 0, but they will be marked (if they are submitted before the solutions are posted on Monday) in order to provide feedback to the student.

Marking scheme:

- All questions of this assignment will be marked for correctness.
- The amount of marks for each question is indicated as part of the question.
- A solution will be posted on Monday after the due date.

---

1. [10 marks] x86-64 function calls and stack

   a. Hand trace the code from our Lab 4 (`main.c` ,`main.s`, `p1.c`, `p1.s`, `p2.c` and `p2.s`) using the test case, i.e., `x = 6`, `y = 9`, `buf[40]`.

- As you do so, draw the corresponding Stack Diagram for the entire program, i.e., until you reach (but have not yet executed) the `ret` instruction of the main function. To do so, you can either print the "Stack Diagram" sheet at the end of this assignment and do the drawing by hand, then scan the result and include it into this assignment document OR do the drawing by electronically annotating the "Stack Diagram" sheet at the end of this assignment then include the result into this assignment document.
- The use of the Register Table is optional: use it only if you find it useful. You do not have to include it as part of your assignment document.
- Indicate the movement of %rsp by crossing its old location and rewriting "%rsp" to indicate its new location (as we have done in our lectures and Lab 4).
- Cross the content of the stack that has been popped.
- When the value of a stack location is changed, cross its old value and write the new value in the same stack location.
- Make sure you include the content of `buf` in your Stack Diagram.
- When drawing your stack diagram, you do not have to show the effect on the stack of the 5 `call` instructions at lines 33, 43, 45 and 57 in `main.s` and at line 47 in `p1.s`. These are calls to `printf(…)`, `puts(…)` and `sprintf(…)`. In other words, you do not have to add the return addresses associated to these 5 calls onto the stack.
- Hint: The solution to Participation Activity 6 (which will be posted after its deadeline on Monday) will give us a great head start!

b. Modify `main.c` by reducing the size of `buf[]` from 40 to 24.

- Remake the code and hand trace it using the following test case, i.e., `x = 6`, `y = 9`, `buf[24]`.
- Repeat the instructions found in the section a. above and create a second drawing of the stack using the Stack Diagram sheet found at the end of this assignment.
- Make sure you include the content of `buf` in your Stack Diagram drawing.
- On the same sheet as your Stack Diagram, asnwer the question:What happens to the "canary value" in this situation?

2. [10 marks] Investigating the size of some x86-64 assembly instructions

Complete the following three tables:

X86-64 Instructions	**Their size (in bytes)**
xorq %rax, %rax	
xorl %eax, %eax	
movq $0, %rax	
movl $0, %eax	

X86-64 Instructions	**Their size (in bytes)**
addl $1, %eax	
leal 1(%eax), %eax	
incl %eax	

X86-64 Instructions	**Their size (in bytes)**
addl $8, %eax	
leal 8(%eax), %eax	

To complete the above tables, you must:

- Write an assembly program and include all the above x86-64 instructions in it. Of course, this program is going to be rather nonsensical. Not a problem!
- Compile your program.
- Use the **`objdump`** tool on the resulting object file you obtained from the compilation process.
- Looking at the result will lead you to the answer, i.e., the size of each of these x86-64 instructions in bytes.
- In each table, **bold** the instruction that is the most space efficient.

- Copy the content of your program (*.s) into this assignment (into this Assignment_5.pdf document).
- Copy the result you obtained from the **`objdump`** tool into this assignment (into this Assignment_5.pdf document).

Base + Displacement	**Stack**	**Purpose**

Register Table :
