Computer Architectures **02LSEOV 02LSEOQ [AA-LZ]**	Delivery date: Thursday 13/12
Laboratory **8**	Expected delivery of lab_08.zip must include: - zipped project folder of the exercise 1 - this document compiled possibly in pdf format.

Solve the following problem by starting from the *template.s* file.

Exercise 1) Experiment the SVC instruction.

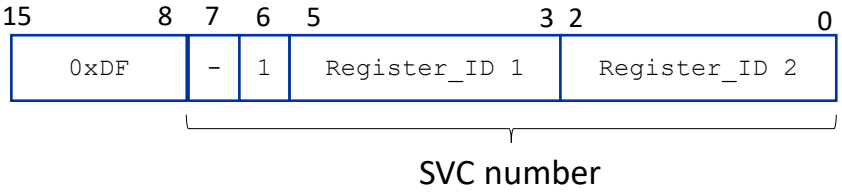
Write, compile and execute a code that invokes an `SVC` instruction when running user routine with unprivileged access level.

In the handler of `SVC` implement the following functionalities according to the `SVC` number:

- 0 to 12: reset the content of register R?, where ? can assume values from 0 to 12
- 13 to 63: nop
- >= 64: the `SVC` call have to implement a module operation according to the `SVC` number.

The ARM instruction set does not include any instruction for module of division computation. In computing, the *modulo* operation finds the remainder after division of one number by another (sometimes called modulus). Given two positive numbers, *a* (the dividend) and *n* (the divisor), *a* module *n* (abbreviated as *a* `MOD` *n*, or *a* % *n*) is the remainder of the Euclidean division of *a* by *n*. (e.g., `13 MOD 6 = 1`).

We want to allow the execution an instruction `MOD` that takes a register as dividend, a register as divisor and it saves the results of the module in R0. The encoding of the personalized instruction is the following:



More in details:

- Bit 6 indicates the SVC functionality as follows:
  - SVC[6] = 0 AND SVC[5] = 0: reset the registers according to the value in SVC[4:0]
  - SVC[6] = 0 AND SVC[5] = 1: NOP operation
  - SVC[6] = 1: Module operation as described before
- Bit 7 is - (where - stands for don't care, meaning either `0` or `1`)
- Field `Register_ID 1` is the index of the register to be used as divisor
- Field `Register_ID 2` is the index of the register to be used as dividend

Q1: In the presented scenario, what is the maximum value that the `SVC` number can assume?

Logically, the maximum value that the SVC number can assume is 127 (module operation between register R7 and itself); however, since the eighth bit in the SVC number is a don't care value, the SVC value can reach (if we consider an unsigned number representation) the value of 255 (or go negative, if we consider a signed representation).

Q2: The list of registers that can be used is limited: which are the possible indexes of registers used as operands of the `MOD` operation?

Since the register fields in the module operation are three bits long, it is possible to address registers R0-R7 (0 to 2^3-1).

Q3: Which strategy can be used to overcome the limitation in terms of usable registers?

To overcome this limitation, it may be possible to use a register for passing the indices of the registers to be used for the module operation, instead of codifying this information in the SVC value (four bits per register are necessary, since the Cortex M3 features 16 general-purpose registers). Another option is to drop the usage of the Thumb instruction set and use the ARM SVC instruction, for which the SVC value is 24-bits long instead of 8.

Q4: What need to be changed in the `SVC` handler if the access level is privileged? Please, report code chunk that solves the issues related to this request.

If the access level is privileged, there is no need for storing the address of the unprivileged stack pointer in a register for the following usage; all the operations can be executed on the privileged stack only. In order to minimize the impact on the code, the MSR instruction only can be modified, by substituting it with an ADD operation saving in R1 the value of the MSP summed to 56 (14 registers saved on the stack, each one 4 bytes long).

```
ADD     R1, SP, #56             ; 14 registers saved on the stack
```

With this single modification, no other instruction needs to be updated to work properly.