Computer Architectures **02LSEOV 02LSEOQ [M-Z]**	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.

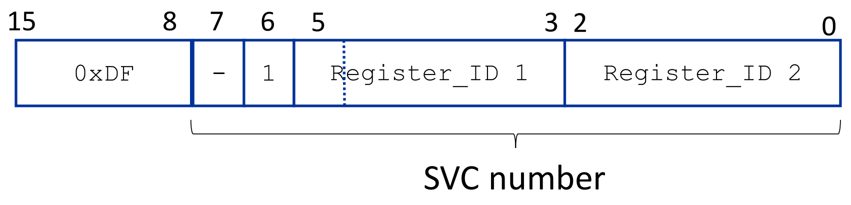
Write, compile and execute a code that invokes a `SVC` instruction when running user routine with unprivileged access level.

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

- 0 to 12: reset the content of registers 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 7 is `-` (where `-` stands for don't care, meaning either `0` or `1`)
- Bit 6 and 5 indicate the SVC functionality as follows:
  - SVC[6] = 0 AND SVC[5] = 0: reset the registers according to the value in SVC[3:0]
  - SVC[6] = 0 AND SVC[5] = 1: NOP operation
  - SVC[6] = 1 AND SVC[5] = X: Module operation as described before
- 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**

Example: the following `SVC` invokes the module computation of `R0` module `R1`

```
SVC     0x41   ; binary value of the SVC number 2_01000001
```

Q1: In the presented scenario, what is the maximum value that the `SVC` number can assume?
255

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?
0-7

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

We could use one bit more for each register index in order to address up to 16 registers.
We could also use the register indexes to address two other registers that contains the indexes of the real registers to be used.

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.

```
SVC_HANDLER         PROC
                       EXPORT  SVC_Handler           [WEAK]
                       MOV R1,SP
                    LDR R0, [R1,#24]
                    LDR R0, [R0,#-4]
                    BIC R0, #0xFF000000
                    LSR R0, #16
                       PUSH {R0-R12,LR}
                       …

                       POP {R0-R12, LR}
                       BX LR
                        ENDP
```