Architetture dei sistemi di elaborazione **02GOLOV** **Laboratory** **8**	Delivery date: Thursday 9/12 Expected delivery of lab_08.zip must include: - the source code (startup.s) for exercise 1, for exercise 2startup.s and main.c; - this document compiled possibly in pdf format.

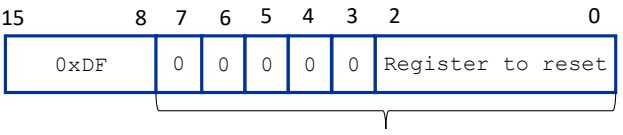
Solve the following problem by starting from the *startup.s* file in the ASM_Template project.

Exercise 1) Experiment the SVC instruction.

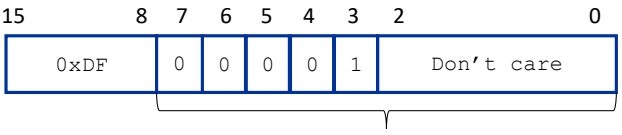
Write a testbench (i.e., a piece of code and data to fully test instruction functionalities) intended to test the following functionalities of a `SVC` instruction handler.. Through this instruction it is requested to implement a RESET, a NOP and a MEMCMP functions. The MEMCMP function is used to compare two memory regions and it returns information about the execution. Assume that the SVC is called from a user routine with unprivileged access level.

In the handler of `SVC`, the following functionalities are implemented according to the `SVC` number:

1. 0 to 7: RESET the content of register R?, where ? can assume values from 0 to 7 and it is the value specified in the SVC number.
2. 8 to 15 and >=128: NOP.
3. 64 to 127: the `SVC` call have to implement a MEMCMP operation, with the following input parameters and return values:
    - the 6 least significant bits of the SVC number indicates the number of bytes to be compared.
    - the initial addresses of the two areas to compare are 32-bit values passed through R0 and R1
    - by again using R0, it returns:
        - 0 if all the bytes in the two areas are the same.
        - 1if the first not equal byte in the first area is greater than the second (in C language, `*ptr1+k>*ptr2+k`).
        - -1 in the other case (`*ptr1+k< *ptr2+k`).

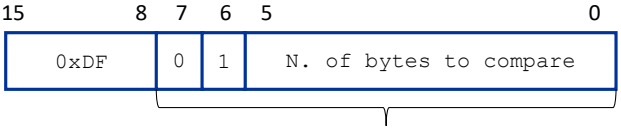


SVC number for Register RESET instruction (0 to 7)



SVC number for NOP (8 to 15)

SVC number for NOP (>= 128)



SVC number for MEMCMP (>= 64 and <=127)

NOTE: in your testbench, you should provide the most appropriate inputs values (SVC numbers, values in R0 and R1) to check that your code matches the requested behaviour. Be also aware that the SVC instruction must be called transparently to your code according to the ARM ABI.

Example: the following `SVC` invokes MEMCMP on two memory areas

```
LDR  R0, =StartAddressA
LDR  R1, =StartAddressB
SVC  0x48  ; 2_01001000 binary value of the SVC number
```

Q1: Describe how the stack structure is used by your project.

E' possibile eseguire la SVCall sia da un caller Privileged che da un caller Unprivileged.
Nel primo caso si usa l'MSP sia per lo stacking che per il push e la pop nel SVC_Handler. In questo caso si accede ai registri R0-R3 all'indirizzo MSP+14*4, MSP+15*4 … (pushati durante lo stacking), mentre si accede ai registri R4-R7 all'indirizzo MSP+4*4, MSP+5*4… (pushati durante il push iniziale nell'handler.
Nel caso di caller Unprivileged, i registri R0-R3 sono invece all'indirizzo PSP, PSP+1*4, …, mentre i registri R4-R7 si trovano nella stessa posizione di prima (MSP+4*4, MSP+5*4).

Q2: What need to be changed in the `SVC` handler if the access level of the caller is privileged? In case report code chunk that solves this request (if any).

Il codice gestisce entrambi i casi, accedendo allo stack corretto e aggiungendo un offset se necessario.

Q3: Is the encoding of the SVC numbers complete? Please comment.

Siccome il comportamento con 15 < SVC number < 64 non è specificato, ho gestito quel caso facendo ritornare l' SVG_Handler senza fare alcuna operazione. Inoltre per evitare problemi nel caso il numero di Byte da confrontare fosse più grande delle aree di memoria allocate per i dati, ho allocato aree da 64 Byte.

Exercise 2) Integrate ASM and C language functionalities

The following function, written in ASSEMBLY language, is invoked from a main C language function:

```
unsigned int variance(unsigned int* V, unsigned int n);
/* where n is the number of V elements */
```

The function returns alternatively:

- the integer truncation of the variance σ^2 of the values stored in V, according to the formula: $\sigma^2 = \frac{\sum_{i=0}^{n-1}(V_i-\mu)^2}{n}$, where μ is the integer mean of the values in V
- the value 0xFFFF if any significant error (identify the most critical ones) is encountered in the computation.

The main C language function takes care of declaring an unsigned integer vector called V composed of N elements (**N chosen by you**). At declaration time, the vector is statically filled by random values (**chosen by you)**.

Please fill the table below. For exercise 1 report the information considering the testbench you have developed. For exercise 2 replace **x** with the value chosen for N assume that the program completes without any error (thus, chose the most appropriate input values to return the variance).

F_{clk} = 12MHz	Execution time (clock cycles)	Code size	Data size
Exercise 1)	2964	368	252
Exercise 2) with N=6	666	444	324