Computer Architectures 02LSEOV 02LSEOQ [AA-LZ]	Delivery date: Wednesday 12/12
Laboratory **7**	Expected delivery of lab_07.zip must include: - zipped project folder of the exercise 1 - this document compiled possibly in pdf format.

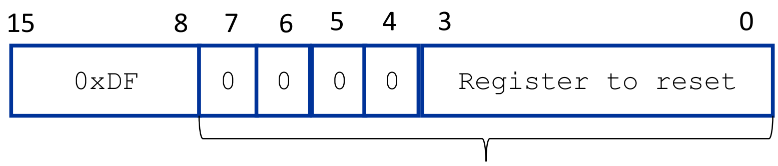
Solve the following problem by starting from the *template.s* file.

Exercise 1) Experiment the SVC instruction.

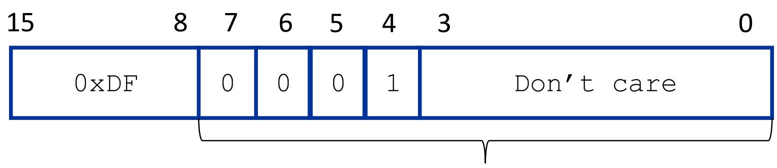
Write, compile and execute a code that invokes a `SVC` instruction when running a **user routine** with **unprivileged access level**. By means of invoking a SuperVisor Call, we want to implement a RESET, a NOP and a MEMCPY functions. The MEMCPY function is used to copy a block of data from a source address to a destination address and return information about the data transfer execution.

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

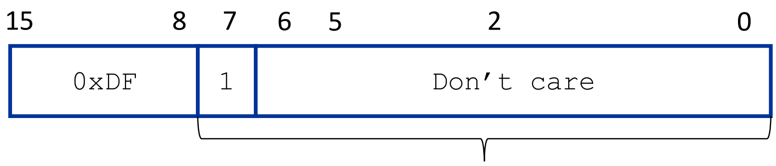
1. 0 to 7: RESET the content of register R**?**, where **?** can assume values from 0 to 7
2. 8 to 15 and >=128: NOP
3. 64 to 127: the `SVC` call have to implement a MEMCPY operation, with the following input parameters and return values:
   - the 6 least significant bits of the SVC number indicates the number of bytes to move
   - source and destination start addresses of the areas to copy are 32 bits values passed through stack
   - by again using the stack, it returns the number of transferred bytes



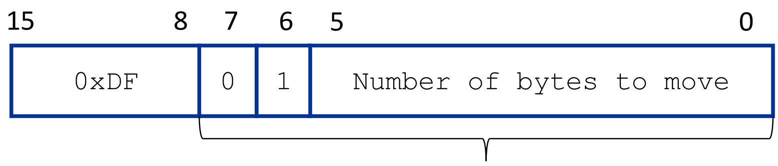
SVC number for Register RESET instruction (0 to 7)



SVC number for NOP (8 and 15)



SVC number for NOP (>= 128)



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

Example: the following `SVC` invokes MEMCPY from a given source to a destination

```
LDR   R0, SourceStartAddress
LDR   R1, DestinationStartAddress
PUSH  R0
PUSH  R1
SVC   0x48   ; 2_01001000 binary value of the SVC number
POP   R0
```

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

Lo stack è diviso in due parti, una usata come Main Stack e l'altra come Proces Stack.
A livello utente si ha un'esecuzione thread level, senza privilegi e quindi usando uno stack pointer di tipo PSP.
Quando si entra nella SVC l'esecuzione passa ad handler level e con privilege, usando uno stack pointer di tipo MSP. Questo implica che vengono usati due strutture stack diverse per l'utente e per l'SVC handler.

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

Per passare ad un access level privilegiato del chiamante è necessario modificare il terzo bit del Control Register. Questo permette al chiamante di usare il Main Stack.
Ciò implica che, a livello del SVC Handler, si accede al Main Stack (quindi con uno stack pointer di tipo MSP) per ricavare i parametri passata via stack dal chiamante e modificare i registri R0-R3.
Per i parametri le uniche modifiche da effettuare sono I valori di offset usati per leggere dentro lo stack.
Per quanto riguarda la modifica dei registri basta sostituire la riga MRS R9,PSP con MRS R9,MSP e modificare I valori di offset utilizzati per accedere ai registri.
Il codice che implementa questa situazione è il seguente:

```
SVC_Handler     PROC
                EXPORT  SVC_Handler                 [WEAK]

                STMFD SP!, {R0-R12, LR}
                TST LR, #0
                MRSEQ   R9, MSP
                MRSNE   R9, PSP
                LDR R8, [R9, #80]
                LDR R8, [R8,#-4]
                BIC R8, #0xFF000000
                LSR R8, #16

                CMP R8,#8
                BLT SVC_Reset ;se minore di 8 passo al reset dei registri

                CMP R8,#15
                BLE SVC_Nop ; se maggiore di 8 e minore di 15 salto a NOP
                CMP R8,#128
                BGE SVC_Nop ; se maggiore o uguale a 128 salto a NOP
                CMP R8,#64 ; se compreso tra 16 e 64 esco
                BLT SVC_End

                LDR R10,[R9,#96] ;R10 contiene SourceStartAddress
                LDR R11,[R9,#100] ;R11 contiene DestStartAddress
                MOV R12,R8
                LSR R12,#2
                MOV R2,R12
SVC_MEMCPY      LDRB R0,[R10]
                STRB R0,[R11]
                ADD R10,R10,#1
                ADD R11,R11,#1
                ADDS R12,R12,#-1
                BNE SVC_MEMCPY
                SUB R12,R2,R12
                STR R12,[R9,#88]

                B SVC_End
```

```
SVC_Reset

                ;R0-R3 -> accedo a Stack Utente PSP
                ;MRS R9,PSP

                CMP R8,#0

                MOVEQ R0,#0
                STREQ R0,[R9,#56]
                LDMFDEQ SP!, {R0-R12, LR}
                BXEQ LR

                CMP R8,#1

                MOVEQ R1,#0
                STREQ R1,[R9, #60]
                LDMFDEQ SP!, {R0-R12, LR}
                BXEQ LR

                CMP R8,#2

                MOVEQ R2,#0
                STREQ R2,[R9, #64]
                LDMFDEQ SP!, {R0-R12, LR}
                BXEQ LR

                CMP R8,#3

                MOVEQ R3,#0
                STREQ R3,[R9, #68]
                LDMFDEQ SP!, {R0-R12, LR}
                BXEQ LR

                ;R4-R7 -> Accedo direttamente ai registri DOPO aver ripristinato i loro valori iniziali
                CMP R8,#4
                LDMFDEQ SP!, {R0-R12, LR}
                MOVEQ R4,#0
                BXEQ LR
```

```
                CMP R8,#5
                LDMFDEQ SP!, {R0-R12, LR}
                MOVEQ R5,#0
                BXEQ LR

                CMP R8,#6
                LDMFDEQ SP!, {R0-R12, LR}
                MOVEQ R6,#0
                BXEQ LR

                CMP R8,#7
                LDMFDEQ SP!, {R0-R12, LR}
                MOVEQ R7,#0
                BX LR

SVC_Nop         NOP
SVC_End         LDMFD SP!, {R0-R12, LR}
                BX LR
```

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

```
No, mancano i valori compresi tra 16 e 63.
```

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

The function returns alternatively:

- the integer average value of the values stored in V, or
- value 0 if any significant error is encountered in the accumulation of the values.

The main C language function takes care of declare an unsigned integer vector called V and composed of N elements. At declaration time, the vector is statically filled by random values.

Please fill the table below.

F = 12MHz	Execution time (clock cycles)	Code size	Data size
Exercise 1)	297.96	564	RO:204; RW:64; ZI:512
Exercise 2)	654.96	504	RO:264; ZI:608