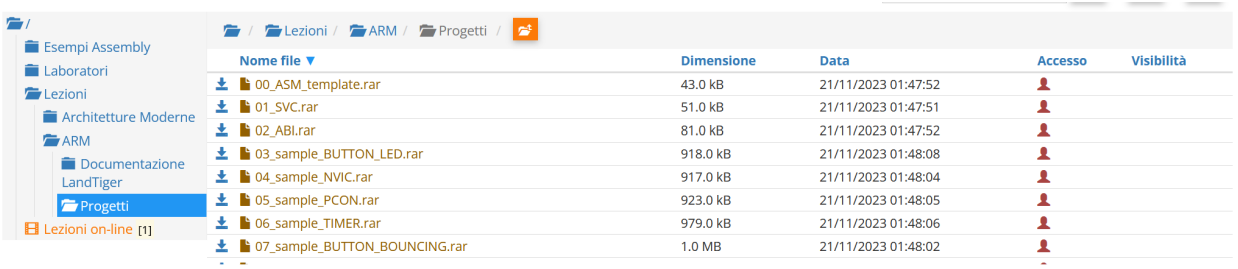
Architetture dei Sistemi di Elaborazione	Delivery date: 30th November 2023
Laboratory 6	Expected delivery of lab_06.zip must include: - Solutions of the exercises 1, 2 and 3 - this document compiled possibly in pdf format.

Starting from the ASM_template project (available on Portale della Didattica), solve the following exercises.



1) Write a program using the ARM assembly that performs the following operations:
   a. Initialize registers R1, R3 and R4 to random signed values
   b. Sum R1 to R3 (R1+R3) and store the result in R2
   c. Subtract R4 to R2 (R4-R2) and store the result in R5
   d. Force, using the debug register window, a set of specific values to be used in the program to provoke the following flag to be updated **once at a time** (whenever possible) to 1:
      - carry
      - overflow
      - negative
      - zero
   e. Report the selected values in the table below.

	Please, report the hexadecimal representation of the values			
Updated flag	R1 + R3		R4 – R2	
	R1	R3	R4	R2
Carry = 1	0x00000001	0x00000001	0x00000005	0x00000002
Carry = 0	0x00000001	0x00000001	0x00000001	0x00000002
Overflow	0x00000001	0x00000003	0x80000001	0x00000004
Negative	0x0000000A	0x00000001	0x00000008	0x0000000B
Zero	0x00000001	0x00000001	0x00000002	0x00000002

Please explain the cases when it is **not** possible to force a **single** FLAG condition:
It's not possible to force a single Zero flag as whenever it's set, it's also set the Carry flag. Same thing goes for the V flag: it can't be obtained without also setting the C flag(if we're adding two negative numbers) or N flag(if we're adding two positive numbers).
Finally, C=0 cant be set as in the subtraction also the N flag is set.
***Take a look at the end of the pdf to see how the flags were set.**

2) Write two versions of a program that performs the following operations:
   a. Initialize registers R2 and R3 to random signed values
   b. Compare the two registers:
      - If they differ, store in the register R5 the minimum among R2 and R3

- Otherwise, perform on R3 a logical left shift of 1 (is it equivalent to what?), sum R2 and store the result in R4 (i.e, r4=(r3<<1)+r2).

First, solve it by resorting to 1) a traditional assembly programming approach using conditional branches and then compare the execution time with a 2) conditional instructions execution approach.
Report the execution time in the two cases in the table that follows.

NOTE, report the number of clock cycles (cc), as well as the simulation time in milliseconds (ms) considering a cpu clock (clk) frequency of 16 MHz.
Refer to the guide "howto_setup_keil" to change the clock frequency in Keil.

	R2==R3 [cc]	R2==R3 [ms]	R2! =R3 [cc]	R2! =R3 [ms]
1) Traditional	7	0.0004375	9	0.0005625
2) Conditional Execution	9	0.0005625	9	0.0005625

3) Write a program that calculates the trailing zeros of a variable. The trailing zeros are computed by counting the number of zeros starting from the least significant bit and stopping at the first 1 encountered: e.g., the trailing zeros of 0b10100000 are 5. The variable to check is in R1. After the count, if the number of trailing zeros is odd, perform the sum between R2 and R3. If the number of trailing zeros is even, perform the difference between R2 and R3. In both cases the result is placed in R4.

Implement the ASM code that performs the following operations:

a. Determines whether the number of trailing zeros of R1 is odd or even.
b. As a result, the value of R4 is computed as follows:
   - If the trailing zeros are even, R4 is the difference between R2 and R3
   - Else, R4 is the sum of R2 and R3
c. Report code size and execution time (with 15MHz clk) in the following table.

Code size [Bytes]	Execution time [ms]	
	If R1 is even	Otherwise
80	0.0028	0.00326666

In the **first** exercise: the numbers inserted generated the indicated flags by the end of the program(thus after the subtraction) and not after every single operation(after an ADD and after a SUB respectively).
In the **third** exercise, the tested numbers are 0b10100000 and 0b10010000.
Time execution is computed by the ratio between CC and Frequency(CC/Freq).