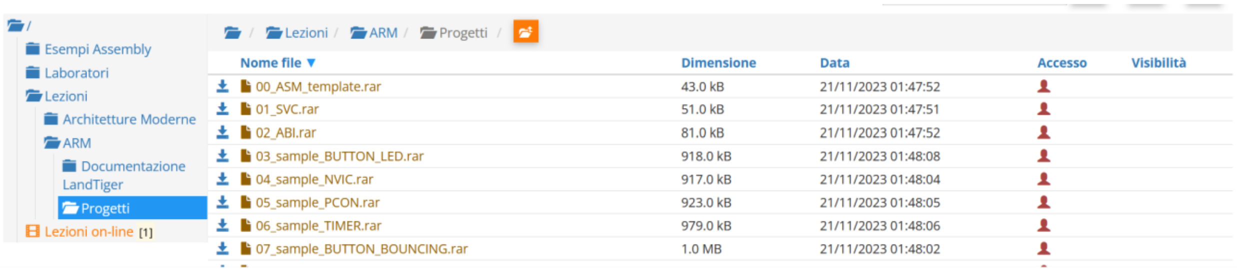
Architetture dei Sistemi di Elaborazione	Delivery date: 30th November 2023
Laboratory 6	Expected delivery of lab_06.zip must include: - Solutions of the exercises 1, 2 and 3 - this document compiled possibly in pdf format.

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



/
- Esempi Assembly
- Laboratori
- Lezioni
  - Architetture Moderne
  - ARM
    - Documentazione LandTiger
    - Progetti
- Lezioni on-line [1]

/ Lezioni / ARM / Progetti /

Nome file ▼	Dimensione	Data	Accesso	Visibilità
00_ASM_template.rar	43.0 kB	21/11/2023 01:47:52	👤	
01_SVC.rar	51.0 kB	21/11/2023 01:47:51	👤	
02_ABI.rar	81.0 kB	21/11/2023 01:47:52	👤	
03_sample_BUTTON_LED.rar	918.0 kB	21/11/2023 01:48:08	👤	
04_sample_NVIC.rar	917.0 kB	21/11/2023 01:48:04	👤	
05_sample_PCON.rar	923.0 kB	21/11/2023 01:48:05	👤	
06_sample_TIMER.rar	979.0 kB	21/11/2023 01:48:06	👤	
07_sample_BUTTON_BOUNCING.rar	1.0 MB	21/11/2023 01:48:02	👤	

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	0x10000000	0xFFFFFFFF	0x000001F4	0x000000B0
Carry = 0	0x00000001	0x00000001	0x00000003	0x00000004
Overflow	0x7FFFFFFF	0x00000002	0x80000000	0x00000001
Negative	0xFFFFFFFF	0xFFFFFFFF	0x0000000C	0x00000154
Zero	0x00000000	0x00000000	0x00000000	0x00000000

Please explain the cases when it is **not** possible to force a **single** FLAG condition:
Overflow:
- Addizione: tra due numeri positivi, forza Negative = 1, dovuto a cambio di segno; tra due numeri negativi (non riportato in tabella), si ottiene Carry = 1, dovuto ad assenza di mancanza di prestito.
- Sottrazione: minuendo positivo e sottraendo negativo, forza Carry = 1, dovuto ad assenza di prestiti mancanti; nel caso opposto, forza Negative = 1.

Carry = 0 in sottrazione: forza Negative = 1, in quanto può avvenire solo con risultato negativo.
Zero = 0 in sottrazione: forza Carry = 1, avviene in quanto non si ha mancanza di prestito

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	11	670e-6	16	1e-3
2) Conditional Execution	12	750e-6	12	750e-6

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 [µs]	
	If R1 is even	Otherwise
564	4.33	5

ANY USEFUL COMMENT YOU WOULD LIKE TO ADD ABOUT YOUR SOLUTION:
Esercizio 3: nella tabella finale, si discrimina in base alla parità del numero di zeri finali del dato; come caso dispari, si utilizza *0b10100000* (*0xA0)*, come caso pari si utilizza *0b10110000* (*0xB0)*; avendo quest’ultimo uno zero in meno del precedente, si effettua un’iterazione in meno e di conseguenza il tempo di esecuzione è (lievemente) inferiore.