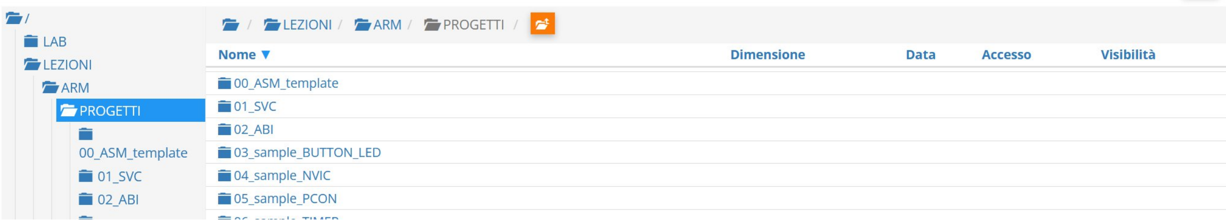
Architetture dei Sistemi di Elaborazione [AAA-GRA]
Laboratory 6

Delivery date:
25th November 2022

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 R3 and R4 to random signed values
   b. Sum R0 to R3 (R0+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 **<u>once at a time</u>** (<u>whenever possible</u>) 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	R0 + R3		R4 – R2	
	R0	R3	R4	R2
Carry = 1	0x00000000	0x00000001	0x00000101	0x00000001
Carry = 0	0x00000000	0x00000001	0x00000000	0x00000001
Overflow	0x80000000	0xC0000000	0x80000000	0x00000000
Negative	0x00000000	0xF0000000	0x00000000	0x00000002
Zero	0x00000000	0x00000001	0x00000001	0x00000001

Please explain the cases when it is **not** possible to force a **single** FLAG condition:

Is not possible after a subtraction to only have the zero flag setted to 1 without the carry flag setted to 1.

We always have c=1 when v=1

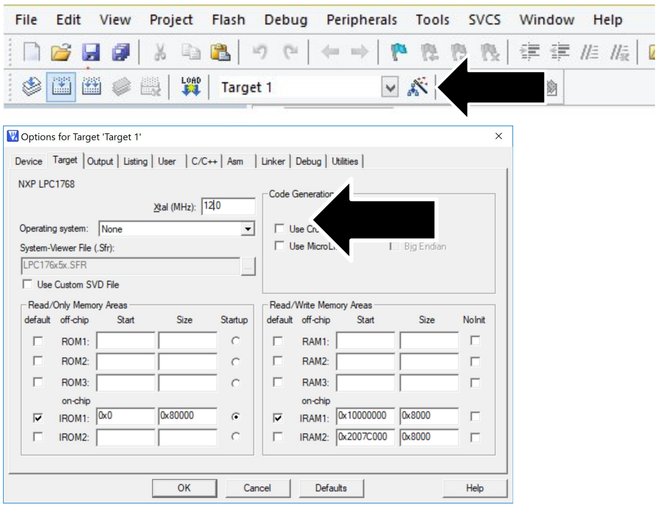
2) Write two versions of a program that performs the following operations:
   a. Initialize registers R2 and R3 to random signed values
   b. Compare the two registers:

- If they differ, store in the register R5 the maximum among R2 and R3
- Otherwise, perform a logical right shift of R3, sum R2 and store the result in R4

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) considering a cpu clock (clk) frequency of 16 MHz, as well as the simulation time in milliseconds (ms).

Notice that the processor clock frequency is setup in the menu "*Options for Target: 'Target 1'*".



	R2==R3 [cc]	R2==R3 [ms]	R2! =R3 [cc]	R2! =R3 [ms]
1) Traditional	13	1,33 x 10^{-3} ms	20	1,67 x 10^{-3} ms
2) Conditional Execution	18	1,50 x 10^{-3} ms	18	1,50 x 10^{-3} ms

3) Write a program that calculates the leading zeros of a variable. The leading zeros are computed by counting the number of zeros starting from the most significant bit and stopping at the first 1 encountered: e.g., the leading zeros of 0b00000101 are 5. The variable to check is in R1. After the count, if the number of leading zeros is odd, perform the sum between R2 and R3. If the number of leading 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 leading zeros of R1 is odd or even.
b. As a result, the value of R4 is computed as follows:
    - If the leading 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 R2 is even	Otherwise
Exercise 3) computation	564	1,25 x 10^{-3} ms	1,42 x 10^{-3} ms

ANY USEFUL COMMENT YOU WOULD LIKE TO ADD ABOUT YOUR SOLUTION: