

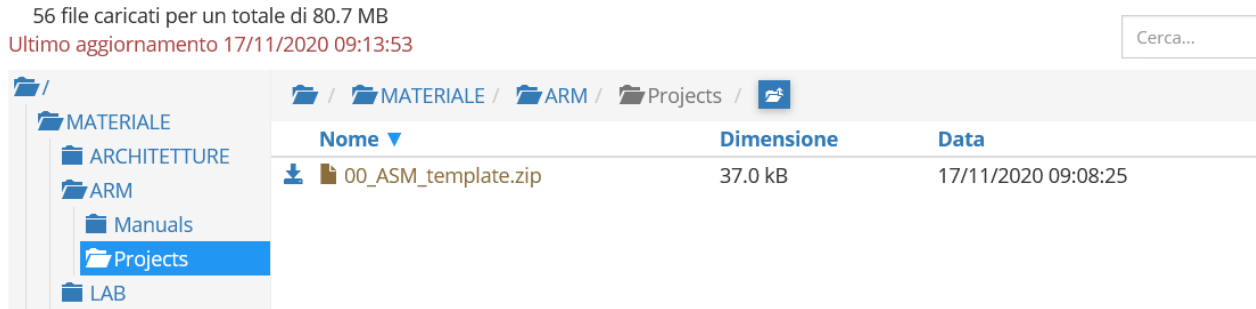
**Architetture dei Sistemi di  
Elaborazione  
02GOLOV  
Laboratory  
6**

Delivery date:  
26/11/2020

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. Sum R0 to R1 (R0+R1) and stores the result in R2
  - b. Subtract R4 to R3 (R3-R4) and stores the result in R5
  - c. Force, using the debug register window, a minimum set of specific values to be used in the program in order to **provoke as many flags (within the APSR) as possible** to be updated to 1:
    - carry
    - overflow
    - negative
    - zero
  - d. Report the selected values in the table below.

Updated flag	Please, report the hexadecimal representation of the values			
	R0 + R1		R3 - R4	
	R0	R1	R3	R4
Carry = 1	0x80000000	0x80000000	0x80000000	0x80000000
Carry = 0	0x80000000	0x00000001	0x00000000	0x00000001
Overflow	0x80000000	0x80000000	0x80000000	0x00000001
Negative	0x80000000	0x00000001	0x00000000	0x00000001
Zero	0x80000000	0x80000000	0x80000000	0x80000000

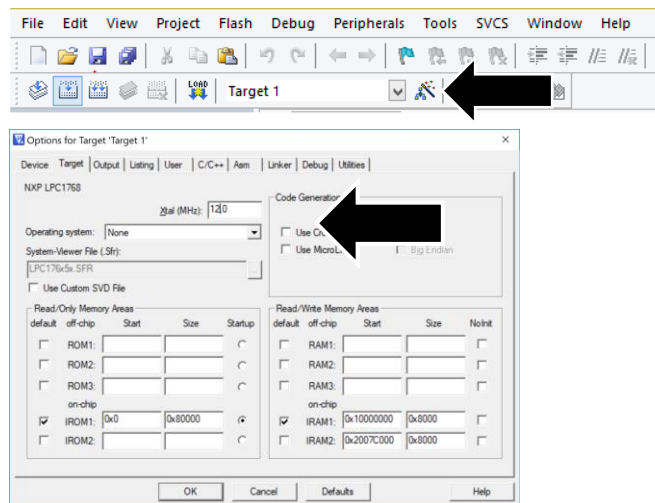
- 2) Write two versions of a program that performs the following operations described below:
  - a. Initialize registers R0 and R1 to signed values chosen by you;
  - b. Compare the content of the two registers:
    - If the stored values are equal, then check if
      - such value perfectly divides by 3, then store value 0xA in R2
      - if not, store value 0x5 in R2
    - If R0 and R1 values differ, store in the R2 the minimum among R0 and R1

The first version must be implemented resorting to a traditional assembly programming approach using conditional branches. Instead, the second version must be implemented using the conditional instructions execution approach. In order to compare exhaustively both versions

report the execution time in the table that follows. **To completely fill each column in the table, select the most appropriate data (i.e., the values to initialize R0 and R1) to explore all the program flow possibilities.**

**NOTE**, report the number of clock cycles (cc) considering a cpu clock (cclk) frequency of 12 MHz.

Notice that the processor clock frequency is setup in the menu “Options for Target: ‘Target 1’”.



<i>Program flow possibilities</i>	R0==R1 and %3==0	R0==R1 and %3!=0	R0!=R1 and R0>R1	R0!=R1 and R0<R1
<i>Programming style</i>	<i>Elapsed time depending on data, measured in clock cycles [cc]</i>			
Traditional	$1.42 \mu s \times 12 \text{ MHz} = 17 \text{ cc}$	$1.58 \mu s \times 12 \text{ MHz} = 19 \text{ cc}$	$0.83 \mu s \times 12 \text{ MHz} = 10 \text{ cc}$	$0.92 \mu s \times 12 \text{ MHz} = 11 \text{ cc}$
Conditional Execution	$1.58 \mu s \times 12 \text{ MHz} = 19 \text{ cc}$	$1.58 \mu s \times 12 \text{ MHz} = 19 \text{ cc}$	$0.83 \mu s \times 12 \text{ MHz} = 10 \text{ cc}$	$0.83 \mu s \times 12 \text{ MHz} = 10 \text{ cc}$

- 3) Write a program able to indicate whether a register contains a value that shows “even” or “odd” parity. The parity refers to the total number of 1-bits in a binary string. For example, the decimal number 4 is showing an **odd parity** (0100 ← a single 1-bit), while the decimal number 5 has an **even parity** (0101 ← two 1-bits).

Implement the ASM code that performs the following operations:

- It determines whether the registers R0 and R1 are showing the same parity,
- As a result, the values of R0 and R1 are updated as following:
  - If R0 and R1 have the same parity (both even or both odd): the program clears (to binary value 0b) the 8 Most Significant Bits – MSB - and sets (to value 1b) the 8 Least Significant Bits – LSB - of R0. **All other bits must remain unchanged.**
  - If R0 and R1 have different parity: the program copies in R1 the values of the flags.
- Report code size (look for the .text section in the map file generated by Keil) and execution time (with 12MHz cclk) in the following table:

	Code size [Bytes]	Execution time [cc]	
		if both Odd or Even	Otherwise
Exercise 3) computation	88	$22.17 \mu s \times 12 \text{ MHz} = 266 \text{ cc}$	$22.17 \mu s \times 12 \text{ MHz} = 266 \text{ cc}$

ANY USEFUL COMMENT YOU WOULD LIKE TO ADD ABOUT YOUR SOLUTIONS: *In exercise 2, in the second version, due to nested IF statements it was not possible to replace all conditional branches to conditional instructions.*