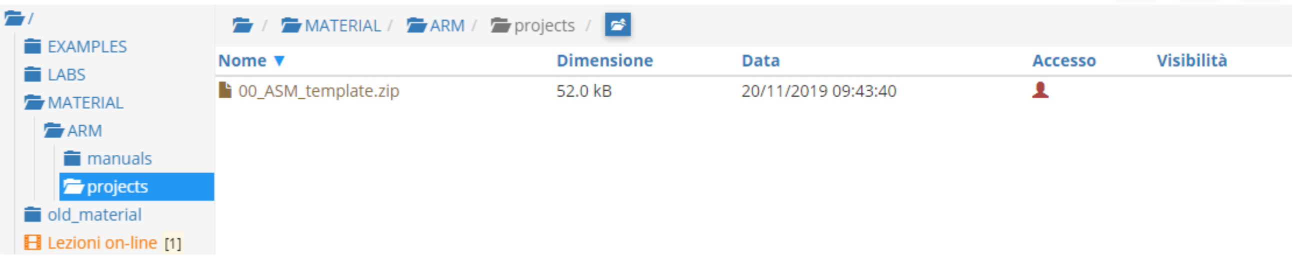
Architetture dei Sistemi di Elaborazione	Delivery date: 26/11/2021
Laboratory 6	Expected delivery of lab_06.zip must include: - Solutions of the exercises 1, 2 and 3 - this document compiled possibly in pdf format.

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



Nome ▼	Dimensione	Data	Accesso	Visibilità
00_ASM_template.zip	52.0 kB	20/11/2019 09:43:40		

1) Write a program using the ARM assembly that performs the following operations:
   a. Sum R0 to R1 (R0+R1) and store the result in R2
   b. Subtract R4 to R3 (R3-R4) and store the result in R5
   c. 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
   d. Report the selected values in the table below.

	Please, report the hexadecimal representation of the values			
Updated flag	R0 + R1		R3 - R4	
	R0	R1	R3	R4
Carry = 1	0x0000000F	0xFFFFFFFB	0x00000016	0x00000005
Carry = 0	0x00000007	0x00000001	0x0000000F	0xFFFFFFFB
Overflow	0xAFFFFFFF	0xAFFFFFFF	0x80000000	0x00000001
Negative	0x00000000	0xFFFFFFFF	0x0000000C	0x0000000D
Zero	0x00000000	0x00000000	0xFFFFFFFF	0xFFFFFFFF

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

Overflow Flag --> Carry Flag

Zero Flag (Sub) --> Carry Flag

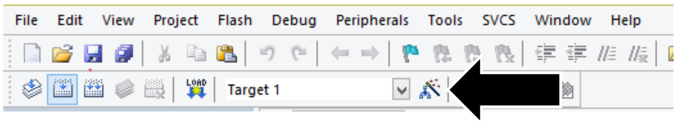
2) Write two versions of a program that performs the following operations:
   a. Initialize registers R2 and R3 to random signed values
   b. Compare the two registers:
      - If they differ, store in the register R4 the minimum among R2 and R3

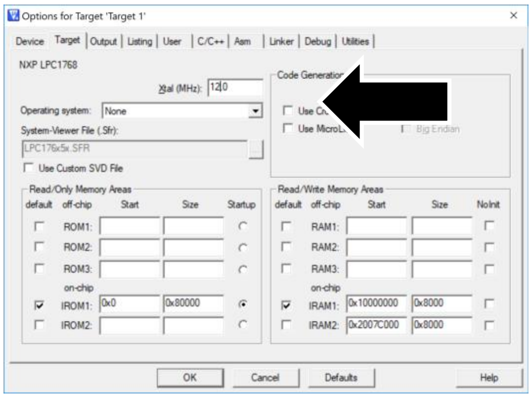
- Otherwise, perform an arithmetic right shift of R3, sum R2 and store the result in R5

First, solve it 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 12 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'*".





NXP LPC1768

Xtal (MHz): 12.0

Operating system: None

System-Viewer File (.Sfr): LPC176x5x.SFR

□ Use Custom SVD File

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
□	ROM1:			○
□	ROM2:			○
□	ROM3:			○
	on-chip			
☑	IROM1:	0x0	0x80000	●
□	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
□	RAM1:			□
□	RAM2:			□
□	RAM3:			□
	on-chip			
☑	IRAM1:	0x10000000	0x8000	□
□	IRAM2:	0x2007C000	0x8000	□

	R0==R1 [cc]	R0==R1 [ms]	R0!=R1 [cc]	R0!=R1 [ms]
1) Traditional	13	0.00108	11	0.00092
2) Conditional Execution	12	0.00100	12	0.00100

3) Write a program that calculates the **Hamming distance** between two values. The Hamming distance is defined as the number of positions at which the corresponding values are different: e.g., the Hamming distance between the values *0b1010101* and *0b1001001* is 3. The initial values are stored in R0 and R1, while the resulting Hamming distance must be stored in R2.

Implement the ASM code that performs the following operations:

a. It determines whether the content of R2 is odd or even.

b. As a result, the values of R0 and R1 are updated as follows:
   - If R2 is even, the program clears the 11th bit of R0 and sets to 1 the 6th bit of R1 (all other bits must remain unchanged)
   - Else, the program copies in R1 the values of the flags.

c. Report code size and execution time (with 15MHz clk) in the following table.

	Code size [Bytes]	Execution time [s]	
		If R2 is even	Otherwise
Exercise 3) computation	84	0.00001780	0.00001800

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