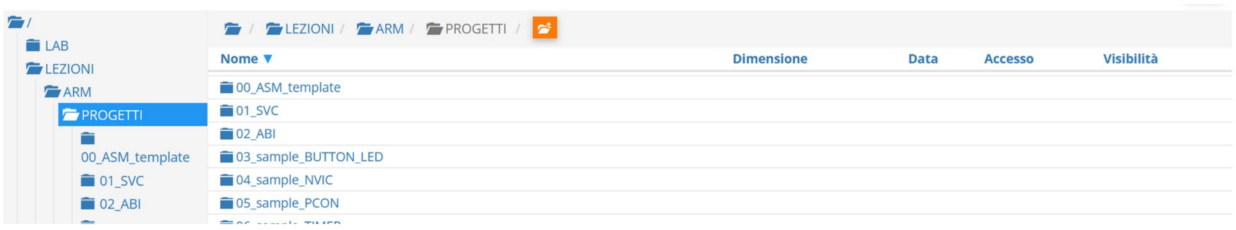
Architetture dei Sistemi di Elaborazione [AAA-GRA] Laboratory 6	Delivery date: 25th November 2022 Expected delivery of lab_06.zip must include: - Solutions of the exercises 1, 2 and 3 - this document compiled possibly in pdf format.

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


/ LAB LEZIONI ARM PROGETTI 00_ASM_template 01_SVC 02_ABI

/ LEZIONI / ARM / PROGETTI /

Nome ▼	Dimensione	Data	Accesso	Visibilità
00_ASM_template				
01_SVC				
02_ABI				
03_sample_BUTTON_LED				
04_sample_NVIC				
05_sample_PCON				

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 **once at a time** (whenever possible) to 1:
      - carry
      - overflow
      - negative
      - zero
   e. Report the selected values in the table below.

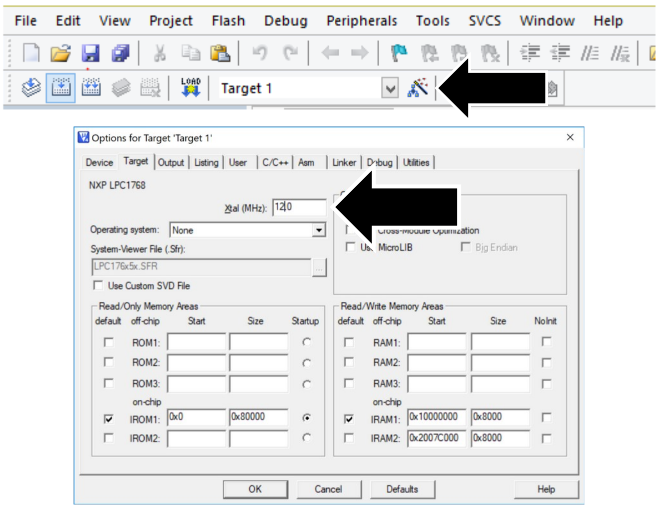
	Please, report the hexadecimal representation of the values			
Updated flag	R0 + R3		R4 – R2	
	R0	R3	R4	R2
Carry = 1	0X00000002	0XFFFFFFFF	0X00000001	0X00000000
Carry = 0	0X00000000	0X00000001	0X00000004	0X00000005
Overflow	0X7FFFFFFF	0X00000001	0X00000004	0X00000005
Negative	0X00000001	0XFFFFFFFD	0X00000004	0X00000005
Zero	0X00000001	0XFFFFFFFF	0X00000001	0X00000001

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

- If Carry equals 0 in SUB also the Negative register and Overflow register are equal to 1
- If Zero flag is enabled in SUB also the Carry is enabled.

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: **<u>NOTE</u>**, 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'*".



NXP LPC1768
Xtal (MHz): 12.0
Operating system: None
System-Viewer File (.Sfr): LPC176x5x.SFR
☐ Use Custom SVD File
☐ Cross-Module Optimization
☐ Use MicroLIB ☐ Big Endian

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	☐

	R2==R3 [cc] (*)	R2==R3 [ms]	R2! =R3 [cc] (*)	R2! =R3 [ms]
1) Traditional	10.72 → 11	0.00067	14.72 → 15	0.00092
2) Conditional Execution	12	0.00075	12	0.00075

(*) I have calculated this value as *cc = f * execution_time*

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 [*replace this with the proper time measurement unit*]	
		If R2 is even	Otherwise 1
Exercise 3) computation	564	0.01875 ms (value #2 with 30 leading zeros)	0.01933ms (value #1 with 31 leading zeros)

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

The execution time is affected by the value of register R2.
In my program the execution time increases with *small number*.
For the 0 case (32 leading zeros) I put a specific comparison, for this reason it is the fastest case.