Computer Architectures **02LSEOV 02LSEOQ [AA_LZ]**	Delivery date: Thursday 29/11
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.

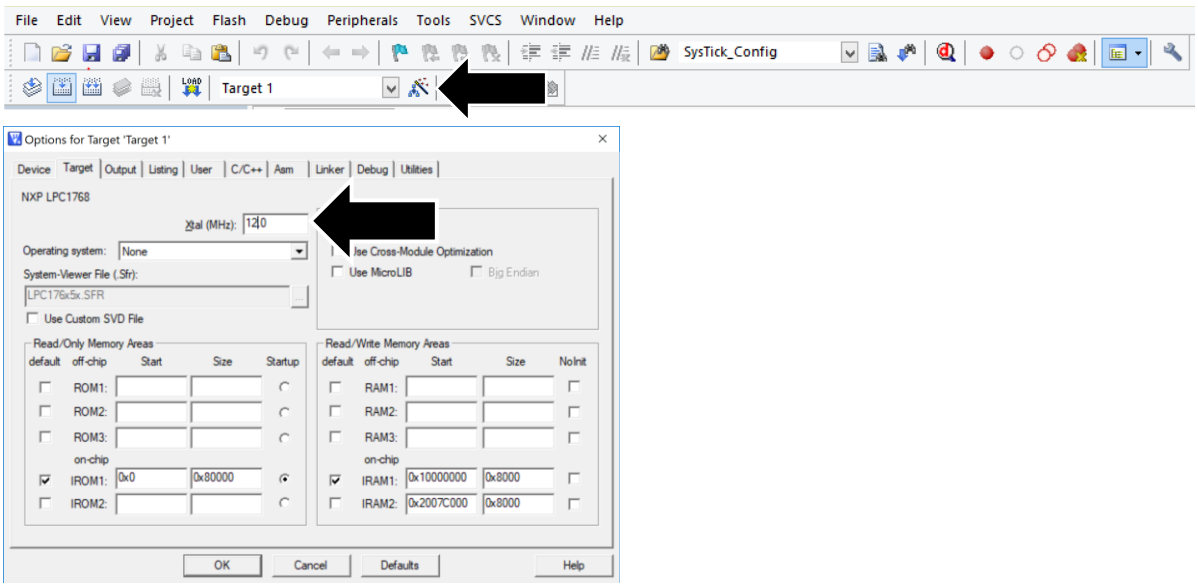
Solve the following problems by starting from the *template.s* file.

1) Write a program using the ARM assembly that makes the following simple operations:
   a. Sum R0 to R1 (R0+R1) and stores the result in R2
   b. Subtract R4 to R3 (R4-R3) and stores the result in R5
   c. Select and force, using debug register window, a set of specific values to be used in the program in order to provoke the following flag to be updated 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
Carry = 1	0xFFFFFFFF	0x2	0x4	0x5
Carry = 0	0x1	0x2	0x5	0x4
Overflow	0x7FFFFFFF	0x1	0xFFFFFFFF	0x7FFFFFFF
Negative	0x0	0xFFFFFFFF	0x5	0x4
Zero	0x1	0xFFFFFFFF	0x5	0x5

2) Write two versions of a program that performs the following operations:
   a. Setup registers R0 and R1 to random signed values
   b. Compare the registers:
      - If they differ, compute and store their average in the register R2
      - Otherwise, rise R0 to the square and store this value in R3

   Solve the problem by using 1) the conditional execution and compare the execution time with a 2) traditional approach. Report the execution time in the two cases in the table that follows: please, 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'*".



# cc	R0==R1	R0!=R1
1) Conditional execution	0.58 µs * 12 MHz = 7	0.58 µs * 12 MHz = 7
2) Traditional	0.58 µs * 12 MHz = 7	0.75 µs * 12 MHz = 9

The values have been computed with constants capable of fitting into the 12-bits format supported by the MOV instruction, into which the LDR is translated after the assembling phase.

3) Write a program able to indicate whether a register contains a value that shows "even" or "odd" parity. In mathematics, an integer is "even" if it is evenly divisible by two, and "odd" if it is not even. In computer science, the parity concept is different and usually indicates whether the total number of 1-bits in the string is even or odd. For example, the number 4 is showing to be odd (0100 ← just a single 1-bit), while the number 5 is even (0101← two 1-bits).
    a. In our case, the target register is R0 and, as a result, the value of R1 is updated as following
        - R1 is written to all 0s in case of even parity in R0
        - R1 is set to all 1s in case of odd parity in R0
    b. Report code size and execution time (with 12MHz cclk) in the following table.

		Execution time	
	Code size	Odd	Even
Parity computation	564 B	18.83 µs	18.83 µs

The values have been computed with constants not capable of fitting into the 12-bits format supported by the MOV instruction, so that the LDR instruction is translated into the access to a literals pool in memory.