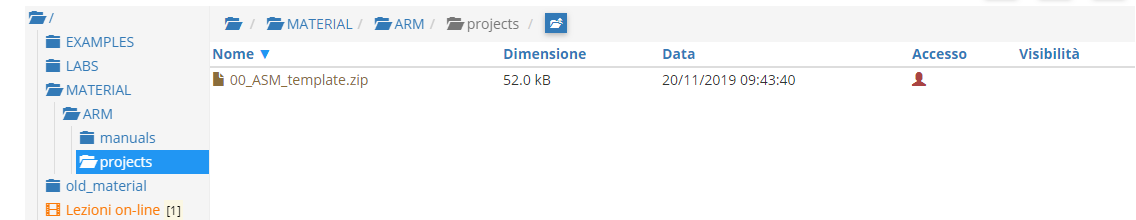
|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione 02GOLOV [M-Z]** | Delivery date:  28/11/2019 |
| **Laboratory**  **5** | Expected delivery of lab\_05.zip must include:   * Solutions of the exercises 1, 2 and 3 * this document compiled possibly in pdf format. |

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



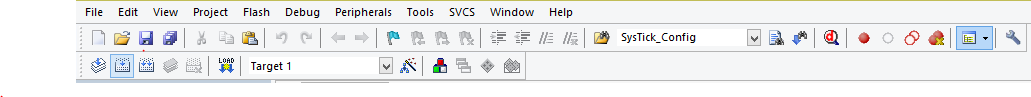
1. Write a program using the ARM assembly that performs the following operations:
   1. Sum R0 to R1 (R0+R1) and stores the result in R2
   2. Subtract R4 to R3 (R3-R4) and stores the result in R5
   3. Force, using the debug register window, a set of specific values to be used in the program in order to provoke the following flag to be updated **once at a time** to 1:
      * carry
      * overflow
      * negative
      * zero
   4. Report the selected values in the table below.

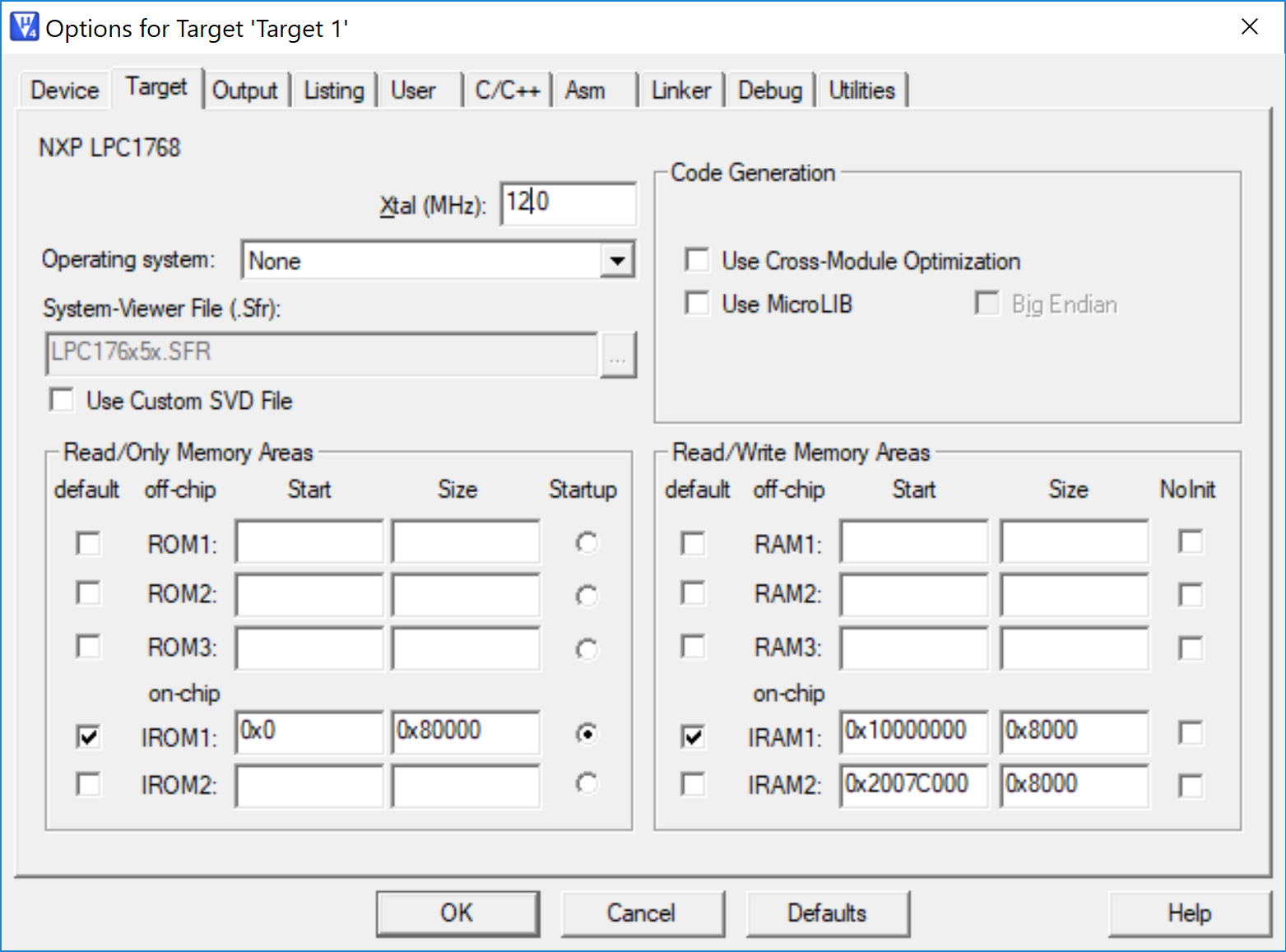
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Please, report the hexadecimal representation of the values | | | |
| Updated flag | R0 + R1 | | R3 - R4 | |
| R0 | R1 | R3 | R4 |
| Carry = 1 | 0xFFFFFFFF | 0x01 | 0x09 | 0x0A |
| Carry = 0 | 0xFF | 0xFF | 0xfffffff0 | 0x0A |
| Overflow | 0xC0000000 | 0x80000000 | 0x40000001 | 0xC0000000 |
| Negative | 0xAA | 0x0A | 0xAA | 0x0A |
| Zero | 0xFFFFFFFF | 0x01 | 0x0A | 0x0A |

1. Write two versions of a program that performs the following operations:
   1. Initialize registers R0 and R1 to random signed values
   2. Compare the two registers:
      * If they differ, store in the register R2 the maximum among R0 and R1
      * Otherwise, rise R0 to the square, sum R1 and store the result in R3

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 (cclk) frequency of 12 MHz.

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





|  |  |  |  |
| --- | --- | --- | --- |
|  | R0==R1 [cc] | R0!=R1 &&R0>R1 [cc] | R0!=R1 &&R0<R1 [cc] |
| 1) Traditional | 11 | 14 | 13 |
| 2) Conditional Execution | 11 | 15 | 14 |

1. 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:

* 1. It determines whether the register R0 and R1 are showing the same parity,
  2. As a result, the value of R0 and R1 is updated as following:
     + If R0 and R1 have the same parity (both even or both odd), the program clears the 16th bit of R0 and sets to 1 the 8th bit of R1 (all other bits must remain unchanged)
     + If R0 and R1 have different parity, the program copies in R1 the values of the flags.
  3. Report code size and execution time (with 12MHz cclk) in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Code size [Bytes] | Execution time  [*sec*] | |
| if both Odd or Even | Otherwise |
| Exercise 3) computation | 116 | 0.00000458 | 0.00000433 |

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