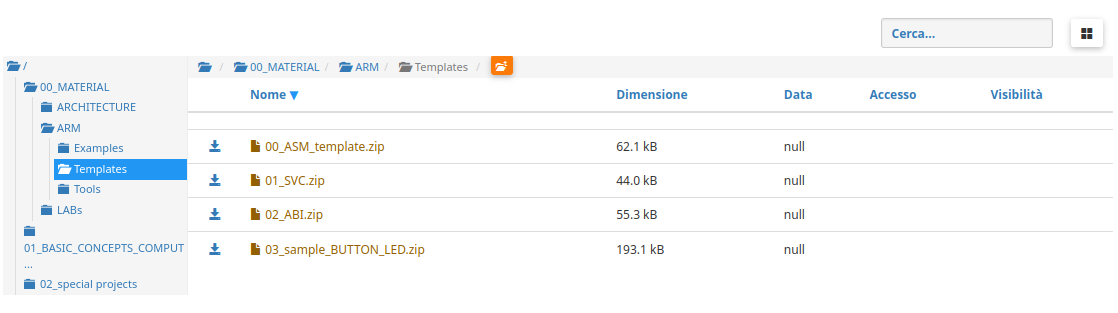
|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione** | Delivery date:  12 November 2024 |
| **Laboratory**  **6** | Expected delivery of lab\_06.zip must include:   * Solutions of the exercises 1, 2, 3 and 4 * this document compiled possibly in pdf format. |

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. Initialize registers *R1*, *R2*, and *R3* to random signed values.
   2. Subtract *R2* to *R1* (*R2* – *R1*) and store the result in *R4*.
   3. Sum *R2* to *R3* (*R2* + *R3*) and store the result in *R5*.

Using the debug log window, change the values ​​of the written program in order to set the following flags to 1, one at a time and when possible:

* + - carry
    - overflow
    - negative
    - zero

Report the selected values in the table below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Hexadecimal representation of the obtained values | | | |
| Updated flag | R2 – R1 | | R2 + R3 | |
| R2 | R1 | R2 | R3 |
| Carry = 1 | -1 | 1 | -1 | -1 |
| Carry = 0 | 1 | -1 | 1 | 1 |
| Overflow | 0x7FFFFFFF | 0xFFFFFFFF | 0x7FFFFFFF | 0x1 |
| Negative | -1 | 0 | -1 | 0 |
| Zero | -1 | -1 | -1 | 1 |

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

Non si può forzare un singolo flag alla volta a 1 soprattutto nel caso dell’overflow (V). Andando ad 1 il flag dell’overflow, va ad 1 anche il flag del negative a causa del carry.

1. Write a program that performs the following operations:
   1. Initialize registers *R6* and *R7* to random signed values.
   2. Compare the two registers:
      * If they differ, store in register *R8* the maximum among *R6* and *R7*.
      * Otherwise, perform a logical right shift of 1 on *R6* (is it equivalent to what? Divisione per 2), then subtract this value from *R7* and store the result in *R4* (i.e., *R4* = *R7* – (*R6* >> 1)).

Considering a CPU clock frequency (clk) of *16 MHz*, report the number of clock cycles (cc) and the simulation time in milliseconds (ms) in the following table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R6 == R7 [cc] | R6 == R7 [ms] | R6 != R7 [cc] | R6 != R7 [ms] |
| Program 3 | 26 | 0.0022 | 62 | 0.0052 |

*Note: you can change the CPU clock frequency by following the brief guide at the end of the document.*

1. Write a program that calculates the leading zeros of a variable. Leading zeros are calculated by counting the zeros starting from the most significant bit and stopping at the first 1 encountered: for example, there are five leading zeros in *2\_00000*101. The variable to be checked is in *R10*. After counting, if the number of leading zeros is odd, subtract *R11* from *R12*. If the number of leading zeros is even, add *R11* to *R12*. In both cases, the result is placed in *R13*.

Implement ASM code that does the following:

* 1. Determine whether the number of leading zeros of *R1* is odd or even (with conditional/test instructions!).
  2. The value of R13 is then calculated as follows:
* If the leading zeros are even, *R13* is the sum of *R11* and *R12*.
* Otherwise, *R13* is the subtraction of *R11* and *R12*.

1. Assuming a *15 MHz* clk, report the code size and execution time in the following table:

|  |  |  |
| --- | --- | --- |
| Code size [Bytes] | Execution time  [*replace this with the proper  time measurement unit*] | |
| If the leading zeroes are even | Otherwise |
| 564 | 3.4 | 4.5 |

1. Create two optimized versions of program 4 (where possible!)
   1. Using conditional execution.
   2. Using conditional execution in IT block.

Report and compare the execution Time

|  |  |  |  |
| --- | --- | --- | --- |
| Program | Code size [Bytes] | Execution time  [*replace this with the proper  time measurement unit*] | |
| If the leading zeroes are even | Otherwise |
| Program 4 (baseline) | 564 | 3.4 | 4.5 |
| Program 4.a | 564 | 3.4 | 4.5 |
| Program 4.b | 564 | 3.4 | 4.5 |

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

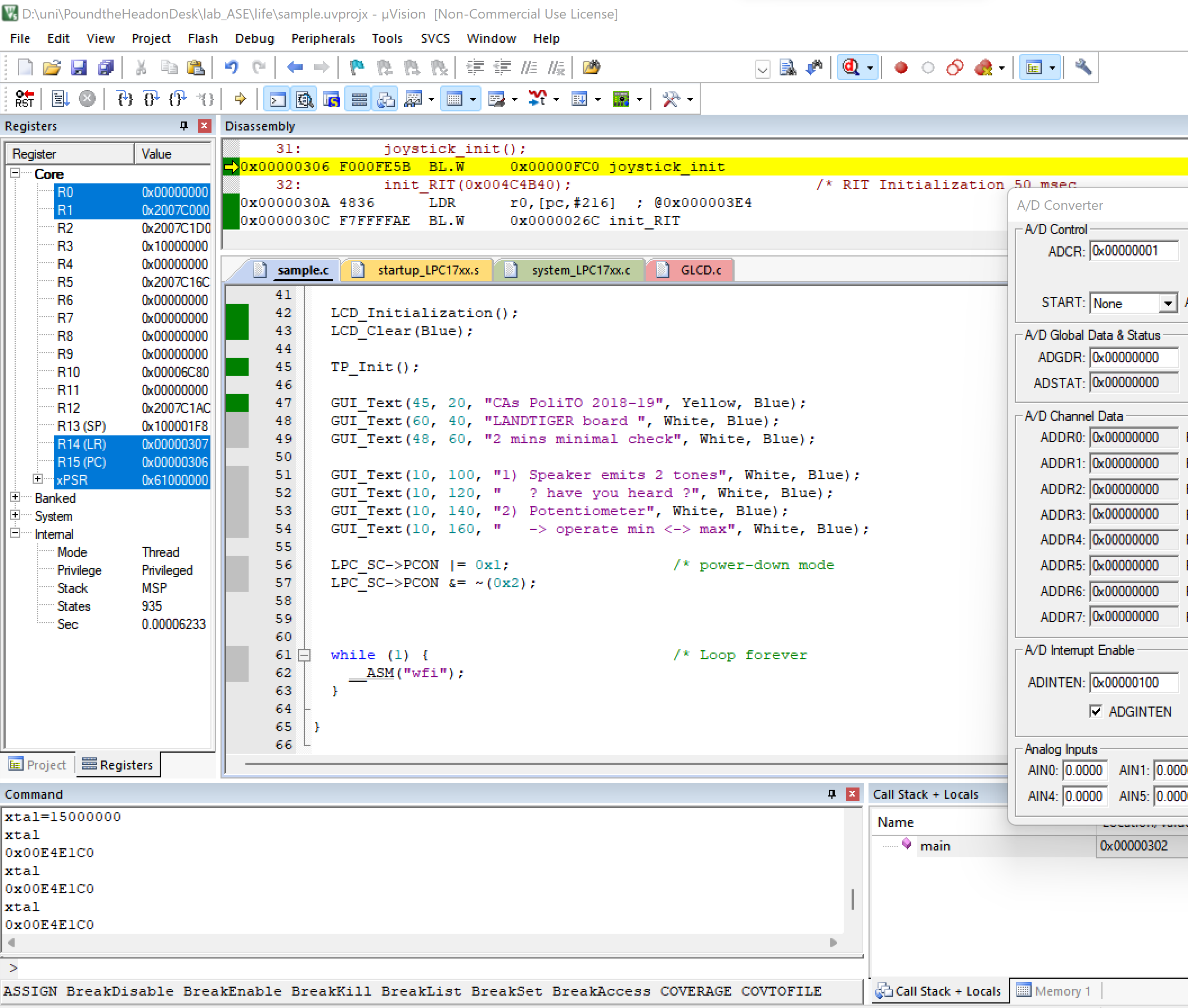
**How to set the CPU clock frequency in Keil**

1. Launch the debug mode and activate the command console.

Immagine che contiene testo

Descrizione generata automaticamente

1. A window will appear:



You can type *xtal* to check its value. To change its value, make a routine assignment, i.e., *xtal=frequency*, keeping in mind that frequency is in Hz must be entered. To set a frequency of *15 MHz*, you must write as follows: *xtal=15000000*.