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

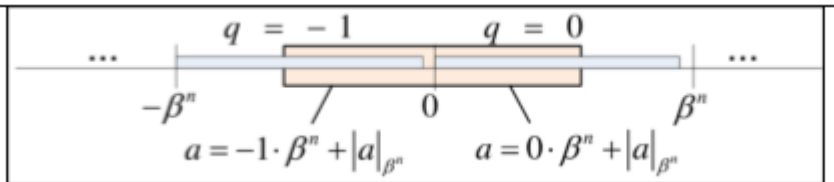
   Using the debug log window, change the values of the written program in order to set the following flags to 1, one at a time and when possible:
   - carry
   - overflow
   - negative
   - zero

Report the selected values in the table below:

Updated flag	Hexadecimal representation of the obtained values			
	R2 – R1		R2 + R3	
	R2	R1	R2	R3
Carry = 1	*0xFFFFFFFF*	*0xF0000011*	*0xFFFFFFFF*	*0x000000FF*
Carry = 0	*0x0FFFFFFF*	*0xFFFFFFFF*	*0x0FFFFFFF*	*0x0FFFFFFF*
Overflow	*0x7FFFFFFF*	*0xFFFFFFFF*	*0x7FFFFFFF*	*0x7FFFFFFF*
Negative	*0x0045ABCD*	*0x0116AF34*	*0x0045ABCD*	*0x98FD54DD*
Zero	*0x0045ABCD*	*0x0045ABCD*	*0x0045ABCD*	*0xFFBA5433*

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

- when performing operations to set the Zero flag, it's not possible not to set the Carry flag too
- it's not possible to set the Overflow flag without setting either the Negative flag or the Carry flag as well as we are crossing the integers range of representability 2^64/2 as shown in figure, crossing the border will set the overflow flag but crossing the border means in any case (N or Z) that the MSB changes so the Negative flag is always set



2) Write a program that performs the following operations:
   a. Initialize registers *R6* and *R7* to random signed values.
   b. 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? to divide by the power of 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:

	R2 == R3 [cc]	R2 == R3 [ms]	R2 != R3 [cc]	R2 != R3 [ms]
Program 2	18.08	0.00113	18.08	0.00113

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

3) 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 *R8 (student note:R13 is reserved)*.

Implement ASM code that does the following:
   a. Determine whether the number of leading zeros of *R10* is odd or even (with conditional/test instructions!).
   b. The value of R8 is then calculated as follows:
      - If the leading zeros are even, *R8* is the sum of *R11* and *R12*.
      - Otherwise, *R8* is the subtraction of *R11* and *R12*.
   a) Assuming a *15 MHz* clk, report the code size and execution time in the following table:

Code size [Bytes]	Execution time [*µs*]	
	If the leading zeroes are even	Otherwise
564	*7.80*	*11.4*

4) Create two optimized versions of program 3 (where possible!)
   a. Using conditional execution.
   b. 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 3 (baseline)	XXX	XXX	XXX
Program 4.a	XXX	XXX	XXX
Program 4.b	XXX	XXX	XXX

Any Useful Comment You Would Like To Add About Your Solution:

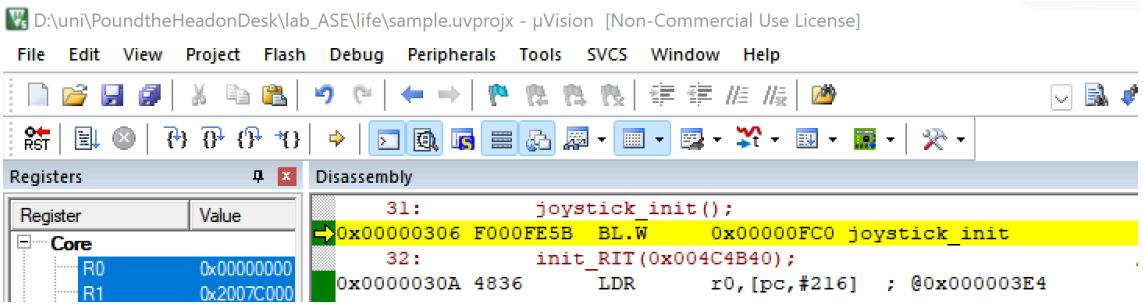
I wrote the program 3 already optimized to no other optimization that i know can be applied.

I gave the solution in a single file since is not specified if we needed to have 4 different files, the runtime is done commenting out the parts of the code not needed for the section of the laboratory request

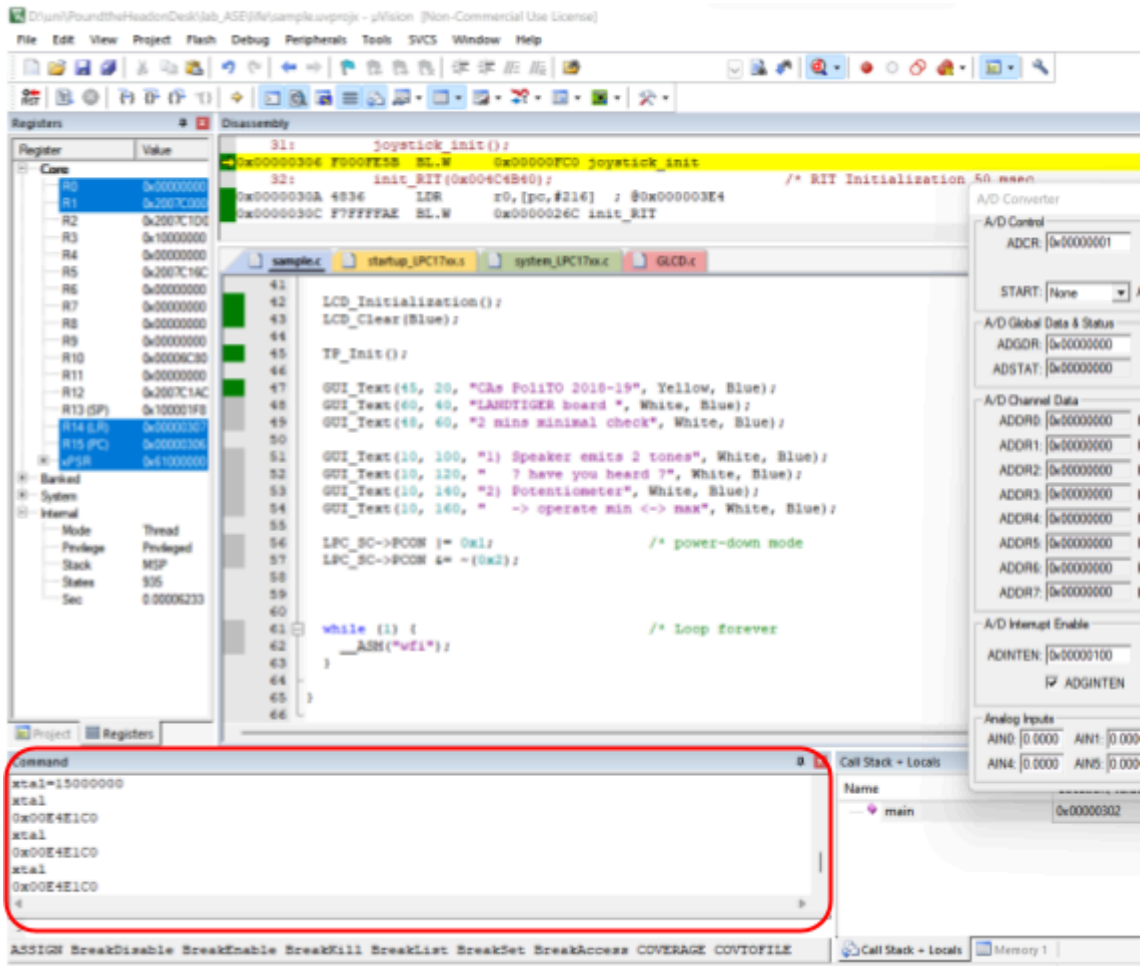
PS i fixed some typos as R13 was used but it is reserved and only 3 programs are requested not 4 versions and optimization but 3 + 2 optimizations

How to set the CPU clock frequency in Keil

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



2) 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*.