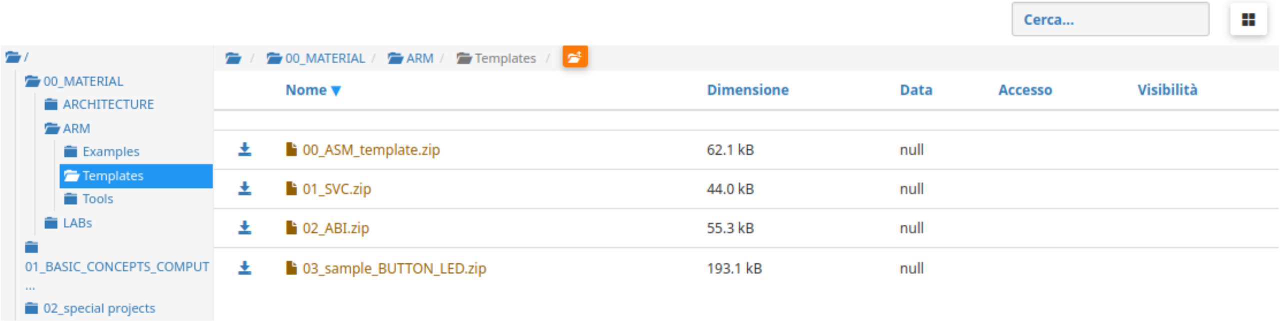
Architetture dei Sistemi di Elaborazione	Delivery date: 12 November 2024
Laboratory 6	Expected delivery of lab_06.zip must include: - Solutions of the exercises 1, 2, 3 and 4 - this document compiled possibly in pdf format.

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



Nome ▼	Dimensione	Data	Accesso	Visibilità
00_ASM_template.zip	62.1 kB	null		
01_SVC.zip	44.0 kB	null		
02_ABI.zip	55.3 kB	null		
03_sample_BUTTON_LED.zip	193.1 kB	null		

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:

	Hexadecimal representation of the obtained values			
Updated flag	R2 – R1		R2 + R3	
	R2	R1	R2	R3
Carry = 1	0x6A	0X7C	0XFFFFFFFF	0X1
Carry = 0	0x80000000	0x1	0X6A	0X7C
Overflow	0x80000000	0x1	0x7FFFFFFF	0X1
Negative	0X1C	0X6D	0XFFFFFFFB	0X02
Zero	0xAC	0XAC	0XFFFFFFFF	0X1

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

It’s impossible to have Carry (C) = 1 and Zero (Z) = 1 at the same time because they represent opposite outcomes. The carry flag means there was an overflow or borrow, while the zero flag means the result is exactly zero. If there's a carry, the result can't be zero, and if the result is zero, there can't be a carry.

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?), 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 2	13	0.00133 ms	13	0.00133 ms

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 casese, the result is placed in *R13*.

   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 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*.

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

Code size [Bytes]	Execution time 0.00100 ms	
	If the leading zeroes are even	Otherwise
	546 bytes	546 bytes

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

   Report and compare the execution Time

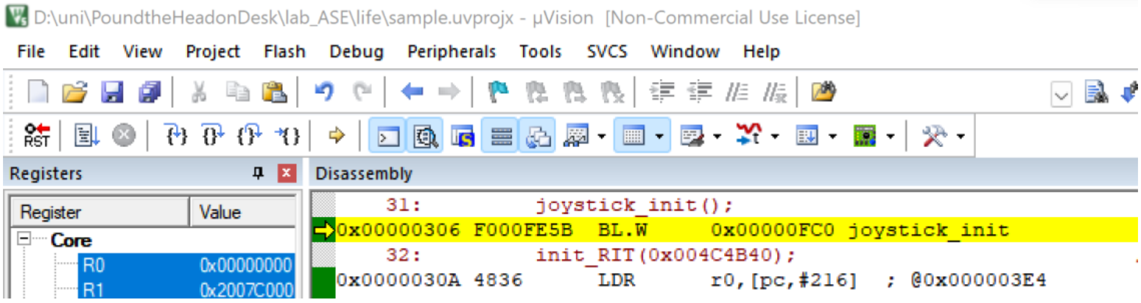
Program	Code size [Bytes]	Execution time [*replace this with the proper time measurement unit*]	
		If the leading zeroes are even	Otherwise
Program 4 (baseline)	546 bytes	0.00100 ms	0.00100 ms
Program 4.a	546 bytes	0.00100 ms	0.00100 ms
Program 4.b	564 bytes	0.00108 ms	0.00108 ms

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

Since the code was already optimized in step 3. There was no need to provide a better solution in program 4.a

How to set the CPU clock frequency in Keil

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

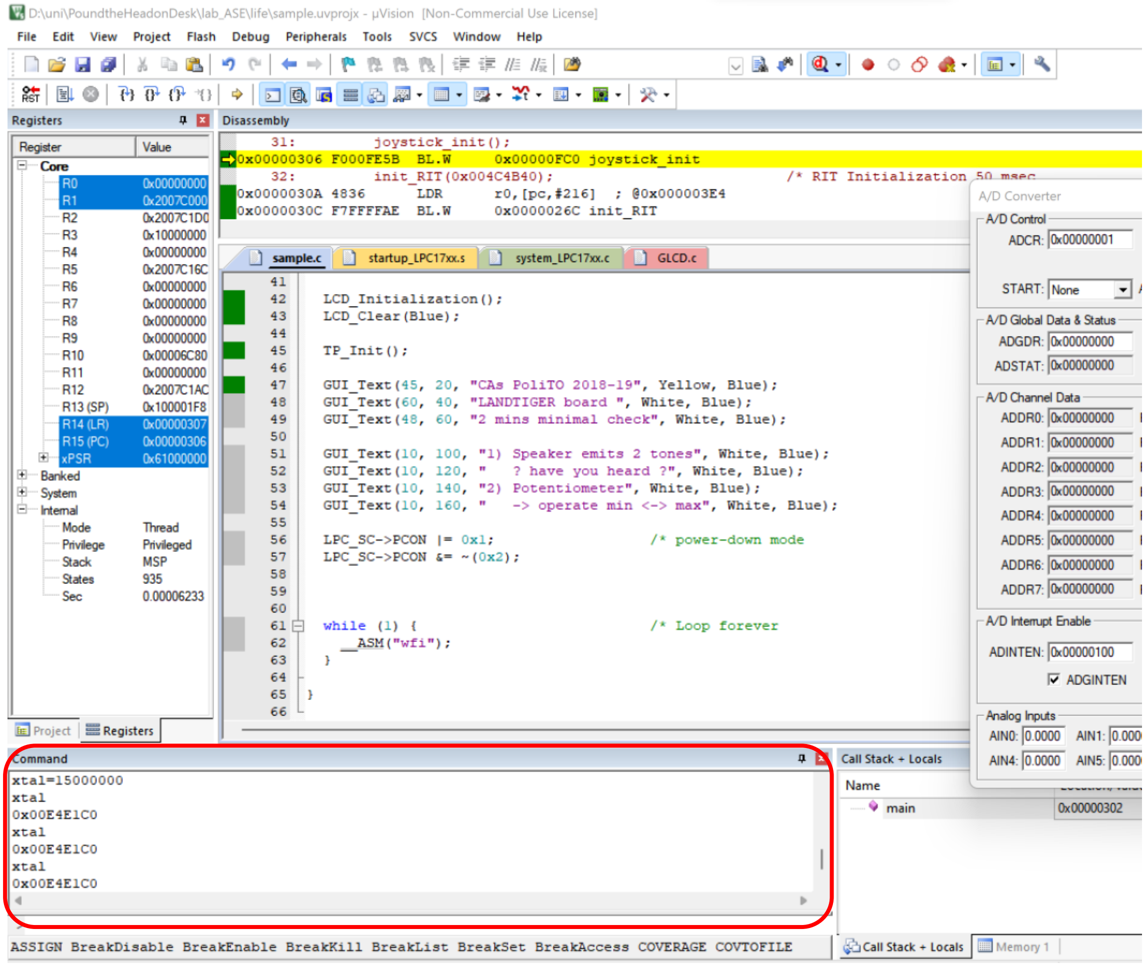


D:\uni\PoundtheHeadonDesk\lab_ASE\life\sample.uvprojx - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RST
Registers
Disassembly

Register	Value
Core	
R0	0x00000000
R1	0x2007C000

```
      31:          joystick_init();
0x00000306 F000FE5B  BL.W       0x00000FC0 joystick_init
      32:          init_RIT(0x004C4B40);
0x0000030A 4836       LDR       r0,[pc,#216]  ; @0x000003E4
```

2) A window will appear:



D:\uni\PoundtheHeadonDesk\lab_ASE\life\sample.uvprojx - µVision [Non-Commercial Use License]
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
RST
Registers
Disassembly
sample.c
startup_LPC17xx.s
system_LPC17xx.c
GLCD.c
Project
Registers
Command
Call Stack + Locals
Memory 1
A/D Converter

Register	Value
Core	
R0	0x00000000
R1	0x2007C000
R2	0x2007C1D0
R3	0x10000000
R4	0x00000000
R5	0x2007C16C
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00006C80
R11	0x00000000
R12	0x2007C1AC
R13 (SP)	0x100001F8
R14 (LR)	0x00000307
R15 (PC)	0x00000306
xPSR	0x61000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	935
Sec	0.00006233

```
      31:          joystick_init();
0x00000306 F000FE5B  BL.W       0x00000FC0 joystick_init
      32:          init_RIT(0x004C4B40);                    /* RIT Initialization 50 msec
0x0000030A 4836       LDR       r0,[pc,#216]  ; @0x000003E4
0x0000030C F7FFFFAE  BL.W       0x0000026C init_RIT
```

```

  LCD_Initialization();
  LCD_Clear(Blue);

  TP_Init();

  GUI_Text(45, 20, "CAs PoliTO 2018-19", Yellow, Blue);
  GUI_Text(60, 40, "LANDTIGER board ", White, Blue);
  GUI_Text(48, 60, "2 mins minimal check", White, Blue);

  GUI_Text(10, 100, "1) Speaker emits 2 tones", White, Blue);
  GUI_Text(10, 120, "   ? have you heard ?", White, Blue);
  GUI_Text(10, 140, "2) Potentiometer", White, Blue);
  GUI_Text(10, 160, "   -> operate min <-> max", White, Blue);

  LPC_SC->PCON |= 0x1;                     /* power-down mode
  LPC_SC->PCON &= ~(0x2);



  while (1) {                              /* Loop forever
    __ASM("wfi");
  }

}

```

A/D Control

ADCR: 0x00000001

START: None

A/D Global Data & Status

ADGDR: 0x00000000

ADSTAT: 0x00000000

A/D Channel Data

ADDR0: 0x00000000

ADDR1: 0x00000000

ADDR2: 0x00000000

ADDR3: 0x00000000

ADDR4: 0x00000000

ADDR5: 0x00000000

ADDR6: 0x00000000

ADDR7: 0x00000000

A/D Interrupt Enable

ADINTEN: 0x00000100

☑ ADGINTEN

Analog Inputs

AIN0: 0.0000 AIN1: 0.000

AIN4: 0.0000 AIN5: 0.000

```
xtal=15000000
xtal
0x00E4E1C0
xtal
0x00E4E1C0
xtal
0x00E4E1C0
```

```
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE
```

Name	
main	0x00000302

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*.