Computer Architectures	Delivery date: 13 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.

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	*0x00FFFF4A*	*0x0000FF4A*	*0x0000FF4A*	*0xFFFF07C2*
Carry = 0	*0x0000FFFF*	*0xFFFFFF44*	*0x00FFFF4A*	*0x000000B2*
Overflow				
Negative	*0x00FFFF44*	*0x0000FFFF*	*0x0000FF4A*	*0xFFFF00B2*
Zero	*0x00000000*	*0x00000000*	*0x00000000*	*0x00000000*

Please explain the cases where it is **not** possible to force a **single** FLAG condition:
For the zero in the subtraction case also the C flag is set to 1, which means that there is no borrow.
For the overflow it's not possible to have only V set: there is always also N 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 a division by 2 if unsigned numbers (for signed needed arithmetic shift)*), 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	*7*	*0.00044*	*10*	*0.00063*

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*.

   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
40	*4.33*	*4.92*

4) Create two optimized versions of program 3 (where possible!)
   a. Using conditional execution.
   b. Using conditional execution in IT block.

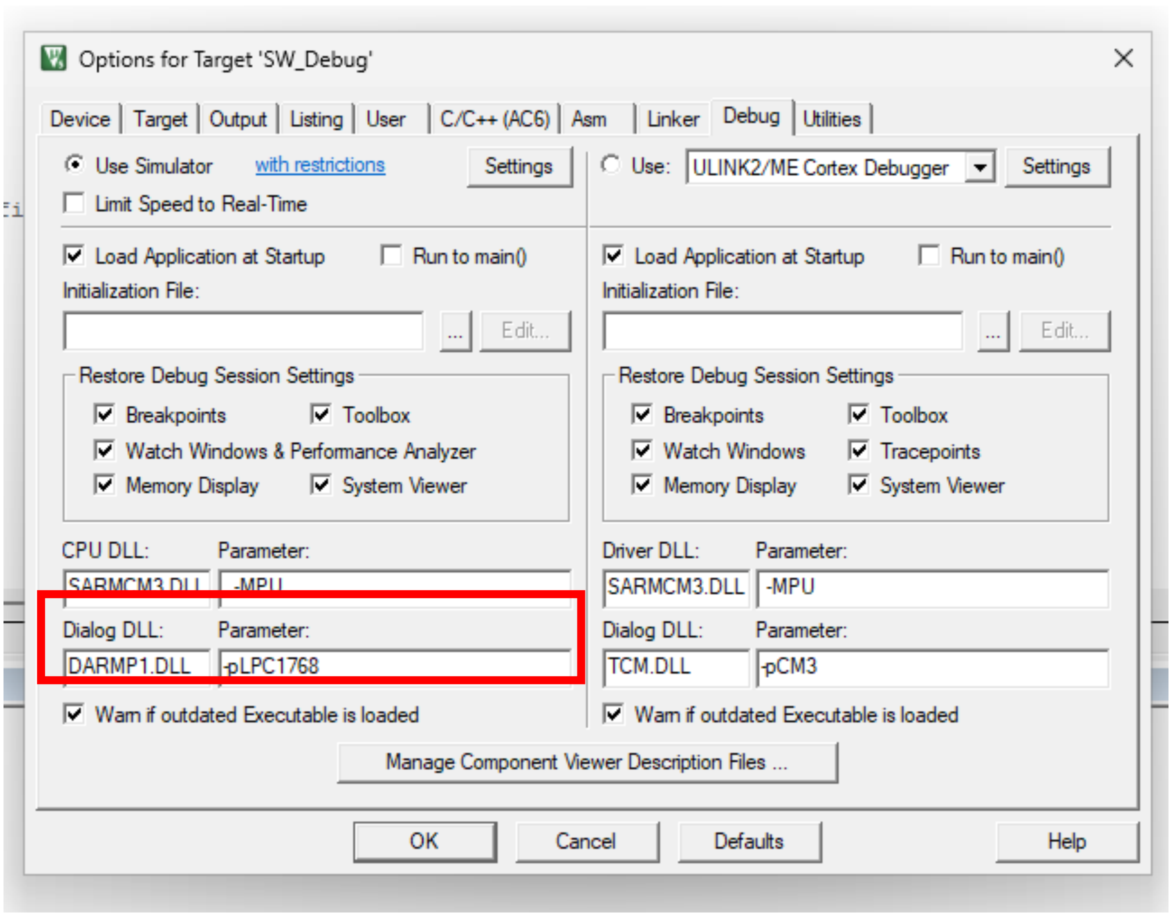
   Report and compare the execution Time

Program	Code size [Bytes]	Execution time [μs]	
		If the leading zeroes are even	Otherwise
Program 4 (baseline)	*40*	*4.33*	*4.92*
Program 4.a	*36*	*3.47*	*4.58*
Program 4.b	*34*	*2.93*	*3.40*

ANY USEFUL COMMENT YOU WOULD LIKE TO ADD ABOUT YOUR SOLUTION:
By progressively changing the implementation of the code, thus passing from conditional branches made with AND, CMP and BNE, to conditional branches with TST and BNE, and finally to a mix of this last solution and IT blocks, the code size decreased along with the execution time.
The last solution is the best I could find, since I put together both IT blocks and TST comparison: originally I substituted both comparisons + conditional branches with IT blocks, but then realised that the first block didn't make much sense, since with it 3 instructions were executed both if the condition was true or not and thus it was convenient to jump over all 3 of them if the condition was false.

How to set the CPU clock frequency in Keil

1) Verify that debug parameters ('Options for Target' window -> 'Debug' tab) are correctly set to match the target device (LPC1768). Modify the 'Dialog DLL' field to DARMP1.DLL and the 'Parameter' field to -pLPC1768, as shown in the image below


Options for Target 'SW_Debug'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

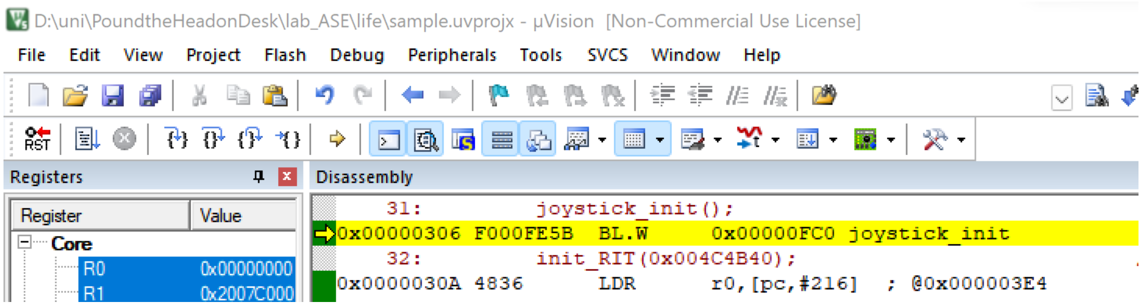
[✓ Use Simulator] with restrictions Settings
☐ Limit Speed to Real-Time
☑ Load Application at Startup ☐ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DARMP1.DLL Parameter: -pLPC1768
☑ Warn if outdated Executable is loaded

Use: ULINK2/ME Cortex Debugger Settings
☑ Load Application at Startup ☐ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows ☑ Tracepoints
☑ Memory Display ☑ System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM3
☑ Warn if outdated Executable is loaded

Manage Component Viewer Description Files ...

OK Cancel Defaults Help

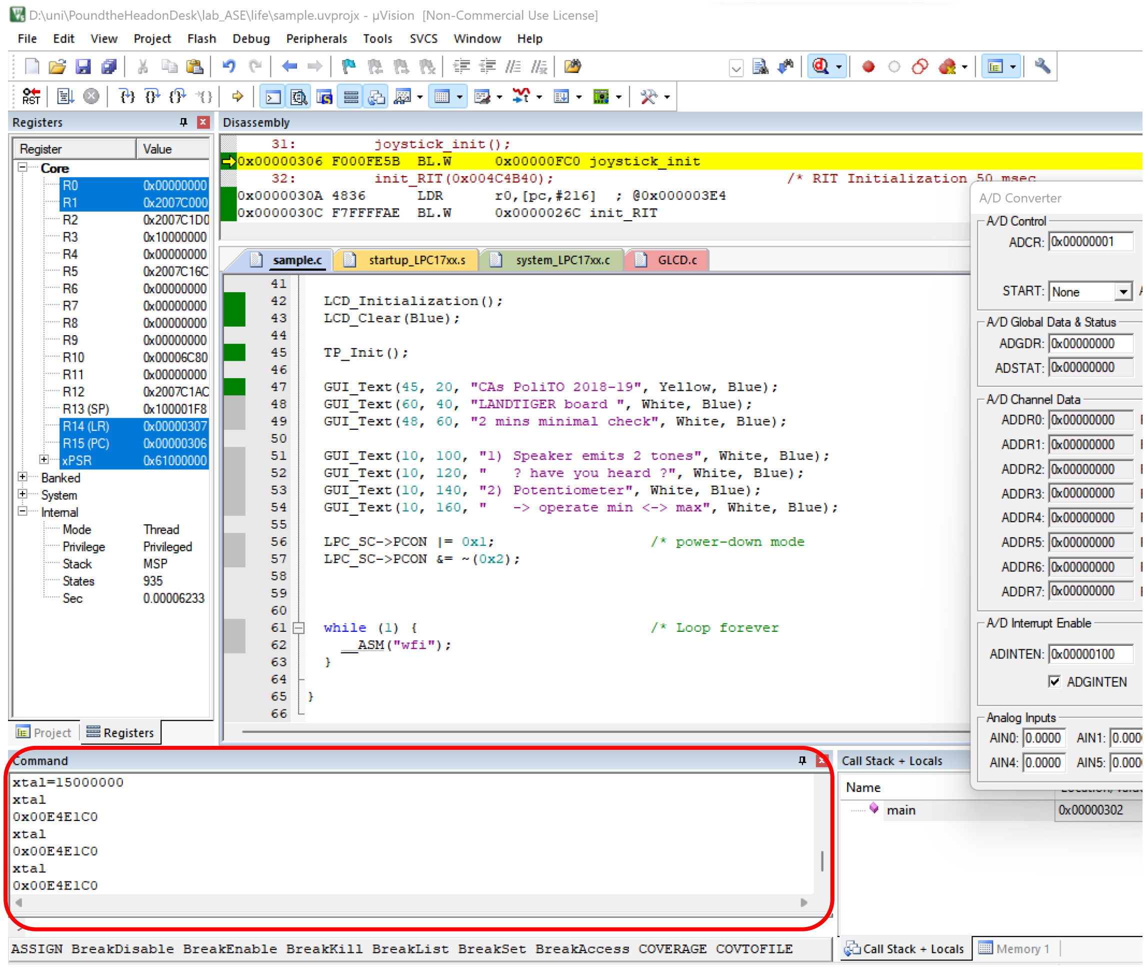
2) Launch the debug mode and activate the command console.


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

3) 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
Registers
Disassembly
sample.c
startup_LPC17xx.s
system_LPC17xx.c
GLCD.c
Project
Registers
Command
Call Stack + Locals
Memory 1

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 Converter

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

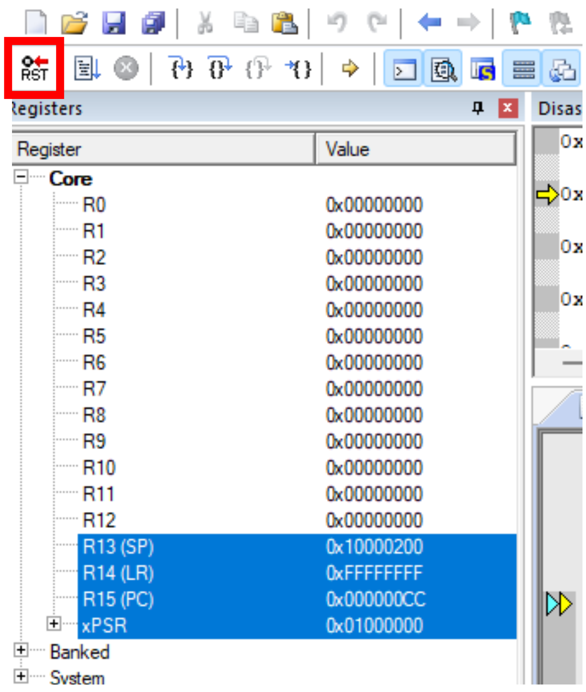
Name	
main	0x00000302

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

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

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*.

4) After having set the frequency, reset the CPU by clicking on the RST button in the top left, above the ‘Registers’ window. The debug session will restart and the simulation will be executed using the desired clock frequency



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x10000200
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x000000CC
xPSR	0x01000000
Banked	
System	