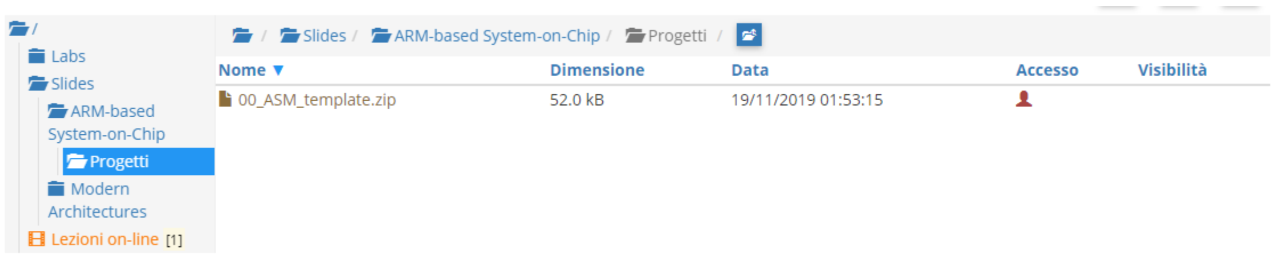
cdArchitetture dei Sistemi di Elaborazione 02GOLOV [AA-LZ] Laboratory 5	Delivery date: <u>Wednesday, November 27, 2019</u> Expected delivery of lab_05.zip must include: - Solutions of the exercises 1, 2 and 3 - this document compiled possibly in pdf format.

Solve the following problems by starting from the ASM_template project. Download from the Portale della Didattica.


/
Labs
Slides
ARM-based System-on-Chip
Progetti
Modern Architectures
Lezioni on-line [1]

/ Slides / ARM-based System-on-Chip / Progetti /

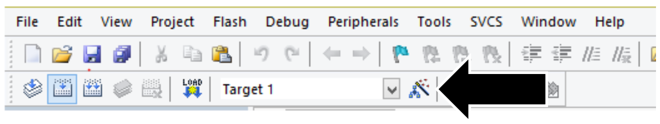
Nome ▼	Dimensione	Data	Accesso	Visibilità
00_ASM_template.zip	52.0 kB	19/11/2019 01:53:15		

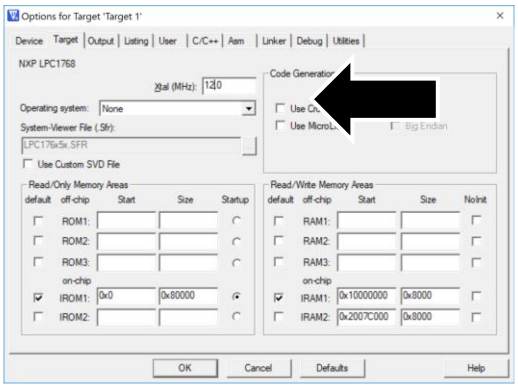
1) Write a program using the ARM assembly that makes the following simple operations:
   a. Sum R0 to R1 (R0+R1) and stores the result in R2
   b. Subtract R4 to R3 (R4-R3) and stores the result in R5
   c. Select and force, using debug register window, a set of specific values to be used in the program in order to provoke the following flag to be updated to 1
      - carry
      - overflow
      - negative
      - zero
   d. Report the selected values in the table below.

	Please, report the hexadecimal representation of the values			
Updated flag	R0	R1	R3	R4
Carry = 1	CD	FFFF FF33	0000000F	0000000E
Carry = 0	CD	00000001	0000000E	0000000F
Overflow	A1111111	B5555555	4AAAAAB	A1111111
Negative	CD	FFFF FF32	00000002	00000001
Zero	CD	FFFF FF33	00000002	00000002

2) Write two versions of a program that performs the following operations:
   a. Setup registers R0 and R1 to random signed values
   b. Compare the registers:
      - If they differ, find and 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

   Solve the problem by adopting a 1) conditional execution approach and compare the execution time with a 2) a traditional assembly programming approach. Report the execution time in the two cases in the table that follows: please, 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'*" .




Options for Target 'Target 1'

NXP LPC1768

Xtal (MHz): 12.0

Operating system: None

System-Viewer File (.Sfr): LPC176x5x.SFR

☐ Use Custom SVD File

Code Generation: ☐ Use Cr… ☐ Use MicroL… ☐ Big Endian

Read/Only Memory Areas

default		Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
✓	IROM1:	0x0	0x80000	●
☐	IROM2:			○

Read/Write Memory Areas

default		Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
✓	IRAM1:	0x10000000	0x8000	☐
☐	IRAM2:	0x2007C000	0x8000	☐

# cc	R0==R1	R0!=R1 && R0>R1	R0!=R1 && R0>R1
1) Conditional execution	13	15	15
2) Traditional	18	16	17

3) Write a program able to indicate whether a register contains a value that shows “even” or “odd” parity. In mathematics, an integer is “even” if it is evenly divisible by two, and “odd” if it is not even. In computer science, the parity concept is different and usually indicates whether the total number of 1-bits in the string is even or odd. For example, the number 4 is showing to be odd (0100 ← just a single 1-bit), while the number 5 is even (0101← two 1-bits).

   Please implement the ASM code that performs the following checks and actions:
   a. It determines if the register R0 and R1 are showing the same c.s. parity,
   b. As a result, the value of R0 and R1 is updated as following
      - If R0 and R1 are both even or odd, then it clears the 16 MSB of R0 and sets to 1s the 8LSB of R1 (all other bits must remain unchanged)
      - Otherwise, it copies in R1 the values of the flags.
   c. Report code size and execution time (with 12MHz cclk) in the following table.

		Execution time [*replace this with the proper time measurement unit*]	
	Code size	if both Odd or Even	Otherwise
Exercize 3) computation	564	0.000015	0.000010

ANY USEFUL COMMENT YOU WOULD LIKE TO ADD ABOUT YOUR SOLUTION:
Execution time depends on values contained in registers R0 and R1. The reason for this is that the cycle counting the numbers of bit set to 1 in the register stops as soon as the value of the register reaches 0. In the worst case (each register set to FFFFFFFF) there will bi 32x2 cycles.
In this case the values were 000000CD for both register to make them both odd and R0=000000CD, r1=00000002 to make them have different parity