Computer Architectures

Programming part T1.2 – February 28, 2019

PLEASE FILL THIS FORM

Student name_____

ID_____ Signature_____

Time the solution was delivered: ___:___

Code compiles: yes [] no []

Code works: yes [] no [] partly []

Please read accurately:

1) The ARM programming part of the exam has a duration of 2 hours
2) You have to develop an ARM project using the KEIL µVision IDE
3) Login in your LABINF area and use the available installation (v4.74) to edit, compile and SW debug your code
4) Use the provided LANDTIGER board and HW debugger to prototype your project
5) You are allowed to access the teaching portal page; this access will be granted by the LABINF infrastructure and any other web page access will be denied and all attempts will provoke the immediate ejection from the exam: LABINF personnel will monitor the network usage along the exam.
6) You can bring a single USB key and use your personal projects and notes.
7) Before the exam time ends you MUST upload a zipped folder of the developed project called 20190130.zip of your project including your project in the "elaborates" section of your Computer Architecture account, in the POLITO teaching portal. Late delivery will not be considered valid and always lead rejection.
8) The professors will reject delivered projects that produce errors during the compiling phase; make sure your project compilation is free of errors.

Exercise 1 (max 30 points)

You are required to implement the following functionalities on the LANDTIGER board equipped with the LPC1768 chip.

1) To reset TIMER1 and initialize if to count up to 3.564 seconds; all LEDs switched OFF.
2) To use button KEY2 to implement the following sequence of operations
   - At the first pressure, it enables TIMER0 to count
   - At the second pressure,
     - It stops TIMER0 and capture the current Timer Counter value and in a VAR variable
     - It launches the execution of the Assembly function described below.
   - If the timer count expires before the second pressure, the counter is reset to 0 and left enabled, as in a cyclical behaviour
3) The ARM assembly language written function, which prototype is:

   ```
   int search_in_pool (unsigned int VAR);
   ```

   - Receives as input the VAR value previously captured in point 2)
   - Internally defines a literal pool composed of N (with N defined constant) elements
   - Searches VAR in the literal pool
     - Returns the value -1 if the value is not included in the literal pool
     - Returns the position of the first element of the pool matching with VAR

Examples:
- if the captured VAR value is 0x78 and the pool contains {0x67, 0x86, 0x13}, the function returns -1;
- if the captured VAR value is 0x86 and the pool contains {0x67, 0x86, 0x13}, the function returns 1;

4) Once the ASM function is executed, the board LEDs LD04-11 (according to schematic names) have to be used to display the result. In particular,
   - the returned 32 bit value is displayed 8 bits at a time, starting from the LSB to the MSB
   - every byte of the returned value is shown on LEDs for 1 seconds by using TIMER3
   - following LSB display, the LEDs are switched all OFF.
5) The system has to show a cyclical behaviour and restarts from point 1) when 4) is completed.