Computer Architectures

Programming part T2.1 - January 30, 2019

PLEASE FILL THIS FORM

Student name________________________________

ID___________ Signature_____________________

Solution delivered in time: yes [] no[]

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 compile 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) The joystick is used to build an signed value to be stored in an variable called VAR1; in particular:
   - UP adds 12_{10} to VAR1
   - DOWN subtracts 8_{10} to VAR1
   - SELECT is used to signal that the value in VAR1 is ready

2) The system has to record N values and, every time a value is ready in VAR1, have to implement the following operations
   - The acquired value is copied to the first free position of a vector called VETT, composed of N elements
   - The value of VAR1 is reset to 0 and restart behaving as in point 1)
   - When the vector VETT is full (N values are acquired),

     1. The following function, written in ASSEMBLY language, needs to be invoked

     ```
     int count_negative_and_odd(int* VETT, unsigned int n);
     /* where n is the number of VETT elements */
     ```

     which returns the number of values stored in VETT that are negative and odd contemporary

     2. The process of filling VETT is restarted from element 0.

3) LED configuration: every time the function `count_negative_and_odd` is executed, the following led configuration need to be reproduced according to the returned value
   - if 0, LD04 (according to schematic names) is repeatedly blinking with the following period

     1.3 second ON – 0.9 seconds OFF
   - otherwise, the binary representation of the returned value has to be displayed on the remaining 7 leds LD05 – LD11 while LD04 is off.