Computer Architectures

PLEASE FILL THIS FORM

Student name\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ID\_\_\_\_\_\_\_\_\_\_\_ Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Time the solution was delivered: \_\_\_:\_\_\_

Code compiles: yes [] no []

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

Programming part T1.1 – September 4, 2019

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 20190904.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 TIMER0, initialize it to count up to 0.231 seconds and enable it to count. The timer has to reset and restart when the timing interval elapses; the interrupt functionalities are not enabled.
2. To use button KEY1 to implement the following sequence of operations
   * At every pressure,
     + stop TIMER0
     + capture the current Timer Counter value
     + put the captured value in the corresponding position of a integer vector called VETT (first value in position 0, second in 1, etc…)
     + resume the TIMER0 counting
   * Every N acquired values, the assembly function described below is called.
3. The ARM assembly language written function, which prototype is:

**int strictly\_monotone\_ascending (int VETT[], unsigned int dim);**

* + Receives as input the values VETT previously captured in point 2)
  + Analyses VETT to determine whether the collected sequence of values are strictly monotone ascending in order to
    - return the value 0xFFFFFFFF if the VETT is monotone, or
    - return the maximum value in case VETT is not monotone

1. 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,
   * All leds on if monotone
   * The LSB of the maximum value if not monotone.
2. The system has to show a cyclical behaviour and restarts from point 1) when 4) is completed.