EMBEDDED SYSTEMS 2016/2017 [9 credits course]

8 T/F questions: 0.5 points for right answers, -0.25 points for wrong ones, and 0 for no answer
3 open questions: up to 2 points for right and complete answers
1 C4uC exercise: up to 12 points for right and complete solutions

Available time: 150 minutes

Name:_____

TRUE or FALSE?

1) An embedded system is always designed by using a *Microcontroller*.

2) To exploit a *VLIW Processor* it is needed a specific compiler.

3) A *Single Purpose Processor* doesn't exploit an *Instruction Set Architecture*.

4) The *Intel 8051* model used in the homelab has only 2 I/O ports.

5) SPI is a multi-master protocol.

6) It is possible to use floating-point arithmetic also on processors without a *Floating Point Unit*.

7) It is possible to transform a FSMD into an equivalent *Combinatorial Network*.

8) The *Instruction Set Architecture* of a *Superscalar* Processor is normally customizable.

OPEN QUESTIONS

1) Describe the problems related to interrupt handling mechanism in a real-time kernel and expose three possible approach to avoid this problems.

2) Given this periodic task set:

	C_i	T_i
τ_1	3	5
τ_2	1	6
τ_3	1	6

Verify the schedulability and draw the schedule according to the EDF algorithm, if possible

3) Given the code from application BlinkApp:

```
configuration BlinkAppC { }
implementation {
components MainC, BlinkC, LedsC;
components new TimerMilliC() as Timer0;
components new TimerMilliC() as Timer1;
components new TimerMilliC() as Timer2;

BlinkC -> MainC.Boot;
BlinkC.Timer0 -> Timer0;
BlinkC.Timer1 -> Timer1;
BlinkC.Timer2 -> Timer2;
BlinkC.Leds -> LedsC;
}
```

draw the component graph of the components, specifying the interfaces on the links.

C4uC (with reference to *DidArch_v1.1*)

[up to 2 point]
Develop a C program that:

- continuously receives sequences of natural numbers from the Input Device
    - discards values out of [1, 100] range and writes the others ASAP to the Output Device
    - such a sequence is terminated by any odd number that shall not be written
    - the minimum delay between inputs is at least 100 ms

[up to 4 points]
Starting from the previous C program, develop a modification/extension that:

- for each sequence
    - find the min value (among the <u>not discarded</u> sequence values <u>and also the terminator value</u>) and writes it on P0 at the end of the sequence itself
        - P0 should be normally set to 0x00
    - evaluate the average (among the <u>not discarded</u> sequence values <u>and also the terminator value</u>) and writes it on P1 at the end of the sequence itself
        - P1 should be normally set to 0x00
        - define how to manage possible overflows

[up to 6 points]
Starting from the previous C program, develop a modification/extension that:

- if the delay between <u>two consecutive not discarded values</u> is greater than 500 ms then write 0xFF on the Output Device, P0 and P1 instead of the previous requested values
    - define how to manage possible overflows on time representation