EMBEDDED SYSTEMS 2015/2016 [6/9 credits courses]

10 T/F questions: 0.5 points for right answers, -0.25 points for wrong ones, and 0 for no answer
5 open questions: up to 2 points for right and complete answers
Available time: 90 minutes

TRUE or FALSE?

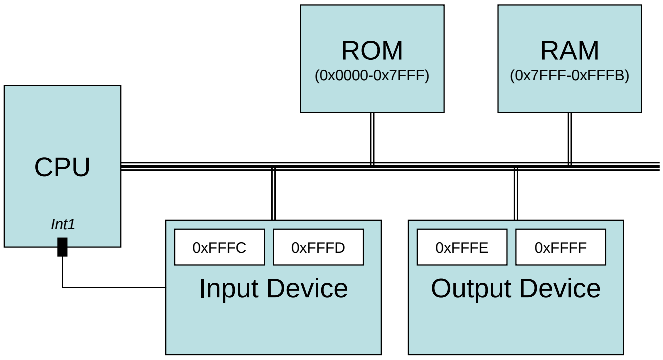
1) Embedded systems development is a discipline with mature and general design methodologies.

2) In the EDA domain, in general, a *synthesis* transforms a *structural* description into a *behavioral* description.

3) ISA of a GPP is normally customizable.

4) A *scratch pad* memory is “invisible” to the compiler (as a cache memory).

5) Given the reference HW architecture below:



with *fixed-interrupt management* it is possible to manage only one ISR.

6) Talking about *Timers* implementation techniques, those based on *Calendar/Clock* circuits are the best ones for hard real-time systems.

7) It is possible to use *floating-point arithmetic* also on processors without a FPU.

8) Given the following *nesC* code related to an event management:

```
event void myTimer.fired()
{
        counter++;
        call Leds.Led0Toggle();

        if (!busy)
        {
                BlinkToRadioMsg* btrpkt = (BlinkToRadioMsg*)(call Packet.getPayload(&pkt, NULL));
                btrpkt->nodeid = TOS_NODE_ID;
                btrpkt->counter = counter;
                if (call AMSend.send(AM_BROADCAST_ADDR, &pkt, sizeof(BlinkToRadioMsg)) == SUCCESS)
                {
                        busy = TRUE;
                }
        }
  }
```

it belongs to a *configuration* component

9) A necessary condition for the re-entrance of a C function is that the code shall call only re-entrant functions.

10) With respect to the following *SystemC* code:

```
SC_MODULE(encode){
       sc_in<bool> clock; //Ports
       sc_in< bool > reset;
       sc_in< bool > input;
       sc_out< sc_bv<3> > output;

       sc_bv<8> trell; //Variables
       sc_bv<3> tmp;
       sc_bv<8> input1;

       void codeGen(); //Function prototipe

       SC_CTOR(encode) // Constructor
       {
              SC_CTHREAD(codeGen,clock.pos());
              watching(reset.delayed() == true);
       }
};

void encode::codeGen(
{
       trell=0x00; //Init
       wait();
       //PROCEDURE
        while(true)
       {
              input1[0]=input.read();
              trell=((trell)<<1)|input1;
              tmp[2]=trell[7]^trell[4]^trell[2]^trell[0];
              output.write(tmp); //output write
              wait();
       }
}
```

encode module is a C++ class.

OPEN QUESTIONS (answers shall not be longer than 1 page)

11) What is *Time-to-Market*?

12) What are the most common *Non-Volatile* memory families?

13) What are the differences between *active* and *passive* roles of the time in an embedded system?

14) Briefly describe the output of the following nesC program:

```
configuration TestAppC
{
}
implementation
{
  components MainC, TestC, LedsC;
  components new TimerMilliC() as Timer;

  TestC -> MainC.Boot;
  TestC.Timer0 -> Timer0;
  TestC.Leds -> LedsC;
}

module TestC
{
  uses interface Timer<TMilli> as Timer;
  uses interface Leds;
  uses interface Boot;
}
implementation
{
  uint_8t temp;

  event void Boot.booted()
  {
    temp=0;
    call Timer.startPeriodic( 250 );
  }

  event void Timer.fired()
  {
    if (temp%3==0)
      call Leds.led0Toggle();
    else if (temp%3==1)
      call Leds.led1Toggle();
    else call Leds.led2Toggle();

    temp++;
  }
}
```

15) Describe the main feature of *Kahn-Process Networks* model of computation.