Embedded Systems Programming
Written Exam

2008-12-20
from 09.00 to 13:00

- **Allowed tools:** An english dictionary.
- **Grading criteria:** You can get at most 20 points.

  To pass you need at least 50% of the points.

  For the highest grade you need more than 90% of the points.
- **Responsible:** Verónica Gaspes, telephone 7380.

- **Read carefully!** Some exercises might include explanations, hints and/or some code. What you have to do in each exercise is marked with the points that you can get for solving it (as **(X pts.)**).
- **Write clearly!**
- **Motivate your answers!**

Good Luck!

1. **(2 pts.)** Write functions that test whether certain bits are set resp. clear in a register:

```
int areSet(unsigned int *port, unsigned int bits)
int areClear(unsigned int *port, unsigned int bits)
```

2. **(2 pts.)** Explain and exemplify how to organize a program fragment that detects an external event using *busy waiting* and using *interrupts*.

3. **(3 pts.)** In the following fragment from laboration 2 there is a critical section. Identify it, explain what may happen, give an example of incorrect behaviour and solve the problem using a mutex.

```
#include "tinythreads.h"

int pp;

void writeChar(char ch, int pos); // defined elsewhere

int is_prime(long i);  // defined elsewhere

void printAt(long num, int pos) {
  pp = pos;
  writeChar( (num % 100) / 10 + '0', pp);
  pp++;
  writeChar( num % 10 + '0', pp);
}

void computePrimes(int pos) {
   long n;
   for(n = 1; ; n++) {
       if (is_prime(n)) printAt(n, pos);
    }
}

int main() {
  spawn(computePrimes, 0);
  computePrimes(3);
}
```

4. The following is the definition of a reactive object that can be used to produce a sequence of prime numbers.

```
#include "tinytimber.h"

typedef struct {
  Object super;
  int lastPrime;
} PrimeCalculator;

#define initPrimeCalculator() {initObject(),2}

int is_prime(int i){
  int n;
  if(i==0 || i==1) return 0;
  for(n=2; n<i; n++){
    if ((i%n)==0) return 0;
  }
  return 1;
}

int next(PrimeCalculator *self, int x) {
  int returnValue = self -> lastPrime++;
  while(!is_prime(self->lastPrime))self->lastPrime++;
  return returnValue;
}
```

(a) **(2 pts.)** Compare what happens if the function `next` is called as a C function, if it is used via `SYNC` and via `ASYNC`.

(b) **(2 pts.)** Write a little TinyTimber application that uses a `PrimeCalculator` to print prime numbers on some device when a user presses a button. Provide the interfaces for the reactive objects that you need, but you do not need to provide implementations!

5. **(3 pts.)** In laboration 4 you used a reactive object to implement a *blinker* that turns on and off a LED. The blinker can be started, stopped and the period can be set. Assuming that you have a class for reactive LEDs, implement a blinker. The blinker should not be hard-coupled to a LED, instead, it should be possible to instantiate blinkers for different LEDs.

6. **(4 pts.)** Say that three periodic tasks A, B and C have periods 20, 30, 40 (and deadline equal to the period). Make 2 diferent assumptions about their utilization and draw the timelines for RM and EDF scheduling for each of these.

7. **(2 pts.)** Write the prime calculator from exercise 4 as an Ada task with an entry `next`.