Embedded Systems Programming
Written Exam

2009-12-21
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, 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. **(1 pts.)** Write a function

   ```
   void toggleBit(unsigned int *port, int bitNr)
   ```

   to change the value of a given bit in a port (if the bit is set it has to be cleared and if it is clear it has to be set). The bit that has to be set or made clear in the port is given by the second argument (`bitNr`) to the function. Bit number 0 is the least significant bit.

2. Consider the example from laboration 1 in which you had to put together three programs into one without any support for concurrency. The individual programs were the following:

   - print prime numbers on the LCD,
   - blink with some symbol in the LCD with a frequency of 1Hz and
   - toggle some other symbol in the LCD in response to input from the joystick.

   Assume that you have the following functions to deal with the LCD:

   ```
   int writeLong(long n); // to print an number on the LCD
   int turnOnS1();  // to turn on one symbol on the LCD
   int turnOffS1(); // to turn off this symbol on the LCD
   int turnOnS2();  // to turn on another symbol on the LCD
   int turnOffS2(); // to turn off this other symbol on the LCD
   ```

   For the separate programs you might have programmed busy-waiting fragments to detect that enough time has passed (for blinking) and to detect input on the joystick:

   ```
   int wait500ms(){
      while(TCNT1 < 15625);
   }

   int waitForJoystick(){
      while(PINB & 0x80);
   }
   ```

   In this exercise you should:

   (a) **(2 pts.)** Program the function for computing primes and printing them on the LCD (print them using `writeLong` above, you do NOT need to define `writeLong`!).

(b) **(2 pts.)** Write the complete program (using the cyclic executive and interleaving by hand) that prints primes, blinks and responds to joystick input.

3. In laboration 2 you worked with a kernel that supports threads. The primitive `spawn` can be used to set up a new thread to execute the body of a function. Usage of the processor is scheduled using *time slicing*. For doing so the kernel keeps a queue of thread blocks that are ready to run (called the `readyQ`). The thread block for the running thread is kept in `current`. In order to allow threads to synchronize (for example to avoid race conditions in the use of shared variables) the kernel implements *mutexes*. Here is the interface for mutexes:

```
typedef struct{
  int locked;
  thread waitQ;
} mutex;

#define MUTEX_INIT {0,0}
void lock(mutex *m);
void unlock(mutex *m);
```

(a) **(2 pts.)** Implement the function `lock`. The purpose of `lock` is to lock the mutex if it is not yet locked **or** to block the current thread in the wait queue of the mutex if the mutex is already locked. Another thread has to be given the processor in the latter case (the function for doing so is `dispatch`).

(b) **(2 pts.)** Implement the function `unlock`. The purpose of `unlock` is to unlock the mutex if there are no thread blocks in the wait queue of the mutex **or** to activate a waiting thread otherwise.

Note The operations for enqueuing and dequeuing are

```
static void enqueue(thread p, thread *queue)
static thread dequeue(thread *queue)
```

4. Using TinyTimber you can organize programs with *reactive objects* while programming in C. As a programmer you have to follow some conventions and TinyTimber guarantees that the methods of a reactive object are executed strictly sequentialy, thus protecting the local state of the object from critical section problems. Here is a rective object implementing a counter:

```
typedef struct {
  Object super;
  int value;
} Counter;

#define initCounter(){initObject(),0}

int inc(Counter * self, int x){
  self -> value++;
}

int reset(Counter * self, int x){
  self -> value = 0;
}

int readValue(Counter * self, int x){
  return self -> value;
}
```

Imagine a program where several objects share the same counter, for example

```
#include "TinyTimber.h"
#include "counter.h"
#include "o1.h"
#include "o2.h"
#include "lcd.h"

Counter c  = initCounter();
Lcd lcd    = initLcd();
O1 client1 = initO1(&c);
O1 client2 = initO1(&c);
O2 reader  = initO2(&c, &lcd);

INTERRUPT(...)
INTERRUPT(...)
STARTUP(CONFLCD;ASYNC(&c,reset,0);...)
```

```
typedef struct {
  Object super;
  Counter *cnt;
} O1;
...

typedef struct {
  Object super;
  Counter *cnt;
  Lcd *lcd;
} O2;
...
```

(a) **(1 pts.)** In the class `O1` a method needs to increment the counter. Say how this should be done and motivate your answer.

(b) **(1 pts.)** In the class `O2` a method needs to read the value of the counter. Show how this should be done and motivate your answer.

(c) **(1 pts.)** What are the problems of using `inc(self->cnt,0)` in a method of `O1`?

5. In this exercise you will write a reactive object for controlling a moving object. There are device drivers for controlling the movement by giving the object a speed (speed 0 when the object is at rest) and for reading a distance using a sensor

```
typedef struct {
  Object super;
} Mover;

#define initMover(){initObject()}

int setSpeed(Mover * self, int speed){
  // give speed the device
}
```

```
typedef struct {
  Object super;
} DistanceSensor;

#define initDistanceSensor(){initObject()}

int getDistance(DistanceSensor * self, int x){

  // read a value from the
  // register associated to the sensor
  // and convert it into centimeters.
  // return this value
}
```

(2 pts.) Program a reactive object that controls the moving object so that it moves forward with a speed of `SPEED`. If an obstacle is detected closer to `MINDIST`cm the object should stop. If the obstacle is removed the object should continue moving forward. For the given speed and distance to obstacle it is enough to check the value in the sensor every `POLLING_TIME` milliseconds.

6. Given a task set with three periodic tasks A, B and C that have periods 50, 40, 30 (and deadline equal to the period).

   (a) **(1 pts.)** What does it mean that A has utilization 24%, B has utilization 25% and C 33%?

   (b) **(1 pts.)** Given these utilizations, what is the total utilization?

   (c) **(2 pts.)** Draw the timeline that results from scheduling the task set according to rate monotonic (RM) and say whether the task set is schedulable.

   (d) **(2 pts.)** Draw the timeline that results from scheduling according to earliest deadline first (EDF) and say whether the task set is schedulable.