Verónica Gaspes
School of IDE

Embedded Systems Programming

Written Exam

April 14th, 2012, from 09.00 to 13.00

- **Allowed tools:** An English dictionary (a paper such, not an electronic one).
- **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.
- **On duty**: Essayas Gebrewahid (Tel: 167327).

- **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!

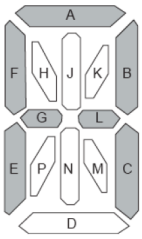

Figure 1: Digit segments in the LCD

Register Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDRx	K	–	–	A	K	-	-	A
LCDDRx+5	J	F	H	B	J	F	H	B
LCDDRx+10	L	E	G	C	L	E	G	C
LCDDRx+15	M	P	N	D	M	P	N	D

Figure 2: LCD data registers

1. In the first three laborations of this course we used a demonstration platform that includes an LCD. The LCD is controlled via the registers `LCDDR0` to `LCDDR19`, each of them one byte wide. Figure 1 shows what items can be turned on/off in each digit while Figure 2 shows what bits of the registers are used to control each of the items. In this exercise we are interested in the fact that four registers are used to control 2 positions.

   (a) **(1 pts.)** What patterns appear in the positions controlled by registers `LCDDR0`, `LCDDR5`, `LCDDR10` and `LCDDR15` if the registers have the following values:

LCDDR0	0x10
LCDDR5	0x41
LCDDR10	0xB1
LCDDR15	0x10

   (b) **(1 pts.)** What patterns appear after the following assignments:

   ```
   LCDDR0  = LCDDR0  | 0x1
   LCDDR5  = LCDDR5  | 0x1
   LCDDR10 = LCDDR10 | 0xE
   LCDDR15 = LCDDR15 | 0x1
   ```

   (c) **(2 pts.)** What patterns appear after the following assignments:

   ```
   LCDDR0  = LCDDR0  & 0x60 | 0x1
   LCDDR5  = LCDDR5  & 0xF0 | 0x1
   LCDDR10 = LCDDR10 & 0xF0 | 0xE
   LCDDR15 = LCDDR15 & 0xF0 | 0x1
   ```

2. Consider the following fragment we discussed in one of the lectures:

```
int old = KEY_STATUS_REG;
int val = old;
while(old==val){
   val = KEY_STATUS_REG;
}
```

   (a) **(2 pts.)** What happens if `KEY_STATUS_REG` is an ordinary variable? What happens if it is an IO register?

   (b) **(2 pts.)** Discuss what might happen in a program that has to test for status changes in several ports and uses fragments like this for doing so.

3. In laboration 2 we programmed with a kernel (tinythreads) that supports threads. Using the kernel function

   ```
   void spawn(void (* function)(int), int argument)
   ```

   a program can start a new thread to execute a call to a `function` with an integer `argument`. The different threads are interleaved automatically by the kernel that calls `yield()` at regular intervals (we call this time slicing). With small enough intervals the program seems to be doing several things at the same time (concurrently).

   The following program uses this kernel:

```
#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);
}
```

(a) **(1 pts.)** What are the global variables in the program? What functions use these variables?

(b) **(1 pts.)** Are these functions executed in different threads? If so, give an example of an interleaving that might produce an erroneous result.

(c) **(2 pts.)** What is the mechanism that tinythreads provides to enforce mutual exclusion? Show how to use it in the program above.

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 sequentially, thus protecting the local state of the object from critical section problems.

   (3 pts.) Program a class for reactive objects that can be used to *protect* (or encapsulate) a port. The port (a pointer to unsigned int) to be encapsulated can be provided on object initialization. Let the type introduced for the class be `Port`. Then, the methods that have to be provided are

```
// set the bits given by the argument mask
int set(Port *self, unsigned int mask)

// clear the bits given by the argument mask
int clear(Port *self, unsigned int mask)

// toggle the bit in position given by argument bitNr
int toggle(Port *self, int bitNr)
```

5. **(3 pts.)** Implement a class for reactive objects that provides methods `turnOn`, `turnOff` and `setPhase`. When it is turned on it

   - calls a function `doA` every T milliseconds (periodically)
   - calls a function `doB` every T milliseconds (periodically), but delayed D milliseconds relative to the activation of `doA` (D is the *phase*)

   The phase can be changed using the method `setPhase`.

6. In Android an app is organized using Activities, Services, ContentProviders, Notifications and other components.

   (a) **(1 pts.)** How do the components exchange data?

   (b) **(1 pts.)** What can notifications be useful for?