Problema 1: Reloj de tiempo real

Se propone trabajar con el reloj en tiempo real con el que está equipada la NDS, para mostrar la fecha/hora actual por pantalla y activar una alarma cuando se llegue a un tiempo especificado.

Los valores de tiempo se definen con un vector de 6 números (bytes), con el siguiente contenido:

Posición	*Campo*	*Rangos*
0	Año	número del 0 al 99 (de 2000 a 2099)
1	Mes	número del 1 al 12 (de enero a diciembre)
2	Día	número del 1 al 31 (según el mes)
3	Hora	número del 0 al 23 (en modo 24 horas)
4	Minuto	número del 0 al 59
5	Segundo	número del 0 al 59

Se dispone de las siguientes rutinas, ya implementadas:

Rutina	*Descripción*
`inicializaciones()`	Realiza inicializaciones del *hardware*
`tareas_independientes()`	Tareas que no dependen de la alarma (ej. captación del movimiento del usuario)
`swiWaitForVBlank()`	Espera retroceso vertical
`mostrar_tiempo(char *tiempo)`	escribe en pantalla el tiempo que se pasa por parámetro
`detectar_alarma(char *t, char *a)`	si el tiempo `t` coinciden con el tiempo de alarma `a`, se activa un proceso de alarma, la ejecución del cual es (aprox.) de 50 milisegundos

Además, hay que realizar la captura del tiempo real por interrupciones. Como el reloj en tiempo real NO genera interrupciones, habrá que utilizar las interrupciones del *timer* 0.

Se dispone de una rutina ya implementada, de nombre `inicializar_timer0()`, que programa la interrupción IRQ_TIMER0 con una frecuencia ligeramente superior a 1 Hz (para evitar perder segundos).

También disponemos de las siguientes rutinas para comunicarnos con el reloj de tiempo real:

Rutina	*Descripción*
`iniciar_RTC()`	Activa el *chip select* del reloj en tiempo real
`enviar_RTC(byte comando)`	Envía un comando al reloj en tiempo real; concretamente hay que enviar el valor `0x26`

`byte recibir_RTC()`	Recibe un byte de datos del reloj en tiempo real; después de enviar el comando `0x26` se recibirán 7 bytes consecutivos con la siguiente información:
	`Year Register: BCD 00h..99h`
	`Month Register: BCD 01h..12h`
	`Day Register: BCD 01h..31h`
	`Day of Week Register: 00h..06h`
	`Hour Register: BCD 00h..23h`
	`Minute Register: BCD 00h..59h`
	`Second Register: BCD 00h..59h`

`parar_RTC()`	Desactiva el *chip select* del reloj en tiempo real

El protocolo de comunicación es el siguiente:

- iniciar RTC
- enviar comando
- recibir, transformar y almacenar los bytes necesarios
- parar RTC

El total de tiempo para realizar esta comunicación supera los 500 microsegundos, por lo tanto, no se aconseja realizarla dentro de una RSI.

Todo el protocolo de comunicación se encapsulará dentro de una rutina de nombre `capturar_tiempo(char *tiempo)`, la cual guardará la información del tiempo real dentro del vector que se pasa por parámetro (por referencia).

En el contexto del problema, la codificación BCD (Binary Coded Decimal) son números decimales de 2 dígitos codificados dentro de un único byte, en el cual se guardan las unidades en los 4 bits de menos peso y las decenas en los 4 bits de más peso. Por ejemplo, el número en BCD `0x39` (0011 1001) representa 3 decenas y 9 unidades, o sea, el valor decimal 39.

Se pide:

Programa principal en C, RSI del *timer* 0 y rutina `capturar_tiempo()` en ensamblador.