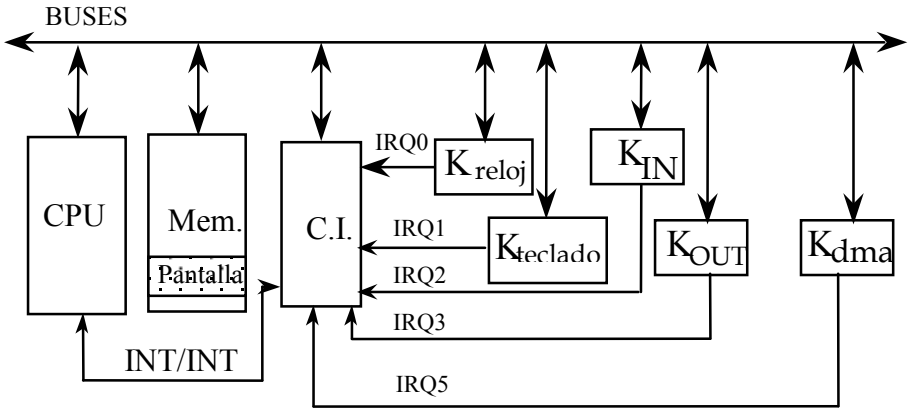
1.- Se quiere realizar de manera automática el control de vehículos en un **parking** céntrico de la ciudad. La estructura hardware del sistema es la que puedes ver en la siguiente figura.



Los controladores de los periféricos que debe controlar el microprocesador son idénticos a los vistos en clase, exceptuando los siguientes:

*** K_{IN}** : es el controlador de un botón situado a la entrada del parking, que es pulsado por el conductor del vehículo que quiere entrar. Dicha pulsación genera una petición de interrupción por la correspondiente línea IRQ (ver figura) proporcionando en su ***registro de datos*** (RDAT_KIN) la matrícula del vehículo que se encuentra a la entrada del parking. Cada vez que se acepta una interrupción es necesario realizar una secuencia de STROBE en su ***registro de control*** (RCON_KIN).

*** K_{OUT}**: se trata de un lector de tarjetas. Cuando una tarjeta es introducida genera una petición de interrupción por la correspondiente línea IRQ y proporciona en ***dos registros de datos*** (RDAT1_KOUT y RDAT2_KOUT) la matrícula del vehículo y la hora a la que entró respectivamente. Cada vez que se acepta una interrupción es necesario realizar una secuencia de STROBE en su ***registro de control*** (RCON_KOUT).

*** K_{DMA}**: es el controlador de DMA y se utiliza para hacer transferencias rápidas de memoria a memoria. Posee los siguientes registros:

- ***registro de control*** (RCON_KDMA): cuando se escribe un 1 en él se inicia la transferencia (se pone a 0 automáticamente).
- ***registro de dirección origen*** (ROR_KDMA): dirección de comienzo del bloque a transferir.
- ***registro de dirección destino*** (RDE_KDMA): dirección de destino a la que se transfiere el bloque.
- ***registro de estado*** (REST_KDMA): cuando su contenido es un 1 indica que la última operación realizada ha sido errónea.
- ***registro de longitud*** (RLON_KDMA): contiene la longitud del bloque a transferir.

*** PANTALLA:** está mapeada en memoria, tiene 25 líneas x 80 columnas, los caracteres se escriben sin atributo, y la dirección de comienzo está en la variable global DIRPAN.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase y en este sistema **la sincronización del teclado se va a realizar por encuesta.**

El **funcionamiento** del sistema debe ser el siguiente. Cuando un vehículo se dispone a entrar en el parking, su conductor pulsará el correspondiente pulsador situado a la entrada. Como resultado se debe proporcionar al vehículo la correspondiente tarjeta de entrada en la que figurarán tanto la matrícula como la hora de entrada. Para ello suponemos ya implementada la rutina **expedir_billete (horaent,matricula)** a la que hay que proporcionar los dos datos que indican los parámetros: hora de entrada y matrícula del vehículo. A continuación se ha de levantar la barrera de entrada para lo cual se tiene la rutina **levantar_barrera_entrada()**. Dicha barrera permanecerá levantada 3 segundos, tras los cuales se procederá a bajarla mediante la rutina **bajar_barrera_entrada()**. Si el número de vehículos en el parking llega al TOPE admitido, se deben inhibir las interrupciones del pulsador situado a la entrada del parking (volviendo a permitirlas cuando sea posible). En lo que a la hora se refiere, se dispone de la rutina **actualizar()** que actualiza la variable global HORA (consta de segundos, minutos y hora) en un segundo.

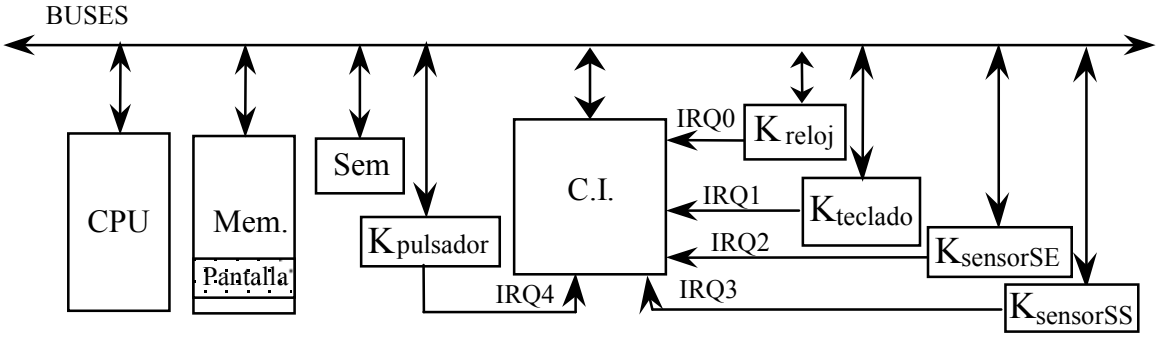
Cuando un vehículo se dispone a salir del parking, el cobrador situado a la salida introduce la tarjeta de entrada del vehículo en el correspondiente lector. El cálculo del importe debe ser automático y para realizarlo se cuenta con la

rutina ya implementada **calcular (horaent,horasal,&importe)** que además de calcular el importe que debe pagar el conductor, lo visualiza en una pequeña pantalla de varios dígitos y lo devuelve en la variable *importe*. Una vez realizado el cobro, el cobrador pulsará la tecla 'L' y se levantará la barrera de salida mediante la rutina **levantar_barrera_salida()**. Dicha barrera permanecerá levantada 5 segundos, tras los cuales se procederá a bajarla mediante la rutina **bajar_barrera_salida()**. Además, cada vez que un coche sale debemos escribir en la última línea de la pantalla del ordenador los datos correspondientes a ese vehículo: matrícula, horaent, horasal, importe para lo cual contamos con la rutina **escribir (matrícula,horaent,horasal,importe)** a la que se le proporciona toda la información necesaria. Tras escribir dicha información generaremos un pequeño pitido mediante la rutina **beep ()**. Queremos que en pantalla siempre estén visualizados los últimos 25 coches que han salido.

Se pide: Escribe en lenguaje algorítmico las rutinas de atención de K_{IN}, K_{OUT}, K_{DMA} y reloj y el programa principal del sistema. Indica qué elementos del vector de interrupciones hay que modificar.

----------o0000000o0----------

2.- Se dispone de un sistema de control de la presencia de **aves** en el área de despegue y aterrizaje de aviones en un aeropuerto.



La estructura hardware del sistema es similar a la vista en clase, como puedes ver en la figura, con la salvedad de que se han añadido 2 sensores (SE, SS) y un pulsador, cuyo funcionamiento se detallará posteriormente. Además, se ha de controlar 1 semáforo, cuyo controlador posee únicamente registro de control, mapeado en memoria, en la dirección SEM. Los periféricos que debe controlar el microprocesador son idénticos a los vistos en clase, exceptuando los dos sensores, el pulsador y el semáforo, cuyas características se exponen a continuación:

- *** Sensor SE**: Sensor de entrada de aves. Detecta cuándo entra algún ave (una o más) en la zona controlada y genera una interrupción. En su registro de datos se puede leer entonces el número de aves que han entrado en ese instante. Una vez leído el registro de datos es necesario hacer STROBE sobre su registro de control.
- *** Sensor SS**: Sensor de salida de aves. Detecta cuándo sale algún ave (una o más) de la zona controlada y genera una interrupción. En su registro de datos se puede leer entonces el número de aves que han salido en ese instante. Una vez leído el registro de datos es necesario hacer STROBE sobre su registro de control.
- *** Semáforo SEM**: Semáforo de aviso a los aviones. Puede estar **Verde**, en cuyo caso significa que no hay aves en la zona controlada y que el avión puede realizar la maniobra correspondiente, o **Rojo** en el caso contrario. Como ya se ha indicado con anterioridad, posee un registro de control mapeado en memoria.
- *** Pulsador**: Genera una interrupción al ser pulsado cada vez que el personal encargado de ahuyentar a las aves captura una de ellas.
- *** Reloj**: Interrumpe 18 veces por segundo.
- *** Teclado**: Cada vez que se pulsa una tecla produce dos interrupciones: la primera indica que se ha pulsado tecla (MAKE) y la segunda indica que se ha liberado (BREAK). Posee un registro de datos en el que se puede leer el código de posición correspondiente a la última tecla pulsada y un registro de control en el que es necesario escribir una secuencia de STROBE cada vez que se acepta una interrupción. No posee registro de estado.
- ***Controlador de interrupciones**: posee los registros: máscara IMR, de petición de interrupción IRR y de interrupción en servicio ISR.

El **funcionamiento** del sistema es el siguiente. La situación o **estado normal** es que no haya aves en la zona controlada y, por tanto, el semáforo estará **verde**. En cualquier otra situación, el semáforo estará rojo por precaución. En el momento en que se detecta la entrada de aves, el sistema pasará a un **estado de prealarma**, estado en el que permanecerá mientras haya aves en la zona controlada (por supuesto, las aves pueden salir "de motu propio") y el operador encargado del sistema no indique la urgencia de ahuyentar a las aves por la inminente maniobra de un avión. En este caso de urgencia, el operador pulsará la **tecla A**, lo que hará que el sistema pase a un **estado de alarma**.

En el estado de alarma, además de mantener el semáforo rojo, se genera una alarma sonora (suponer que tenemos la rutina ***genera_alarma()***, ya escrita) de aviso al personal encargado de ahuyentar a las aves. Esta "brigada de limpieza" debe atrapar a las aves presentes en la zona controlada con una red, apretando el pulsador tantas veces como aves atrapadas en dicha red. De nuevo, por supuesto, las aves pueden salir "de motu propio" ahuyentadas por la presencia humana. Cuando por fin la brigada ahuyentadora consiga que se haya vaciado de aves la zona controlada, el sistema pasa a un **estado de post-alarma**, en el cual se mantendrá durante **2 minutos**, por precaución. Si durante este tiempo se detecta de nuevo la entrada de aves, se vuelve al estado de alarma. Si no, pasado ese tiempo sin aves, el sistema vuelve al estado normal.

Se pide: escribir en lenguaje pseudo algorítmico las rutinas de servicio: sensor SE, sensor SS, pulsador, teclado y reloj

----------o0000000o----------

3.- En un pequeño pueblo, tienen un problemático **puente**: dada su estrechez dispone de un sólo carril, con lo cual la dirección de paso debe ser única en cada instante. Dicho puente se encuentra en una carretera de doble dirección, con lo cual, a ambos lados del puente han instalado dos semáforos, tal y como se muestra en la Figura 3a. Queremos que el control de estos semáforos se haga automáticamente. La estructura hardware se puede ver en la Figura 3b.

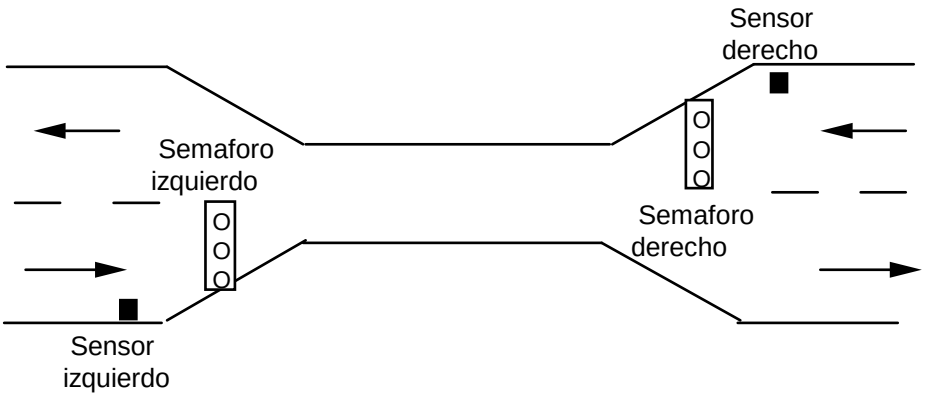


Figura 3a: Esquema del puente.

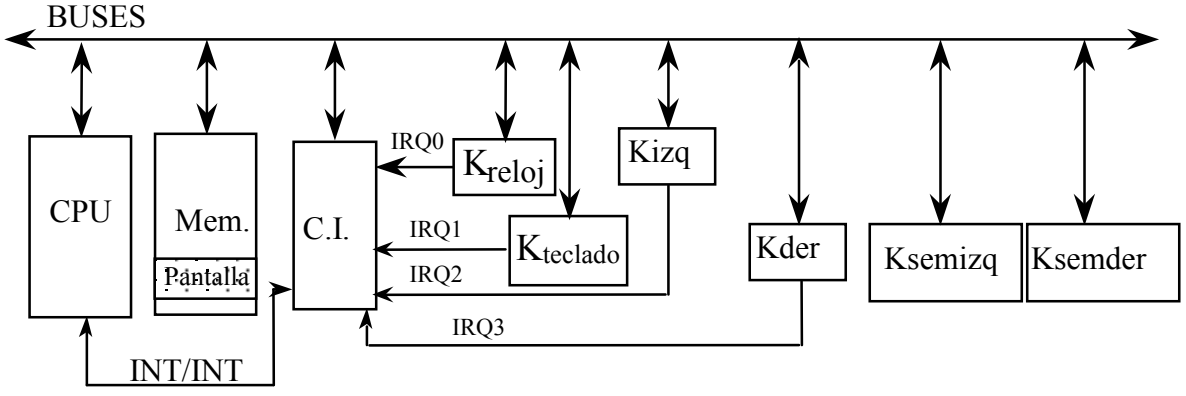


Figura 3b: Estructura hardware del sistema.

Los periféricos que se tienen que controlar, aparte de los vistos en clase, son los siguientes:

*** K_{izq} y K_{der}** Son los controladores de los dos sensores que se encuentran a los lados del puente. Cuando hay un coche parado a su par, esperando que el semáforo se ponga verde, generan peticiones de interrupción con un intervalo de 2 segundos. En la figura 3b se puede ver por qué línea IRQ interrumpe cada uno.

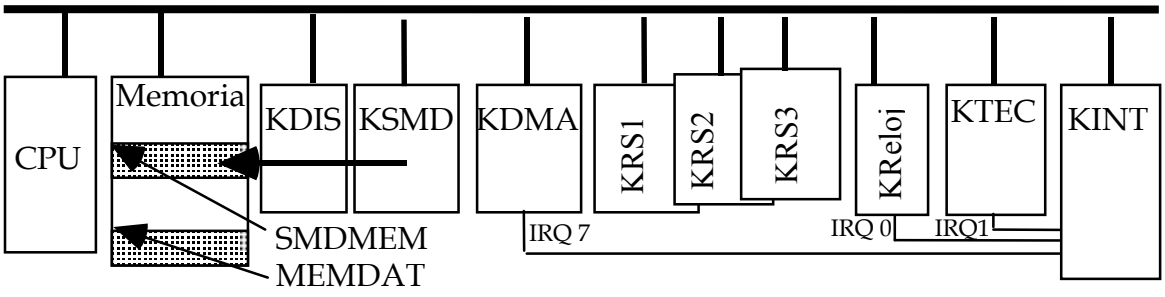
* **K_{semizq}** y **K_{semder}** Son los controladores de los semáforos que se encuentran a ambos lados del puente. Cada uno tiene un **registro de datos** y un **registro de control**. **(RDAT_IZQ, RKON_IZQ, RDAT_DER** y **RKON_DER)**. En el registro de datos se escribe el color del semáforo (rojo, naranja o verde), y cuando queramos cambiar dicho color hay que enviar una secuencia STROBE sobre el bit 4 del correspondiente registro de control.

El **funcionamiento** del sistema debe ser el siguiente. Al inicializar el sistema, los dos semáforos estarán en rojo, y nunca debe ocurrir que ambos estén verdes al mismo tiempo. Partiendo de esta situación inicial, si se detecta algún coche a la izquierda (o a la derecha), el semáforo de ese lado debe ponerse verde, y permanecer así 10 segundos. Si en ese período de tiempo hay coches parados a la derecha (o a la izquierda), no es posible poner verde el semáforo de este lado, pues tal y como ya se ha dicho, los dos semáforos no pueden estar verdes al mismo tiempo. Una vez transcurridos los 10 segundos, cambiaremos el color del semáforo de la izquierda (o de la derecha) a naranja, y lo mantendremos así durante 2 segundos. Mientras un semáforo está naranja se dará prioridad a las peticiones que se produzcan en el lado contrario. Por tanto, si durante esos 2 segundos se detectan coches a la derecha (o a la izquierda), lo que se debe hacer es, esperar 1,5 segundos y después poner el semáforo de la izquierda (o de la derecha) en rojo y el de la derecha (o el de la izquierda) en verde. Si durante los mencionados 2 segundos no se detectan coches en el lado contrario, el semáforo de la izquierda (o el de la derecha) se pondrá rojo, volviendo de este modo a estar los dos en rojo. Por otro lado, existe la posibilidad de poner fin a la ejecución del programa de control, mediante la tecla 'S'. La sincronización del teclado se hará por encuesta.

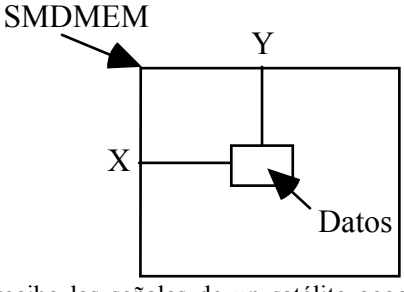
Escribir en lenguaje algorítmico: (a) las siguientes rutinas de servicio: K_{izq} y K_{der} , y la de reloj; y **(b)** el programa principal.

----------o0000000o----------

4.- L@s alumn@s de Arquitectura de la Facultad de Informática de la UPV-EHU han recibido de la organización del **Rally Granada-Dakkar** la petición de desarrollar un sistema de "posicionamiento global por satélite" (GPS), para que los pilotos no se pierdan por el desierto. Un sistema GPS sirve para obtener las coordenadas, latitud (X) y longitud (Y) del punto en el que se está por referencia a un sistema de satélites geoestacionarios. Si además dispone de un mapa digital en memoria, puede dar información relativa a ese punto (altura, vegetación, caminos...). El sistema dispone de los siguientes dispositivos de entrada/salida:



Lector/scanner de mapa digital **SMD**. Es un aparato que permite introducir los datos contenidos en un mapa digital. Dispone de un controlador, **Ksmd** que deja los datos del mapa en un array de 256*256 mapeado en memoria a partir de la posición **SMDMEM**. Cada elemento del array contiene 4 bytes con información (altura, tipo de terreno...) asociada a ese punto (X, Y).



Tres receptores de satélite **RS_i,** con **i: [1..3]**.: Cada uno de ellos recibe las señales de un satélite geoestacionario y dispone de un controlador **KRS_i** con los siguientes registros:

$RegDatSat_i$: Código de posición enviado por el satélite **i.**
$RegEstSat_i$: Registro de estado. Para saber si el dato está disponible hay que esperar a que esté a 1.
$RegConSat_i$: Registro de control. Para avisar de que el dato ha sido leído hay que escribir en él un 1 (strobe).

Controlador de display, **Kdis**: Escribe en un pequeño display de cristal líquido los valores de las coordenada X e Y y la información asociada al punto y un mensaje. Para ello utiliza los registros:

RegDispX: Coordenada X
RegDispY: Coordenada Y
RegDispDat: Datos asociados al punto X, Y
RegDispMen: Mensaje
RegDispCon: Registro de control. Se pone un 1 para avisar que el dato está disponible

Controlador de acceso directo a memoria, **Kdma**: Dispone de los registros

Reg@OrigDMA: dirección de donde se va a leer el primer dato
Reg@DestDMA: dirección donde se va a cargar el primer dato
RegNumDat: número de bytes a transferir
RegConDMA: poner a 1 para iniciar el DMA
RegEstDMA: si está a 1 la transferencia se ha completado con éxito.

Teclado **TEC**: Funciona tal y como hemos visto en clase, pero en este caso sólo se van a utilizar 3 teclas, tal y como veremos más adelante.

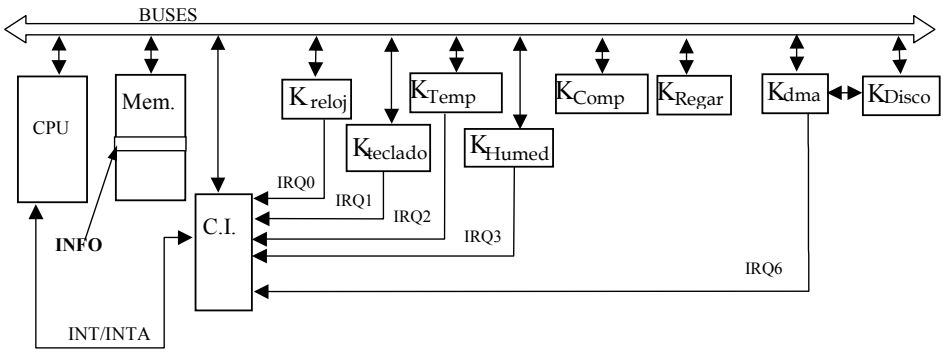
Funcionamiento. El teclado se comunica por interrupción y dispone de las siguientes teclas: **A=** Posición, **B**= Leer nuevo mapa y **C**= Finalizar. Cuando se teclea A: Posición, se leen **POR ENCUESTA** los códigos de los tres satélites y se dejan en la variable **datsat[i]**. Cada encuesta no debe durar más de 30 segundos. Cuando se dispone de los códigos, la rutina ***calcula-posición*** *(datsat, Px, Py)* obtiene a partir de los datos de los satélites, y la devuelve en los parámetros Px y Py. Obtenidos Px y Py hay que escribir en pantalla:

- las coordenadas Px y Py
- la información asociada a esas coordenadas contenida en el mapa de memoria
- un mensaje cuyo contenido será
  - "EXACTAMENTE", si hay datos de tres satélites.
  - "APROXIMADAMENTE", si sólo hay conexión con dos satélites.
  - "SIN COBERTURA" si se reciben menos de 2 satélites. En este caso Px=0 y Py=0

Cuando se teclea B: Leer nuevo mapa, se transfiere la información del controlador del lector/scanner de mapa digital Ksmd a un array de 256*256 elementos de 4 bytes situado en la zona de trabajo de la rutina calcula-posición, a partir de la dirección de memoria **MEMDAT** (que es la dirección de trabajo de la rutina *calcula-posición*). Cuando se teclea C: Finalizar, se termina el programa. En todo caso no se atenderá a una tecla hasta que no se haya terminado de realizar la tarea asociada a la tecla anteriormente pulsada.

Se pide: (a) el programa principal y **(b)** las rutinas de atención a las interrupciones de reloj, teclado y DMA.

5.- Se desea diseñar un **sistema automático de la temperatura y humedad de un invernadero**. El sistema está basado en la arquitectura de la siguiente figura, similar a la vista en clase, a la que se han añadido unos sensores de temperatura y humedad.



KTEMP: Dispositivo que interrumpe cada vez que la temperatura cambia, dispone de un registro de datos (RDAT_TEMP) en que nos almacenará la nueva temperatura. Tras leer el registro de datos será necesario realizar una secuencia de STROBE en el registro de control (RCON_TEMP).

KHUMED: Dispositivo que interrumpe cuando la humedad relativa sea inferior a un valor mínimo (HUMEDAD_ESCASA).

KCOMP: Compuertas del invernadero. Disponen de un registro de control (RCON_COMP) en que se debe indicar que se quiere hacer: 1 para abrir las compuertas y 0 para cerrarlas. Para simplificar, suponed que no pasa nada si se intenta cerrar las compuertas estando ya cerradas o si se intenta abrir estando ya abiertas.

KREGAR: Válvulas de regadío. Disponen de un registro de control (RCON_REGAR) en que se debe indicar que se quiere hacer: 1 para abrir las válvulas y 0 para cerrarlas. Al igual que en el caso anterior, para simplificar, suponed que no pasa nada si se intenta cerrar las válvulas estando ya cerradas o si se intenta abrir estando ya abiertas.

KDMA: Es el controlador del DMA y se utiliza para realizar transferencias entre la memoria y una unidad externa de disco. Posee los siguientes registros:

- **Registro de control** (RCON_KDMA): es el registro sobre el que se debe indicar la operación que se desea realizar, escribiendo un uno (1) para dar comienzo a la transferencia (se pone a 0 de manera automática al finalizar la transferencia).
- **Registro de dirección** (RDIR_KDMA): en el que se indicará la dirección de memoria a partir de la cual deberá comenzar a copiar los datos en la transferencia al disco.
- **Registro de longitud** (RLON_KDMA): se debe indicar en este registro el tamaño en bytes del bloque de información que se debe transmitir.
- **Registro de estado** (REST_KDMA): una vez finalizada la transferencia, el contenido de este registro indica si se ha realizado sin problemas (1) o si, por el contrario, se ha producido un error en la misma (0).

KDisco: Es el controlador de disco. Supondremos que existe una rutina llamada *ProgramarKDisco()* que se encargará de su programación.

El **funcionamiento** del sistema es el siguiente. Cuando la temperatura supere un valor máximo (TEMP_ALTA), se deben abrir las compuertas del invernadero. Estas compuertas se deben cerrar cuando la temperatura descienda a un valor inferior a una temperatura de cierre (TEMP_CIERRE). Si la temperatura baja por debajo de un valor crítico (TEMP_MUY_BAJA), se debe producir una alarma (mediante la rutina ya programada *SonarAlarma*()) para avisar al operario. En esta caso, el sistema se suspende y no realizará ningún control hasta que el operario haya resuelto el problema: se lo indicará al sistema pulsando la tecla 'C'. Tras esto el sistema automático debe volver al estado inicial con las compuertas del invernadero cerradas. Si se detecta que la humedad relativa es inferior al mínimo, se deben abrir las válvulas de regado durante 5 minutos. Durante estos cinco minutos no se realizará ningún control de temperatura ni de humedad.

La información de cada cambio de temperatura o humedad se debe almacenar en una variable del sistema (que está en la dirección de memoria *INFO*). Para ello, puedes utilizar la función ya programada *AlmacenaEvento(Número_evento, Evento, Valor, Hora)*, que guardará en esa variable global el histórico de lo que ha ocurrido. La descripción de los parámetros es la siguiente: *Número_evento* es el número de evento correlativo, *Evento* puede tener uno de estos dos valores (TEMPERATURA o HUMEDAD), *Valor* es el valor de la temperatura o el valor 0 (en el caso de HUMEDAD), *Hora* es la hora a la que ha ocurrido el evento.

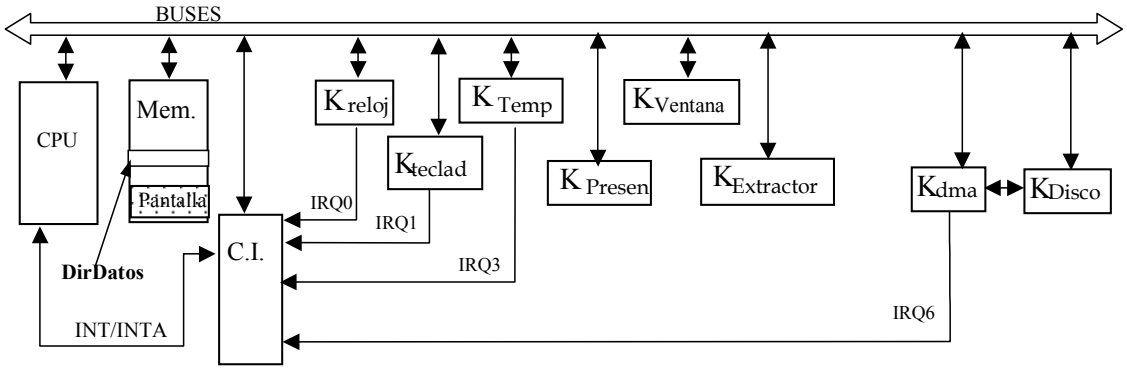
En cualquier momento el operario puede finalizar el sistema pulsando la tecla 'S'. En este caso, antes de terminar, se debe guardar la información de los eventos del sistema en disco. Suponed que cada evento ocupa 40 bytes. Si la transferencia falla, se debe reintentar hasta tres veces. Tras el tercer fallo se enviará un mensaje de error mediante la ruina ya programada *ErrorEnTransferencia()* y, como ya se ha dicho, finalizará el sistema.

La hora actual se encuentra en la variable global *HORA*. Para su tratamiento puedes utilizar las siguientes funciones ya implementadas: *InicializaHora(HORA)*, para inicializar la hora al comienzo del sistema, y *ActualizaHora(HORA)*, para incrementar en un segundo la hora.

Los registros de todos los dispositivos tienen direcciones dentro del espacio de E/S, independientes de memoria. Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema en particular, la sincronización del **teclado** se debe realizar por **encuesta**.

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier suposición que realices en la resolución del problema.

6.- Dada la temperatura excesiva de algunas aulas y despachos de la facultad y del aulario, el vicerrectorado nos ha pedido que realicemos un estudio de la viabilidad de poner un ***sistema de control automatizado de la temperatura*** en una de las salas. Para ello el sistema se apoyará en una serie de dispositivos que se han de añadir a la sala del edificio: *sensores de temperatura* que nos indicarán la temperatura de la sala; *sensores de presencia,* que nos indicará si una sala está o no ocupada; *motores* para abrir o cerrar las ventanas; y e*xtractores* de ventilación forzada. A continuación se describe el esquema hardware del sistema y los periféricos que intervienen:



KTemp: Sensor de temperatura de la sala. Interrumpirá regularmente para informarnos de la temperatura. Dispone de un registro de datos, RDAT_KTemp, en el que se indicará la temperatura de la sala. Tras leer el registro, se debe activar una secuencia de *strobe* sobre el bit 3 del registro de control, RCON_KTemp.

KPresen: Sensor de presencia de la sala. En su registro de datos RDAT_KPresen nos indicará si detecta o no personas en la sala (1 sala ocupada, 0 sala vacía). Este dispositivo se debe gestionar por **encuesta**.

KVentana: Los motores de las ventanas de la sala están centralizados en este dispositivo. Para abrir o cerrar las ventanas hay que indicar en el registro de datos (RDAT_KVentana) la acción a realizar (1 Abrir, 0 Cerrar) y enviar la secuencia de *strobe* sobre el registro de control (RCON_Kventana).

KExtractor: De funcionamiento similar al KVentana. En este caso el registro de control es RDAT_KExtractor (1 Activar, 0 Desactivar) y el registro de control es RCON_KExtractor.

KDMA: Es el controlador de DMA y se utiliza para realizar transferencias entre la memoria y una unidad Disco. Posee los siguientes registros:

> **Registro de control** (RCON_KDMA). Registro sobre el que se debe indicar el comienzo de la transferencia, escribiendo un uno (1) (se pone a 0 de manera automática al finalizar la transferencia).
>
> **Registro de dirección** (RDIR_KDMA). En este registro se indicará la dirección de memoria a partir de la cual deberá comenzar a copiar los datos.
>
> **Registro de longitud** (RLON_KDMA). Se debe indicar en este registro el tamaño en bytes del bloque de información que se debe transmitir.
>
> **Registro de estado** (REST_KDMA). Una vez finalizada la transferencia, el contenido de este registro indica si se ha realizado sin problemas (1) o si, por el contrario, se ha producido un error en la misma (0).

KDisco: Es el controlador de Disco. Posee un conjunto de registros, pero para obviar su programación supondremos que existe una rutina llamada *Programar_KDisco()* que se encargará de ello.

El **funcionamiento del sistema** es el siguiente. El sistema debe tratar de estabilizar la temperatura de la sala, solamente cuando esté ocupada, intentando mantener su temperatura en un nivel normal. Si la sala no está ocupada, o en algún momento se desocupa, no se realizará ningún control. Cuando se detecte alguna persona en la sala, el sistema comenzará a controlar la temperatura. Si la temperatura de la sala sobrepasa una TEMP_ALTA se deben abrir las ventanas para reducir la misma. Si transcurrido 5 minutos la temperatura de la sala no ha descendido de TEMP_ALTA, se deben activar los extractores de ventilación forzada. Si a pesar de todo, y transcurridos 5 minutos más, la temperatura sigue estando por encima de TEMP_ALTA, se debe activar una alarma en el equipo de control, mediante la función ya programada *Alarma("Temperatura excesiva").* En este momento, el supervisor hará desalojar la sala y cuando pulse la tecla 'A' se desactivará la alarma mediante la función *DesactivarAlarma()*, se cerrarán las ventanas y los extractores se

desactivarán. Si antes de producirse la alarma, la temperatura baja de TEMP_ALTA, se cerrarán las ventanas y/o se desactivarán los extractores. Si en algún momento la sala se desocupa, hay que dejar de realizar los controles, cerrando las ventanas y/o desactivando los extractores.

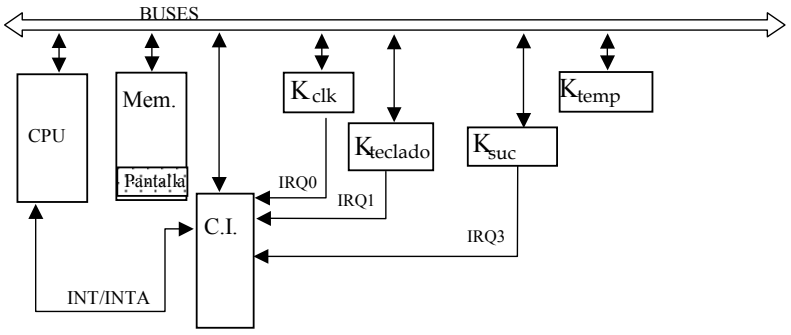
Cada minuto se debe registrar la temperatura de la sala mediante la función ya programada *RegistrarTemperatura(Temperatura, Hora)*, donde *Temperatura* es la última temperatura detectada en el sistema y *Hora* es la hora global del sistema. Cada noche a las 00:00 horas, se debe realizar un volcado en disco de la información que se ha registrado con *RegistrarTemperatura*. La información se ha guardado en memoria (a partir de la dirección *DirDatos*) y cada registro es de tamaño fijo (100 bytes). Si se produce un error en esta transferencia, se debe sacar un mensaje al operador mediante la función *MensajeError("Error en DMA")*, pero el sistema seguirá funcionando con normalidad. Para llevar el control de la hora del sistema, se dispone de una variable global HORA. Dicha variable de debe de inicializar mediante la función ya programada *InicializaHora*(). Para actualizar la hora se dispone de la función *ActualizaHora*(), que incrementa en un segundo la variable HORA.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema en particular, la sincronización del **teclado** se debe realizar por **interrupción**. Todos los registros de los dispositivos están mapeados en E/S.

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier suposición que realices en la resolución del problema.

----------o0000000o----------

7.- Se desea diseñar un sistema que gestione el **tratamiento del agua de una piscifactoría** a nivel del mar, basándose en la estructura hardware que se muestra en la siguiente figura:



Los controladores de los periféricos novedosos son los siguientes:

Ksuc: Es el controlador que detecta alto nivel de suciedad en la superficie de la piscifactoría. Interrumpe cada vez que el nivel de suciedad supera una cota preestablecida. Cada vez que interrumpa es necesario realizar un STROBE sobre el registro de control (*RCON_KSUC*).

Ktemp: Es el controlador que mide la temperatura del agua. Cada vez que en el registro de estado *REST_KTEMP* haya un 1, en el registro de datos (*RDAT_KTEMP*) están dispuestos los grados Celsius que tiene el agua.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema la sincronización de **teclado** se realiza **por interrupción** y el **análisis de la temperatura** del agua se realizarán **por encuesta**. Los registros asociados al controlador de teclado son *RDAT_KTEC* y *RCON_KTEC* para el caso del registro de datos y el de control, respectivamente.

El **funcionamiento del sistema** es el siguiente. Si no hay problemas con la temperatura en la piscina, cada cinco minutos hay que analizar la temperatura del agua. Si el agua está entre –5º Celsius y +15º Celsius se supone que los peces viven con toda normalidad y no hay que tomar medidas especiales. Sin embargo, si la temperatura baja de los –5º hace falta calentarla, ya que en caso contrario puede ser letal para los peces si pasa cierto tiempo así. Para calentar el agua, se añaden 100 litros de agua caliente mediante la rutina *agua_caliente()*. Si una vez añadidos los 100 litros, la temperatura sigue fría, se añade una y otra vez agua caliente mediante *agua_caliente()*. El hecho de que la temperatura sea superior a 15º también resulta letal para los peces al cabo de un tiempo. En este caso, hay que enfriar el agua mediante la rutina *hielos()* que añade 100 litros de hielo. Aquí también, si añadidos los 100 litros la temperatura sigue elevada, habrá que añadir hielos una y otra vez hasta que llegue al intervalo permitido. Como puede apreciarse, mientras haya problemas hay que estar analizando la temperatura varias veces. Tras solventar el problema, a partir de este momento, se analizará la temperatura a los cinco minutos.

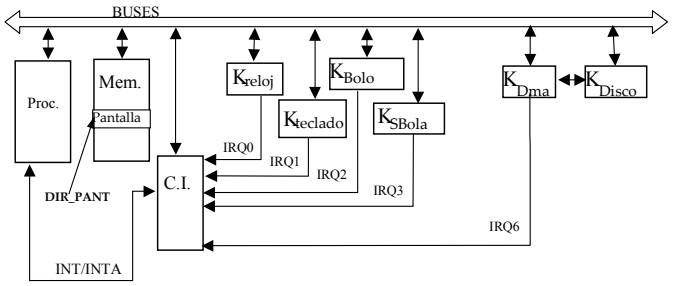
Además de la temperatura, la suciedad en el agua puede causar el fallecimiento múltiple de los peces. Como ya se ha indicado, el controlador de la suciedad interrumpe en cuanto detecta que la suciedad en la superficie supera una cota. En este caso, se activa una alarma sonora (rutina *alarma()*) que sirve para que los operarios de la factoría se pongan a retirar la suciedad que se recoge mediante unas redes que recorren la superficie de la piscina (las redes se ponen en marcha mediante *activar_redes()*). Esa suciedad la van volcando los operarios a unos contenedores a orillas de la piscina (suponer que hay suficientes contenedores y que cuentan con la capacidad necesaria). En cuanto el director de los operarios estime que se ha limpiado convenientemente la superficie de la piscina, pulsará la tecla 'D' que controla el sistema, lo que conllevará que se desactive la alarma con *parar_alarma()*, las redes mediante *desactivar_redes()* y se sustituyan los contenedores que almacenan la suciedad por otros, mediante *cambiar_containers()*. De cualquier manera, para evitar contaminación acústica, la alarma activada sólo estará en marcha a lo sumo 30 segundos desde que se puso en marcha, por lo que si se supera este tiempo también habrá que desactivarla.

Se pide:

(a) Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier supuesto que hagas para la resolución del problema.

(b) ¿Qué cambios conllevaría la sincronización del teclado por encuesta?

8.- Queremos diseñar un sistema que controle las **partidas de una bolera**. Para ello disponemos de un sensor que detecta si la bola ha sido lanzada y un dispositivo capaz de contar y levantar los bolos derribados.



K_BOLO: Es el controlador del dispositivo utilizado para levantar los bolos. Tiene dos registros: un registro de control (RCON_KBOLO) y un registro de datos (RDAT_KBOLO). Para poner el dispositivo en marcha debemos realizar una secuencia de strobe en el registro de control, y éste levantará todos los bolos que hayan sido derribados. Una vez finalizado su trabajo el dispositivo genera una petición de interrupción. En ese momento podremos leer en el registro de datos el número de bolos que ha tenido que levantar.

K_SBOLA: Es el controlador del sensor que detecta que la bola ha sido lanzada. La bola pasa junto a éste cuando está a punto de llegar a los bolos. En ese momento el sensor genera una petición de interrupción.

K_DMA: Es el controlador de DMA y se utiliza para hacer transferencias rápidas de memoria a disco. Posee los siguientes registros:

- **registro de control** (RCON_KDMA): cuando se escribe un 1 en él se inicia la transferencia (se pone a 0 automáticamente).
- **registro de dirección** (RDIR_KDMA): dirección de comienzo del bloque a transferir.
- **registro de longitud** (RLON_KDMA): contiene la longitud del bloque a transferir.
- **registro de estado** (REST_KDMA): cuando su contenido es un 1 indica que la última operación realizada ha sido errónea.

K_DISCO: Es el controlador de disco. Para realizar la transferencia hay que inicializar este controlador. Supondremos que existe una rutina llamada *ProgramarKDisco()* que se encargará de su programación.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema la sincronización de **teclado** se realiza **por encuesta**. La pantalla está mapeada en memoria a partir de la dirección DIR_PANT.

El **funcionamiento** del programa es el siguiente. Hasta que el propietario de la bolera no nos lo permita no podremos jugar, es por ello por lo que antes de la partida habrá una barrera delante de los bolos. Una vez hayamos pagado, éste

pulsará la tecla 'H' del teclado. De esta forma la barrera se levantará y podremos comenzar la partida. Para levantar la barrera podemos utilizar la rutina *levantar_barrera()*.

Cada partida constará de 10 lanzamientos y después de cada uno se contarán los bolos derribados. En cada lanzamiento, el sensor SBOLA detectará el momento en el que la bola se acerca a la posición de los bolos y generará una interrupción. Tendremos que esperar 10 segundos a que los bolos se estabilicen en el suelo y una vez transcurrido ese tiempo el dispositivo BOLO deberá levantar los bolos. Después de cada lanzamiento deberemos mostrar en pantalla la suma de todos los bolos derribados hasta el momento. Para ello disponemos de la rutina *mostrar_derribados(Num_Bolos, Num_Lanzamiento)*, que recibe dos parámetros: el número de bolos derribados hasta el momento, y el numero de lanzamientos realizados. De esta forma, todas las puntuaciones irán mostrándose en pantalla. Cada puntuación ocupará 4 bytes en memoria.

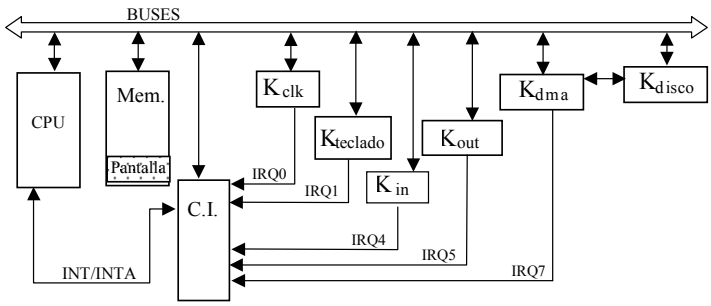
Una vez que se hayan realizado 10 lanzamientos la partida terminará, pero antes habrá que guardar todas las puntuaciones en disco. Para esta transferencia de memoria a disco utilizaremos el DMA. Si la transferencia no se realiza correctamente deberemos intentarlo de nuevo. Si en 3 intentos no hemos conseguido realizar la transferencia llamaremos a la rutina *error_de_transferencia()* y terminaremos la partida. Para que las partidas no se prolonguen demasiado, el jugador tiene un limite de tiempo para realizar su lanzamiento. Si desde que se han levantado los bolos, pasan 5 minutos sin que la bola sea lanzada, la partida se terminará. En este caso la puntuación no será transferida al disco.

En cualquier caso, una vez finalizada la partida la barrera deberá ser bajada otra vez, hasta que el propietario vuelva a pulsar la tecla 'H'. Para ello utilizaremos la rutina *bajar_barrera()*.

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier supuesto que hagas para la resolución del problema.

----------o0000000000o----------

9.- En un parque temático de diversiones, van a incorporar un **pasillo del horror** en donde las personas irán entrando y saliendo una a una, quedando la estructura hardware del sistema de la siguiente manera:



Los controladores de los periféricos novedosos son los siguientes:

Kin: Es el controlador de la barrera de entrada. Cuando una persona quiere entrar, debe pulsar un botón que hay junto a la barrera, lo que hace que este controlador genere una petición de interrupción. Cuenta con un registro de datos (RDAT_Kin) en donde se muestra el peso de la persona que quiere entrar y un registro de control (RCON_Kin) donde hay que hacer un STROBE cada vez que se atienda una de sus interrupciones.

Kout: Es el controlador de la puerta de salida. Cuando la persona que está dentro del pasillo empuja la puerta final del mismo, este controlador solicita interrumpir. Cuenta con un registro de control (RCON_Kout) donde hay que hay que hacer un STROBE cada vez que se atienda una de sus interrupciones.

Kdisco: Es el controlador de disco. Posee un conjunto de registros, pero para obviar su programación supondremos que existe una rutina, *Programar_KDisco(tipo_operación)* que se encargará de ello, según el tipo de operación que quiera realizarse (*LECTURA* de disco o *ESCRITURA* en disco).

Kdma: Es el controlador de DMA y se utiliza para realizar transferencias entre la memoria y la unidad de disco. Posee los siguientes registros:

Registro de control (RCON_Kdma). En este registro se escribe un 1 para dar comienzo a la transferencia (se pone a 0 de manera automática al finalizar la transferencia).

Registro de dirección (RDIR_Kdma). En este registro se indicará la dirección de memoria a partir de la cual deberá comenzar a transferir los datos.

Registro de longitud (RLON_Kdma). Se debe indicar en este registro el tamaño en bytes del bloque de información que se debe transmitir.

Registro de estado (REST_Kdma). Una vez finalizada la transferencia, el contenido de este registro indica si se ha realizado sin problemas (0) o si se ha producido un error en la misma (1).

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema la sincronización de teclado se realiza **por encuesta**. Los registros asociados al controlador de teclado son RDAT_Kteclado y RCON_Kteclado, registro de datos y de control, respectivamente.

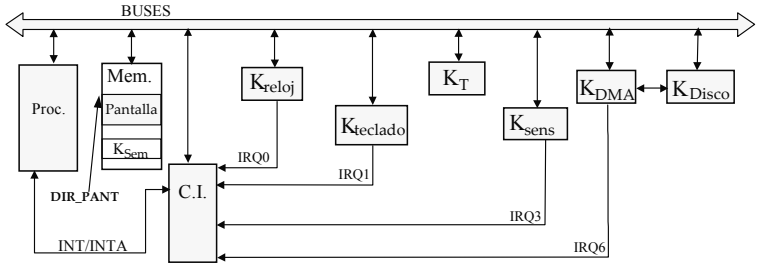
El **funcionamiento del sistema** es el siguiente. Si una persona quiere entrar en el pasillo, pulsará el botón que hay junto a la barrera de la entrada. Si el pasillo está ocupado, no se le permite la entrada, ya que no se abre la barrera de entrada, por lo que tendrá que pulsar el botón una y otra vez hasta que se abra la barrera. Por el contrario, si el pasillo está libre, se abre la barrera y se le deja pasar mediante la rutina ya implementada *permitir_entrada()*. Además, se toma constancia del peso de la persona que ha entrado para unas estadísticas que se guardarán posteriormente.

Una vez dentro del pasillo, cada 5 segundos se le irán realizando diversos sustos, tales como oír risotadas siniestras, soltar llamaradas, irrumpir bichos malignos, etc. De ello se encarga la rutina *susto()*, que cada vez que se ejecuta escoge aleatoriamente uno de los sustos archivados en el sistema. La persona tiene un plazo de 3 minutos para atravesar el pasillo y salir por la puerta de salida. Si pasan los 3 minutos y no ha salido por la puerta, se abre una trampilla en el suelo por medio de la rutina *trampilla()*, que hace que el usuario sea expulsado del pasillo cayendo por un tobogán. Pero si antes de los 3 minutos logra salir por la puerta de salida, se debe guardar en memoria un dato de 40 bytes con la información siguiente: cuántas personas han salido por la puerta hasta el momento, el peso de la persona que acaba de salir y el tiempo medido en segundos que le ha costado atravesar el pasillo. Para ello, disponemos de la rutina *registrar_datos(dirección, persona, peso, duración)*. Cuando se pone en marcha el sistema, la información correspondiente a la primera persona que sale por la puerta de salida se almacena en la dirección *Dirinicio*, y a partir de esa dirección, en posiciones consecutivas, se debe almacenar la información correspondiente a las personas que a continuación consiguen salir por la puerta.

Si el operario que supervisa el sistema pulsa la tecla 'D' y el pasillo está vacío, se copian en disco los datos que están guardados en memoria. Si cuando ha pulsado 'D' había alguien en el pasillo, habrá que realizar dicha copia en cuanto se vacíe el pasillo, bien sea por la trampilla o por la puerta. Mientras se hace la copia no se permite el acceso a nadie al pasillo. Si al finalizar la copia algo ha ido mal, se reintentará una y otra vez, hasta que la copia se realice correctamente. Una vez realizada la copia, se vuelven a guardar los datos en memoria a partir de *Dirinicio*, se vuelve a inicializar el número de personas que han salido hasta el momento y se permite nuevamente la entrada a más personas al pasillo, siempre de una en una.

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier supuesto que hagas para la resolución del problema.

10.- Queremos diseñar un sistema que **control**e las cabinas de pago automático (las denominadas de **telepago**) sitas en los peajes de salida de los tramos de precio fijo de una autopista. Para ello, el sistema dispone, en cada cabina, de los dispositivos indicados en la figura siguiente.



Las características de los controladores de los periféricos son las siguientes:

K_T: Es el controlador de telepago. Tiene tres registros: registro de control (RCON_KT), registro de estado (REST_KT) y registro de datos (RDAT_KT). El registro de estado toma el valor 1 cuando se detecta un coche que posee el sistema de telepago. En el registro de datos —de 5 bytes—, estará el número de identificación (*NI*) del coche detectado. En el registro de control se debe realizar una secuencia de

strobe, gracias a la cual el valor del registro de estado pasará a ser 0 y el controlador quedará listo para detectar el siguiente coche. La sincronización de este periférico se realizará por **encuesta**.

K_Sem: Es el controlador del semáforo que hay encima de la cabina. Tiene un registro de control (RCON_KSem) mapeado en memoria. Cuando se escribe un 1 en él, el semáforo se pone en rojo, y en verde cuando se escribe un 0.

K_sens: Es el controlador de un sensor situado junto a la barrera de salida de la cabina. Solicita una interrupción cuando detecta que un coche ha pasado completamente frente a él.

K_DMA: Es el controlador de DMA y se utiliza para hacer transferencias de memoria a disco. Posee los siguientes registros:

- **registro de control** (RCON_KDMA): cuando se escribe un 1 en él se inicia la transferencia (se pone a 0 automáticamente).
- **registro de dirección** (RDIR_KDMA): dirección de comienzo del bloque a transferir.
- **registro de longitud** (RLON_KDMA): longitud en bytes del bloque a transferir.
- **registro de estado** (REST_KDMA): cuando su contenido es un 0 indica que la última operación realizada ha sido errónea.

K_DISCO: Es el controlador de disco. Para realizar la transferencia hay que programar este controlador, para lo que tenemos la rutina *ProgramarKDisco()*.

Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. En este sistema la sincronización de **teclado** se realiza **por interrupción**.

El **funcionamiento** del sistema es el siguiente. Cuando a la cabina controlada llega un coche que dispone del sistema de telepago, si el semáforo está verde, lo detectará el controlador K_T. Como el pago se realizará automáticamente (gracias al número de identificación del coche), el sistema deberá levantar la barrera que hay a la salida de la cabina, mediante la rutina ***levantar_barrera()***, para dejar pasar al coche.

La barrera de salida se mantendrá levantada el tiempo necesario para que el coche pase completamente, y 2 segundos más, por razones de seguridad, tras lo cual se bajará, mediante la rutina ***bajar_barrera()***. Para simplificar el sistema, supondremos que es necesario esperar a que baje la barrera para detectar el coche siguiente.

Para gestionar el cobro automático del peaje, cada vez que pasa un coche por la cabina, se debe guardar en memoria la información siguiente: *NI* número de identificación —de 5 bytes— y la hora de paso (que está en la variable global *HORA*) —ésta también de 5 bytes—. Para escribir esa información en memoria, disponemos de la rutina ***escribmem (NI, HORA, númerococches)***. La propia rutina se encarga de calcular en qué posición de memoria debe escribir la información en cada momento, teniendo en cuenta el número de coches que han pasado por la cabina y que la dirección de comienzo en la que se escribe la información correspondiente al primer coche que pasa al inicializar el sistema es INFO_DIR. En lo que respecta a la variable *HORA*, disponemos de las dos rutinas siguientes: ***InicializarHora()***, que asigna a la variable *HORA* el valor actual indicado por el sistema operativo, y ***ActualizarHora()***, que actualiza la variable *HORA* en un segundo.

El color del semáforo de la cabina se controla por medio del teclado:

- Al pulsar la tecla R, el semáforo se pondrá en rojo y el sistema no aceptará que pasen coches por esa cabina (para simplificar el sistema, supondremos que la persona que debe pulsar la tecla lo hará cuando no haya coches en la cabina). Además de cambiar el color del semáforo, el sistema debe pasar a disco, mediante DMA, toda la información que se ha ido almacenando en memoria acerca de los coches que han pasado a lo largo de todo el intervalo de tiempo en que el semáforo ha estado verde. Al finalizar la transferencia por DMA, si ha ocurrido algún error, el sistema deberá intentarlo otras dos veces más como máximo, pero si después del tercer intento no se ha conseguido finalizar con éxito la transferencia, entonces se deberá ejecutar la rutina ***error_transferencia()*** para que aparezca un mensaje en pantalla y se deberá finalizar el programa. Si no ocurren errores en la transferencia, se continuará con el funcionamiento normal (semáforo rojo).
- Al pulsar la tecla V, el semáforo se pondrá en verde y el sistema recuperará el funcionamiento normal (semáforo verde). La información acerca de los coches que pasen por la cabina en el nuevo intervalo de tiempo en que el semáforo esté verde se escribirá otra vez a partir de la dirección INFO_DIR.

Se pide: Escribe en lenguaje algorítmico todas las rutinas de atención que consideres necesarias, así como el programa principal. Comenta cualquier supuesto que hagas para la resolución del problema.