Arquitectura de Computadores I

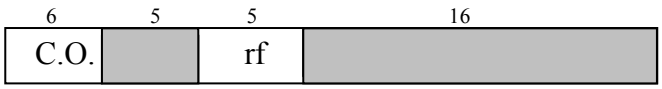
Ejercicios de Autoevaluación - 3

1.- [1 punto] Responde a las siguientes preguntas teniendo en cuenta el procesador sencillo BIRD que se ha visto en la asignatura.

a) ¿Por qué suele ser más rápida la ejecución de las instrucciones en unidades de control cableadas que en las microprogramadas? En contrapartida, ¿qué ventaja principal tienen las unidades de control microprogramadas frente a las cableadas?

b) Se quiere incorporar al juego de instrucciones la instrucción `NEG Rf`. Esta instrucción invierte los bits contenidos en el registro que tiene como operando, uno a uno, y deja el resultado en el propio registro `Rf` (para ello suponer que hay una operación de la UAL que invierte los bits de un operando dado): `Rf:=` $\overline{\texttt{Rf}}$ `{inversión bit a bit}`

Su formato es el siguiente:



Escribe la parte del microprograma correspondiente a la ejecución de esta instrucción. En el caso de que se deba realizar alguna modificación en la estructura de la Unidad de Proceso, especifícala claramente en el esquema que se te proporciona en el cuadernillo de respuestas. ¿Cambia algo la parte del control correspondiente a las demás instrucciones?

2.- [2 puntos] El sistema de memoria de un computador tiene las siguientes características:

- Direccionamiento al byte, con palabras de 8 bytes.
- **Memoria virtual** paginada de 1 MB, con páginas de 512 bytes. Para la traducción de direcciones se dispone de un TLB con tiempo de acceso de 30 ciclos en caso de fallo y 1 ciclo en caso de acierto. El TLB está inicialmente vacío.
- **Memoria principal** de 256 kB, formada por 4 módulos entrelazados. El tiempo de acceso a MP es de 12 ciclos (1 desde el buffer de entrelazado).

Se pide:

a) Esquema de la traducción de direcciones. ¿Cuántas páginas como máximo puede tener un programa? ¿Cuántas páginas tiene la memoria principal? ¿Cuántas entradas tiene la tabla de páginas? ¿De qué tamaño es cada entrada? ¿Cuál es el tamaño de cada entrada del TLB?

b) Esquema de la dirección de memoria principal y de su organización. ¿Cuál es el tamaño en bytes de un módulo?

c) En este computador se ejecuta el siguiente programa:

```
for (i=2;i<64;i++)
        x=C[4]+(A[i]*A[i+2]);
```

```
                movi r1, #1296
                movi r3, #2
                load r2, [r1+680]
bucle:          load r5, [r1]
                load r6, [r1+16]
                mul r5, r5, r6
                add r10, r2, r5
                add r1, r1, #8
                addi r3, r3, #1
                subi r4, r3, #64
                bnz r4, bucle
```

El programa comienza en la dirección lógica 480. El vector A comienza en la dirección lógica 1280 y el vector C en la dirección lógica 1944. Las instrucciones y los elementos de los vectores son de tamaño 1 palabra. Las variables x e i (índice del bucle) están almacenadas en registros de la máquina (ver código en ensamblador).

Traduce las referencias que genera el programa en la primera pasada por el bucle y calcula el tiempo de acceso al sistema de memoria para cada una de esas referencias. Para ello, indica claramente, mediante una tabla, los diferentes pasos que sigues para el cálculo del tiempo de acceso. El contenido de la tabla de páginas es el siguiente

Página Lógica:	2	14	0	1	32	3	....
Página Física:	10	8	0	2	12	1	....

d) ¿A qué se debe el que haya fallo o acierto en el TLB? Calcula el número de fallos en el TLB que se producen al realizar la traducción de todas las referencias lógicas del programa.

3.- [0,8 puntos] Queremos conectar a nuestro PC un periférico que utiliza la línea IRQ5. El controlador de este periférico tiene 2 registros de 8 bits cuyas direcciones de E/S (independientes de memoria) son:

`Registro de datos: @ 0x279` `Registro de Control: @ 0x281`

El funcionamiento del sistema es el siguiente. Cada vez que este periférico interrumpe hay que leer el contenido del registro de datos y, a continuación, hacer una secuencia de `strobe` sobre el bit 3 del registro de control. Si el bit 4 del registro de datos vale 1, sacará el carácter '1' por la pantalla. Si, por el contrario, ese bit vale 0, sacará el carácter '0' por la pantalla y acabará la ejecución del programa. **La sincronización con este periférico se realiza por encuesta.**

Para escribir en la pantalla utilizaremos la función `EscribeCar`. La pantalla está mapeada en memoria, tiene 60 filas y 132 columnas y los caracteres se escriben sin atributo. La codificación de cada carácter a imprimir en la pantalla se realiza en 2 bytes.

De acuerdo al funcionamiento del periférico, implementa en C las siguientes cuestiones:

a) La rutina que realiza la encuesta: `unsigned char Encuesta().` Esta función devuelve el contenido del registro de datos del controlador.

b) El programa principal. Entre otras cosas, este programa llamará a la función `Encuesta()` para realizar la encuesta del periférico y cogerá el valor devuelto por la función. El programa realizará el tratamiento indicado para ese valor mientras no se produzca la condición de fin de programa (el bit 4 del registro de datos valga 0).

NOTA: Para aclarar el código C, explica lo que hace cada línea de código que escribas. Para hacer este ejercicio, dispones de las siguientes rutinas:

unsigned short ***InicioPant****;*
unsigned short ***DirPant****();*

#define RDAT_PER 0x279 *#define RCON_PER 0x281*
#define REG_IMR_C8259 0x21 *#define REG_IRR/ISR_C8259 0x20*

unsigned char ***InPort*** *(puerto);* *void* ***EnableInts****();*
void ***OutPort*** *(puerto, valor);* *void* ***DisableInts****();*

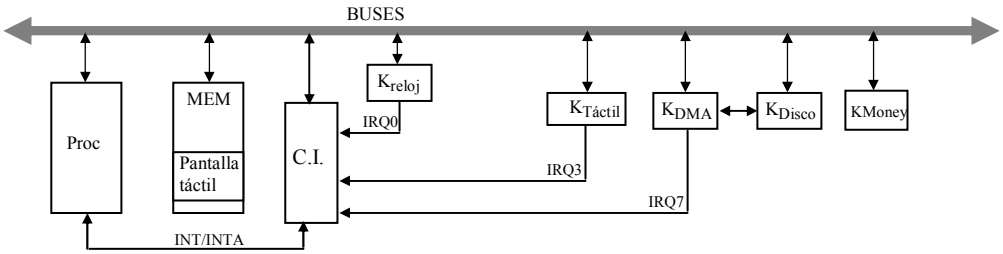
unsigned char ***LeeByteFis*** *(unsigned short Seg, unsigned short Desp);*
void ***EscribeByteFis*** *(unsigned short Seg, unsigned short Desp, unsigned char car);*

unsigned short ***LeePalFis*** *(unsigned short Seg, unsigned short Desp);*
void ***EscribePalFis*** *(unsigned short Seg, unsigned short Desp, unsigned short pal);*

void ***Desinhibir_Int_KPer****();*

4.- [2,5 puntos] Se piensa ubicar una **máquina lúdica** en la primera planta de la Facultad. Esta máquina cuenta con una pantalla táctil por medio de la cual se hace la interacción principal con el juego en cuestión. Queremos diseñar un sistema que la controle. Para ello, el sistema dispone de los dispositivos indicados en la figura.

Figura



Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. La pantalla táctil está mapeada en memoria. Salvo el controlador de la ranura del dinero, que se controla por encuesta, el resto de controladores se gestionan por interrupciones. Las características de los controladores de los periféricos son las siguientes:

KMoney: Es el controlador de la ranura de dinero. Tiene un registro de estado (REST_KMONEY) que se pone a 1 si se detecta que se ha introducido una moneda de 50 céntimos por la ranura y se pone a 0 automáticamente en cuanto se traga o se expulsa la moneda. Supón que el resto de monedas no las acepta (o no caben por la ranura o las devuelve). Este controlador se sincroniza **por encuesta**.

KTáctil: Es el controlador de la pantalla táctil. Solicita una interrupción cada vez que detecta que el usuario toca un punto de la pantalla. Tiene un registro de datos (RDAT_KTáctil) — de 3 bytes, debido a que la resolución de la pantalla es de 800 * 600—, en el que aparece identificado el píxel central de donde ha realizado la pulsación el usuario. Tiene un registro de control RCON_KTáctil, en el que hay que hacer un STROBE cada vez que interrumpa este controlador.

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): contiene la 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 una transferencia hay que programar este controlador, para lo que tenemos la rutina *ProgramarKDisco()*.

El **funcionamiento** del sistema es el siguiente. Al inicializar el sistema y mientras no esté nadie jugando una partida, se mostrará en la pantalla táctil una foto diferente cada 20 segundos de entre una secuencia de fotos. La cantidad existente de fotos diferentes es de CANT y para mostrar una en concreto existe la rutina ya implementada ***mostrar_foto(número)*** a la que se pasa como parámetro el número de la foto que se quiere enseñar. La secuencia lógica es desde la foto 0 hasta la última, vuelta a la foto 0 para recomenzar la secuencia, y así sucesivamente.

Pero en cuanto el controlador de la ranura de dinero detecta que se ha introducido una moneda de 50 céntimos, mediante la rutina ***iniciar_juego()*** da comienzo la partida, que dura 10 minutos de juego. Esa misma rutina se encarga de tragar la moneda. Durante esos 10 minutos, el usuario irá pulsando en puntos distintos de la pantalla con el fin de hacer blanco a unos gnomos que van apareciendo. Cada vez que se pulsa la pantalla táctil en ese intervalo, hay que llamar a la rutina ya implementada ***pasar_posicion(fila, columna, &puntuación)***, a la que se pasan la fila (un valor entre 0 y 599) y la columna (un valor entre 0 y 799) correspondiente al píxel central implicado en la pulsación y devuelve actualizada la puntuación que va logrando el jugador, en base a sus aciertos o fallos a la hora de hacer blanco. Al comienzo de cada partida, la puntuación inicial es de 0.

Una vez que pasan los diez minutos, acaba la partida y se guarda en memoria la puntuación obtenida en la misma mediante la rutina ***guarda_puntuación(num_partida, puntuación)***, a la que se pasan el número de la partida y la puntuación obtenida en ella. Esas puntuaciones se van guardando en memoria a partir de la dirección DIRPUNT y cada una de ellas ocupa 12 bytes. Tras ello, se queda un minuto a la espera de que se introduzca otra moneda de 50 céntimos. En caso de que no ocurra eso, vuelve a mostrar cada 20 segundos una foto diferente de la secuencia.

Cuando se hayan acabado cien partidas en el sistema, toda la información almacenada en memoria acerca de las puntuaciones obtenidas deberá ser transferida a disco, por medio del DMA. Cuando se programa esta transferencia, se manda un aviso a la pantalla mediante la rutina ***mostrar_aviso_de_copia()***. Al finalizar la transferencia de DMA, si ha ocurrido algún error, el sistema deberá intentar realizar la transferencia una y otra vez, hasta que no ocurra ningún error (para simplificar, supondremos que la transferencia habrá acabado correctamente antes de que pase el minuto en que está sin mostrar las fotos). Sin embargo, si se introduce una moneda de 50 céntimos mientras se está realizando el DMA, no se le hará caso y dicha moneda será devuelta al usuario mediante la rutina ***devolver_dinero()***. Si la transferencia de DMA acaba bien, se indica que el sistema ha acabado de hacer la copia mediante ***copia_finalizada()*** y las nuevas puntuaciones que se vayan logrando volverán a almacenarse a partir de DIRPUNT.

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. Se valorará representar el comportamiento del sistema mediante un autómata de estados y transiciones.

5.- [0,7 puntos] Contesta las siguientes preguntas:

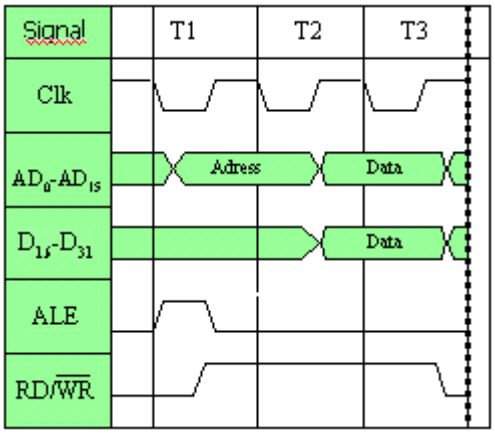
- **a)** ¿Por qué no se comunican directamente los periféricos con la CPU? ¿Qué utilizan para ese fin? ¿Cómo funciona?
- **b)** ¿Qué hacen las funciones `Eoi()` e `Iret()`? ¿Cuál de ellas no se utiliza cuando programamos la rutina de atención al reloj? ¿Por qué?
- **c)** Haz un esquema del vector de interrupciones del PC. ¿Para qué se utiliza? ¿Qué direcciones de memoria se corresponden con la entrada correspondiente a la rutina de atención al teclado?

6.- [1 punto] La siguiente figura presenta las señales y los pasos a seguir para llevar a cabo una lectura a través de un protocolo de bus presentado en clase. Responde a las siguientes preguntas:

(a) Indica qué tipo de protocolo es y cuáles son sus principales características ¿Cuáles son sus principales ventajas y desventajas respecto a otros protocolos estudiados en clase?

(b) Si la frecuencia de reloj es de 500 MHz, ¿cuál es el ancho de banda alcanzado?

(c) ¿Cuánto tiempo se necesita para transferir 20 GB?



7.- [0,5 puntos] Tenemos un computador cuyo procesador trabaja con un reloj de 1 GHz y queremos calcular la sobrecarga que sufre el procesador al tener que realizar una operación de Entrada/Salida por interrupción con un disco duro. El mecanismo de detección de la interrupción, salto a la rutina de atención correspondiente y ejecución de la misma supone un tiempo total de 1.000 ciclos de reloj. El disco duro transfiere datos en bloques de 16 palabras (siendo cada palabra de 64 bits), con una velocidad de funcionamiento de 32 MB/s. El disco está trabajando de manera continua e interrumpe al procesador cada vez que tiene listo un bloque de datos.

8.- [1,5 puntos] Responde a las siguientes preguntas sobre aritméticos.

a) Indica paso a paso cómo se obtiene el resultado de multiplicar los números $A = 11$, $B = -7$ utilizando el algoritmo "suma o salta" estando el multiplicador recodificado según el algoritmo de Booth en base 2. Los operandos están expresados en base 2, complemento a la base y con el menor número posible de bits.

b) Haz la división (C/D) de los siguientes números $C = 23$ y $D = 3$, con restauración. Los operandos están expresados en base 2 con el menor número posible de bits. ¿Qué resto se obtiene?

c) Dibuja el esquema de un sumador *Carry-Select* de 12 bits, teniendo en cuenta que debes utilizar bloques de 3, 4 y 5 bits consecutivamente. Indica cómo se realiza la suma de los siguientes números en ese sumador: $Z1 = EA3_H$, $Z2 = 15D_H$. ¿Qué retardo tiene? ¿Y en el peor de los casos? Justifica tus respuestas