# Arquitectura de Computadores I

Ejercicios de Autoevaluación - 1

**1.- [1 punto]** Contesta las siguientes preguntas.

**a)** ¿En qué consisten la microprogramación vertical, la microprogramación horizontal y la nanoprogramación?

**b)** Queremos ampliar el repertorio de instrucciones del procesador BIRD diseñado en clase para introducir la instrucción "`storepci VARIABLE`". Esta instrucción carga el contenido del registro `pci` en la dirección de memoria definida por `VARIABLE`: `VARIABLE:=pci`. Su formato es el siguiente:

| 6 | 10 | 16 |
|---|---|---|
| C.O. | | VARIABLE |

Escribe la parte del microprograma correspondiente a la ejecución de esa instrucción. En caso de que se deba realizar alguna modificación en la Unidad de Proceso, especificala claramente en el esquema que se proporciona en el cuadernillo de respuestas del examen.

**2.- [1,5 puntos]** Queremos realizar el siguiente cálculo: X=A/B-D*E, con al mínimo número de bits en complemento a dos, para este caso particular: A=27, B=6, D=-5, E=7.

- La división es sin restauración.
- La multiplicación se realiza con Booth en base 2 y el algoritmo de suma o salta.
- La resta final se realiza con un circuito *carry-skip* de 12 bits con RCAs de 4 bits.

Asimismo, responde a las siguientes preguntas:

**a)** ¿Se produce *overflow* en algún momento?
**b)** ¿Cuál es el resto de la división?
**c)** Para este caso particular, ¿se te ocurre alguna forma de calcular la multiplicación de forma más eficiente?
**d)** Dibuja el esquema del sumador-restador para hacer la resta final. Calcula el tiempo necesario para hacer la resta en este caso concreto y en el caso peor.

**3.- [1,5 puntos]** Contesta las siguientes preguntas:

**a)** Haz un esquema del controlador de interrupciones del PC. ¿Qué registros tiene? ¿Para qué se utiliza cada registro? ¿Cuál es el cometido de las señales INT e INTA? Escribe el código de la rutina que inhibe las interrupciones de un periférico que interrumpe por la línea IRQ3.

**b)** ¿Qué sucede cuando se ejecuta la rutina *RecuperaVI*? Escribe el código de esa rutina para el periférico del apartado anterior y la llamada que haría a esa función el programa principal.

*void **RecuperaVI** (int n, unsigned short valIP, unsigned short valCS);*

**NOTA**: Para hacer este ejercicio, dispones de las siguientes definiciones y rutinas ya implementadas:

*#define REG_IRR 0x20* *#define REG_IMR 0x21*

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

*unsigned char **LeeByteFis** (unsigned short Seg, unsigned short Desp);*
*unsigned short **LeePalFis** (unsigned short Seg, unsigned short Desp);*
*void **EscribeByteFis** (unsigned short Seg, unsigned short Desp, unsigned char valor);*
*void **EscribePalFis** (unsigned short Seg, unsigned short Desp, unsigned short valor);*

**4.- [2 puntos]** En un sistema de memoria con direccionamiento al byte, las direcciones para acceder a la memoria tienen la siguiente estructura:

- Memoria virtual paginada, con direcciones lógicas divididas en los campos siguientes:

| 20 bits | 10 bits |
|---|---|

- Memoria principal entrelazada, con direcciones físicas divididas en los campos siguientes:

| 15 bits | 3 bits | 2 bits |
|---|---|---|

Para la traducción de direcciones se dispone de un TLB con tiempo de acceso de 35 ciclos en caso de fallo y 1 ciclo en caso de acierto. El TLB está inicialmente vacío. El tiempo de acceso a MP es de 10 ciclos (1 desde el buffer de entrelazado).

**Se pide:**

**a)** Esquema de traducción de direcciones. ¿Cuántas páginas puede tener un programa? ¿Cuál es el tamaño de una página de memoria principal?

**b)** ¿Cuántas palabras tiene la memoria principal? ¿Cuál es el tamaño en bytes de cada módulo?

**c)** En este computador se ejecuta el siguiente programa:

```
for (i=1;i<65;i++)
      B[i+1]=A[1]*A[i+4];
```

```
        movi r1, #8
        movi r2, #63
        load r3, [r1+4084]
bucle:  load r4, A[r1+12]
        mul r4, r3, r4
        store r4, B[r1]
        addi r1, r1, #4
        subi r2, r2, #1
        bge r2, bucle
```

El programa comienza en la dirección lógica 1020. El vector A comienza en la dirección lógica 4088 y el vector B en la dirección lógica 2048. Las instrucciones y los elementos de los vectores son de tamaño 1 palabra.

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: | 1 | 3 | 5 | 0 | 8 | 4 | 2 | .... |
| Página física: | 0 | 10 | 6 | 1 | 7 | 3 | 2 | .... |

**d)** En el proceso de traducción de direcciones, ¿por qué se utiliza el TLB? ¿Cuál será el contenido del TLB tras el proceso de traducción de las referencias del apartado c?

**e)** Si suponemos un sistema de memoria segmentado, en el que el programa está formado por dos segmentos (instrucciones y datos), ¿cuántos ciclos son necesarios para traducir todas las referencias del programa? También en este caso existe un TLB con los mismos tiempos de acceso.

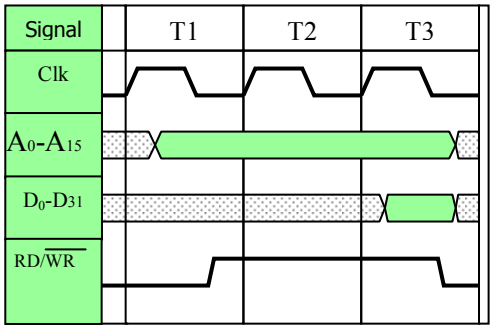
**5.- [0,5 puntos]** Tenemos un computador cuyo procesador trabaja con un reloj de 1 GHz y queremos calcular la sobrecarga que sufre el procesador al tener que realizar una operación de Entrada/Salida E/S por interrupción. El mecanismo de detección de la interrupción, salto a la rutina de atención correspondiente y ejecución de la misma supone un tiempo total de 1.500 ciclos de reloj. El disco 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

Conviene recordar que la sobrecarga que sufre el procesador en una operación de E/S es el porcentaje de tiempo que le dedica a la operación de E/S frente al tiempo total que tiene.

**6.- [1 punto]** Contesta las siguientes preguntas:

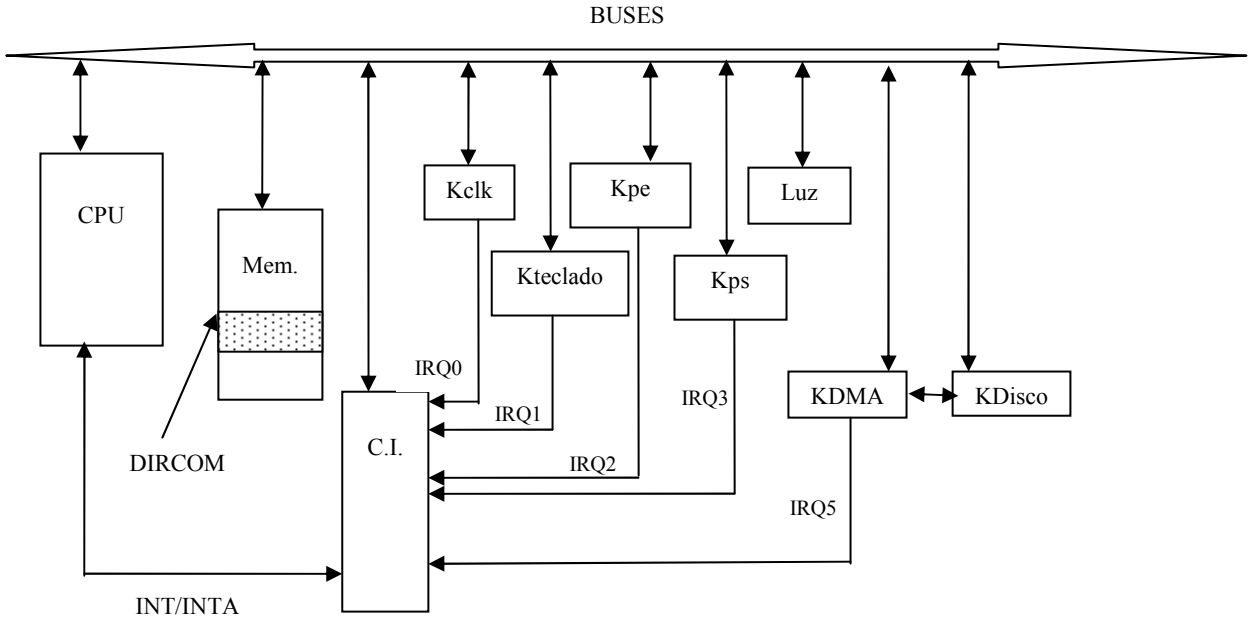
**(a)** Explica brevemente, aunque con suficiente detalle, el protocolo de arbitraje por encuesta (*polling*). Haz un esquema del mismo.

**(b)** Si con el protocolo de esta figura se alcanza un ancho de banda de 66,67 MB/s, ¿cuál es la frecuencia de reloj utilizada? ¿Cuánto tiempo será necesario para transferir un fichero de 150 GB?




**(c)** Teniendo en cuenta el protocolo del apartado (b), si la multiplexación del bus de direcciones supone aumentar el ciclo de bus a 4 ciclos de reloj, ¿merece la pena el cambio?

**7.- [2,5 puntos]** Se desea diseñar un sistema que gestione una **cámara frigorífica industrial**. Los controladores darán cuenta de la cantidad de alimentos (en kilogramos) contenidos en la cámara, así como de la presencia de alguna persona dentro. El sistema dispone de los dispositivos indicados en la Figura.




Los controladores de reloj, teclado e interrupciones son los mismos que los vistos en clase. Salvo el controlador del **sensor ps,** que se controla **por encuesta**, el resto de controladores se gestionan por interrupción. Las características de los controladores de los periféricos son las siguientes:

**Kpe**: es el controlador de la puerta de entrada. Este dispositivo dispone además de una balanza para pesar los alimentos a introducir. El operario dejará los alimentos en la balanza, ésta los pesa y pulsará el botón de entrada situado junto a la puerta. Al pulsar el botón, permite la entrada de la persona que está a la espera de entrar en la cámara e indica en un registro de datos (RDAT_KPE) el peso de los alimentos introducidos.

Este controlador también dispone de un registro de control (RCON_KPE) que permite bloquear y desbloquear la puerta. Para bloquear la puerta hay que escribir el código 0x07 en el registro de control, mientras que para desbloqueada se escribe el código 0x70 en dicho registro.

**Kps**: es el controlador de la puerta de salida, que sólo se abre hacia fuera. Al igual que la puerta de entrada, posee una balanza para indicar el peso de los alimentos retirados. Al pulsar un botón junto a la puerta, permite la salida de la persona que está a la espera de salir de la cámara e indica en su registro de datos (RDAT_KPS) el peso de los alimentos retirados. Posee un registro de control (RCON_KPS) sobre el que hay que hacer STROBE cada vez que se acepta una petición de este controlador. Como ya se ha comentado, **la sincronización de este periférico se realiza por encuesta.**

**Luz**: es una señal luminosa indicadora de presencia o no de operarios en la cámara. Será verde cuando no haya personas en el interior, por lo que se puede acceder a ella, y será Rojo en caso contrario. Para su gestión posee un registro de control mapeado en memoria.

**KDMA:** 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.

**Kdisco:** 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. Los operarios entrarán y saldrán de uno en uno de la cámara frigorífica, por lo que en el momento en que entre un operario, la puerta de entrada debe bloquearse y ponerse la señal luminosa en rojo, indicando que la cámara está ocupada y no puede entrar nadie más. La puerta de salida nunca se bloqueará. Cuando el operario salga, se pondrá la señal luminosa en verde y se desbloquea la puerta de entrada, para permitir la entrada de otro operario.

Como medida de precaución, para evitar congelarse, un operario puede estar 15 minutos como máximo en el interior de la cámara. Si pasado este tiempo el operario no ha salido, se activará una alarma sonora con la rutina ***activar_alarma().*** La alarma la escuchará todo el personal para que acudan en caso de que la persona haya sufrido algún accidente y no pueda salir por sus propios medios, por lo que la puerta de entrada deberá desbloquearse. En el momento en el que hayan salido todas las personas, el responsable del almacén pulsará la tecla 'E' que activará la rutina ***desactivar_alarma()*** y el sistema volverá a funcionar de la forma indicada.

Por otra parte, se quiere controlar el movimiento de alimentos que se hacen diariamente. Por ello, cada vez que un operario entre o salga de la cámara, hay que guardar en memoria cierta información mediante la rutina ya implementada ***guardar(dirección, peso, ENTRADA/SALIDA)***, que guarda en la dirección que se indica la cantidad de kilos que se introducen (ENTRADA) o se sacan (SALIDA). Esta información ocupa 40 bytes y se almacena a partir de la dirección DIRCOM de memoria. Al acabar la jornada, supondremos que a las 00:00' (12 de la noche) no hay nadie en la cámara, se copiará a disco la información que estaba en memoria mediante DMA. En caso de que haya error al copiar la información a disco, se debe intentar una vez más. Si el error persiste, el programa finalizará mostrando al operador un mensaje de error mediante la rutina ***ErrorDMA()***. Al comienzo de la jornada, la información volverá a almacenarse a partir de DIRCOM.

En lo que respecta a la hora, disponemos de dos rutinas: ***InicializarHora()***, que asigna a la variable *HORA* el valor dado por el sistema operativo en el instante concreto, y ***ActualizarHora()***, que actualiza la variable global *HORA* en un segundo.

**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.