ESTRUCTURA DE COMPUTADORES

Recuperación Tercer examen parcial **16 de junio de 2015**

Apellidos y nombre	DNI	Grupo

Ejercicio 1 (6 Puntos)

Las ventanas de un edificio están gobernadas por un sistema de control cuya interfaz muestra dos registros, (ver Figura 1). El sistema dispone de un sensor de humedad y un sensor de luz, y está conectado a un actuador que le permite subir o bajar la persiana de la ventana. Mediante el indicador de posición (bits P_4 a P_0 del registro de estado) se detecta la posición de la persiana con un valor de 0 a 31, siendo 31 cuando está completamente subida y 0 cuando está completamente bajada.

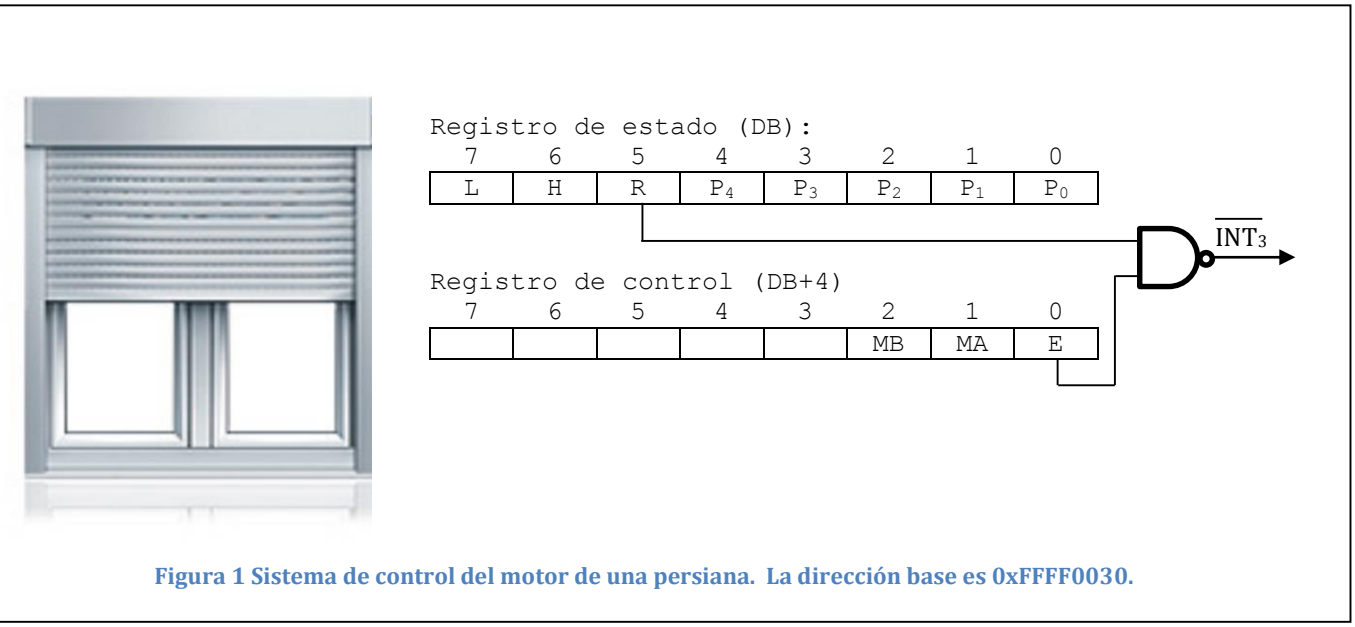


Figura 1 Sistema de control del motor de una persiana. La dirección base es 0xFFFF0030.

Este sistema está conectado a un computador MIPS R2000 a través de una interfaz apropiada. Los registros de esta interfaz se encuentran ubicados a partir de la dirección base 0xFFFF0030 y su descripción es la siguiente:

- Registro **ESTADO** (DB):
  - Bits 4 a 0 - **Indicador de posición (sólo lectura):** valor de 0 a 31.
  - Bit 5 - **R (lectura/escritura)**: La interfaz lo pone a 1 cuando hay un cambio en alguno de los bits H o L o en los del indicador de posición, para cancelarlo se escribe un cero.
  - Bit 6 - **H (sólo lectura)**: Se pone a 1 cuando detecta lluvia y se pone a 0 cuando deja de llover.
  - Bit 7 - **L (sólo lectura)**: Se pone a 1 cuando hay luz y a 0 cuando no la hay.

- Registro **CONTROL** (DB+4) :
  - Bit 0 - **E**: Se pone a 1 para activar la interrupción en la interfaz y a 0 para inhibirla. Cuando E es 1 y el bit R también, se activa la interrupción 3 del procesador (INT_3 *).
  - Bit 1 - **MA**: Se pone a 1 para subir la persiana (si ya está subida no tiene efecto). El sistema detiene automáticamente el motor cuando ha terminado de subirla.
  - Bit 2 - **MB**: Se pone a 1 para bajar la persiana (si ya está bajada no tiene efecto). El sistema detiene automáticamente el motor cuando ha terminado de bajarla.
    Poniendo a cero ambos bits MA y MB paramos el motor.

1. (0.5 puntos) Programe las instrucciones que habilitan las interrupciones en la interfaz descrita, habilita la línea de interrupción 3 en el procesador y deja el procesador en modo usuario e interrupciones generales habilitadas. Los demás bits del registro de estado del procesador deben quedar inalterados. En la Figura 2 aparece información del formato de los registros del coprocesador cero.

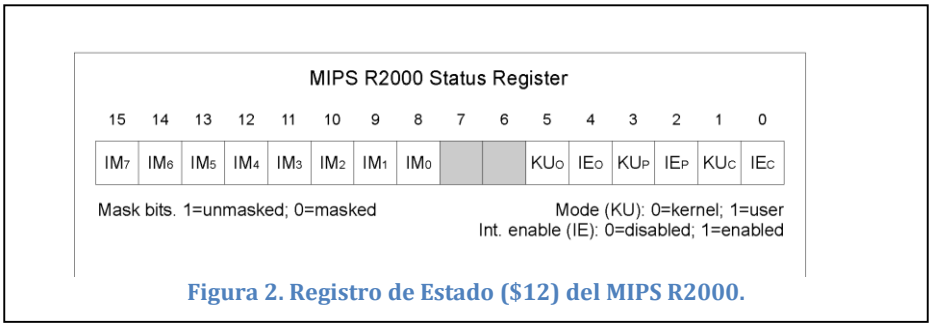


Figura 2. Registro de Estado ($12) del MIPS R2000.

```
        la $t0, 0xFFFF0030
        li $t1, 0x1             #  bit E = 1
        sb $t1, 4($t0)
        mfc0 $t0, $12           # lee el registro de estado
        ori $t0, $t0, 0x0803    #  IM3=KUC=IEC=1
        mtc0 $t0, $12           # escribe en reg. estado
```

2. (1 punto) Se han implementado las siguientes llamadas al sistema para este sistema de control:

Función	Índice	Argumentos	Resultado
Estado_persiana	$v0 = 20	------	$v0 = posición (0..31)
Posicionar persiana	$v0=21	$a0= posición (0..31)	Mueve la persiana a la posición Suspende el proceso hasta que la ventana se ha posicionado.
Sensor de luz	$v0=22	------	$v0=valor del sensor(0..1)
Sensor de lluvia	$v0=23	------	$v0=valor del sensor(0..1)

Programe una rutina de usuario que utilizando estas llamadas realice lo siguiente:

SI hay luz y no llueve ENTONCES
    Subir persiana
SI NO
    Bajar persiana
FIN SI

```
            li $v0, 22
            syscall
            beqz $v0, bajar
            li $v0, 23
            syscall
            bnez $v0, bajar
            li $v0, 21
            li $a0, 31 #subir
            syscall
            b fin
bajar:      li $v0, 21
            add $a0,$zero,$zero # bajar
            syscall
            jr $ra
```

3. (1 punto) **Sólo P3** Para implementar las llamadas al sistema se han declarado las siguientes variables del sistema (de tipo byte):
   - *pos*: contiene la posición de la ventana (0..31)
   - *luz*: contiene el valor del sensor de luz (0..1)
   - *lluvia*: contiene el valor del sensor de lluvia.(0..1)
   - *espera*: un valor distinto de cero indica si el proceso de usuario está esperando a que se posicione la ventana (llamada 31).
   - *npos*: contiene la posición que desea el usuario cuando hace la llamada 31.

A modo de ejemplo de uso de estas dos últimas variables, a continuación se muestra parte del código de la llamada al sistema 21

```
funcion21:   sb $a0, npos
             lb $t0, pos
             beq $a0, $t0, retexc
             sb $v0, espera #($v0 vale 31 que es distinto de 0)
             jal suspende_este_proc # Suspende el proceso llamador

             la $t0, 0xFFFF0030
             bgt $a0, $t0, subir
             li $t1, 0x5 #MB para bajar y E
             sb $t1, 4($t0)
             b retexc
subir:       li $t1, 0x3 #MA para subir y E
             sb $t1, 4($t0)

             b retexc
```

Completar el código de la función21 para que cuando npos sea menor que pos, escriba en el registro de control el valor necesario para bajar la persiana y si es mayor el de subir (las interrupciones deben permanecer habilitadas).

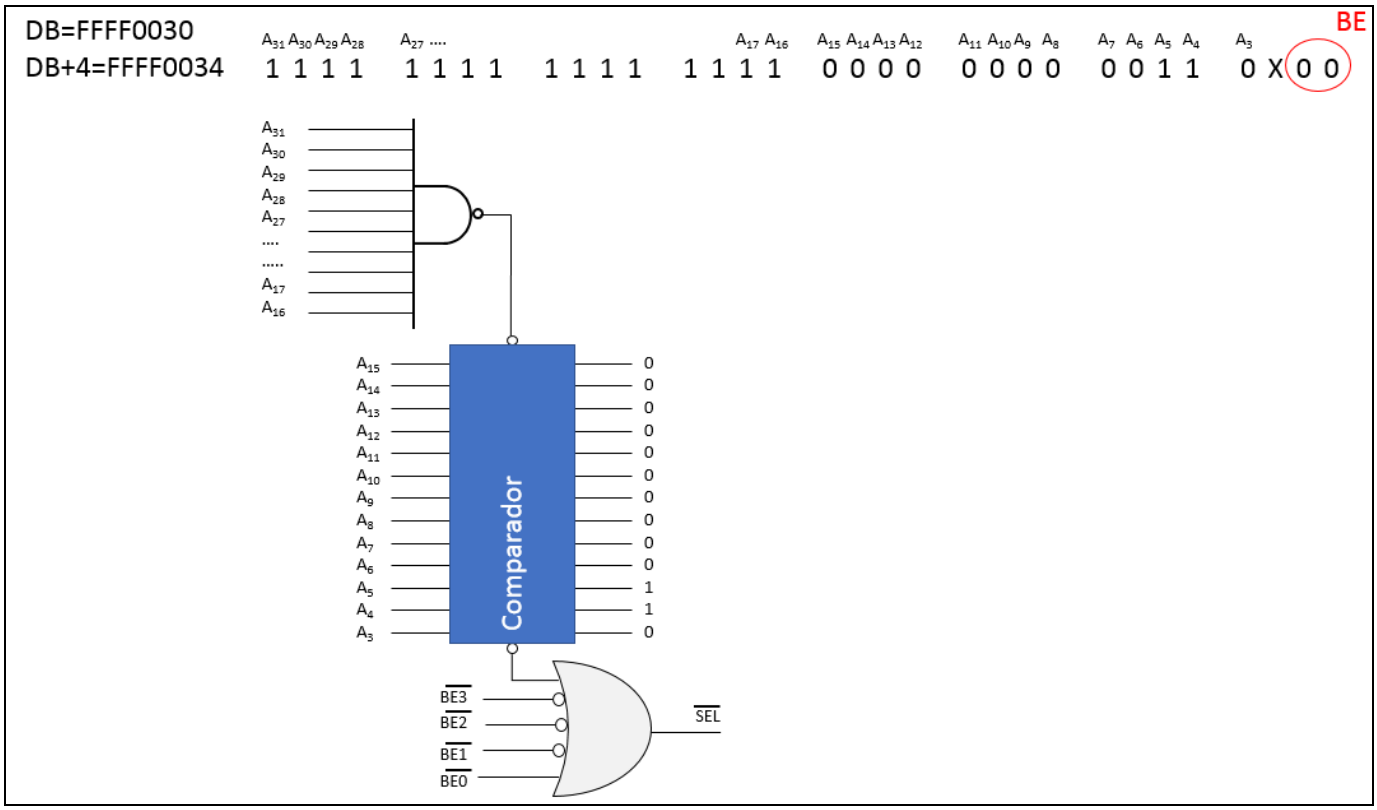
4. (2 puntos) Implementar la rutina de servicio de la INT3 que, utilizando las variables definidas en el apartado 3 realice lo siguiente:
   - Cancele el bit R, lea el registro de estado y actualice las variables *luz, lluvia y pos.*
   - Si *espera* es distinto de cero debe comparar *pos* y *npos*, en caso de que coincidan debe parar la persiana, poner *espera* a 0 y despertar el proceso (utilizando la subrutina *despierta_este_proc*). Las interrupciones deben permanecer habilitadas.

```
Int3:        la $t0, 0xFFFF0030
             sb $zero,0($t0)
             lb $t1,0($t0)
             andi $t0,$t1,0x80
             beqz $t0,escluz #también se puede usar desplazamientos
             li $t0,1
escluz:      sb $t0,luz
             andi $t0,$t1,0x40
             beqz $t0,esclluvia
             li $t0,1
esclluvia:   sb $t0,lluvia
             andi $t0,$t1,0x1F
             sb $t0,pos
             lb $t1,espera
             beqz $t1,retexc
             lb $t1,npos
             bne $t0,$t1,retexc
             sb $zero,espera
             la $t0, 0xFFFF0030
             li $t1,0x1 #MA=0,MB=0,E=1
             sb $t1, 4($t0)
             jal despierta_este_proc # Activa el proceso
             b retexc
```

5. (0,5 puntos) Implementar la llamada al sistema 20 (debe usar la variable del sistema pos):

```
funcion20:   lb $v0, pos
             b retexc
```

6. (1 punto) **Sólo P3** Dibujar el circuito de la función de selección de dispositivo del interfaz de la figura 1, para que la dirección base sea la indicada (0xFFFF0030) y sólo sea accesible con instrucciones lb y sb, utilizar como entradas a vuestra función las líneas de dirección $A_{31}..A_i$ y las líneas $/BE_3../BE_0$. Esta función /SEL debe dar cero cuando la dirección corresponda al registro de estado o al registro de control, y además sea una dirección de byte y 1 en cualquier otro caso.
El circuito utilizará una puerta NAND de 16 entradas y un comparador para el resto de líneas de dirección.



Ejercicio 2 (2.5 Puntos)

Un adaptador de un dispositivo con acceso directo a memoria (ADM) se encuentra conectado a un procesador MIPS R2000 en la dirección base 0xFFFF0050. Los registros de 32 bits de los que dispone son los siguientes:

Dirección	Registro	Comentario
DB	Estado	Bit R en posición 0, sólo lectura
DB+0x4	Ordenes	Lectura DMA poner a uno los bits 1 y 3, para lectura PIO poner a uno sólo el bit 3
DB+0x8	Bloque	Indica el bloque a leer (LBA)
DB+0xC	Puntero	Puntero a la dirección de memoria
DB+0x10	Contador	Número de palabras a transferir
DB+0x14	Datos	Contiene la palabra leída o a escribir para transferencias PIO

1. (1 punto) Escribe el código que permite programar una transferencia de lectura por ADM desde el dispositivo a la memoria. La operación de lectura empieza cuando los bits 1 y 3 del registro *Ordenes* se escriben a uno, y los demás a cero. Se desea leer el bloque cuyo identificador es 0x00B4C000. El tamaño del bloque es de 2048 bytes pero el bus de datos es de 32 bits, por tanto la transferencia se realiza al máximo ancho de banda. El registro contador se decrementa de uno en uno con cada transferencia. La dirección de almacenamiento del bloque en la memoria está etiquetada con Mem_Block.

```
        la $t0, 0xFFFF0050
        li $t1, 512              # Número de transferencias (2048/4)
        sw $t1, 0x10($t0)        # Registro Contador
        la $t1, Mem_Block
        sw $t1, 0xC($t0)         # Registro Puntero
        li $t1, 0x00B4C000
        sw $t1, 0x8($t0)         # Registro Block_ID
        li $t1, 0xA              # Orden lectura, bits 1 y 3 a 1
        sw $t1, 0x4($t0)         # Registro Command
```

2. (1.5 puntos) **Sólo P3** Escribe el código que permite realizar una transferencia por PIO y consulta de estado desde el dispositivo a la memoria. La operación de lectura empieza cuando el bit 3 del registro *Ordenes* se escribe a uno, y los demás a cero. A partir de ese momento cada vez que llegue un nuevo dato se activará el bit R que se cancelará de forma automática al acceder al registro de *Datos* Se desea leer el bloque cuyo identificador es 0x00B4C000. El tamaño del bloque es de 2048 bytes pero el bus de datos es de 32 bits y, por tanto, permite la transferencia de 4 bytes simultáneamente. La dirección de almacenamiento del bloque en la memoria está etiquetada con Mem_Block.

```
        la $t0, 0xFFFF0050
        li $t2, 512              # Número de transferencias (2048/4)
        la $t3, Mem_Block
        li $t1, 0x00B4C000
        sw $t1, 0x8($t0)         # Registro Block_ID
        li $t1, 0x8              # Orden lectura PIO, bit 3 a 1
        sw $t1, 0x4($t0)         # Registro Command
bucle:  lw $t1, 0($t0)
        andi $t1,$t1,0x1         # bit R
        beqz $t1,bucle
        lw $t1,0x14($t0)         # registro de datos
        sw $t1,0($t3)            # escribimos en memoria
        addi $t3,$t3,4           # siguiente dirección
        addi $t2,$t2,-1          # una palabra menos
        bnez $t2,bucle
```

Ejercicio 3 (1.5 Puntos)

La Figura 3 muestra la estructura de los buses de un determinado computador. Todos los anchos de banda indicados son efectivos, exceptuando los buses SATA, los cuales codifican un byte en 10 bits. El disco duro almacena un archivo que contiene una película de 3 minutos comprimida en sistema MPEG-2, a 10 Mbps. La película descomprimida está formada por escenas de 1280×720×24 bits a 60 escenas/segundo; y audio 5.1 a 16 bits por canal y muestreada a 44 KHz. Para completar la reproducción de la película se superpone una música de audio estéreo (de las mismas características que la del vídeo) codificada en un archivo de 3 minutos MP3 a 192Kbps que se encuentra en el CD, generando conjuntamente un sistema de audio 7.1

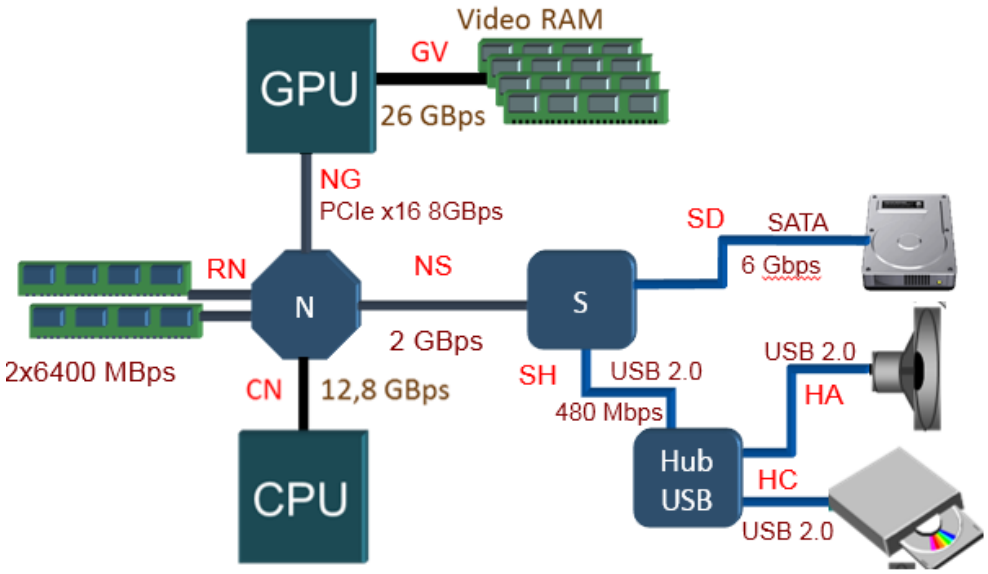


Figura 3. Estructura de los buses y periféricos de un computador.

1. (0,5 puntos) **Sólo P3** Calcule el tamaño de la película comprimida y del audio MP3 del CD. Justifique los cálculos.

```
Película: 10 Mb/s × 180 s = 1800 Mb = 225 MB
Audio CD: 192Kb/s x 180 s = 34,56 Mb = 4,32 MB
```

2. (1 punto) Para reproducir la película se lee simultáneamente el MPEG-2 del HD y el MP3 del CD y se llevan a memoria. La GPU lee de memoria, descomprime el vídeo y lo pasa a la Video RAM, y descomprime los audios para enviar el 7.1 al sistema de audio. Todo esto se hace de manera **concurrente y en tiempo real** (el proceso total tarda 3 minutos).

Calcule la velocidad en MBps requerida por los flujos que se indican a continuación.

```
MPEG2 = 10/8=1,25 MBps
MP3 = 0,192/8=0,024 MBps
VideoDesc = 1280x720x24/8x60= 165,888 MBps
Audio7.1Desc = (7+1)*0,044*16/8=0,704 MBps
```

Calcule el porcentaje de utilización de los buses que se indican a continuación.

```
Bus RN = (MPEG2×2+MP3x2)/12800 = 0,02%
Bus NS = (MPEG2+MP3+Audio71Desc)/ 2000  = 0,10%
Bus SH = (MP3+ Audio71Desc)/(480/8)= 1,21%
Bus NG = (MPEG2+MP3+Audio71Desc)/8000 = 0,02%
BUS GV = VideoDesc/26000 = 0,64%
```