Estructura de Computadores

Rec Parcial 2 **2019**

Nombre: **Grupo**

1 ***(3 puntos)*** Un sistema basado en procesador MIPS R2000 posee una cache L1 dual configurada como sigue:

- **Cache de Instrucciones**: 256B, correspondencia asociativa por conjuntos de 2 vías, tamaño de bloque de 16 Bytes y usa FIFO para los reemplazos
- **Cache de Datos**: 256B, correspondencia directa, tamaño de bloque de 16 bytes. Emplea ubicación en escritura (*write-allocate*) con actualización posterior (*write-back*).

a) *(0.5 puntos)* Indique el número de bits de los campos de la dirección de memoria para ambas caches

Cache de Instrucciones	
Etiqueta	25
Conjunto	3
Desplazamiento	4

Cache de Datos	
Etiqueta	24
Línea	4
Desplazamiento	4

b) *(0.5 puntos)* Calcule el tamaño de la memoria de control requerido por cada una de las caches

	Cache de Instrucciones	Cache de Datos
Número de entradas en la memoria de control	16	16
Número de bits de cada entrada (indique el nombre de los campos)	27 (25 Etiq + 1 V + 1 FIFO)	26 (24 Etiq + 1 V + 1 M)
Tamaño total de la memoria de control (en bits)	16x27=432	16*26=416

c) El siguiente programa ordena el array v1 y deja el resultado en el array v2. Para ello, en cada una de las 8 iteraciones del bucle b1 calcula el elemento que debe ir en la posición i, comparándolo (bucle b2) con los i-1 elementos restantes, por lo que el bucle b2 dará 7, 6, 5, 4, 3, 2, 1 y 0 vueltas, respectivamente, en cada una de las 8 iteraciones del bucle b1.

```
        .data 0x10000000
v1:     .word 34,21,56,48,4,120,17,65   # 8 palabras (32 bytes)
v2:     .space 32               # 8 palabras

        .text 0x00400000
__start:lui $t0,0x1000          # Puntero a v1
        lui $t1,0x1000
        addi $t1,$t1,32         # Puntero a v2
        li $t2,8                # contador bucle externo
b1:     lw $t3,0($t0)           # min=primer elemento de la interación i
        or $t7,$zero,$t0        # dir min
        or $t8,$zero,$t3        # copiamos valor del primer elemento
        addi $t5,$t0,4          # dirección del elemento a comparar
        addi $t4,$t2,-1         # contador bucle interno
b2:     beq $t4,$zero,fin       # si no hay más elementos a comparar, fin
        lw $t6,0($t5)           # lectura siguiente elemento a comparar
        bge $t6,$t3,mayor
        or $t3,$zero,$t6        # nuevo min
        or $t7,$zero,$t5        # dir min
mayor:  addi $t5,$t5,4          # pasamos al siguiente elemento
        addi $t4,$t4,-1
        b b2
fin:    sw $t3,0($t1)           # almacenamos el min en v2
        sw $t8,0($t7)           # copiamos el primer elemento de la
                                # iteración i en el lugar del mínimo hallado
        addi $t2,$t2,-1
        addi $t1,$t1,4
        addi $t0,$t0,4
        bne $t2,$zero,b1        # Sigue iterando hasta fin de v1
```

c.1) *(0.8 puntos)* Obtenga, para la **cache de instrucciones,** sabiendo que para los datos del enunciado el programa ejecuta **315 intrucciones.**

Número de bloques de código	6		
	Nº bloque	**Etiqueta**	**Conjunto**
Primer bloque	0x0040000	0x0008000	0
Último bloque	0x0040005	0x0008000	5
Total de FALLOS de código (justifique)	Un fallo por bloque: 6 Fallos		
Tasa de ACIERTOS (Con cuatro dígitos decimales. Indique el cálculo)	(315-6)/315=0,9810 98,10%		

c.2) *(0.4 puntos)* Indique los números de bloque del primer y último bloque correspondientes a los dos arrays, así como las líneas en las que se almacenan en la **cache de datos (en hex)**

	Primer bloque	**Último bloque**	**Primera línea**	**Última línea**
v1	0x1000000	0x1000001	0	1
v2	0x1000002	0x1000003	2	3

c.3) *(0.5 puntos)* Calcule (indicando los cálculos) para la **cache de datos**:

Total de ACCESOS	8x(1 lw + 2 sw)+(7+6+5+4+3+2+1)x(1 lw)=52
Total de FALLOS	**Fallos de Inicio:** 4 de inicio **Fallos de colisión/capacidad:** 0
Tasa de aciertos (Con cuatro dígitos decimales. Indique el cálculo)	(52-4)/52=0,9230=92,30%
Número de reemplazos	0 (todos los bloque se almacenan en cache sin necesitar reemplazos)

- c.4) *(0.3 puntos)* Si cambiásemos la cache de datos a escritura directa con política de no ubicación (*write-through no-allocate*), ¿de qué manera afectaría a los resultados del apartado c.3? Justifique la respuesta.

Todos los sw del vector v2 serían fallos ya que ese array nunca se lee, por tanto el número de fallos pasaría a ser de 10, y la tasa de acieros (52-10)/52=80,77%

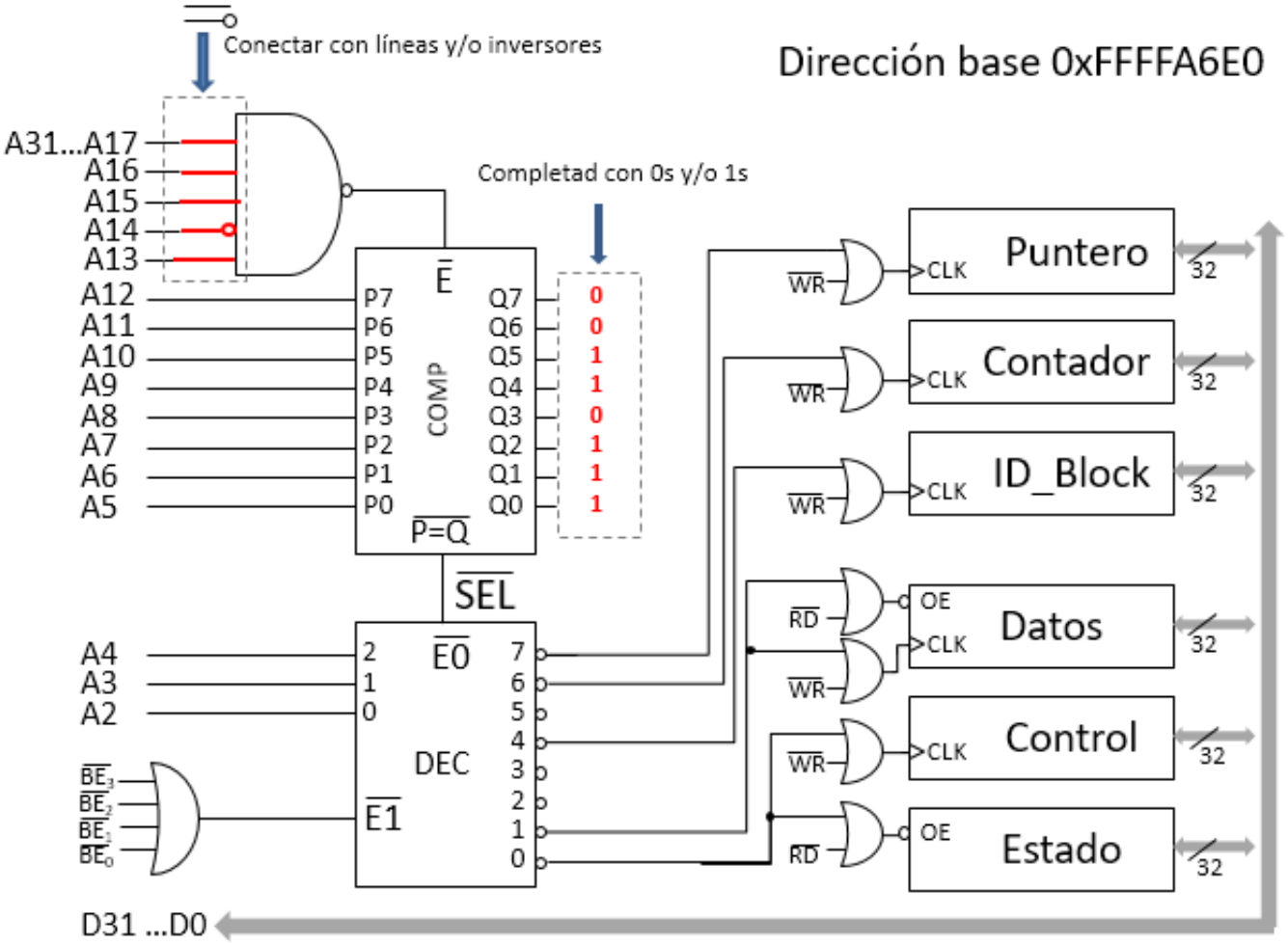
2 ***(2.5 puntos)*** La figura muestra el esquema de una interfaz de un disco magnético. Esta interfaz se conecta a una CPU MIPS R2000. El disco soporta únicamente transferencia por DMA. Los registros Estado y Control poseen los siguientes bits significativos:

Registro **CONTROL**:

- **A** (bit 0), a 1 ordena al interfaz el inicio de una operación de lectura/escritura sobre el disco magnético
- **IE** (bit 1) a 1 habilita la interrupción INT3*. Si IE=1, la interrupción se emitirá cada vez que R sea igual a 1
- **R/W** (bit 2), indica al interfaz si se trata operación de lectura (R/W= 0) o de escritura (R/W=1) sobre el disco magnético
- **CL** (bit 3), a 1 cancela R (R=0)

Registro **ESTADO**:

- **R** (bit 0) se activa a 1 cuando la transferencia del bloque a/desde memoria ha concluido



***Nota**: Todos los registros son de tamaño 32 bits*

a) *(0.5 puntos)* Complete el esquema para que se ajuste a la dirección base indicada.

b) *(0.3 puntos)* Calcule la dirección (DB+X) de cada uno de los registros del interfaz

Registro	Dirección
ESTADO	0
CONTROL	0
DATOS	4

Registro	Dirección
ID_BLOCK	16
CONTADOR	24
PUNTERO	28

c) *(0.3 puntos)* ¿Cuál es la utilidad de las líneas BE en el esquema anterior?

Fuerzan a que los accesos sean de word (lw o sw) si se accede a byte o a half el registro no se seleccionaría

d) *(1.4 puntos)* En el driver del disco magnético controlado a través del interfaz del esquema anterior se define la siguiente función que escribe un bloque en disco:

Función	Índice	Argumentos
wr_block	$v0= 400	$a0: Puntero a buffer de memoria $a1: Identificador del bloque

La sincronización con el dispositivo se realiza por **INTERRUPCIÓN** al nivel de bloque, el tamaño del bloque es de 1024 bytes y las transferencias de DMA son de 32 bits. La función `wr_block` deberá configurar adecuadamente el **DMA,** la operación y habilitar la interrupción en el interfaz. Se pide:

Nota: *Debe tenerse en cuenta que se está en un contexto en el que **múltiples procesos** pueden estar ejecutándose concurrentemente*

d.1) *(0,5 puntos)* Genere el código que implementa la función `wr_bloque`

```
wr_bloque:la $t0,0xFFFFA6E0
          sw $a0,28($t0)
          sw $a1,16($t0)
          li $t1,256 #1024/4
          sw $t1,24($t0)
          li $t1,0x7 #W+IE+A
          sw $t1,0($t0)
          jal suspende_proceso




          j retexc
```

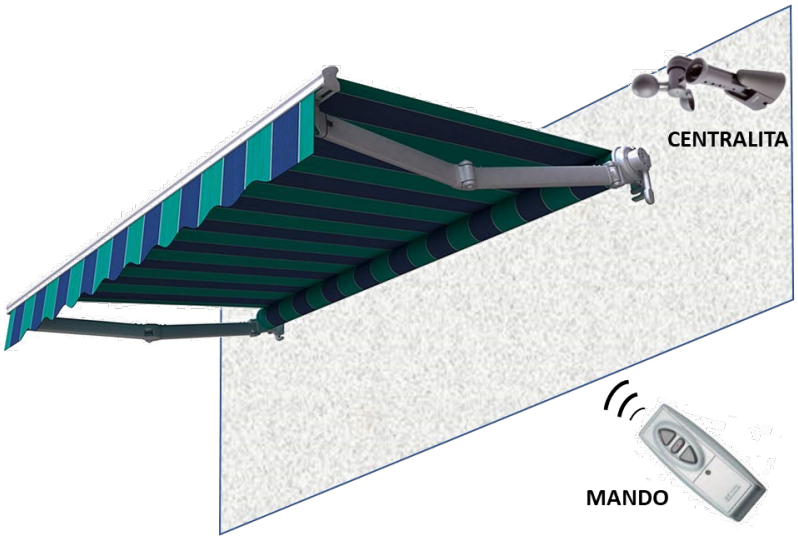
d.2) *(0,5 puntos)* Genere el código que implementa la interrupción 3.

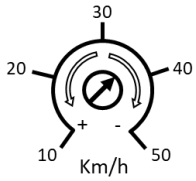
```
Int3:     la $t0,0xFFFFA6E0
          li $t1,0x8 #clear+noint
          sw $t1,0($t0)
          jal activa_proceso




          j retexc
```

d3) *(0.4 puntos)* Escriba un fragmento de código de usuario que invocaría a esta llamada al sistema para escribir el bloque 4334 del disco en la zona de memoria etiquetada como buffer.

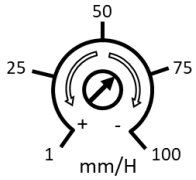
```
        .data
buffer: .space 1024

        .text
         la $a0,buffer
         li $a1,4334
         li $v0,400
         syscall
```

3 ***(3 puntos***) La figura adjunta muestra un toldo de brazo articulado motorizado y una centralita de control que dispone de tres sensores: de viento, de lluvia y de luminosidad, así como de un mando a distancia. La centralita tiene un procesador embebido (tipo MIPS) que gestiona la apertura/cierre del toldo siguiendo las indicaciones del mando a distancia o según el estado del tiempo. Se dispone de tres periféricos: **SENSORES**, **MANDO A DISTANCIA** y **MOTOR TOLDO**, los cuales se describen a continuación.

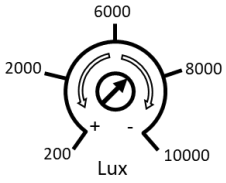




SENSOR DE VIENTO: Se trata de un anemómetro que mide la velocidad del viento (km/h) en tiempo real. Permite ajustar un valor de referencia mediante un potenciómetro situado en el sensor. Valor inicial: 30 Km/h. Cuando dicha velocidad supera el valor de referencia, el sensor emite una señal de alarma que servirá para cerrar el toldo.



SENSOR DE LLUVIA: Ídem que el de viento, pero mide la magnitud de la lluvia en mm/H. Referencia inicial: 30 mm/H. Cuando la intensidad de la lluvia supera el valor de referencia, el sensor emite una señal de alarma que servirá para cerrar el toldo.



SENSOR DE LUMINOSIDAD: Mide la intensidad de la radiación solar (lux). Permite ajustar dos valores de referencia: uno para detectar exceso de radiación solar <Ref. inicial: 3000 lux> y otro para detectar el crepúsculo y noche <Ref. inicial: 200 lux> Cuando la radiación solar supera el valor establecido en la primera referencia, el sensor emite una alarma para abrir el toldo. Si, por el contrario, el valor baja por debajo de la segunda referencia, entonces emite una alarma para cerrar el toldo.

Los tres sensores se gestionan a través de una única interfaz formada por 2 registros, situados a partir de DB = 0xFFFF2000

- Registro **ESTADO**: DB + 0 (8 bits: sólo lectura)
  - R (bit 0): se activa a 1 cuando se recibe una alarma procedente de alguno de los sensores. Se cancela (R=0) mediante el bit CL del registro de control. SI IE=1 emite una interrupción, asociada a la INT2 del MIPS.
  - IND (bit 4): Un '1' indica alarma por viento.
  - RAIN (bit 5): Un '1' indica alarma por lluvia.
  - SUN (bit 6): Un '1' indica alarma por radiación solar
  - NIGHT (bit 7): Un '1' indica Noche o Crepúsculo.
- Registro **CONTROL**: DB + 4 (8 bits; sólo escritura)
  - IE (bit 7): Un '1' habilita la interrupción
  - CL (bit 6): Un '1' cancela el bit R (R=0).

MANDO A DISTANCIA: Se trata de un mando con tres botones <Subir / Parar / Bajar>. Cada vez que se pulsa un botón se transmite un código a la Centralita por radiofrecuencia. Si se pulsan varios botones a la vez, no hace nada. La interfaz de este periférico dispone de 2 registros situados en DB=0xFFFF1000.

- Registro **ESTADO**: DB + 0 (8 bits: sólo lectura)
  - R (bit 7): se activa a 1 cada vez que se pulsa alguno de los botones del mando a distancia y envía el código correspondiente. Se cancela (R=0) cuando se lee el registro de Datos.
- Registro **DATOS**: DB + 4 (8 bits; sólo lectura)
  - UP (bit 4): Un '1' indica código del botón Subir.
  - DOWN (bit 5): Un '1' indica código del botón Bajar
  - STOP (bit 6): Un '1' indica código del botón Parar

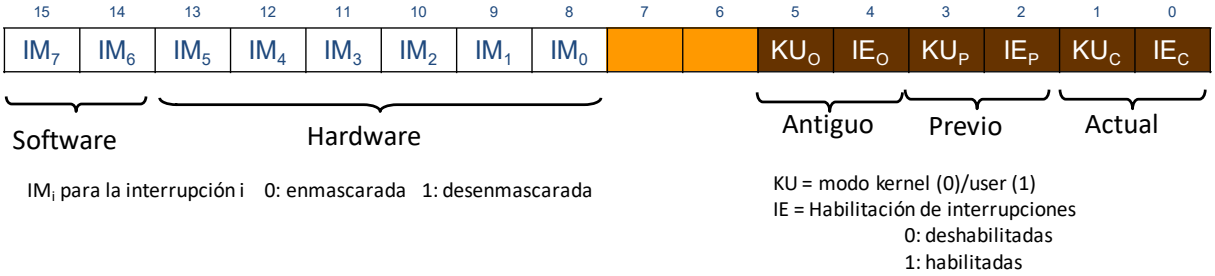
MOTOR TOLDO: Periférico de E/S directa que acciona el motor para subir o bajar el toldo. También dispone de dos finales de carrera para detener el motor en dos posiciones determinadas (0: toldo enrollado; 1: toldo extendido). Como elemento de seguridad, el motor siempre se detiene automáticamente tras 3 minutos. La interfaz de este periférico dispone de un solo registro situado en DB=0xFFFF3000.

- Registro **MOTOR**: DB + 0 (8 bits)
  - FC0(bit 0, lectura): Un '1' indica que se ha llegado al final de carrera 0 (enrollado)
  - FC1(bit 1, lectura): Un '1' indica que se ha llegado al final de carrera 1 (extendido)
  - UP (bit 4, escritura): Un '1' activa subir toldo. Se detendrá en 3 min. o en el final de carrera 0.
  - DO (bit 5, escritura): Un '1' indica bajar toldo. Se detendrá en 3 min. o en el final de carrera 1.

El programa principal del sistema se basa en una inicialización y un bucle sin fin:

```
          .text 0x00400040
__start:  ……                    # Código de inicialización del sistema
          ……
          jal espera_3_min       # rutina que espera 3 minutos

Bucle:    ……                     # esperar código del mando a distancia
             ….                  # actuar según el código
          j Bucle                # bucle sin fin
```

La figura adjunta muestra el contenido del Registro de Estado ($12) del MIPS



a) *(0.5 puntos)* Escriba el código de inicialización al arranque del sistema, que habilite las interrupciones necesarias, recoja el toldo y deje la CPU en modo 'kernel'.

```
__start:    la $t0,0xFFFF2000 # DB de sensores
            li $t1,0x80
            sb $t1,4($t0)     # E=1 en sensores
            li $t1, 0x0401
            mtc0 $t1,$12      # Desenmascara int 2, habilita interrupciones y modo kernel
            la $t0, 0xFFFF3000
            li $t1, 0x10
            sb $t1, 0($t0)    # Sube el toldo si o si


            jal espera_3_minutos
```

b) *(0.5 puntos)* Escriba las subrutinas para subir, bajar o parar el motor. No se debe activar el motor para subir si el toldo ya está arriba. No se debe activar el motor para bajar si el toldo ya está abajo.

```
Subir_toldo:
        la $t0, 0xFFFF3000
        lb $t1, 0($t0)
       andi $t1, $t1, 0x01
       bnez $t1, fin_subir
        li $t1, 0x10
       sb $t1, 0($t0)



fin_subir:  jr $ra
```

```
Bajar_toldo:
        la $t0, 0xFFFF3000
        lb $t1, 0($t0)
       andi $t1, $t1, 0x02
       bnez $t1, fin_bajar
        li $t1, 0x20
       sb $t1, 0($t0)



fin_bajar:  jr $ra
```

```
Parar_toldo:
        la $t0, 0xFFFF3000
        sb $zero, 0($t0)



fin_parar: jr $ra
```

c) *(0.5 puntos)* Complete el código del bucle sin fin del programa principal. Este fragmento debe esperar a recibir algún código del mando a distancia y actuar en consecuencia: Subir, bajar o parar el toldo. Imagine que puede emplear una nueva pseudo-instrucción de salto a subrutina condicional:

Branch on Equal and Link: ***beql rs, inmed, etiqueta*** Si (rs == inmediato) { jal etiqueta}

```
Bucle:  la $t0, 0xFFFF1000
        lb $t1,   0 ($t0)
        andi $t1, $t1, 0x80
        beqz $t1, Bucle
        lb $t1, 4(t0)
        andi $t1, $t1, 0x70
        beql $t1, 0x10, Subir_toldo
```

```
        beql $t1, 0x20, Bajar_toldo
        beql $t1, 0x40, Parar_toldo
        j Bucle
```

d) (0.3 puntos) Escriba el código equivalente de la pseudo-instrucción 'beql' anterior.

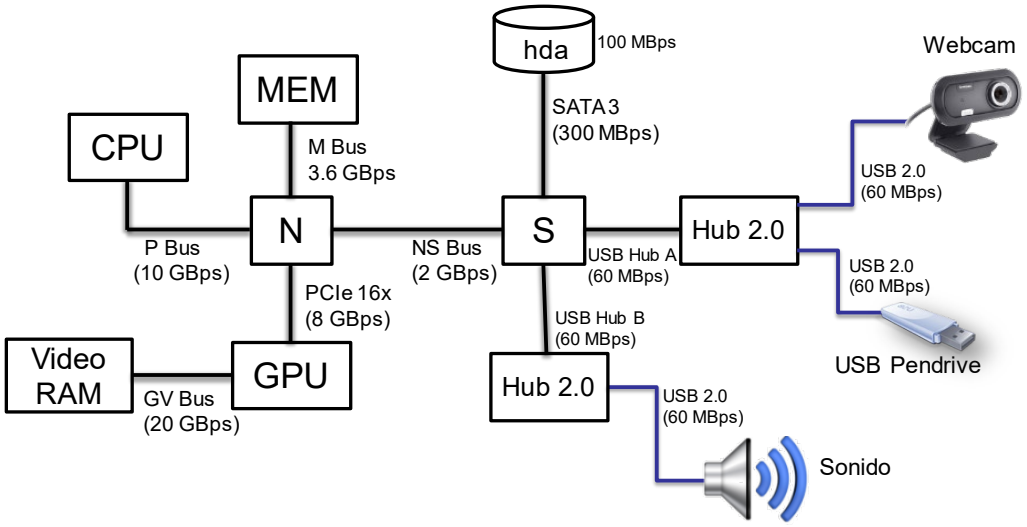
```
beql $rs, inmed, etiqueta
```

```
.set noat

        li $at, inmed
        bne $rs, $at, seguir
        jal etiqueta
    seguir:

.set at
```

e) *(1.2 puntos)* Escriba el código de la rutina de servicio de la interrupción INT2, el cual debe responder adecuadamente a los distintos tipos de alarmas provenientes de los SENSORES. Puede utilizar, si lo desea, la pseudo-instrucción ***beql rs, inmed, etiqueta*** definida anteriormente.

```
Int2:   la $t0, 0xFFFF2000    # DB Sensores
        li $t1, 0xc0
        sb $t1, 4($t0)        # Cancela, int habilitada

        lb $t1, 0($t0)

        andi $t2, $t1, 0x10
        beql $t2, 0x10, Subir_toldo    # Viento
        b retexc
        andi $t2, $t1, 0x20
        beql $t2, 0x20, Subir_toldo    # Lluvia
        b retexc
        andi $t2, $t1, 0x40
        beql $t2, 0x40, Bajar_toldo    # Sol
        b retexc
        andi $t2, $t1, 0x80
        beql $t2, 0x80, Subir_toldo    # Noche

        b retexec
```

4 ***(1.5 puntos)*** Un alumno aplicado está grabando su presentación del Proyecto Fin de Grado en su computador, cuyo esquema de conexión se muestra en la figura adjunta. La imagen y sonido lo adquiere de una Webcam USB 2.0 que lo transmite en formato MPEG 4 codificado a 42 Mbps. El vídeo codificado se transfiere a la memoria (MEM) del computador por ADM. Al mismo tiempo, la GPU lee la secuencia codificada desde memoria, la descomprime y envía las imágenes a la RAM de Vídeo, y el audio al equipo de sonido. Simultáneamente, la CPU está tomando el vídeo codificado y lo está grabando en un archivo situado en el Pendrive.



***Nota**: Todos los anchos de banda mostrados en el esquema son efectivos*

a) *(0.5 puntos)* Suponiendo que el vídeo descomprimido tiene una resolución de 1280x720x24 bits y 30 escenas por segundo y que el sonido es estéreo (audio 2.0), con muestreo a 48 KHz y 16 bits/muestra, calcule el ancho de banda (en MBps) requerido para:

Transferir el video comprimido desde la webcam a la memoria (MEM):
42 Mbps / 8bits = 5,25MBps
Escribir las imágenes de vídeo desde la GPU a la RAM de vídeo:
1280×720× (24/8Bytes) × 30fps = 82,944 MBps
Enviar el audio desde la GPU al equipo de sonido:
2 canales × 48000 muestras por segundo × 16/8 bytes por muestra = 0.192 MBps

b) *(0.25 puntos)* Indique la ocupación (%) de los buses siguientes durante la grabación:

Bus USB Hub A: (5,25 + 5,25) / 60 = 17,5%
Bus USB Hub B: 0.192 / 60 = 0,32%
GV Bus: 82,944 / 20000 = 0,41%
Bus NS: (5,25 + 5,25+ 0,192) /2000= 0,5346%
Bus M: (5,25 + 5,25+ 5,25) /3600= 0,4375%

c) *(0.25 puntos)* Indique cuánto ocupará el archivo del videoclip grabado si tuviera una duración de 15 minutos. Indíquelo en MB

5,25 MB/s x 60 s/min x 15 min= 4725 MB

d) *(0.25 p)* Indique cuánto ocuparía el mismo videoclip si se hubiera grabado **sin comprimir** (audio + vídeo). Indíquelo en GB.

(82,944 + 0,192) x 60s/min x 15 min= 74,822 GB

e) *(0.25 puntos)* Asumiendo que tanto el disco duro (**hda**) y el pendrive tienen espacio suficiente ¿en cuál se debería hacer la grabación del **videoclip sin comprimir**? Razone la respuesta.

En el disco dura (hda) pues el Pendrive no permite tanto ancho de banda (82.944 MBps vídeo + 0.192 MBps audio)