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

* **Cache de Instrucciones**: 512B, correspondencia directa, tamaño de bloque de 16 Bytes
* **Cache de Datos**: 1024B, correspondencia asociativa por conjuntos de 4 vías, tamaño de bloque de 16 bytes, algoritmo de remplazo LRU. Emplea política de escritura directa sin ubicación (*write-through no-allocate*).

1. *(0.5 puntos)* Indique el número de bits de los campos de la dirección de memoria para ambas caches

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cache de Instrucciones** | |  | **Cache de Datos** | |
| **Etiqueta** |  |  | **Etiqueta** |  |
| **Línea** |  |  | **Conjunto** |  |
| **Desplazamiento** |  |  | **Desplazamiento** |  |

1. *(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** |  |  |
| **Número de bits de cada entrada (indique el nombre de los campos)** |  |  |
| **Tamaño total de la memoria de control (en bits)** |  |  |

1. El siguiente programa calcula el producto de la matriz *ma* por el vector *v1* y deja el resultado en el vector *v2*. La matriz *ma* tiene dimensión de 4x4 words, con cada una de las filas almacenadas en direcciones consecutivas de memoria, mientras que los vectores *v1* y *v2* son arrays de 4 words. Se hacen dos bucles anidados de 4 iteraciones cada uno, de forma que cada componente del resultado es el producto escalar de una fila de *ma* por el vector *v1.*

.data 0x10000000

ma: .word 34,21,56,48 # 4 filas x 4 col

.word 4,120,17,65

.word 10,27,5,6

.word 5,12,1,5

.data 0x10010000

v1: .word 2,4,6,8

.data 0x10020000

v2: .space 16 # 4 palabras

.text 0x00400000

\_\_start:lui $t0,0x1000 # Puntero a ma

lui $t2,0x1002 # Puntero a v2

ori $t3,$zero,4 # contador bucle externo b1

b1: ori $t4,$zero,4 # contador bucle interno b2

lui $t1,0x1001 # Puntero a v1

or $a0,$zero,$zero # inicializamos producto a cero

b2: lw $a1,0($t0) # componente ma(i,j)

lw $a2,0($t1) # componente v1(j)

mult $a1,$a2 # producto

mflo $a1 # resultado ma(i,j)\*v1(j)

add $a0,$a0,$a1 # acumulamos resultado

addi $t4,$t4,-1 # contador bucle interno b2

addi $t0,$t0,4 # incrementamos punteros ma y v1

addi $t1,$t1,4

bne $t4,$zero,b2 # si no hemos acadado repetimos b2

sw $a0,0($t2) # guardamos v2(i)

addi $t3,$t3,-1 # contador bucle externo b1

addi $t2,$t2,4 # incrementamos puntero v2

bne $t3,$zero,b1 # si no hemos acabado repetimos b1

c.1) *(0.8 puntos)* Obtenga, para la **cache de instrucciones:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Número de bloques de código** |  | | **Instrucciones ejecutadas** | |  |
|  | **Nº bloque** | **Etiqueta** | | **Línea** | |
| **Primer bloque** |  |  | |  | |
| **Último bloque** |  |  | |  | |
| **Total de FALLOS de código (justifique)** |  | | | | |
| **Tasa de ACIERTOS (Con cuatro dígitos decimales. Indique el cálculo)** |  | | | | |

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Primer bloque** | | | **Último bloque** | | |
|  | **Nº bloque** | **Etiqueta** | **Conjunto** | **Nº bloque** | **Etiqueta** | **Conjunto** |
| **ma** |  |  |  |  |  |  |
| **v1** |  |  |  |  |  |  |
| **v2** |  |  |  |  |  |  |

c.3) *(0.5 puntos)* Calcule (indicando los cálculos) para la **cache de datos**: *Para el cálculo del número de fallos recuérdese que se aplica una política de* ***NO-ubicación*** *en escritura.*

|  |  |
| --- | --- |
| **Total de ACCESOS** |  |
| **Total de FALLOS** |  |
| **Tasa de aciertos (Con cuatro dígitos decimales. Indique el cálculo)** |  |
| **Número de reemplazos** |  |

* c.4) *(0.3 puntos)* Si cambiásemos la cache de datos a escritura directa con política de ubicación (*write-through allocate*), ¿de qué manera afectaría a los resultados del apartado c.3? Justifique la respuesta.

2

1. ***(5.5 puntos)*** La figura muestra el esquema del interfaz que controla una cámara de vigilancia. Esta interfaz se conecta a una CPU MIPS R2000 modificada para incluir un mapa separado de direccionamiento de la entrada/salida (I/O-Mapped I/O). El juego de instrucciones de este procesador incorpora instrucciones adicionales para lectura (**inb** / **inw**) y escritura (**outb** / **outw**) en puertos del mapa de E/S. La sintaxis de dichas instrucciones es la misma que la de las instrucciones load y store del mapa de memoria. La cámara soporta transferencias tanto por PIO como por ADM. Los registros Estado y Control poseen los siguientes bits significativos:

Registro **ESTADO** (32 bits):

* Bit 0: **R**: En modo ADM, vale 1 cuando se completa la operación de captura y la imagen se halla almacenada en memoria. En modo PIO, vale 1 cuando el buffer interno del interfaz contiene todos los pixels de la imagen.

Registro **CONTROL** (32 bits):

* Bit 1-0: **RA** (Relación de aspecto)

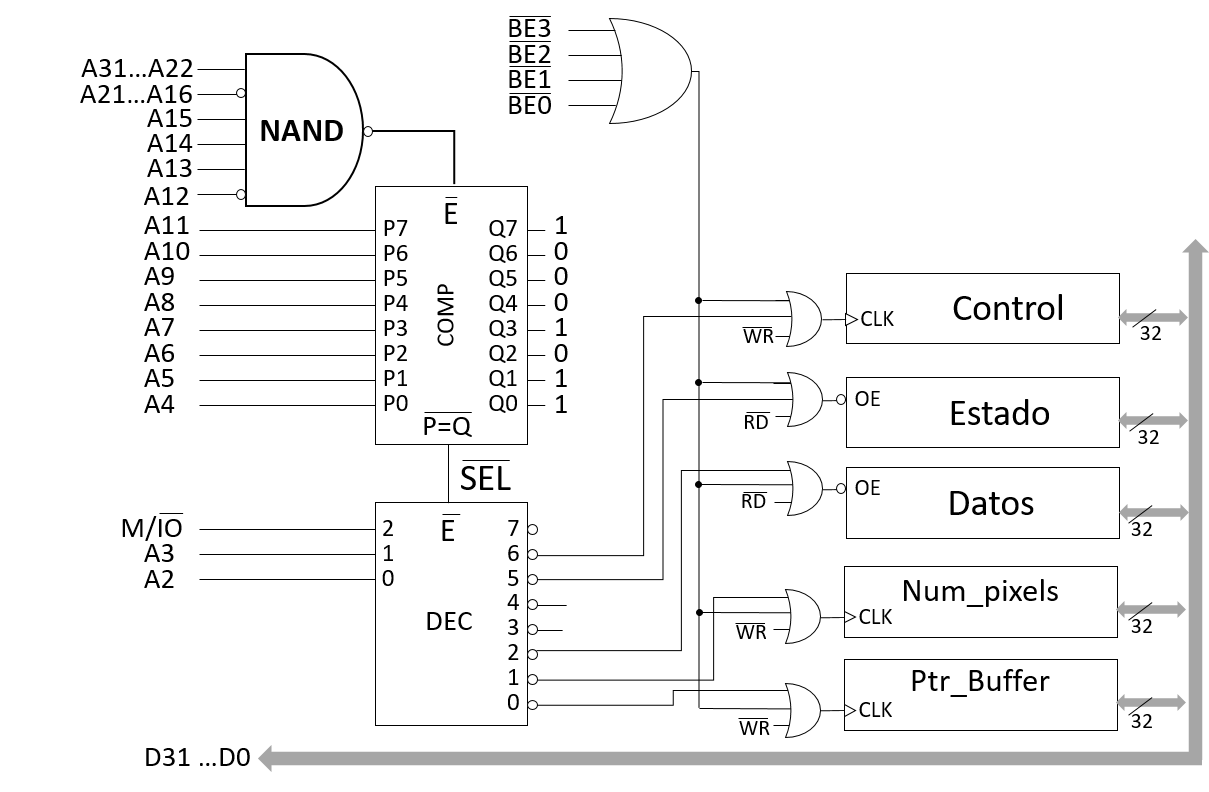
|  |  |
| --- | --- |
| 00 | 1:1 |
| 01 | 4:3 |
| 10 | 3:2 |
| 11 | 16:9 |

* Bit 2: **CL** (cancelación R): un 1 hace que el bit R se ponga a cero
* Bit 3: **IE** : mientras el bit R del registro de estado vale 1, activa la línea de interrupción INT3.
* Bit 4: **M** (modo de transferencia): a 1 indica ADM, a 0 indica PIO
* Bit 6: **B/C** (modo captura): a 0, captura en B/N (pixel – 8 bits), a 1, captura en color (pixel – 32 bits)
* Bit 7: **A** (capturar imagen): al escribir un 1 se ordena la captura de imagen

Registro **DATOS** (32 bits): Se usa en modo PIO para leer los píxeles de la imagen. Permite lectura de datos de 8 y 32 bits

Registro **NUM\_PIXELS** (32 bits): Establece el número de pixels de la imagen a capturar

Registro **PTR\_BUFFER** (32 bits): Sólo se emplea en modo ADM. Contiene la dirección inicial del buffer de memoria en el que se ha de almacenar la imagen.



3

1. *(0.5 puntos)* Cuál es la dirección base del interfaz de la cámara?
2. *(0.5 puntos)* Determine la dirección (DB+X) de cada uno de los registros del interfaz, el mapa en el que se direccionan (MEM o E/S) y las instrucciones con las que se accederían

|  |  |  |  |
| --- | --- | --- | --- |
| **Registro** | **Dirección (DB+X)** | **Mapa direccionamiento** | **Instrucciones** |
| PTR\_BUFFER |  |  |  |
| NUM\_PIXELS |  |  |  |
| DATOS |  |  |  |
| ESTADO |  |  |  |
| CONTROL |  |  |  |

1. *(0.5 puntos)* ¿Cuál es la utilidad de las líneas BE con la que se seleccionan los puertos Num\_pixeles y Ptr\_Buffer en el esquema anterior?
2. *(1 punto)* Programe la llamada al sistema Inicializar que se describe a continuación:

|  |  |  |
| --- | --- | --- |
| Función | Índice | Argumentos |
| Inicializar | $v0= 50 | Configura el registro de control del interfaz con relación de aspecto (1:1) - véase detalle en descripción registro Control, interrupción **inhibida**, modo de transferencia PIO y modo de captura en B/N. Además, habilita la línea de interrupción 3 en el MIPS y deja el procesador en modo usuario e interrupciones generales habilitadas. Los demás bits del registro de estado del procesador deben quedar inalterados |

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



4

1. *(1.5 puntos)* En el driver de la cámara controlada a través del interfaz del esquema anterior se define la siguiente función que lee la imagen capturada por la cámara:

|  |  |  |
| --- | --- | --- |
| Función | Índice | Argumentos |
| leer\_img\_bn | $v0= 100 | $a0: Puntero a buffer de memoria  $a1: Número de píxeles de la imagen  $a2: Relación de aspecto de la imagen – Véase detalle en descripción registro Control |

La sincronización con la cámara se realiza por **CONSULTA DE ESTADO** al nivel de imagen. La función leer\_img\_bn deberá configurar adecuadamente el interfaz para capturar una **imagen en B/N** (tamaño pixel igual a 8 bits) y realizar una **transferencia en modo PIO,** Se pide:

|  |
| --- |
| leer\_img\_bn:    j retexc |

1. *(1.5 puntos)* En el driver de la cámara controlada a través del interfaz del esquema anterior se define la siguiente función que lee la imagen capturada por la cámara:

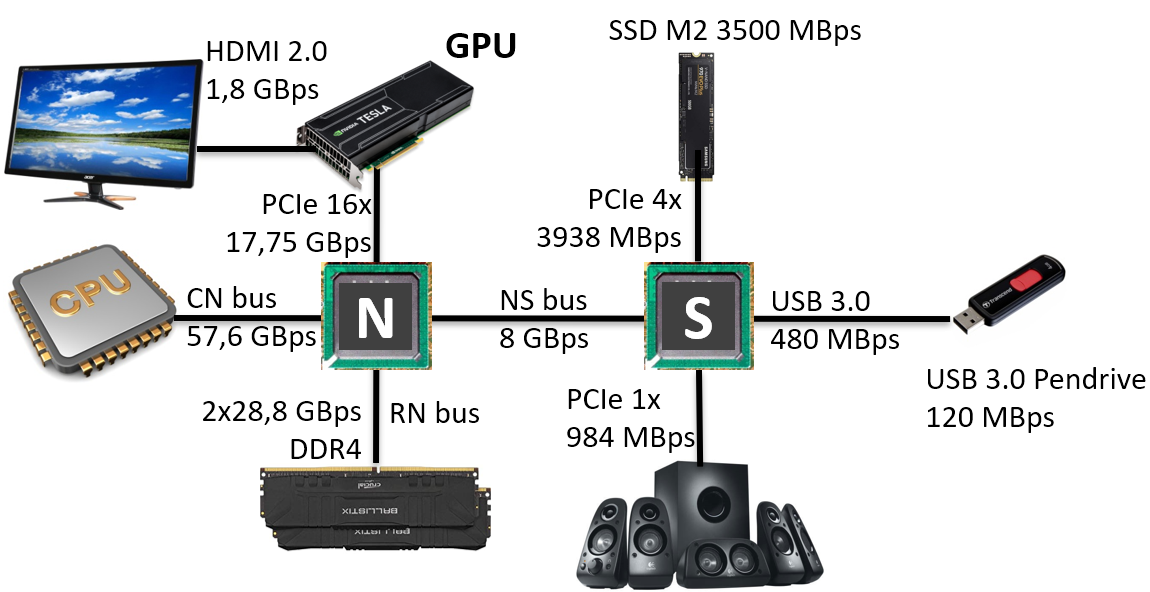
|  |  |  |
| --- | --- | --- |
| Función | Índice | Argumentos |
| leer\_img\_color | $v0= 100 | $a0: Puntero a buffer de memoria  $a1: Número de píxeles de la imagen  $a2: Relación de aspecto de la imagen – 1:1 (0); 4:3 (1); 3:2 (2); 16:9 (3) |

La sincronización con la cámara se realiza por **INTERRUPCIÓN**. La función leer\_img\_color deberá configurar adecuadamente el **ADM,** la operación de captura de **imagen en color** y habilitar la interrupción en el interfaz. En cada ciclo de bus del ADM se transfiere un pixel en B/N (8 bits) o en color (32 bits), según el modo de captura que se haya fijado en el registro de control. Considérese que **múltiples procesos** pueden estar ejecutándose concurrentemente, estando disponibles las funciones fijar\_contexto, suspende\_este\_proceso, y activa\_proceso\_en\_espera.Se pide:

5

|  |  |
| --- | --- |
| f.1) (*0.6 puntos)* Escriba el código que implementa la función leer\_img\_color | f.2) *(0.6 puntos)* Escriba el código que implementa la interrupción 3. |
| leer\_img\_color:  j retexc | Int3:  j retexc |
| f.3) *(0.3 puntos)* Completa el fragmento de código de usuario que invocaría a esta llamada al sistema para ordenar la captura de una imagen en color de 1600x1200 pixels (formato 4:3) y almacenarla en la zona de memoria etiquetada como buffer. | |
| .data buffer: .space s   .text   la $a0, s  li $a1, s  li $a2, s  li $v0, s  syscall | |

1. ***(1.5 puntos)*** El computador del esquema reproduce una película en formato MP4 de 40Mbps almacenada en el pendrive. Para hacer esto, se transfiere por DMA del pendrive a memoria y a la vez de memoria a la GPU (USB3 🡪 M, M 🡪 GPU). La GPU descomprime el vídeo que guarda en su memoria gráfica y el audio se envía al sistema de sonido por DMA (GPU 🡪 M, M 🡪 PCie 1x). Además, se guarda una copia descomprimida del video (sin audio) en el SSD por DMA (GPU 🡪 M, M 🡪 PCIe 4x). Todas las transferencias se hacen de forma sincronizada y de forma simultánea.



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

6

1. *(0.45 puntos)* Suponiendo que el vídeo descomprimido tiene una resolución UHD de 3840x2160x32 bits y 24 escenas por segundo y que el sonido es multicanal (audio 7.1), con muestreo a 48 KHz y 16 bits/muestra, calcule el ancho de banda (en MBps) requerido para:

|  |
| --- |
| Transferir la película comprimida desde el pendrive a la memoria:  Escribir el vídeo descomprimido desde la GPU a la RAM de vídeo:  Enviar el audio desde la GPU al equipo de sonido: |

1. *(0.35 puntos)* Indique la ocupación (%) de los buses siguientes:

|  |
| --- |
| Bus USB 3.0:  Bus PCIe x4:  Bus PCIe x1:  Bus NS:  Bus PCie x16: |

1. *(0.2 puntos)* Indique cuánto ocupará el archivo del video descomprimido si tuviera una duración de 1 minuto. Indíquelo en MB

|  |
| --- |
|  |

1. *(0.3 p)* Mientras se reproduce la película (y se graba el vídeo descomprimido) se va a transferir un archivo de 1GByte (1x109) del pendrive al disco SSD por DMA (USB 3.0 🡪 M, M 🡪 PCIe 4x). Asumiendo que la reproducción del video tiene prioridad. Indique:

|  |
| --- |
| Tiempo de transferencia del archivo:  Ocupación (%) de bus NS: |

1. *(0.2 puntos)* Los buses PCIe que se utilizan en este computador son de la versión 3.0, con una codificación 128/130. ¿Cuál será la frecuencia de reloj?

|  |
| --- |
|  |

7