1. ***(3 puntos)*** Un sistema basado en procesador MIPS R2000 dispone de una **cache L1 unificada** para Instrucciones y Datos, cuya configuración es de 64KB, 4 vías y tamaño de bloque 16B. La política de fallo en escritura es de Ubicación (*write allocate*) y la política de acierto en escritura (actualización) es de escritura Directa (*write through*). El algoritmo de reemplazo es LRU.

a) *(0,5 puntos)* Indique los campos (nombre y tamaño) en que se descompondría la dirección de memoria principal

|  |  |  |
| --- | --- | --- |
| 31 14 | 13 4 | 3 0 |
| 18 bits | 10 bits | 4 bits |
| *Etiqueta* | *Conjunto* | *Desplazamiento* |

b) *(0,25 puntos)* Calcule los siguientes parámetros de la cache

|  |  |
| --- | --- |
| Número de líneas | 4K |
| Número de conjuntos | 1K |

c) *(0,25 puntos)* Suponiendo que el conjunto 0xF0 contiene el bloque etiquetado como 0x3C800, indique lo siguiente:

|  |  |
| --- | --- |
| Dirección de bloque | 0xF2000F0 |
| Rango de direcciones de MP que comprende el bloque | 0xF2000F00 … 0xF2000F0F |

d) El siguiente fragmento de código realiza la suma de dos vectores A y B de números enteros, almacenando el resultado sobre el vector A { A←A+B}

|  |
| --- |
| .data 0x100C2000  A: .word 1,2,3, …,256 #vector de 256 enteros  .data 0x200E2000  B: .word 1,2,3, …,256 #vector de 256 enteros  .text 0x00400000  \_start:li $t4,256 # *carga contador*  lui $t0,0x100C # *carga puntero a vector A*  ori $t0,$t0,0x2000  lui $t1,0x200E # *carga puntero a vector B*  ori $t1,$t0,0x2000  buc: lw $t2,0($t0) # *lee A[i*]  lw $t3,0($t1) # *lee B[i*]  add $t2,$t2,$t3 # *A[i]+B[i]*  sw $t2,0($t0) # *almacena en A[i] el resultado de la suma*  addi $t0,$t0,4 # *incrementa puntero a vector A*  addi $t1,$t1,4 # *incrementa puntero a vector B*  addi $t4,$t4,-1 # *decrementa contador*  bnez $t4,buc # *mientras contador≠0, seguir en el bucle*  .end |
| d.1) *(1,25 puntos)* Calcule los siguientes valores tanto para Instrucciones como para Datos   |  |  |  | | --- | --- | --- | |  | **CÓDIGO** | **DATOS** | | **Número de bloques que lo contienen** | 4 | 64 | | **Dirección del primer bloque** | 0x0040000 | **Vector A:** 0x100C200  **Vector B:** 0x200E200 | | **Dirección del último bloque** | 0x0040003 | **Vector A:** 0x100C23F  **Vector B:** 0x200E23F | | **Conjunto al que se mapea el primer bloque** | 0x000 | **Vector A:** 0x200  **Vector B:** 0x200 | | **Número de FALLOS** | 4 | 64 | | **Número de ACCESOS** | 5+8x256=2053 | 768 | | **Número de reemplazos de bloque** | 0 | 0 | | **Número de escrituras a MP (palabras)** | 0 | 256 | | **TASA DE ACIERTOS** |  | | |

*Nota:* ***Justifique*** *los valores indicados en la tabla*

d.2) *(0,25 puntos)* Suponiendo que el tiempo de acceso a la cache es de 1ns y que el acceso a memoria principal es de 300ns, calcule cuál sería el tiempo medio de acceso a memoria en la ejecución del anterior fragmento de código

Tm= 0,9759x1ns + (1-0,9759)x300ns= 8,21ns

d.3) *(0,25 puntos)* Comente cómo se vería afectada la tasa de aciertos calculada anteriormente si se empleara correspondencia directa en lugar de correspondencia asociativa por conjuntos de 4 vías. Razone la respuesta

Puesto que los bloques de los vectores A y B mapean a los mismos conjuntos, el empleo de correspondencia directa causaría la colisión de ambos, haciendo que todos los accesos a datos sean fallos. En consecuencia, la tasa de aciertos se reduciría considerablemente, al contabilizar como aciertos únicamente los accesos a código (hasta el 72,63%).

d.4) *(0,25 puntos)* Comente qué efecto tendría el empleo de las políticas *no-write allocate* y *write back* en los valores de la tabla del apartado d.2

El empleo de no-write allocate no tendría ningún efecto, puesto que no hay fallos en escritura. Sin embargo, el empleo de write back, dado que no hay reemplazos, haría que el número de escrituras a memoria principal fuera cero.

1. ***(2,5 puntos)*** En la figura se muestra el esquema de un interfaz de E/S para ser conectado a una CPU MIPS R2000 que ha sido modificada para incluir dos espacios de direccionamiento: el espacio de memoria y el espacio de entrada/salida. Por ello, su conjunto de instrucciones se amplía con las correspondientes instrucciones de lectura/escritura en puertos de E/S: InputW/InputH/InputB y OutputW/OutputH/OutputB, con una sintaxis similar a las loads/stores de memoria. Dicho interfaz ha sido concebido para trabajar con dispositivos de bloques (discos magnéticos) y soporta transferencias PIO y ADM. Los registros Estado y Control poseen los siguientes bits significativos:

Registro **CONTROL**:

* **MOD** (bits 7 y 6). Permite seleccionar entre el modo PIO (MOD=00) y el modo ADM (MOD=11)
* **A** (bit 3), a 1 ordena al interfaz el inicio de un operación de lectura/escritura sobre el periférico (disco magnético)
* **IE** (bit 2) a 1 habilita la interrupción int0\*. IE=1 se asocia con que el periférico está preparado para iniciar la transferencia (caso de PIO) o que la transferencia ha finalizado (caso de DMA)
* **R/W** (bit 1), indica al interfaz si se trata operación de lectura (R/W= 0) o de escritura (R/W=1) sobre el dispositivo de bloques
* **CL** (bit 0), a 1 hace R=0

Registro **ESTADO**:

* **R** (bit 3) se activa a 1 cuando el bloque está listo para empezar a ser transferido a/desde memoria (modo PIO) o bien cuando la transferencia a/desde memoria a concluido (modo ADM)

1. *(0,25 puntos)* Calcule la dirección base (DB) del interfaz

0xFFFEC820

1. *(0,5 puntos)* Calcule la dirección (DB+X) de cada uno de los registros del interfaz e indique el espacio de direccionamiento (memoria ó E/S) en el que se hallan ubicados

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Registro** | **Dirección** | **Espacio** |  | **Registro** | **Dirección** | **Espacio** |
| ESTADO | DB+4 | E/S |  | ID\_BLOCK | DB | M |
| CONTROL | DB+4 | E/S |  | CONTADOR | DB+4 | M |
| DATOS | DB+12 | E/S |  | PUNTERO | DB+8 | M |

1. *(1 punto)* El driver del dispositivo de bloques controlado a través del interfaz del esquema anterior dispone de la siguiente función:

|  |  |  |
| --- | --- | --- |
| Función | Índice (en $v0) | Argumentos |
| Read\_Disk | 400 | $a0: Puntero a buffer de memoria  $a1: Número de ciclos de transferencia  $a3: Identificador del bloque |

Se desea implementar la función Read\_Disk de modo que la transferencia se realice en modo **ADM** y que la sincronización con el dispositivo se realice mediante **INTERRUPCIÓN**. Suponemos que la sincronización se realiza al nivel de bloque y no palabra a palabra. También se asume que los registros $a0, $a1 y $a3 han sido, en el momento de la llamada a la función Read\_Disk desde la aplicación, debidamente inicializados con la dirección inicial del buffer en memoria, el número de ciclos de transferencia que requiere el bloque y el identificador del bloque que se desea leer de disco, respectivamente. El código deberá configurar adecuadamente el ADM sobre el interfaz y habilitar la interrupción int0\*. **Nota**: *Al terminar, tenga en cuenta que se está en un contexto en el que múltiples procesos pueden estar ejecutándose concurrentemente*

Read\_Disk: la $t0,0xFFFEC820

sw $a0,8($t0) # incializa reg. PUNTERO a memoria

sw $a1,4($t0) # inicializa reg. CONTADOR

sw $a3,0($t0) # inicializa reg. ID\_BLOCK

li $t1,0xCC

OutputW $t1,4($t0) # configura modo ADM, habilita interrupción

# y ordena inicio de operación de lectura

jal suspende\_proceso # suspende el proceso actual

j retexc

1. *(0,75 puntos)* Aunque la forma habitual de hacer la sincronización cuando se emplea ADM es mediante interrupción (como se ha asumido en la pregunta anterior), supongamos que se decidiese realizar en esta ocasión mediante CONSULTA DE ESTADO. Indique la secuencia de código que se requeriría para hacer la sincronización y realizar la lectura de la primera palabra del bloque que se encuentra almacenado en memoria, a partir de la dirección Mem\_Block, al término del ADM.

la $t0,0xFFFEC820

buc: lw $t1,4($t0) # lectura registro ESTADO

andi $t1,$t1,0x08 # máscara para consultar R (bit 3 reg. ESTADO)

beqz $t1,buc # bucle de consulta

li $t1,1

sw $t1,4($t0) # cancela R (R=0) poniendo CL (bit 0 reg. CONTROL)=1

lw $t1,Mem\_Block # lectura primera palabra del bloque en memoria

1. **(2.5 puntos)** En una fábrica de envasado de refrescos, se dispone de un sistema de inspección de botellas de vidrio vacías situado entre la lavadora y la llenadora. Las botellas recién lavadas llegan por una cinta transportadora y son inspeccionadas mediante un sistema de análisis de imagen. Si la botella está correcta, seguirá hacia la llenadora. Si todavía está sucia debe ser llevada de vuelta a la lavadora, y si se ha deteriorado hay que retirarla para reciclado. Un esquema del inspector de botellas se muestra en la figura adjunta.

El sistema se compone de tres dispositivos: El inspector de botellas, el separador y un visualizador. El inspector comprueba, una a una, todas las botellas y cuando la botella activa un sensor a su salida emite el resultado de la inspección. El separador abre o cierras las trampillas para que cada botella siga el curso apropiado. El visualizador muestra en todo momento la cuenta de botellas que están OK y las que se han separado para reciclaje.



El sistema se controla con un MIPS R2000, al que están conectados los tres periféricos mediante las interfaces siguientes:

**INSPECTOR** (Dir. Base 0xFFFF1000)

* Registro de **CONTROL** (Sólo escritura, 8 bits, DB+0)
  + Bit 0 - **A**: 1 para activar la inspección de botellas. 0: Para detener el inspector pero las botellas pueden pasar.
  + Bit 6 – **E**: Se pone a 1 para habilitar la interrupción en la interfaz y a 0 para inhibirla.
  + Bit 7 – **C**: Bit de cancelación: Si se escribe un 1 entonces la interfaz pone el bit R a 0.
* Registro de **ESTADO** (Sólo lectura, 8 bits, DB + 4)
  + Bit 7 – **R**: (Ready) La interfaz lo pone a 1 cada vez que una botella inspeccionada aparece por la salida del inspector. Si E=1, entonces se activa la interrupción INT3 del MIPS.
  + Bits 1,0: **RES-** Resultado de la inspección:

00: OK - 01: Sucia (a lavar) - 10: Deteriorada (a reciclar)

**SEPARADOR** (Dir. Base 0xFFFFA000)

* Registro de **CONTROL** (Sólo escritura, 8 bits, DB+0)
  + Bit 1,0 – TRAMPILLAS

00 – Abre las dos trampillas (OK)

01 – Cierra trampilla 1 (a lavado)

10 – Abre trampilla 1 y cierra trampilla 2 (a reciclado)

11 – Sin efecto.

**VISUALIZADOR** de 2 líneas (Dir. Base 0XFFFF7000)

* Registro de **DATOS 1** (Sólo escritura, 32 bits, DB+0): Número a visualizar por la línea 1
* Registro de **DATOS 2** (Sólo escritura, 32 bits, DB+4): Número a visualizar por la línea 2

Se pide:

1. (0,5 puntos) Programe la siguiente función de sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| Función | Índice | Argumentos | Resultado |
| *Activar\_inspector* | $v0 = 30 | ----- | Inicializa el sistema de interrupciones y habilita la INT3. Activa el inspector. Pone a cero y visualiza los contadores de botellas. |

Se supone que hay definidas unas variables del sistema con los contadores de botellas, como sigue:

.kdata

Botellas\_OK: .word 0

Botellas\_reciclado: .word 0

*Activar\_inspector:*

*li $t0, 0x080C # Desenmascara INT3, habilita interrupciones (en estado Previo)*

*mtc0 $t0, $12*

*la $t0, 0xFFFF1000 # DB del inspector*

*li $t1, 0xC1 # A=1, E=1, C=1*

*sb $t1, 0($t0) # Activa inspector*

*sw $zero, Botellas\_total # Inicializa las variables*

*sw $zero, Botellas\_OK*

*la $t0, 0xFFFF7000*

*sw $zero, 0($t0) # Visualiza los contadores*

*sw $zero, 4($t0)*

*b retexc*

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



1. (1.5 puntos) Programe la rutina del sistema asociada con la INT3. Esta rutina debe comprobar el resultado de la inspección de cada botella y activar el separador como corresponda. También debe actualizar las variables de contadores de botellas y mostrarlos por el visualizador.

*INT3: la $t0, 0xFFFF1000*

*li $t1, 0xC1 # A=1, E=1, C=1*

*sb $t1, 0($t0) # cancelar INT3*

*lb $t1, 4($t0) # Reg estado inspector*

*andi $t1, $t1, 0x03 # extraigo los bits del resultado de la inspección*

*la $t0, 0xFFFFA000*

*sb $t1, 0($t0) # los dos bits de la inspección sirven para activar las trampillas*

*andi $t1, $t1, 0x02 # para comprobar si ha pasado la inspección*

*bnez $t1, no\_OK*

*lw $t0, Botellas\_OK # Botellas\_OK+= 1*

*addi $t0, $t0, 1*

*sw $t0, Botellas\_OK*

*la $t0, 0xFFFF7000*

*sw $t0, 0($t0) # Visualiza Botellas\_OK*

*b retexc*

*no\_OK: lw $t0, Botellas\_reciclado # Botellas\_reciclado += 1*

*addi $t0, $t0, 1*

*sw $t0, Botellas\_reciclado*

*la $t0, 0xFFFF7000*

*sw $t0, 4($t0) # Visualiza Botellas\_reciclado*

*b retexc*

1. **(1.0 punto)** En un sistema como el que se indica en la figura se está reproduciendo en vivo el vídeo tomado desde la cámara. La cámara codifica vídeo+audio en formato MPEG con una tasa de 30 Mbits/s. Se utiliza la GPU para descomprimir el vídeo y el audio. De esta forma, se mueve el vídeo comprimido por DMA desde la Cámara a la Memoria (MEM), mientras que la CPU lo lee de Memoria y lo transmite a la GPU, que procesará y escribirá las imágenes descomprimidas en Memoria de Vídeo (Vídeo RAM) y el sonido en el dispositivo de audio (Sonido), ambos también por DMA.



1. (0.5 puntos) Asumiendo que el vídeo HD es de 1920x1080x24 bits a 30 fps (frames per second) y el audio es Surround 5.1 de 16 bits por canal y 48khz de frecuencia de muestreo, indique el ancho de banda requerido, en MB/s, para:

Lectura del vídeo MPEG desde la Cámara a la Memoria:

Escritura de los cuadros de imagen desde la GPU a la Memoria de vídeo:

Escritura de la información de audio desde la GPU al dispositivo de sonido:

En HD: 30 Mbps /8bits = 3,75 MB/s.  
Imagen: 1920x1080x(24/8Bytes) x 30fps = 186,624 MB/s  
audio: 6 canales x 48000muestras/s x (16/8bytes) = 0.576 MB/s

1. (0.5 puntos) Estando en la situación anterior, se procede a transferir un archivo de 1 GB (109 bytes) desde el Pendrive hasta el disco HDA. Para ello se lee del Pendrive por DMA a la Memoria (MEM) y desde ésta al disco duro también por DMA y de forma concurrente. ¿Cuánto tardará la transferencia del archivo completo?. Indicar también la ocupación (%) del bus NS

El factor limitante es el bus USB Hub, que está ocupado en la lectura del vídeo comprimido (3,75 Mbps) y en la transferencia del audio (0.576 MBps) de forma que el ancho de banda disponible es 60 – 3,75 – 0.576 = 55,674 MBps  
Por lo tanto la transferencia del archivo durará: 103 (MB)/ 55,674 (MB/s) =17.96 s.

NS: (3,75+0.576 + 2x 55.674 )/ 2000 = 4,326 / 2000 = 5.784%

1. **(1.0 punto)** Sea un disco duro magnético formado por tres platos. El área útil de las seis superficies es una corona circular de 3” de diámetro exterior y 1” de diámetro interior. El área útil se ha distribuido en 4 zonas o anillos. Cada anillo contiene 100.000 cilindros. La distribución de sectores (de 512 bytes de capacidad) es la siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Anillo 0 | Anillo 1 | Anillo 2 | Anillo 3 |
| Sectores/pista | 1000 | 800 | 600 | 400 |

El disco gira a 9000 rpm, el tiempo medio de posicionamiento de 10 ms, el *track-to-track time* de 2 ms y dispone de una conexión SATA de 3 Gbps con codificación 8/10.

1. (0.5 puntos) Calcula los siguientes parámetros del disco:

Capacidad del disco en número de sectores: 6 caras x 100.000 ciclindros x (1000 + 800 + 600 + 400 sectores) = 1.680.000.000 sectores

Capacidad del disco en GB (109 B): 1.680.000.000 sectores x 512 bytes = 860,16 GB

Densidad lineal de pistas en tpi (tracks per inch): 100.000 pistas / 0.25” = 400.000 tpi

1. (0.5 puntos) Con el disco duro conectado al computador mediante el bus SATA. ¿cuál es el tiempo medio para leer un archivo de 2 MB (2x106 B): ubicado en la zona 0? ¿Y en la zona 3? En ambos casos, supón que el archivo está ubicado en el disco de forma óptima.

El archivo ocupará 2·10^6/512 = 3906,25~ **3907 sectores**

Una vuelta de disco dura 6.67 ms; media vuelta 3.33

En zona 0: puede alojarse en un cilindro (capacidad 6x1000 = 6000 sectores)

tiempo = 10 + 3.3 + 6.67 x 3907 sectores /1000 = 39.36 s

En zona 3: hacen falta dos cilindros, pues cada uno contiene 2400 sectores

Por tanto, 10 + 3.3 + 6.67 x 3907/400 + 2 ms = 41.36 s