

Estructura de Computadores

Parcial 2

11 Junio - 2018

Nombre:

Grupo:

1

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

- **Cache de Instrucciones:** 2 KB, correspondencia directa, tamaño de bloque de **16 Bytes**.
- **Cache de Datos:** 2 KB, correspondencia asociativa por conjuntos de 2 vías, tamaño de bloque de **64 bytes**. Emplea ubicación en escritura (*write-allocate*) con actualización posterior (*write-back*) y usa LRU para los reemplazos.

a) (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	

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

c) El siguiente programa obtiene la tercera fila (vector Fila) de una matriz de convolución. El vector Fila se obtiene, tras haber sido inicializadas todas sus componentes a cero, almacenando sobre el mismo las componentes de un vector X desplazadas 2 posiciones a la derecha. A modo de ejemplo, suponiendo que $[x_0, x_1, x_2, x_3]$ fuesen los cuatro elementos del vector X y que $[0, 0, 0, 0, 0, 0, 0]$ fuese el vector Fila, inicializado a cero, el vector Fila resultante de la matriz de convolución sería $[0, 0, x_0, x_1, x_2, x_3, 0, 0]$

```
.data 0x2C000000
X:      .word 1, 2, 3, 4, ... 256      # Vector de 256 valores enteros
Fila:   .space 2048                    # Espacio para vector de 512 valores enteros

__start:
        .text 0x00400000
        ori $s0, $zero, 512           # Contador de elementos vector Fila
        lui $t0, 0x2C00               # Puntero al vector Fila (parte superior)
        ori $t0, $t0, 0x0400          # Puntero al vector Fila (parte inferior)
L0:     sw $zero, 0($t0)               # Inicializa a cero Fila[i]
        addi $t0, $t0, 4               # Incrementa puntero
        addi $s0, $s0, -1              # Decrementa contador
        bne $s0, $zero, L0            # Sigue iterando mientras contador > 0

        ori $s0, $zero, 256           # Contador de elementos vector X
        lui $t1, 0x2C00               # Puntero al vector X
        lui $t0, 0x2C00               # Puntero al vector Fila (parte superior)
        ori $t0, $t0, 0x0400          # Puntero al vector Fila (parte inferior)
L1:     lw $t2, 0($t1)                 # Lee X[i]
        sw $t2, 8($t0)                # Almacena X[i] en Fila[i+2]
        addi $t0, $t0, 4               # Incrementa puntero a vector Fila
        addi $t1, $t1, 4               # Incrementa puntero a vector X
        addi $s0, $s0, -1              # Decrementa contador
        bne $s0, $zero, L1            # Sigue iterando mientras contador > 0
        .end
```

c.1) (0.5 puntos) Obtenga, para la **cache de instrucciones**:

Número de bloques de código	Primer número de bloque	Último número de bloque	Total de FALLOS de código
Total de ACCESOS a código (Indique el cálculo)			
Tasa de aciertos (Con cuatro dígitos decimales. Indique el cálculo)			

c.2) (0.5 puntos) Indique el número de bloques que ocupa cada vector, así como los números de bloque del primer y último bloque de cada uno y sus correspondientes números de conjunto en los que se almacenan en la **cache de datos (en hex)**

	Num. de bloques de datos	Primer bloque	Último bloque	Primer conjunto	Último conjunto
X					
Fila					

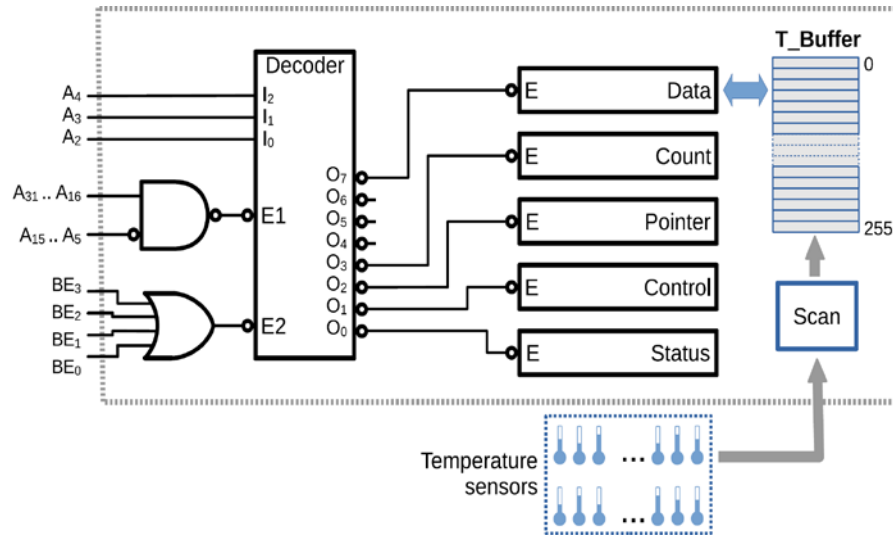
c.3) (0.5 puntos) Calcule, para la **cache de datos**:

Total de ACCESOS a datos	
Total de FALLOS de datos	Fallos de inicio: Fallos de colisión/capacidad: Total Fallos:
Tasa de aciertos	
Número de reemplazos de bloque	
Número de palabras (32 bits) escritas a memoria	

c.4) (0.5 puntos) ¿Cuál sería el número de fallos de la cache de datos si se empleara una correspondencia directa en lugar de asociativa de 2 vías? Justifique la respuesta.

2

(2.5 puntos) La figura muestra los detalles del adaptador de un dispositivo de adquisición de temperaturas conectado a un procesador MIPS R2000. Dicho dispositivo puede ser configurado para registrar las temperaturas de hasta 256 sensores ubicados en las estancias de un edificio. Los valores de temperatura se hallan codificados como enteros con signo de 32 bits. Todos los registros del interfaz son de tamaño 32 bits.



La operación del dispositivo se inicia mediante una orden de *UPDATE*, escrita en el registro de Control. En respuesta a dicha orden, el sistema escaneará las medidas de temperatura de hasta 256 sensores y las almacenará en un buffer interno (*T_Buffer*), en posiciones consecutivas. Una vez *T_Buffer* se actualice con las medidas de temperatura, existen dos posibles modos de transferir su contenido al espacio de memoria de usuario: modos *PIO* y *ADM*. En modo *PIO*, el dispositivo se encontrará *Ready* tan pronto *T_Buffer* se actualice, de modo que la transferencia de datos desde *T_Buffer* a memoria de usuario debe hacerse por programa. En modo *ADM*, una vez *T_Buffer* se actualiza, los datos se transfieren por *ADM* a un buffer en memoria de usuario, de modo que el dispositivo se encontrará *Ready* al término de la transferencia por *ADM*. El adaptador contiene los siguientes registros:

STATUS (lectura y escritura)

- **RDY** (bit 0). Bit *Ready*. Es puesto a 1 por el hardware dependiendo del modo de transferencia
 - Modo *PIO*: un vez que *T_Buffer* ha sido actualizado con las medidas de temperatura
 - Modo *ADM*: cuando el contenido de *T_Buffer* ha sido totalmente transferido a la memoria de usuario por *ADM*.

Para hacer $RDY = 0$, basta con escribir un 0 sobre dicho bit (obsérvese que no hay un bit específico de cancelación en este interfaz).

- **IEN** (bit 1). Bit *Interrupt Enable*. Si $IEN = 1$, la puesta a 1 del bit RDY causa la activación de $INT3^*$.

CONTROL (solo escritura)

- **UPD** (bit 0). Bit *UPDATE*. Cuando se pone a 1 por software, el sistema inicia el registro de temperaturas de los distintos sensores.
- **MOD** (bit 1). Bit *MODO*. A 0 para *PIO* y a 1 para *ADM*. Se ignora cuando $UPD = 0$.

POINTER (solo escritura)

En modo *ADM*, este registro debe actualizarse con la dirección de inicio del buffer en memoria de usuario, donde el contenido de *T_Buffer* es transferido por *ADM*. No tiene ningún uso en modo *PIO*.

COUNT (solo escritura)

Debe actualizarse con el número de sensores de temperatura a registrar por el sistema con cada orden *UPDATE*.

DATA (lectura y escritura)

Este registro (contiene un valor de temperatura cada vez) se usa solo en modo *PIO*, durante la transferencia por programa del contenido de *T_Buffer*. El registro almacena temporalmente el siguiente valor de temperatura a ser transferido desde *T_Buffer*. Cada vez que se lee el registro, éste se actualiza automáticamente con el siguiente valor de temperatura almacenado en *T_Buffer*.

- a) (1 punto) Calcule la dirección base del adaptador y los desplazamientos de cada uno de los registros (en hex).

Dirección Base (DB):		0x	
Registro	Desplazamiento	Registro	Desplazamiento
Status	BA + 0x	Count	BA + 0x
Control	BA + 0x	Data	BA + 0x
Pointer	BA + 0x		

- b) (0.5 puntos) De acuerdo a su descripción, se trata de un dispositivo de *bloques* o de *caracteres*? Justifique la respuesta.

- c) (0.6 puntos) Escriba el código de la función del sistema encargada de dar una orden *UPDATE* para escanear *N* sensores y, una vez actualizado *T_Buffer*, almacenar su contenido en la dirección de memoria proporcionada por el usuario **empleando modo *PIO***. Asuma que *N* se pasa en el registro \$a0 y el que el puntero al buffer de memoria de usuario lo hace en \$a1. La transferencia se debe **sincronizar mediante *consulta de estado***.

PIO_Update:

b retexc

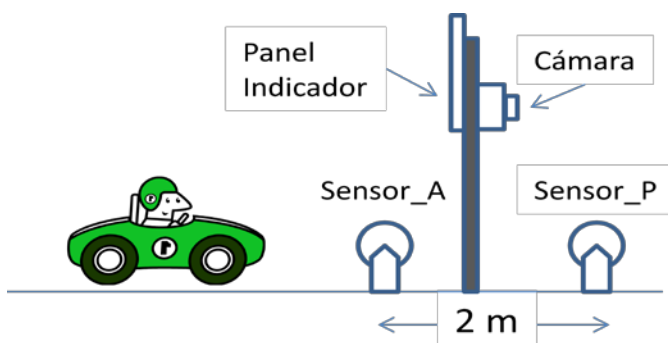
- d) (0.4 puntos) Escriba el código de la función del sistema encargada de dar una orden *UPDATE* para escanear N sensores y, una vez actualizado T_Buffer , almacenar su contenido en la dirección de memoria proporcionada por el usuario **empleando modo ADM**. Asuma que N se pasa en el registro $\$a0$ y el que el puntero al buffer de memoria lo hace en $\$a1$. La espera a que la transferencia por *ADM* se complete, se hará asumiendo un **entorno multitarea**.

DMA_Update:

b retexc

3

(3 puntos) Se dispone de un sistema de control de velocidad sobre una carretera de un solo carril, el cual se halla controlado por un MIPS R2000. Básicamente, el sistema se encarga de detectar el paso de vehículos (contabilizando su número) y de calcular su velocidad. En caso de que se supere el límite de velocidad, se hará una foto al vehículo. Para la detección de velocidad, el sistema dispone de un módulo **DETECTOR DE PASO** constituido por dos sensores (ver dibujo). El Sensor_A se halla ubicado 2 metros antes que el Sensor_P. La velocidad se calcula midiendo el tiempo que el vehículo tarda en recorrer los 2 m que separan ambos sensores. Cuanto menor sea dicho tiempo, mayor será la velocidad. Si la medida de velocidad supera la velocidad máxima establecida, el módulo **CÁMARA** realizará una foto del vehículo y registrará su velocidad (diferencia en tiempos de paso por los sensores).

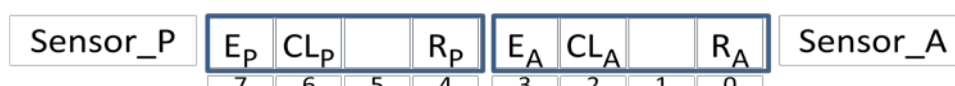


El sistema consta además de otros dos módulos: un **SENSOR DE LLUVIA** y un **PANEL INDICADOR**. El primero fija la velocidad máxima permitida (120 km/h ó 90 km/h) en función del estado del pavimento, la cual se almacena en una variable del sistema, al tiempo que se visualiza en el **PANEL INDICADOR**.

La estructura de cada uno de los referidos módulos de interfaz se indica a continuación:

DETECTOR DE PASO (Dir. Base 0xFFFF01000)

- Registro de **SENSORES** (lectura y escritura, 8 bits, DB+0)
 - Bit 0 – **RA**: (bit Ready – solo lectura) La interfaz lo pone a 1 con el paso de cada vehículo por el **Sensor_A**
 - Bit 2 – **CLA**: (bit Cancel – solo escritura) Se pone a 1 para cancelar R_A
 - Bits 3 – **EA**: (bit Enable – solo escritura) Se pone a 1 para habilitar la interrupción asociada al Sensor_A en la interfaz y a 0 para inhibirla. Si $R_A=1$, entonces se activa la interrupción **INT0*** del MIPS
 - Bits 4, 6 y 7: Idem. que bits 0, 2 y 3, pero para el **Sensor_P**, asociado a la interrupción **INT1*** del MIPS



CÁMARA (Dir. Base 0xFFFF02000) Este dispositivo usa E/S Directa

- Registro de **CONTROL** (solo escritura, 8 bits, DB+0)
 - Bit 3 – **F**: (bit Foto) Se pone a 1 para ordenar la captura de una foto
- Registro de **VELOCIDAD** (sólo lectura, 32 bits, DB+4): Tiempo de paso (en ms) entre los dos Sensores

SENSOR DE LLUVIA (Dir. Base 0xFFFF03000)

- Registro de **CONTROL** (lectura y escritura, 8 bits, DB+0)
 - Bit 3 – **E**: Se pone a 1 para habilitar la interrupción en la interfaz y a 0 para inhibirla.
- Registro de **ESTADO** (sólo lectura, 8 bits, DB + 1)
 - Bit 7 – **R**: (*Ready*) La interfaz lo pone a 1 cada vez que se detecta una variación en el estado del pavimento. R se pone a cero al leer ESTADO. Si E=1, entonces se activa la interrupción **INT5*** del MIPS.
 - Bits 1,0: **RES**- Resultado de la detección:
 - 01: Pavimento SECO
 - 10: Pavimento MOJADO

PANEL INDICADOR (Dir. Base 0xFFFF04000) Este dispositivo usa E/S Directa

- Registro de **VELOCIDAD** (sólo escritura, 32 bits, DB+0): Valor de la velocidad máxima permitida.
- Registro de **INDICADOR** (solo escritura, 8 bits, DB+4)
 - Bits 1,0: **MSG**- Mensaje de texto a visualizar:
 - 01: NO muestra mensaje (si pavimento SECO)
 - 10: "CONDUZCA CON PRECAUCIÓN" (si pavimento MOJADO)

Se supone la existencia de las siguientes variables del sistema

	.kdata	
Reloj:	.word 0	# medida de tiempo real en milisegundos (ms)
tiempo_paso_A:	.word 0	# almacena el valor de tiempo de paso (ms) por el Sensor_A
contador_vehiculos:	.word 0	# contabiliza el número de vehículos que pasan
límite_velocidad:	.word 0	# velocidad máxima según estado del pavimento

Se pide:

a) (1 punto) Programe las siguientes funciones del sistema:

Función	Índice	Argumentos	Resultado
<i>inicializar</i>	\$v0 = 800	-----	Habilita las dos interrupciones asociadas al DETECTOR DE PASO, así como la interrupción del SENSOR DE LLUVIA. Inicializa la variable <i>límite_velocidad</i> a 120 km/h y la visualiza en el PANEL INDICADOR. El resto de variables, salvo la de <i>Reloj</i> , las inicializa a cero. El PANEL INDICADOR no debe mostrar ningún mensaje adicional (MSG=01). Habilita globalmente las interrupciones y actualiza las máscaras de las interrupciones INT0, INT1 e INT5 en el procesador.
<i>get_vehiculos</i>	\$v0=900		Retorna en \$v0 el valor de la variable <i>contador_vehiculos</i>

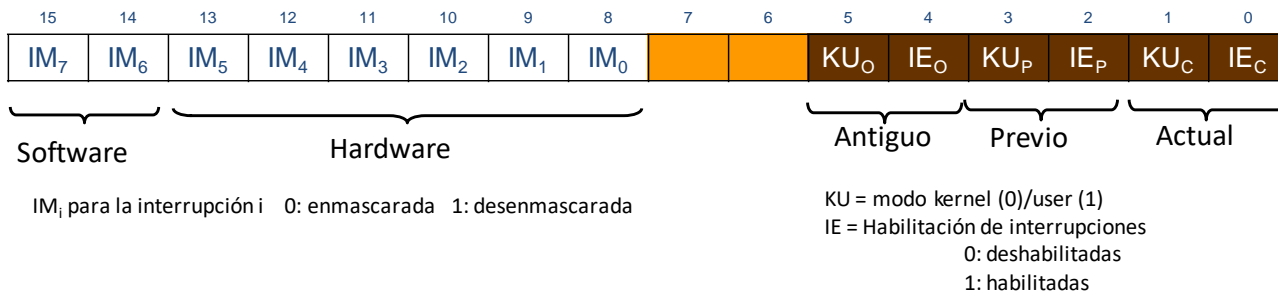
inicializar:

b retexc

get_vehiculos:

b retexc

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



- b) (1 punto) Escriba el código de las rutina de servicio de las interrupciones INT0* (asociada al Sensor_A) e INT1* (asociada al Sensor_B). El servicio de interrupción INT0* deberá únicamente almacenar el tiempo de paso (en *ms*) por el Sensor_A en la variable *tiempo_paso_A*. El servicio de interrupción de INT1* deberá leer el tiempo de paso por el Sensor_P y calcular la diferencia entre éste y el tiempo de paso por el Sensor_A (previamente almacenado por INT0 en la variable *tiempo_paso_A*). Si la diferencia calculada es inferior al valor de la diferencia en los tiempos de paso correspondiente al límite de velocidad establecido por la variable *límite_velocidad* (véase la tabla más abajo), se deberá ordenar a la CÁMARA la captura de una foto y proceder a registrar en la misma la diferencia de los tiempos de paso (en *ms*) calculada. En cualquiera de los casos, se deberá incrementar la variable *contador_vehiculos*.

Tabla velocidad / diferencia_tiempos de paso	
Límite de velocidad	Diferencia tiempo de paso sensores
120 km/h	60 ms
90 km/h	80 ms

La tabla proporciona las diferencias en los tiempos de paso equivalentes a cada límite de velocidad. Para cada uno de estos límites, si la diferencia de tiempos de paso calculada es **inferior** a la proporcionada por la tabla, significa que se ha excedido el límite de velocidad

int0:

b retexc

int1:

b retexc

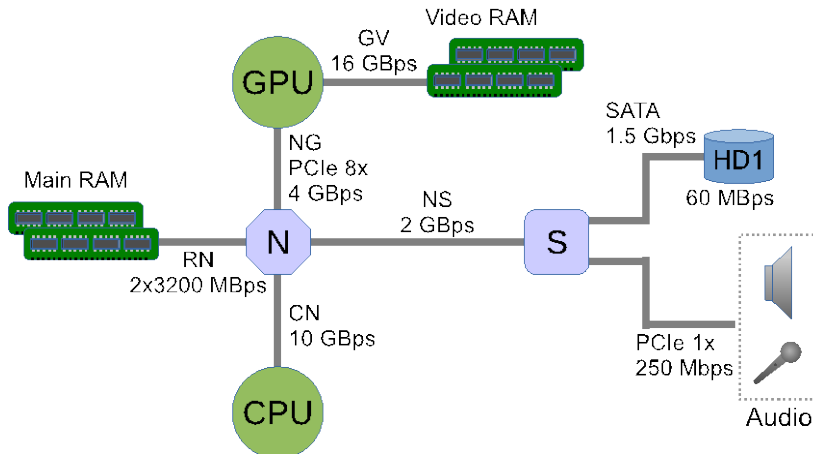
- c) (1 punto) Programe el código de la rutina de servicio de interrupción INT5* del SENSOR DE LLUVIA, la cual deberá detectar el estado del pavimento, actualizar en consecuencia la variable `Lími te_vel oci dad` (Seco: 120 km/h; Mojado: 90 km/h) y visualizarla, junto al mensaje de texto correspondiente, en el PANEL INDICADOR.

Int5:

b retexc

4

(1 punto) Se quiere evaluar la capacidad de un cierto computador (ver figura más abajo) para soportar una aplicación de *Karaoke*. El *videoclip* (audio/vídeo) de la canción reside en un archivo comprimido en el disco HD1. Durante la reproducción (*playback*) el archivo es transferido por ADM desde HD1 a memoria principal. De forma concurrente, la GPU va leyendo desde memoria el archivo y descomprimiendo el audio y el vídeo. Al mismo tiempo, la voz del cantante es capturada por el sistema de Audio y transferida por ADM a memoria principal. Este flujo de audio es leído por la GPU desde memoria y combinado con el audio de la canción. La GPU envía a continuación el audio resultante al sistema de audio para su reproducción y el vídeo descomprimido a la RAM de vídeo.



Los anchos de banda dados en la figura son todos efectivos, excepto para el SATA y PCIe 1x, los cuales usan codificación 8b/10b. La velocidad del computador necesita ser suficiente para soportar las siguientes especificaciones:

- El *videoclip* con la canción está comprimido en un archivo MPEG codificado a 32 Mbps.
- El *videoclip* descomprimido está formado por video de 1920x1200x24 bits, para ser reproducido a 30 fps; y audio 5.1 a 24 bits por canal, muestreado a 96 KHz.
- El micrófono de audio está

muestreado a 48 KHz, con 16 bits por muestra.

- a) (0.4 puntos) Calcular el ancho de banda requerido por cada una de las transferencias necesarias por la aplicación de *Karaoke*. Expresar todos los resultados en KBps, MBps o GBps, según el caso.

Vídeoclip comprimido de la canción:

Micrófono de audio:

Vídeo descomprimido:

Audio combinado descomprimido:

- b) (0.6 puntos) Justifica si los buses pueden soportar o no los requisitos de ancho de banda para las transferencias simultáneas durante la reproducción.

Bus SATA	Bus PCIe 1x
Bus NS	Bus RN

Bus NG

Bus GV

5

(0.5 puntos) Calcule el tiempo de transferencia de un archivo de 10 MB ($K=10^3$) desde un disco magnético, cuyos parámetros se muestran más abajo. El archivo se halla almacenado de forma óptima en la Zona 1, esto es, en sectores consecutivos dentro de una misma pista y en cilindros consecutivos dentro de la zona.

Tamaño de sector: 512 bytes

Velocidad de rotación: 7500 RPM

Tiempo medio de posicionamiento: 6 ms

Trac-to-track seek time: 1 ms

Densidad lineal: 10000 tpi

Número de caras: 6

Radio interno: 0.5"

Radio externo: 1.5"

Formato: ZCAV, 2 zonas

Zona	Límites	Sectores/pista
0	1.5" - 1.0"	800
1	1.0" - 0.5"	400