Estructura de Computadores

Parcial 2 **30-mayo-2022**

Nombre: **Grupo**

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

- **Cache de Instrucciones**: 1024B, correspondencia directa, tamaño de bloque de 16 Bytes
- **Cache de Datos**: 512B, correspondencia asociativa por conjuntos de 2 vías, tamaño de bloque de 32 bytes, algoritmo de remplazo LRU. Emplea política de escritura directa con ubicación (*write-through allocate*).

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

Cache de Instrucciones	
Etiqueta	
Línea	
Desplazamiento	

Cache de Datos	
Etiqueta	
Conjunto	
Desplazamiento	

b) *(0.5 puntos)* Suponiendo que en el conjunto 0x2 de la **Cache de Datos** se halla almacenado un bloque con etiqueta 0x20801, indíquese el Número de dicho Bloque, así como la primera y última dirección de memoria principal comprendidas en el mismo.

Conjunto	**Etiqueta**	**Número de Bloque**	**Rango direcciones MP**	
0x2	0x20801		**Primera**	
			Última	

c) El siguiente programa inicializa a '0' el vector Y y a continuación copia 8 componentes del vector X, concretamente las componentes con índices 0, 32, 64, 96, 128, 160, 192 y 224, de forma consecutiva sobre las 8 primeras componentes del vector Y. El programa ejecuta el siguiente código en alto nivel y ensamblador, respectivamente:

```
byte X[]= {-128,-127,-126,…,-1,0,1,2,3, …,127};
byte Y[512];
. . .
for (int i=0; i<512; i=i++)
    Y[i]= 0;
for (int i=0, j=0; i<256; i=i+32, j++)
    Y[j]= X[i];
```

```
-------------------------------
        .data 0x20F00000
X:      .byte -128,127,…,-1,0,1,2,…,127   # vector de 256 componentes
Y:      .space 512                        # vector de 512 componentes

          .text 0x00400000
__start:  lui $t0,0x20F0
          Ori $t0,$t0,0x0100    # Puntero a Y
          li  $t1,512           # Inicializa contador
b1:       sb  $zero,0($t0)      # Inicializa a '0' vector Y
          addi $t0,$t0,1        # Incrementa puntero a Y
          addi $t1,$t1,-1       # Decrementa contador
          bnez $t1,b1           # Mientras contador<>0, saltar a b1
          lui $t0,0x20F0        # Puntero a X
          ori $t1,$t0,0x0100    # Puntero a Y
          add $t2,$t1,255       # Puntero al final del vector X

b2:       lb  $t3,0($t0)       # Lectura componente i del vector X
          sb  $t3,0($t1)        # Almacena en componente j del vector Y
          addi $t0,$t0,32       # Incrementa i en 32
          addi $t1,$t1,1        # Incrementa j en 1
          ble  $t0,$t2,b2       # Mientras no alcanza el final del
          .end                  # vector X, saltar a b2
```

c.1) *(0.4 puntos)* Obtenga, para la **cache de instrucciones:**

Número de bloques de código		**Instrucciones ejecutadas**	
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 dos vectores, 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**
X						
Y						

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 ubicación (write allocate) en escritura.*

Total de ACCESOS	
Total de FALLOS	
Número de reemplazos	
Número de escrituras a MP (tamaño de palabra de los módulos de MP= 64 bits)	

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

d) *(0.4 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)		

2 ***(2 puntos)*** La figura muestra el esquema del interfaz de una impresora. 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), añadiendo para ello la señal M/IO (=1, MEM / =0, E/S). 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. Los registros del interfaz son los siguientes:

Registro **CONTROL** (32 bits):

- Bit 1-0: **QP** (calidad de impresión)

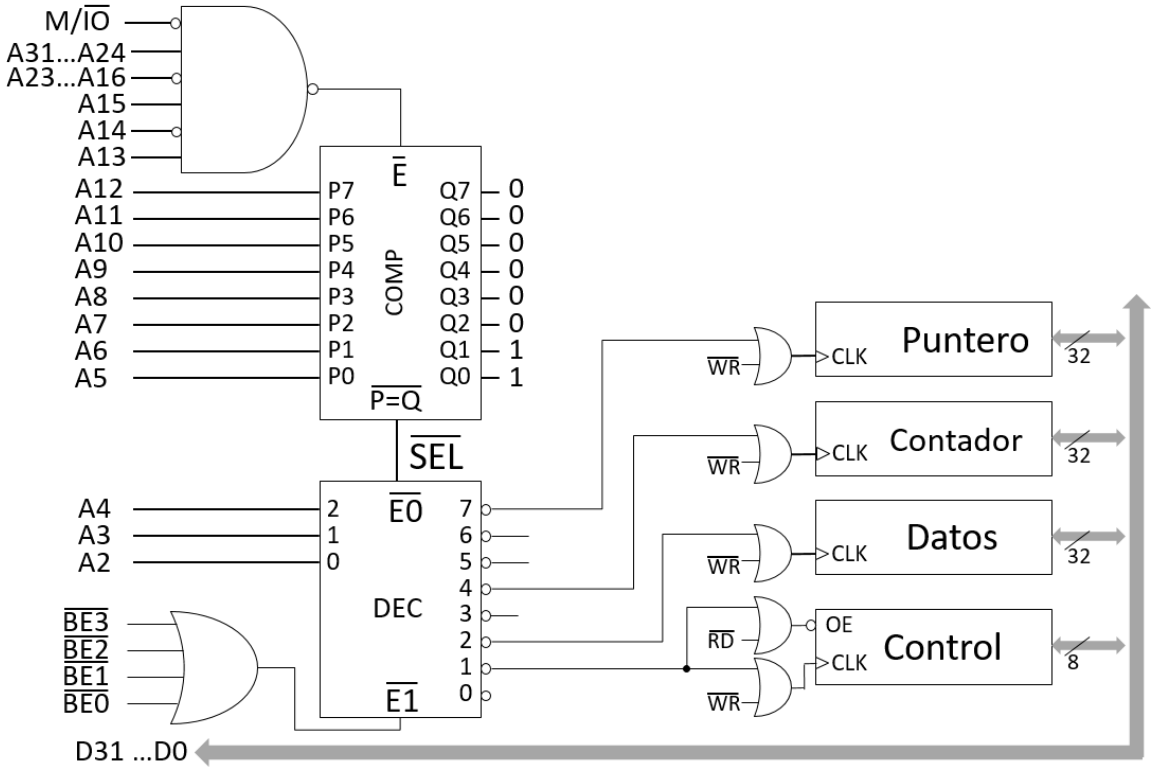
00	Automática
01	Rápida
10	Normal
11	Foto

- Bit 3: **P** (imprimir): al escribir un 1 se ordena el envío de la imagen o archivo a la impresora.
- Bit 4: **M** (modo): igual a 1 selecciona modo ADM; igual a 0 selecciona modo PIO.
- Bit 7: **R** (bit *Ready*): el valor 1 indica que la impresora está preparada para recibir datos a imprimir. La cancelación de R (R=0) se realiza escribiendo un 0 directamente sobre el bit R.

Registro **DATOS** (32 bits): Se usa en modo PIO para enviar a la impresora los datos a imprimir.

Registro **CONTADOR** (32 bits): Sólo se emplea en modo ADM. Establece el tamaño en bytes de la imagen o archivo a imprimir.

Registro **PUNTERO** (32 bits): Sólo se emplea en modo ADM. Contiene la dirección inicial del buffer de memoria en el que se ha encuentra la imagen o archivo a imprimir.



a) *(0.5 puntos)* Cuál es la dirección base del interfaz de la impresora?

b) *(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
PUNTERO			
CONTADOR			
DATOS			
CONTROL			

c) *(0.2 puntos)* Teniendo en cuenta que el registro de Control es de 8 bits ¿Qué habría que modificar en el circuito de selección de los puertos para acceder a dicho registro en la misma dirección, pero empleando instrucciones de byte?

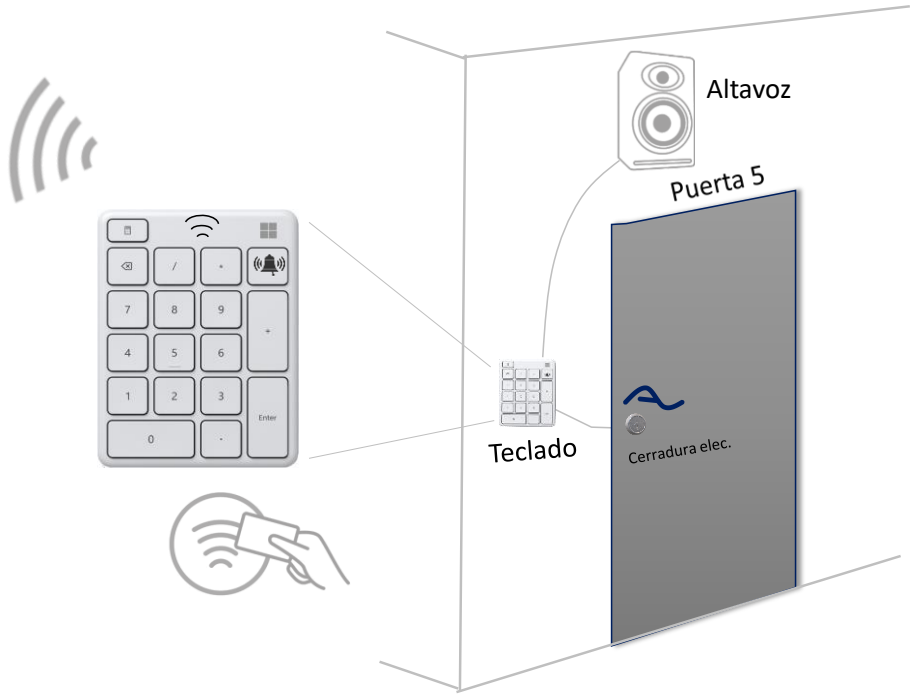
d) *(0.8 puntos)* En el driver de la impresora controlada a través del interfaz del esquema anterior se define la siguiente función para imprimir un archivo:

Función	Índice	Argumentos
Imprimir_ADM	$v0= 100	$a0: Puntero a buffer de memoria $a1: Tamaño en bytes del archivo $a2: Calidad de impresión

La sincronización con la impresora se realiza por **CONSUTA DE ESTADO**, lo que significa que hay que esperar a que la impresora esté preparada (R=1) para mandarle cualquier orden de imprimir (incluido con ADM). En esta interfaz no hay interrupción de fin de ADM. La función `imprimir` deberá configurar adecuadamente el interfaz de la impresora para realizar una transferencia en **MODO ADM** e imprimir el archivo con la calidad especificada como parámetro.

```
Imprimir_ADM:

          j retexc
```

3 ***(3.75 puntos***) Para controlar el acceso a las entradas de una empresa se ha instalado un control de acceso en cada puerta, compuesto por un teclado numérico son sensor RFID para tarjetas, una cerradura electrónica y, en el interior, se coloca un botón para abrir la puerta y un altavoz. Todos estos elementos están controlados por el teclado, el cual se comunica con la unidad central por transmisión inalámbrica. La figura siguiente muestra todos los componentes.



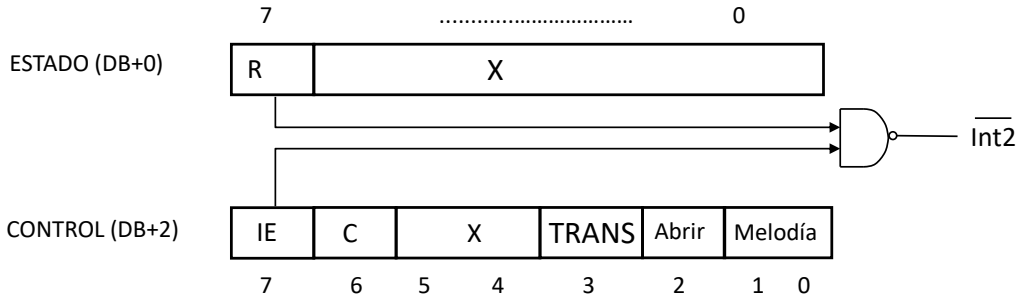
El sistema funciona como sigue: el usuario acerca su tarjeta o teclea su código de usuario, más la tecla <Enter>. El código <USUARIO – PUERTA> se envía a la unidad central que comprueba si ese usuario está habilitado para acceder por esa puerta. En caso afirmativo, la unidad central ordena abrir la puerta y emitir un 'BIP' por el altavoz. La puerta se cerrará sola tras unos segundos. En caso contrario, la puerta no se abre y se emite un sonido 'BOOP'. EL usuario tiene la opción de pulsar el botón <Timbre> que activa una melodía para que alguna persona dentro pueda abrir (NO hay intercomunicador en este caso). Desde el interior la puerta se abre pulsando un botón interno de <Abrir>.

La unidad de control central es un periférico de un computador basado en MIPS R2000. Esta unidad es capaz de controlar hasta 16 controles de acceso (16 puertas), transmitiendo o recibiendo los mensajes de ellas. Consta de los siguientes registros:

CONTROL_CENTRAL: dirección base DB= 0xFFFF1000, con los siguientes registros:

Registro **ESTADO**: (Lectura, 8 bits, en DB):

- Bit 7 – **R**: Se pone a 1 cada vez que se hay un mensaje procedente del teclado de una puerta
  - Este bit debe cancelarse poniendo el bit **C** a 1. Si el bit **IE**=1, se activa la salida de interrupción que está conectada a la INT2* del MIPS.



Registro **CONTROL**: (Lectura/Escritura, 8 bits, en DB+2):

- Bit 7 – **IE**: Igual a 1 habilita la INT2*.
- Bit 6 – **C**: Igual a 1 cancela bit R.
- Bit 3 – **TRANS**: Igual a 1 transmite la orden (bits 2,1,0) a la puerta indicada (registro PUERTA)
- Bits 2 – **ABRIR.** Igual a 1 abre la cerradura de la puerta indicada.

- Bits 1,0 **Melodía** en el altavoz en la puerta indicada:
  00 – Silencio
  01 – BIP de apertura
  10 – BOOP de código erróneo
  11 – Melodía del TIMBRE

Registro **PUERTA**: (Lectura/Escritura, 16 bits, en DB+4):
Indica la puerta de la que se ha recibido el mensaje o a la que va dirigida la orden. El código de la puerta es un número descodificado, es decir, un bit a '1' en uno de los 16 bits del código. Por ejemplo, el código de la puerta 5 es 0000 0000 0010 0000 (el bit 5 a uno)

Registro **USUARIO**: (Lectura, 16 bits, en DB+8):
Código de usuario que se ha recibido desde el teclado de la puerta indicada. Un valor de 0 significa código del timbre. Los códigos de usuario son números binarios de 0001 a 9999, por lo que con 16 bits es suficiente para codificarlo.

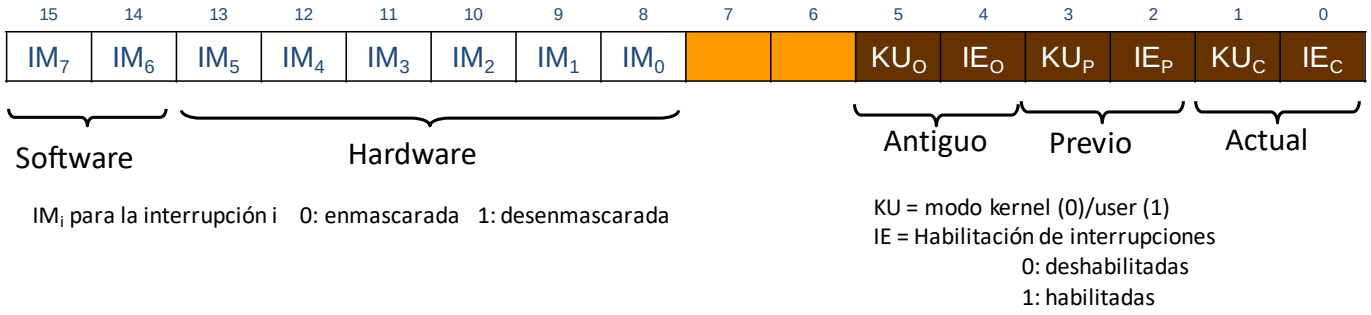
El sistema de control de acceso de las puertas se controla desde un programa de usuario para MIPS R2000. El sistema dispone de una tabla de usuarios cuyas entradas son de tamaño *word* y cada una consta de dos componentes: <usuario> registrado (primer *halfword*) y <puertas> en las que tiene permitido el acceso (segundo *halfword*). El usuario '0' corresponde al timbre y está habilitado inicialmente en todas las puertas. Otros usuarios pueden estar habilitados sólo en algunas puertas. Por ejemplo, si el usuario 0x100 está habilitado en las puertas 0,1,2,3 y 5, en la tabla aparecerá <usuario> como <0x0100> y <puertas> como <0000000000101111 = 0x002F>, observe que justo los bits 0,1,2,3 y 5 están puestos a uno. El código en la correspondiente entrada de la tabla aparecerá como <0x0100002F>.

Las variables definidas en el kernel son:

```
                    .kdata
mensaje:            .word  0            # Código <USUARIO-PUERTA> del último mensaje recibido
Numero_usuarios:    .word  4            # Número de usuarios registrados
Usuarios:           .word  0x0000FFFF   # El usuario 0 está habilitado en todas las puertas
                    .word  0x00010081   # El usuario 1 está habilitado en las puerta 0 y 7
                    .word  0x00020111   # El usuario 2 está habilitado en las puertas 0,4 y 8
                    .word  0x02000222   # El usuario 0x200 lo está en las puertas 1, 5 y 9
```

El programa de usuario tiene un código de inicio y el salto a '*main'*, donde hay un bucle sin fin en el que se espera un mensaje desde alguna puerta y se actúa de acuerdo con él a través de la invocación de distintas llamadas al sistema. Se dispone de las subrutinas "*suspende_este_proceso*" y "*activa_proceso_en_espera"*, vistas en clase, para la multitarea. También, se pueden usar los registros $t0, $t1, $t2 y $t3 en el manejador de excepciones.

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



a) *(1 punto)* Escriba el código de inicio del programa de usuario tal que se emita un sonido 'BOOP' en TODAS las puertas, que permanecerán cerradas. La interrupción 2 quedará desenmascarada en la UCP, pero inhabilitada en el periférico. El procesador debe quedar en modo usuario con las interrupciones habilitadas globalmente.

```
                .text    # Código de inicio del programa
__start:

        la $k1, salvareg
        jal main
        li $v0, 10
        syscall          # exit
```

b) *(0.5 puntos)* Programe la siguiente función del sistema:

Función	Índice	Argumentos	Resultado
Espera_mensaje	$v0 = 600	-----	Espera a que se reciba un mensaje de alguna puerta. Entre tanto, la multitarea puede realizar otros procesos. Debe habilitar la int2 y dormir hasta que se reciba un nuevo mensaje.

```
Espera_mensaje:



    b retexc
```

c) *(0.75 puntos)* Escriba el código de la rutina de servicio de la interrupción INT2*, que gestiona la recepción del mensaje de alguna puerta. El mensaje recibido se dejará en la variable 'mensaje' con el código de 32 bits <USUARIO--PUERTA>. Se deberá cancelar e inhabilitar la INT2*, y también despertar al proceso anteriormente suspendido.

```
Int2:
```

b retexc

Nota: No se debe preocupar por los mensajes, más o menos simultáneos, llegados desde múltiples puertas, pues el control central los registra y serializa. Con cada llamada a '*Espera_mensaje'* se activa la INT2* y se entrega el siguiente mensaje en cola.

d) *(1 punto)* Programe la siguiente función del sistema:

Función	Índice	Argumentos	Resultado
Abrir_puerta	$v0 = 610	$a0: <Usuario> $a1: <-Puerta>	Transmite la orden correspondiente a la puerta indicada en $a1. Se asume que el usuario ($a0) es válido. Si es el usuario 0 (timbre), se toca la melodía en la puerta indicada, pero no se abre. Si el usuario es distinto de cero, se abre la puerta con un sonido BIP. Debe dejar las interrupciones como estaban.

Abrir_puerta:

b retexc

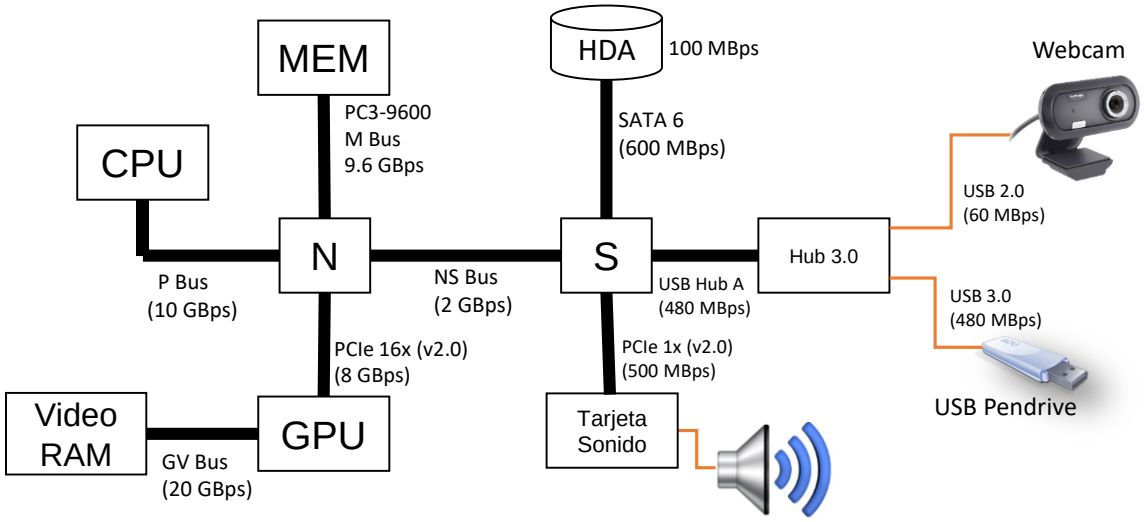
e) *(0.5 puntos)* Complete la siguiente función del sistema:

Función	Índice	Argumentos	Resultado
Es_UsuarioValido	$v0 = 620	$a0: <Usuario> $a1: <-Puerta>	$v0≠0: usuario SI habilitado en la puerta indicada. $V0=0: usuario NO habilitado

Esta *syscall* debe recorrer la tabla de usuarios y comprobar si el usuario está en ella. Si lo está, se comprobará si la puerta indicada es alguna de las permitidas para ese usuario. Considera qué puede suceder si se hace una 'AND' entra la puerta actual y el código de las puertas permitidas. Recuérdese que en las entradas de la tabla el <usuario> corresponde al primer *halfword* y las <puertas> corresponden al segundo.

```
Es_UsuarioValido:
        li $v0,0
        lw $t1, numero_usuarios
        la $t0, usuarios          # Tabla de usuarios
buc2:   lh $t2, 2($t0)    # usuario actual
        ……..      # completar




        addi $t1, $t1, -1
        bnez $t1, buc2   # bucle que recorre la tabla de usuarios
        b retexc
encontrado:
        ……..      # completar



       b retexc
```

4 ***(1.25 puntos)*** El computador del esquema reproduce una secuencia de vídeo en formato MP4 codificado a 40Mbps y almacenada en el pendrive. Para hacer esto, se transfiere por ADM 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 ADM (GPU → M, M → PCie 1x). Todas las transferencias se hacen de forma sincronizada y simultánea.



a) *0.3 puntos)* Suponiendo que el vídeo descomprimido tiene una resolución de 2560x1440x24 bits y 30 escenas por segundo y que el audio es 4.1, con una frecuencia de muestreo a 48 KHz y 16 bits/muestra, calcule el ancho de banda (en MBps) requerido para:

Transferir el video comprimido desde el pendrive a la memoria (MEM): Enviar las imágenes de vídeo desde la GPU a la RAM de vídeo: Enviar el audio desde la GPU al equipo de sonido:

b) *(0.6 puntos)* Indique la ocupación (%) de los buses siguientes, suponiendo que las transferencias se realizan sin problemas:

Bus USB 3.0: Bus GV: Bus PCIe x1: Bus NS: Bus PCIe x16: Bus M:

c) *(0.15 puntos)* Indique cuánto ocupará el archivo del video descomprimido si tuviera una duración de 1 minuto. Indíquelo en MB

d) *(0.2 puntos)* Supongamos que queremos guardar una copia de 60 minutos de video descomprimido (sin audio) en el disco HDA por ADM (GPU → M, M → HDA). Calcúlese el tamaño de la información a copiar y el tiempo que se tardaría en hacer la copia. Teniendo en cuenta que las capacidades de la memoria (MEM) y del disco HDA son de 16 GB y 1TB, respectivamente, indique qué problema se observa en la configuración mostrada y qué solución se le podría dar.
