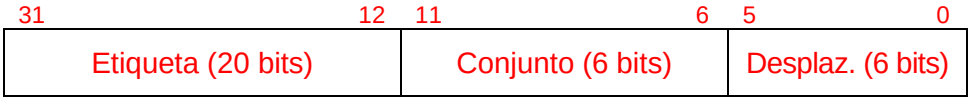
Estructura de Computadores

Recuperación Parcial 2 **19 Junio - 2017**

Nombre: **Grupo:**

1 ***(3 puntos)*** Un sistema basado en el procesador MIPS R2000 dispone de una **CACHE L1 DUAL** (una L1 de Instrucciones y una L1 de Datos), cuya configuración es de 8 KB cada una, 2 vías y tamaño de bloque 64B. La política de fallo en escritura es de Ubicación (*write allocate*) y la política de acierto en escritura (actualización) es de escritura Posterior (*write back*). El algoritmo de reemplazo es LRU.

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



b) *(0,25 puntos)* Calcule los siguientes parámetros de la cache L1 de Datos

Número de líneas	128
Número de conjuntos	64

c) *(0,5 puntos)* Calcule el tamaño en bits de la **memoria de control de la cache de Datos**. Indique claramente el **número de líneas o entradas** que contiene y el **tamaño en bits de cada entrada**, especificando el nombre de cada uno de los **campos que la integran**:

128 líneas × [V (1) + M (1) + Etiqueta (20) + LRU (1)]= 128 líneas × 23 bits = 2944 bits

d) El siguiente programa realiza la suma de los elementos V[1] a V[4095] del vector V de números enteros y almacena el resultado en la cabecera del vector (elemento V[0]): $V[0] = \sum_{i=1}^{i=4095} V[i]$

```
        .data 0x2F000800
A:      .word 1,2,3, …,4096     # vector de 4096 enteros (32 bits)

        .text 0x00400000
_start:li $t4,4095              # carga contador
       li $t2,0                 # inicializa suma=0
       lui $t0,0x2F00           # carga puntero a vector A
       ori $t0,$t0,0x0800
buc:   lw $t1,4($t0)            # lee V[i]
       add $t2,$t2,$t1          # suma=suma+V[i]
       addi $t0,$t0,4           # incrementa puntero a vector V
       addi $t4,$t4,-1          # decrementa contador
       bnez $t4,buc             # mientras contado<>0, seguir en el bucle
       lui $t0,0x2F00           # carga puntero a vector V
       ori $t0,$t0,0x0800
       sw $t2,0($t0)            # V[0]=suma
       .end
```

d.1) *(1,25 puntos)* Calcule los siguientes valores tanto para Instrucciones como para Datos

	CÓDIGO	**DATOS**
Número de bloques que lo contienen	1	256
Número de bloque de los dos primeros bloques (hex)	0x0010000	0x0BC0020 0x0BC0021
Conjunto al que se mapea el primer bloque (hex)	0x00	0x20
Etiqueta del primer bloque	0x00400	0x2F000
Número de FALLOS	1	257
Número de ACCESOS *(Mostrar el cálculo)*	4 + 5x4095 + 3 = 20482	4096
Número de reemplazos de bloque	0	129
TASA DE ACIERTOS	$H_{L1I}= 1 - \frac{1}{20482} = 0{,}999$	$H_{L1D}= 1 - \frac{257}{4096} = 0{,}937$
TASA DE ACIERTOS PROMEDIO *(Mostrar el cálculo)*	$H_{L1}= \frac{0{,}999\times20482 + 0{,}989\times4096}{24578} = 0{,}989$	

d.2) *(0,25 puntos)* Asumiendo la existencia de un nivel L2 de cache (Unificada I+D), con tasa de acierto H_{L2}= 0,9, y suponiendo que los tiempos de acceso de los niveles L1 y L2 son 1 ns y 4 ns, respectivamente, 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 programa

Tm= 0,989 x 1ns + (1-0,989) x {0,9 x 4ns + (1-0,9) x 300ns}= 1,36 ns

d.3) *(0,25 puntos)* Suponga que la L1 de Datos emplea correspondencia directa en lugar de correspondencia asociativa por conjuntos de 2 vías, manteniedo la capacidad. Comente cómo se vería afectada su **tasa de aciertos** y el **tamaño de su memoria de control**. Razone la respuesta

La tasa de aciertos no se verá afectada, dado que el fallo que se produce al acceder a V[0] es de capacidad, y este no se eliminaría cambiando la función de correspondencia, sino que requeriría aumentar el tamaño de la cache.

El tamaño de la memoria de control disminuye al reducirse el tamaño de la etiqueta (-1 bit) y eliminarse la necesidad de contador LRU (-1 bit).

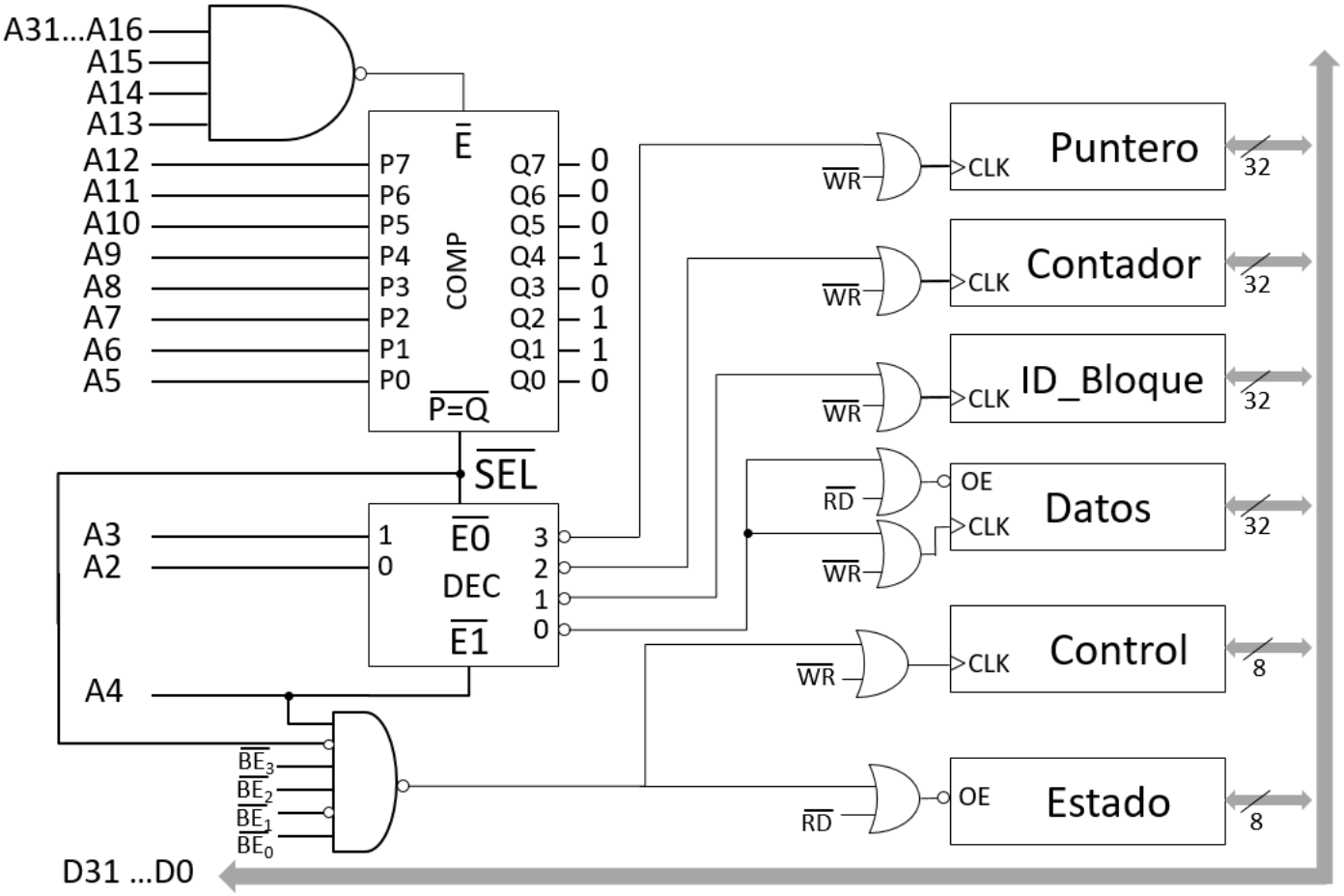
2 ***(2,5 puntos)*** La figura muestra la interfaz de un disco magnético. Esta interfaz se conecta a una CPU MIPS R2000. La interfaz soporta transferencia por PIO y por DMA. Los registros Estado y Control, que son de 8 bits, poseen los siguientes bits significativos:

Registro **CONTROL**:

- **PIO/DMA** (bit 7), a 0 indica el modo PIO y a 1 el modo DMA.
- **A** (bit 4), a 1 ordena al interfaz el inicio de una operación de lectura/escritura sobre el disco magnético.
- **CL** (bit 2), a 1 hace R=0.
- **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 0), indica al interfaz si se trata operación de lectura (R/W=1) o de escritura (R/W=0) sobre el disco magnético.

Registro **ESTADO**:

- **R** (bit 7) se activa a 1 cuando la transferencia del bloque a/desde memoria ha concluido



***Nota**: Los registros Control y Estado son de 8 bits, el resto son de 32 bits*

a) *(0,3 puntos)* Calcule la dirección base (DB) del interfaz

DB= 0xFFFFE2C0

b) *(0,7 puntos)* Calcule la dirección (DB+X) de cada uno de los registros del interfaz

Registro	Dirección
ESTADO	DB+17

Registro	Dirección
ID_BLOQUE	DB+4

Nota: en realidad ESTADO y CONTROL al no depender de A2 y A3 tienen vinculadas varias direcciones a parte de las indicadas, DB+21, DB+25… pero esto no afecta al funcionamiento y simplifica el dibujo.

CONTROL	DB+17
DATOS	DB

CONTADOR	DB+8
PUNTERO	DB+12

c) *(1 punto)* En el driver del disco magnético controlado a través del interfaz del esquema anterior se define la siguiente función que permite leer o escribir un bloque de 512 bytes:

Función	Índice	Argumentos
RW_Block	$v0= 444	$a0: Puntero a buffer de memoria $a1: Identificador del bloque $a2: 1 lectura – 0 escritura

La sincronización con el dispositivo se realiza por **INTERRUPCIÓN** al nivel de bloque. La función `RW_Block` deberá configurar adecuadamente el interfaz y habilitar la interrupción int3* en el interfaz. Suponiendo que el identificador del bloque que se desea escribir en el disco es 0xABCD0123, se pide:

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

c.1) (0,4 puntos) Escriba el código empleado en la invocación de la función `RW_Block` para **leer** de disco dicho bloque y almacenarlo en la zona reservada por la etiqueta Buffer.

```
        .data 0x20000000
Buffer: .space 512     # 512 bytes

        .text 0x00400000

        la $a0,Buffer
        li $a1, 0xABCD0123
        li $a2,1
        li $v0,444
        syscall
```

c.2) *(0,6 puntos)* Escriba el código que implementa la función `RW_Block` si se usa el modo DMA; tenga en cuenta que se transfiere una palabra en cada ciclo y el registro contador es de ciclos.

```
RW_Block:  la $t0, 0xFFFFE2C0
           sw $a0,12($t0)
           li $t1,128
           sw $t1,8($t0)
           sw $a1,4($t0)
           li $t1,0x92
           or $t1,$t1,$a2
           sb $t1,17($t0)
           jal suspende_proceso



           j retexc
```

d) *(0,5 puntos)* Describa brevemente las acciones que debería realizar la rutina de servicio de INT3* tanto en este caso (que usa DMA) como en el caso de usar PIO.

Acciones en modo DMA:

En este caso: Cancelar e inhibir la interrupción y activar proceso

Acciones en modo PIO:

En caso de usar modo PIO, transferir el bloque con un bucle leyendo del registro de datos y luego cancelar e inhibir la interrupción y activar proceso

3 **(3 puntos)** Se quiere desarrollar un sistema de apertura de una cerradura mediante un teclado, de forma que cuando se introduzca la clave correcta de n cifras, seguida de “Llave”, se abra la cerradura. Para ello se dispone de los siguientes dispositivos:

Teclado numérico con dos leds y un altavoz en la dirección base **0xFFFF0000**, conectado a la interrupción **INT0** del MIPS, y que posee los siguientes registros :

Registro de órdenes y estado (Lectura/escritura. Dirección = Base)

- R (bit 0, sólo lectura). Indicador de dispositivo preparado: R = 1 cada vez que se pulsa una tecla. Para la cancelación (hacer R = 0) es necesario realizar un acceso de lectura en el registro de datos.
- E: (bit 1, sólo escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)
- LED Rojo: (bit 3, sólo escritura). Valor 1 enciende el LED, 0 lo apaga.
- LED Verde: (bit 4, sólo escritura). Valor 1 enciende el LED, 0 lo apaga.
- Sonido: (bits 7..5, sólo escritura). Suena un sonido durante 1 segundo con un tono de 1 al 7, siendo el valor 0 sin sonido.

Registro de datos (Sólo lectura. Dirección = Base + 4)

- COD (bits 7...0). Código de la tecla del 0 al 9 ó 15 para “Llave”. Leer de este registro provoca que R = 0.

Cerradura electrónica en la dirección base **0xFFFF0010** y conectada a la interrupción **INT4** del MIPS:

Registro de órdenes y estado (Lectura/escritura. Dirección = Base)

- R (bit 0, sólo lectura). Indicador de cambio de estado: **R = 1 cada vez que la cerradura se abre o se cierra**.
- P (bit 1, sólo lectura). Indica si la cerradura está abierta (P=1) o cerrada (P=0).
- A (bit 2, sólo escritura). Abre la cerradura cuando se escribe un 1. La cerradura se cierra automáticamente con un temporizador.
- C (bit 3, sólo escritura). Escribiendo un 1 hace R = 0.
- E: (bit 4, sólo escritura). Habilitación de la interrupción (mientras E = 1, el valor R = 1 activa la línea de interrupción del dispositivo)

Se definen dos variables del kernel:

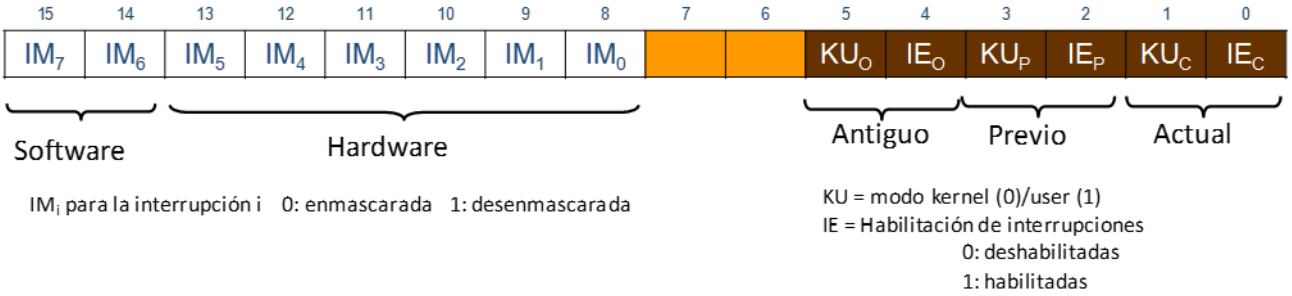
```
                .kdata
clave:          .word 0x1234
codpulsado:     .word 0
```

Se pide:

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

Función	Índice	Argumentos	Resultado
inicializar	$v0=550	-----	• Habilita las interrupciones 0 y 4 en el coprocesador 0 del MIPS, dejando el resto de bits inalterados • Habilita las interrupciones en los dispositivos. • Apaga el led verde • Enciende el led rojo • Inicializa a cero la variable **codpulsado**.
set_clave	$v0=560	$a0 = la nueva clave	Fija la clave con el valor de $a0, la clave es un número hexadecimal en el que cada cifra sólo puede valer de 0 a 9. Asumimos que la clave tiene este formato, no es necesario verificarlo.

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



```
inicializar:    la $t0,0xFFFF0000
                li $t1,0x0A          #LED rojo y E
                sb $t1,0($t0)
                la $t0, 0xFFFF0010
                li $t1,0x10          #Bit E
                sb $t1,0($t0)
                sw $zero,codpulsado
                mfc0 $t1,$12
                ori $t1,$t1,0x1100   #IM4, IM0
                mtc0 $t1,$12

                j retexc
```

```
set_clave:      sw $a0,clave

                j retexc
```

b) *(1 punto)* Programe el código de servicio de la rutina de interrupción **int4**, que encenderá el led verde (apagando el rojo) y emitirá un tono 5 cuando se abra la cerradura, mientras que encenderá el led rojo (apagando el verde) y emitirá un tono 4 cuando se cierre la cerradura. Se pueden usar $t0, $t1 y $t2.

```
int4:           la $t0, 0xFFFF0010
                li $t1, 0x18         #bits E y C para cancelar
                sb $t1, 0($t0)
                lb $t1, 0($t0)       #leemos estado
                andi $t1,$t1,0x02    #bit P
                beqz $t1,cerrada
                la $t0, 0xFFFF0000
                li $t1, 0xB2         #Tono 5, Led verde y E
                sb $t1, 0($t0)
                j retexc
cerrada:        la $t0, 0xFFFF0000
                li $t1, 0x8A         #Tono 4, Led rojo y E
                sb $t1, 0($t0)









                j retexc
```

c) *(1 punto)* Programe el código de la rutina de servicio de interrupción **int0** que actualiza la variable **codpulsado** con las teclas pulsadas y que **verifica la clave** tras pulsar "Llave", **procediendo a abrir la puerta si la clave es correcta**. Véase el ejemplo:

Tecla pulsada	Valor de **codpulsado**
	0x0000
5	0x0005
7	0x0057
3	0x0573
llave	Después de verificar, se pone de nuevo a 0x0

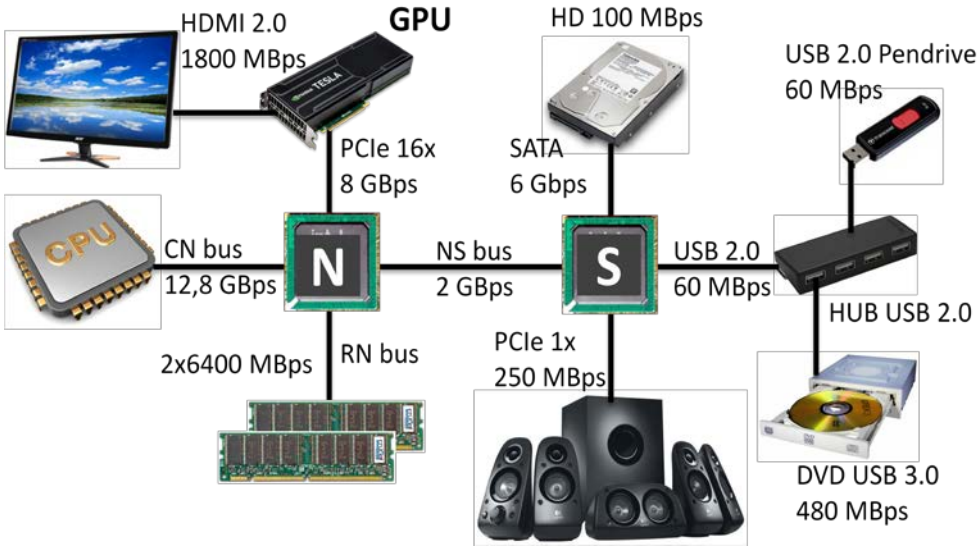
El comportamiento debe ajustarse al siguiente algoritmo:

```
Si la tecla es llave
   Si codpulsado == clave
      Abrir cerradura
      codpulsado = 0
   Sino
      Emitir tono 7
      Encender led rojo
      codpulsado = 0
Sino
   Emitir tono 1
   Añadir tecla a codpulsado (por la derecha, mediante desplazamientos a izquierda)
```

Se pueden usar $t0, $t1 y $t2.

```
int0:           la $t0, 0xFFFF0000
                lb $t1, 4($t0)        #esta lectura cancela R
                li $t2, 15            #código de la llave
                beq $t1,$t2,probar
                lw $t2,codpulsado
                sll $t2,$t2,4         #desplazamos 4 bits a izquierda
                or $t2,$t2,$t1        #añadimos el dígito pulsado
                sw $t2,codpulsado
                li $t1,0x22           #tono 1 y E
                sb $t1, 0($t0)
                j retexc
probar:         lw $t1, clave
                lw $t2, codpulsado
                sw $zero, codpulsado
                beq $t1,$t2,acierto
                li $t1,0xEA           #tono 7, Led rojo y E
                sb $t1, 0($t0)
                j retexc
acierto:        la $t0, 0xFFFF0010
                li $t1, 0x14          #bits E y A para abrir
                sb $t1, 0($t0)
```

4 ***(1 punto)*** En un sistema como el que se indica en la figura se está reproduciendo en tiempo real un vídeo almacenado en el DVD. Este codifica 2 minutos de película (vídeo+audio) en formato MPEG, con un tamaño de archivo de 960 MB. Se utiliza la GPU para descomprimir el vídeo y el audio. De esta forma, se mueve el vídeo comprimido por DMA desde el DVD a Memoria, mientras que la CPU lo lee de Memoria y lo transmite a la GPU, que procesará y enviará las imágenes descomprimidas a un monitor 4K a través de un conector HDMI, mientras que el sonido es enviado al equipo de audio, ambos también por DMA. La película descomprimida está formada por escenas de 4096×2160×24 bits a 60 escenas/segundo; por su parte, el audio es de 5.1 canales de 24 bits, y se halla muestreado a 96 Khz.



a) (0.5 puntos) Indique los siguientes anchos de banda requeridos, en MBps:

Lectura del vídeo MPEG desde el DVD a la Memoria: $\frac{960\ MB}{120\ seg} = 8\ MBps$

Reproducción de las imágenes enviadas desde la GPU al Monitor:

$$4096 \times 2160 \times 3\ bytes \times 60\ fps = 1593\ MBps$$

Reproducción del audio enviado desde la GPU al equipo de audio:

$$96000\ Hz \times 3\ bytes \times 6\ canales = 1{,}73\ MBps$$

b) (0.5 puntos) Estando en la situación anterior, se procede a transferir un archivo de 2 GB (10^9 bytes) desde el disco HD hasta el Pendrive, garantizando en todo momento la correcta reproducción de vídeo y audio. Para ello se lee del disco HD por DMA a la Memoria (MEM); la CPU lo va leyendo desde la memoria y envía desde ésta al Pendrive, también por DMA **de forma concurrente**. ¿Cuánto se tardará en transferir el archivo del disco HD al Pendrive? Indique también la ocupación (%) de los buses NS y USB 2.0

El archivo de 2000 MB deberá atravesar el bus USB 2.0, el cual está siendo usado al mismo tiempo por el flujo de tiempo real correspondiente al archivo comprimido MPEG, que precisa de un ancho de banda de 8MBps. Como este flujo es prioritario para la correcta reproducción del vídeo, el ancho de banda del USB 2.0 disponible para el archivo será 60MBps-8MBps= 52MBps, por lo que el tiempo de transferencia del archivo será $\frac{2000\ \text{MB}}{52\ \text{MBps}} = 38{,}46\ seg$;

Ocupación USB 2.0 = $\frac{8\ \text{MBps} + 52\ \text{MBps}}{60\ \text{MBps}} = 100\%$; Ocupación NS = $\frac{8\ \text{MBps} + 2\times 52\ \text{MBps} + 1{,}73\ \text{MBps}}{2000\ \text{MBps}} = 5{,}7\%$

5 ***(0.5 puntos)*** Sea un disco duro magnético formado por cuatro platos. El área útil de los platos está delimitada por un radio interno de 1" y un radio externo de 3", y se encuentra distribuida en 4 zonas o anillos. Cada anillo contiene 10.000 cilindros. La distribución de sectores (de 512 bytes de capacidad) es la siguiente:

	Zona 0	Zona 1	Zona 2	Zona 3
Sectores/pista	1000	850	650	500

Suponiendo que el disco gira a 15000 rpm, que el tiempo medio de posicionamiento es de 6 ms, y que el *track-to-track time* es de 1 ms, se pide.

a) *(0.25 puntos)* Calcule los siguientes parámetros del disco:

Capacidad del disco en GB (10^9 B):
$8\ caras \times 10000\ cilindros \times (1000 + 850 + 650 + 500) sectores/pista \times 512B$ =122,88 GB
Latencia media rotacional en ms: $\frac{60\ \text{seg}}{15000\ \text{rpm}} \times \frac{1}{2} = 2\ \text{ms}$
Ancho de banda de transferencia de la Zona 3 en MBps: $\frac{500 \times 512\text{B}}{4\ \text{ms}} = 64\ \text{MBps}$

b) *(0.25 puntos)* Calcule el tiempo medio necesario para leer un archivo de 10 MB (10×10^6 B), ubicado en la Zona 3? Suponga que el archivo está almacenado en el disco de forma óptima.

El archivo estará almacenado en sectores y cilindros consecutivos
Número de sectores ocupados por archivo = $\frac{10\ \text{MB}}{512\ \text{B}} = 19531{,}25 \cong 19532$ sectores
Número de pistas = $\frac{19532\ \text{sectores}}{500\ \text{sectores /pista}} \cong 40$ pistas ; Número de cilindros= $\frac{40\ \text{pistas}}{8\ \text{caras}} = 5$ cilindros
Tiempo medio de lectura = 6 ms + 2 ms + (5-1)×1 ms + $\frac{19532\ sectores \times 512\ B}{64\ MBps}$ = 12 ms + 156,26 ms = 168,26 ms