

Universidad de Murcia
Facultad de Informática

TÍTULO DE GRADO EN
INGENIERÍA INFORMÁTICA

Estructura y Tecnología de Computadores

Tema 7: Gestión de la entrada/salida

Apuntes

CURSO 2010 / 11

VERSIÓN 2.0

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores



Índice general

7.1. Introducción	1
7.2. Clasificación de los dispositivos de E/S	2
7.3. Programación de la entrada/salida	2
7.3.1. Puertos, controladoras y canales	3
7.3.2. E/S mapeada en memoria vs. E/S aislada	4
7.3.3. Técnicas de comunicación CPU–E/S	4
7.4. El papel del sistema operativo	9
7.5. Implementación de la E/S	11
7.5.1. Concepto de bus	11
7.5.2. Elementos de diseño de un bus	12
7.5.3. Parámetros de los buses	14
7.5.4. Protocolos de acceso al bus	15
7.5.5. Mecanismos de control de acceso	23
B7.Boletines de prácticas	24
B7.1.Programación de la E/S en MIPS	24
B7.1.1. Objetivos	24
B7.1.2. Prerequisitos	24
B7.1.3. Plan de trabajo	25
B7.1.4. Dispositivos mapeados en memoria.	25
B7.1.5. Programación de la E/S por sondeo	26
B7.1.6. Excepciones e interrupciones en MIPS	27
B7.1.7. Registros del coprocesador 0	28
B7.1.8. Programación de la E/S por interrupciones.	29
B7.1.9. Ejercicios	32
E7.Ejercicios	33
E7.1. Gestión de la E/S	33
E7.2. Solución a ejercicios seleccionados	40

7.1. Introducción

En los temas anteriores, hemos estudiado el procesador y el sistema de memoria, dos de los componentes básicos de un ordenador. En este último tema, en cambio, vamos a estudiar los distintos elementos que conforman el **sistema de entrada/salida** de un ordenador, el cual permite la comunicación entre los diferentes dispositivos conectados al sistema (figura 1). Básicamente, el sistema de E/S es el encargado de proporcionar información a la CPU y de permitir a ésta comunicarse con el exterior y con otros dispositivos.

El sistema de E/S de un computador moderno puede ser bastante complejo. Por ello, nos vamos a centrar sólo en algunos aspectos básicos, como son:

- Clasificación de los diferentes periféricos de E/S.
- Programación de los dispositivos de E/S.
- Papel del sistema operativo en las comunicaciones.
- Implementación de la E/S: buses y protocolos de acceso.

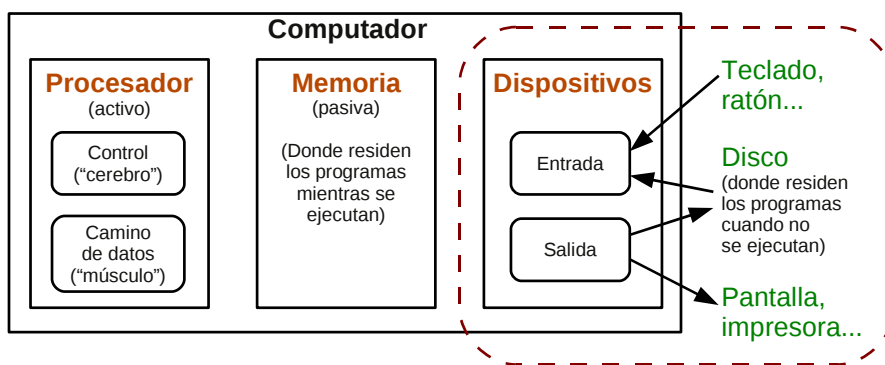


Figura 1: Esquema de Von Neumann de un ordenador.

7.2. Clasificación de los dispositivos de E/S

Debido a la gran cantidad de dispositivos que podemos utilizar para comunicar un ordenador con el exterior, se hace necesario algún tipo de clasificación para poder realizar un estudio más en profundidad. De hecho, existen varias clasificaciones de los dispositivos de E/S, algunas de las cuales se muestran a continuación.

Si nos centramos en el uso al que se destina un dispositivo, podemos llegar a la siguiente clasificación:

- **Almacenamiento:** discos duros, diversos tipos de disquetes (unidades ZIP, etc.), memorias flash, CDRoms, DVDs, grabadoras de CD/DVD, unidades de cinta, etc.
- **Interfaz con el usuario:** teclados, ratones, tabletas gráficas, joysticks, etc.
- **Visualización y multimedia:** tarjetas gráficas, monitores, impresoras, tarjetas de sonido, altavoces, etc.
- **Comunicaciones:** tarjetas de red (Ethernet, inalámbricas), módems, etc.
- **Adquisición de datos:** cámaras de vídeo, micrófono+tarjeta de sonido, etc.

Si realizamos la clasificación en función de la tasa de transferencia de datos o ancho de banda, observamos que existe una gran diversidad (desde unos pocos bytes a MB o incluso GB):

Dispositivo	Tipo	Interacción	Ancho de banda (KB/s)
Teclado	Entrada	Humana	0,01
Ratón	Entrada	Humana	0,02
Impresora de líneas	Salida	Humana	1,00
Impresora láser	Salida	Humana	100,00
Memoria flash	Almacenamiento	Máquina	20.000,00
Disco duro	Almacenamiento	Máquina	60.000,00
Tarjeta de red	Entrada/Salida	Máquina	125.000,00
Adaptador gráfico	Salida	Humana	3.000.000,00

Por lo tanto, una de las principales características de los dispositivos de E/S es su heterogeneidad (usos muy diversos, grandes diferencias en cuanto a velocidad), debida, en gran medida, a la tecnología utilizada para su construcción.

7.3. Programación de la entrada/salida

Aunque, como acabamos de comentar, desde el punto de vista tecnológico existe una gran heterogeneidad en los dispositivos de entrada/salida, la comunicación y programación siguen ciertas reglas comunes.

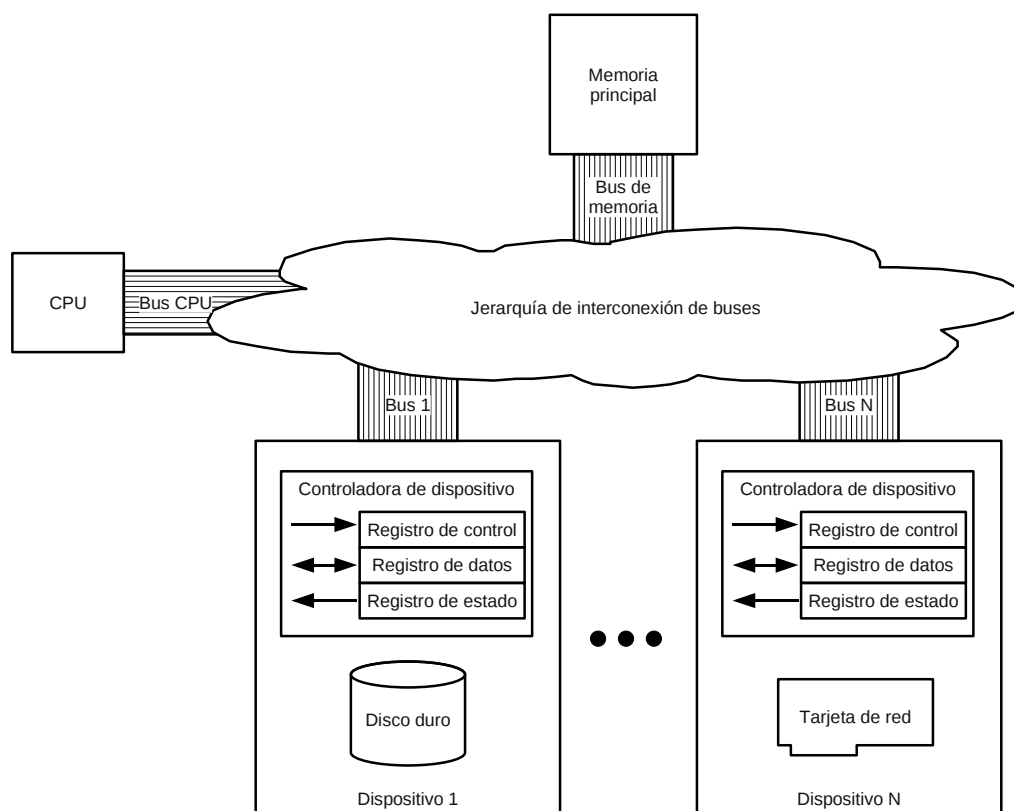


Figura 2: Esquema de comunicación de la CPU con la E/S a través de los puertos.

En esta sección nos centraremos en el estudio de la E/S desde el punto de vista del programador. Para ello empezaremos estudiando cómo se direcciona y accede a los puertos de los dispositivos para, a continuación, ver el tipo de instrucciones que se utilizan en dicho acceso y, por último, estudiar los mecanismos de interacción entre la CPU y los dispositivos: sondeo (*polling*), interrupciones o interrupciones + DMA (acceso directo a memoria).

7.3.1. Puertos, controladoras y canales

Todos los periféricos cuentan con una serie de *puertos de E/S*. Estos puertos están implementados como registros externos a la CPU a través de los cuales se comunican la CPU y los dispositivos (figura 2). Estos puertos de E/S se encuentran integrados en un chip llamado *controladora del dispositivo*.

Normalmente, en cualquier controladora tendremos los siguientes registros que nos facilitan la comunicación y el control del dispositivo:

- **Registro de datos:** en este registro se realiza la lectura/escritura del dato a transferir entre la CPU y el dispositivo.
- **Registro de control:** en él la CPU escribe las órdenes a realizar: leer, escribir, inicialización, configuración del dispositivo, etc.
- **Registro de estado:** nos permite saber en qué estado se encuentra el dispositivo. Por ejemplo, nos informa si está disponible para aceptar una nueva orden o se encuentra procesando una y debemos esperar (información de *ready/not ready*). Este registro suele disponer de bits adicionales para indicar otra información sobre el estado del dispositivo.

En cuanto a la comunicación, ésta se realiza a través de *canales* o *buses*, que se definen como el conjunto de líneas de comunicación que conectan los distintos componentes de un ordenador (CPU–memoria, dispositivos lentos–dispositivos rápidos, etc.). Los canales o buses se suelen organizar de forma jerárquica (figura 2) por cuestiones de eficiencia, tal y como se verá en la sección 7.5.

7.3.2. E/S mapeada en memoria vs. E/S aislada

Existen dos modos de **direccionamiento y acceso** a los puertos de los dispositivos por parte de la CPU: la **E/S mapeada en memoria** y la **E/S aislada**. La elección de uno u otro modo de acceso determinará el tipo de instrucciones a utilizar a la hora de programar la E/S.

En la *E/S mapeada en memoria* parte del espacio de direcciones se asocia a los dispositivos de E/S (son direcciones de memoria normales pertenecientes al espacio de direccionamiento convencional), pero una operación de lectura o escritura en dichas posiciones reservadas a los dispositivos es ignorada por la memoria, siendo capturada por la controladora del dispositivo correspondiente.

La principal ventaja de este modo de direccionamiento es la gran cantidad de instrucciones ya disponibles (todas las de acceso a memoria normales) así como una menor complejidad de la CPU al no tener que implementar instrucciones de E/S específicas (en consonancia con el concepto de procesador RISC). Entre los posibles inconvenientes está la utilización de parte del “valioso” espacio de direcciones convencional, aunque en la actualidad, con las arquitecturas de 32/64 bits, esto ha dejado de tener importancia.

Por otra parte, en el caso de tener una *E/S aislada*, son necesarias instrucciones especiales (no valen las de carga y almacenamiento de memoria normales) en las que se especificarán el dispositivo y el puerto donde se quiere leer o escribir. En general, lo que se hace en la E/S aislada es asignar un *número de puerto* concreto a cada registro de la controladora de un determinado dispositivo, por lo que dicho número determina a la vez tanto el registro a leer/escribir como el dispositivo con el que nos queremos comunicar. Estos números de puerto conforman, por tanto, un espacio de direcciones de E/S dedicado.

Como ejemplo de este modo de acceso tenemos a los procesadores de la familia Intel 8086¹. En estos procesadores, el acceso a los registros de la controladora de un dispositivo se realiza mediante las instrucciones IN y OUT:

```
IN reg,dir_puerto
OUT dir_puerto,reg
```

La principal ventaja de la E/S aislada es que no utiliza una parte de la memoria normal, lo que era útil en los primeros procesadores de 8/16 bits con un tamaño de memoria bastante limitado. Por contra, entre los inconvenientes podemos destacar la mayor complejidad de la CPU, al tener que implementar instrucciones adicionales de E/S, y el menor repertorio de instrucciones disponibles.

7.3.3. Técnicas de comunicación CPU–E/S

La lectura o escritura de un dato de o en un dispositivo requiere, por lo general, varias operaciones separadas de E/S en los distintos puertos del dispositivo. Por ejemplo:

- Escribir un código de orden en el registro de control.
- Leer el estado del dispositivo del registro de estado.
- Leer/escribir el dato de/en el registro de datos.

Estas operaciones son realizadas por la CPU en un cierto orden y unos momentos determinados. En esta sección vamos a ver las distintas opciones que tenemos.

¹Esta familia de procesadores también dispone de E/S mapeada en memoria, como es el caso de la memoria de vídeo.

Sondeo (*polling*)

La técnica más sencilla de comunicación entre la CPU y un dispositivo es la denominada técnica de *polling*, también llamada sondeo, encuesta o escrutinio. En ella, el procesador es el que realiza todo el trabajo, «sondeando» continua o periódicamente el registro de estado del dispositivo para, una vez detectado un cambio de estado, obrar en consecuencia. Por ejemplo, si pensamos en el ratón de un ordenador, podemos implementar esta técnica haciendo que el procesador verifique periódicamente el estado del ratón, para ver si el usuario lo ha movido, en cuyo caso el sistema operativo informa al programa asociado del cambio de situación.

El principal problema de esta técnica es que, cuando la E/S es mucho más lenta que la CPU, podemos estar sondeando muchas veces el dispositivo comprobando que no ha cambiado, lo que implica una gran pérdida de tiempo y un uso innecesario de la CPU².

Normalmente, los bucles de *espera activa* (encuesta continua) sólo son permisibles en dispositivos dedicados (sistemas empujados). Para computadores normales, que usan sistemas operativos de tiempo compartido, lo más habitual es la encuesta periódica, como es el caso del ratón.

Veamos a continuación un ejemplo para determinar el rendimiento de este método de comunicación. Supongamos que, en una CPU a 500 MHz, una operación de sondeo consume 400 ciclos de reloj (llamar a la rutina de sondeo, acceder al dispositivo y volver). Determinemos el porcentaje de tiempo que la CPU gasta realizando sondeos para los siguientes dispositivos:

- Ratón: debe escrutarse 30 veces/seg. para no perder el movimiento del usuario.

Ciclos por segundo consumidos en el sondeo del ratón: $30 * 400 = 12.000$ ciclos/seg. consumidos → % CPU: $12.000 / (500 * 10^6) = 0,0024$ % (impacto despreciable).

- Disquete: transfiere datos en unidades de 2 bytes y tiene un ancho de banda de 50 KB/seg. No pueden perderse datos.

Ciclos por segundo consumidos en el sondeo del disquete: $50 \text{ KB/s} / 2 \text{ bytes por transferencia} = 25.600$ encuestas/seg. → $25.600 * 400 = 10.240.000$ ciclos/seg. consumidos → % CPU: $10.240.000 / (500 * 10^6) = 2,048$ % (impacto apreciable, pero permisible si no hay muchos dispositivos).

- Disco duro: transfiere datos en unidades de 16 bytes y su ancho de banda es de 8 MB/seg. Tampoco pueden perderse datos.

Ciclos por segundo consumidos en el sondeo del disco duro: $8 \text{ MB/s} / 16 \text{ bytes por transferencia} = 524.288$ encuestas/seg. → $524.288 * 400 = 209.715.200$ ciclos/seg. consumidos → % CPU: $209.715.200 / (500 * 10^6) = 41,95$ % (impacto inaceptable).

Como se puede observar, aunque en determinadas situaciones la técnica de sondeo es perfectamente válida, para otros dispositivos es necesario recurrir a técnicas de comunicación más sofisticadas.

Interrupciones

La comunicación entre la CPU y los dispositivos de E/S mediante el uso interrupciones permite a la CPU delegar parte del trabajo al dispositivo y reducir así el impacto que las transacciones de E/S tienen sobre la CPU. Veamos mediante una analogía la diferencia entre la utilización de interrupciones y el uso del sondeo (analogía jefe–secretaria):

1. Encargamos a nuestra secretaria un trabajo (= orden de transacción de E/S).
2. Ahora, tenemos dos opciones:

²Esto es un problema añadido en sistemas portátiles, en los que la CPU sigue funcionando y gastando batería sin hacer nada productivo.

- a) Nos quedamos mirándola continuamente, esperando a que termine (= sondeo continuo) o bien hacemos otra cosa, pero cada X minutos nos acercamos a preguntarle si ha terminado (= sondeo periódico).
 - b) Continuamos haciendo otras cosas, sin preocuparnos ya de vigilar y pedimos a la secretaria que nos avise por teléfono cuando haya terminado el trabajo (= interrupción externa a la CPU).
3. En ambos casos, una vez detectamos que la subtarea ha sido realizada, utilizar sus resultados adecuadamente (= tratamiento de los datos: lectura/escritura de los mismos desde/a buffers adecuados, etc.)

Un aspecto importante a tener en cuenta es que mientras que en el método de encuesta el tratamiento de los datos lo hacen las instrucciones que siguen al bucle de espera activa, una interrupción puede llegar en cualquier momento, mientras que la CPU esta haciendo otra cosa (son asíncronas respecto al programa en ejecución). Por lo tanto, hay que habilitar mecanismos para que:

- La CPU deje automáticamente lo que esté haciendo y pase a ejecutar la *rutina de servicio de la interrupción* (RSI), también llamada manejador.
- Se salve la información mínima necesaria para que luego pueda recuperarse el estado de ejecución en el que estaba el programa justo en el momento en que llegó la interrupción.
- Al finalizar la RSI, se vuelva justo a dicho estado y se reanude el proceso interrumpido con normalidad.

La principal ventaja de las interrupciones frente al sondeo es que la CPU no gasta ciclos útiles en comprobar si el dispositivo está preparado. Simplemente, cuando éste lo esté, y lo indique mediante la interrupción correspondiente, se ejecutará la rutina de servicio asociada. Por lo tanto, también será necesario identificar qué dispositivo ha enviado la interrupción. Para esto, hay dos opciones:

- Mediante la asignación de un pin de entrada de interrupción en la CPU por cada dispositivo (como en el MIPS).
- Mediante un mecanismo de dos pasos: la activación de un único pin de interrupción, igual para todos los dispositivos, seguido de la identificación del dispositivo mediante la inserción de un código (número de interrupción) en el bus de datos de la CPU (como ocurren en los procesadores Intel IA-32).

Una vez que tenemos claro cómo se puede identificar al dispositivo que ha realizado la interrupción, nos queda por estudiar la implementación de los mecanismos de los que hemos estado hablando: salto a la rutina de servicio, salvar el estado de la CPU y reanudación del proceso interrumpido.

En cuanto al *salto a la rutina de servicio*, la duda que se nos plantea es dónde hay que saltar, ya que la única información que tenemos es qué dispositivo ha realizado la petición. De nuevo, existen dos opciones:

- Saltar siempre al mismo lugar del código (dirección física fija) al producirse la interrupción. Las primeras instrucciones de ese código serían las encargadas de mirar qué dispositivo provocó la interrupción y, a partir de esa identificación, bifurcar al tratamiento adecuado. Éste es el caso del MIPS, donde siempre se salta a la dirección 0x80000180.
- Saltar a un lugar variable, dependiendo del número de interrupción producida. Por ejemplo, en los procesadores Intel IA-32 se busca la dirección correspondiente en un array de direcciones indexadas por dicho número (interrupciones vectorizadas)³.

³Algunas versiones modernas de MIPS también incorporan esta modalidad.

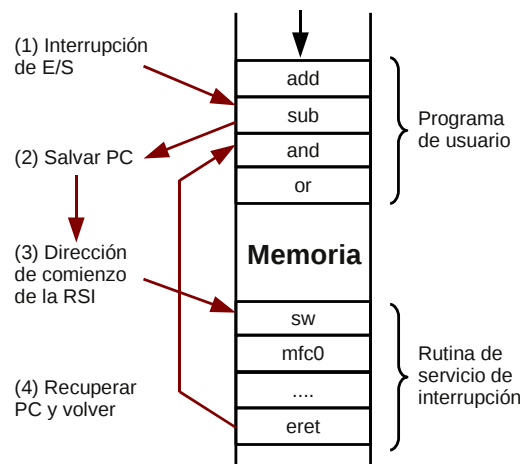


Figura 3: Resumen del esquema de llamada a una rutina de servicio.

Una vez que hemos saltado a la rutina de servicio, el siguiente paso consiste en *salvar el estado de la CPU*. Como mínimo, hay que salvar siempre el valor del PC. Además, en las máquinas que utilizan un registro de estado para las condiciones (p.e., *flags* en Intel IA-32), también hay que salvar dicho registro. Por supuesto, todos aquellos registros modificados por la rutina de servicio deben ser salvados, pero esto lo hace la propia rutina de forma análoga a como se hace en una rutina normal con los registros preservados entre llamadas (ver tema 3), pero para todos los registros ya que una rutina de servicio es asíncrona a la ejecución normal del programa y puede ser invocada en cualquier momento.

El último paso, una vez ejecutada la RSI, consiste en *recuperar el estado y reanudar el proceso interrumpido*. Para ello debemos recuperar los registros que la rutina de servicio apiló antes de ser utilizados y actualizar el valor del PC con el del EPC guardado, lo cual provoca la reanudación del proceso. La figura 3 muestra de forma resumida el proceso completo.

Veamos a continuación un ejemplo que nos permita analizar cuál sería el rendimiento al utilizar el mecanismo de interrupciones. Supongamos un gasto de 500 ciclos por cada transferencia (tiempo de ejecución de la rutina de servicio); también que el disco duro se usa durante un 5 % del tiempo. Consideremos, por otro lado, la misma CPU y disco duro que en el caso anterior. Entonces tenemos que:

- Interrupciones de disco duro/seg.: $8 \text{ MB/s} / 16 \text{ bytes por transf.} = 524.288 \text{ interrupciones/seg.}$
- El consumo de ciclos/seg. será: $524.288 \text{ intr/seg.} * 500 \text{ ciclos/intr.} = 262.144.000 \text{ ciclos/seg.}$
- % CPU (durante las transferencias): $262.144.000 / (500 * 10^6) = 52,4 \%$
- % CPU (total; sólo 5 % tiempo activo): $52,4 \% * 5 \% = 2,62 \%$

Es decir, puesto que no hay que estar sondeando continuamente el disco duro y éste se usa sólo durante el 5 % del tiempo, el porcentaje total de uso de CPU se reduce al 2,62 %, muy inferior al 41,95 % que obteníamos usando sondeo.

En verdad, en el caso del sondeo, si también supusiéramos que el disco duro sólo se usa durante un determinado porcentaje de tiempo, el porcentaje de uso de la CPU también se podría reducir considerablemente. La razón es que, puesto que es la propia CPU la que inicia las transferencias con el disco duro, ésta no necesita estar sondeando siempre, sólo cuando hay una operación en curso para saber cuándo ha terminado. Sin embargo, hay otros dispositivos, como puede ser una tarjeta de red, donde una transferencia puede llegar en cualquier momento. En estos casos, el uso del sondeo haría que se tuviera que estar consultado continuamente el estado del dispositivo, mientras que el uso de interrupciones haría que sólo se atendiera al dispositivo cuando realmente hubiera trabajo que hacer.

Como conclusión podemos decir que una interrupción de E/S es como una excepción (necesita rutina de tratamiento, capacidad de recuperación, etc.), pero con la gran diferencia de que la interrupción de E/S es *asíncrona* respecto al programa ya que, a priori, no se sabe en qué momento se producirá (no se produce al ejecutar una instrucción, como ocurre con las excepciones), necesitando, por tanto, saber qué dispositivo la provocó. En el resto del tema utilizaremos la siguiente terminología (muy variable en la literatura):

- **Excepción⁴**: una instrucción provoca una situación anormal o no permitida durante su ejecución (p.e., violación de acceso, *overflow*, fallo de página, etc.).
- **Interrupción**: un dispositivo externo a la CPU indica a la CPU que necesita su atención.
- **Trap**: es una instrucción especial que el programador puede usar para provocar una llamada a una rutina de servicio (llamada al sistema operativo).

Interrupciones y DMA

El sondeo y las interrupciones son adecuados para dispositivos con escaso ancho de banda ya que permiten reducir el coste de la controladora y de la interfaz, dejando casi todo el peso de las transferencias a la CPU y al sistema operativo: cada transferencia de una palabra a/desde el dispositivo implica una secuencia de instrucciones que la CPU tiene que ejecutar para transferir la palabra desde/a memoria (mediante la rutina de sondeo o la de servicio de interrupción).

Con dispositivos de gran ancho de banda, sin embargo, las transferencias constan principalmente de grandes bloques de datos donde los mecanismos como el sondeo o las interrupciones no son métodos adecuados, pues se mantendría ocupado el procesador demasiado tiempo. En estos casos, se utiliza el *acceso directo a memoria* (DMA) que permite que la transferencia de datos desde el dispositivo a la memoria o viceversa se realice sin la intervención del procesador.

El DMA se implementa con un chip especializado (controladora de DMA), que transfiere datos entre un dispositivo de E/S y la memoria principal independientemente del procesador. La introducción del DMA lleva aparejada la aparición de múltiples buses del bus (CPU y DMA), lo que a nivel hardware implica la necesidad de disponer de un sistema de arbitraje del bus (sección 7.5.5).

Los pasos de los que consta una transferencia DMA son los siguientes (ver figura 4):

1. El procesador inicializa la controladora de DMA: identidad del dispositivo, operación a realizar, dirección de memoria donde leer o escribir, número de bytes a transferir y sentido del desplazamiento (hacia direcciones crecientes o decrecientes de memoria).
2. La controladora de DMA va pidiendo el bus y, cuando lo consigue, va realizando tantas operaciones de transferencia entre el dispositivo y la memoria como sean necesarias.
3. Finalmente, la controladora de DMA genera una interrupción informando al procesador de la finalización de la transferencia o de una condición de error.

Siguiendo con nuestra analogía jefe–secretaria:

1. Encargamos un trabajo bastante grande a nuestra secretaria (p.e., varios informes) (= programación de la controladora de DMA).
2. Dejamos que haga todos los informes en su despacho, sin interrumpirnos cada vez que termine uno de ellos para enseñárnoslo. En vez de ello, cada vez que termina uno, lo va acumulando en una esquina de nuestra mesa, sin interrumpirnos (= acceso directo a memoria por parte de la controladora).
3. Una vez que termina de realizar y colocar el último informe, nos avisa de que ya lo tenemos todo en nuestra mesa, listo para ser utilizado (= interrupción de notificación de finalización).

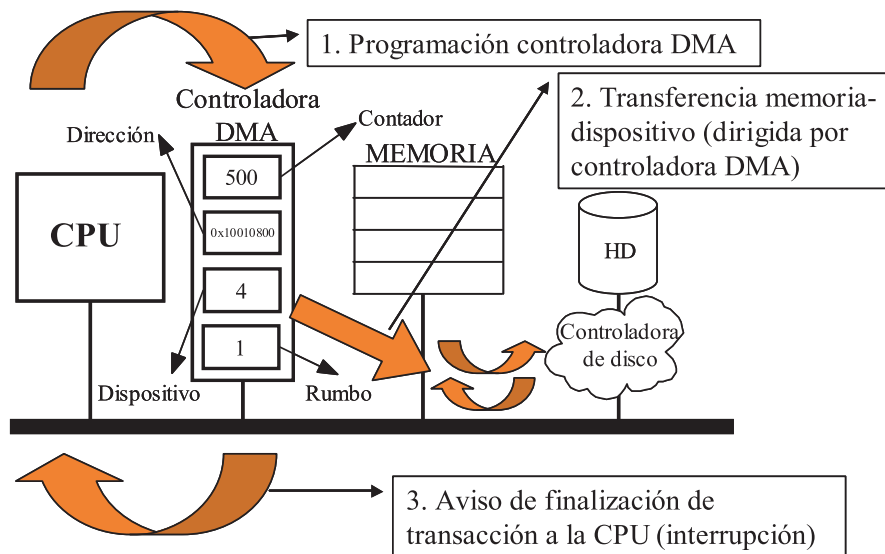


Figura 4: Secuencia de acciones en una transferencia DMA.

Al igual que hemos hecho antes, veamos a continuación el rendimiento que podemos obtener utilizando la técnica de DMA. Supongamos la misma CPU y disco duro de los ejemplos anteriores y también que el proceso de inicio de la controladora DMA (programación de los puertos correspondientes) tarda 1.000 ciclos y que la RSI de tratamiento, una vez finalizada la transferencia, tarda 500 ciclos. Además, supondremos que cada transferencia DMA es de 4KB. Si el disco estuviese continuamente transmitiendo:

- 8 MB por seg. / 4KB por transf. = 2048 transf. DMA por segundo
- Ciclos CPU/seg.: $(1.000+500) * 2048 = 3.072.000$ ciclos/seg.
- % CPU (durante las transferencias): $3.072.000 / (500 * 10^6) = 0,62 \%$

Como sólo transmite el 5 % del tiempo:

- % CPU (total; sólo 5 % tiempo activo): $0,62 * 5 \% = 0,031 \%$ (prácticamente despreciable⁵)

Finalmente, la figura 5 muestra mediante un diagrama el procesamiento de una transferencia de entrada/salida utilizando los tres mecanismos estudiados.

7.4. El papel del sistema operativo

Como ya se comentó al principio de este tema, existe una gran variedad de dispositivos de E/S, con lo que su control puede llegar a ser bastante complicado. Por suerte, el sistema operativo va a ser el encargado de ocultarnos esa heterogeneidad y proporcionarnos una interfaz homogénea y sencilla para el manejo de los dispositivos. El sistema operativo es, por tanto, el encargado de mantener el control del estado de los dispositivos, conocer las direcciones de los puertos para leer/escribir los datos, realizar las órdenes asociadas a cada dispositivo, manejar los errores, etc.

Una vez que arranca el ordenador y se realizan algunas comprobaciones iniciales, el primer código que pasa a ejecutarse es el del SO que, entre otras muchas tareas, carga los *drivers* de los distintos dispositivos, es decir, carga las rutinas de petición de E/S y las posibles RSI asociadas. Por lo tanto, sólo el SO tiene

⁴A veces, por abuso del lenguaje, se usa «excepción» como un término genérico para los 3 casos.

⁵El ejemplo supone que el DMA no interfiere con la CPU, lo cual puede no ser muy realista, pues los accesos a memoria de la CPU pueden colisionar con los del DMA.

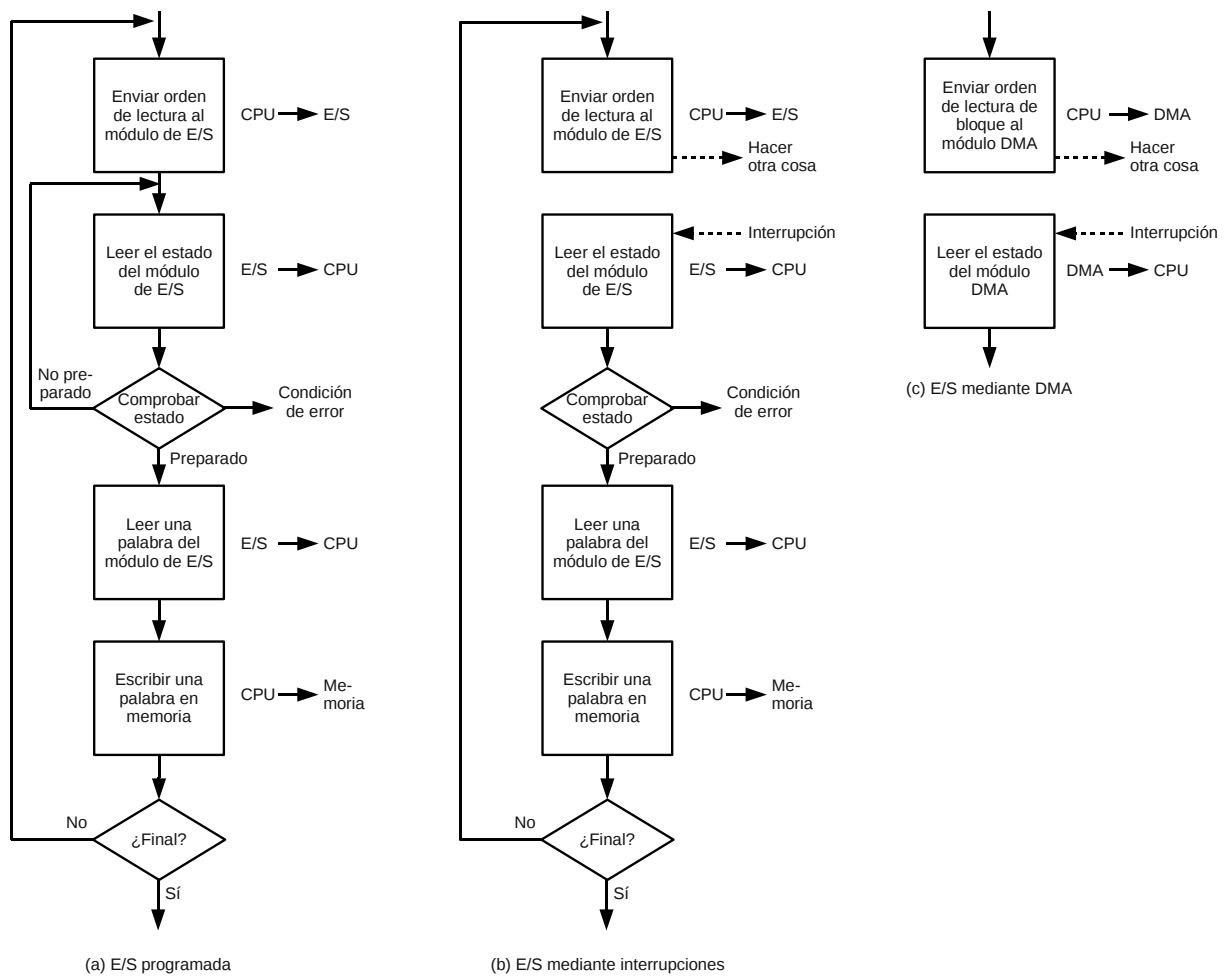


Figura 5: Resumen de los mecanismos de sondeo, interrupciones y DMA.

conocimiento de los puertos, órdenes, etc., necesarios para acceder a los dispositivos. El usuario solicita sus servicios a través de mecanismos sencillos con parámetros bien definidos (llamadas al sistema o *syscall*). Además, y como medida de seguridad, el sistema operativo es el único que puede acceder a la E/S; el resto de programas tendrá que utilizar las llamadas preestablecidas por el mismo como única forma de acceder a la E/S (los usuarios no podrán, por ejemplo, utilizar directamente los puertos).

7.5. Implementación de la E/S

7.5.1. Concepto de bus

Los diferentes subsistemas que componen un ordenador (procesador, memoria principal, memoria caché, tarjetas de expansión, dispositivos de almacenamiento, etc.) necesitan comunicarse entre sí. Pues bien, esta comunicación se lleva a cabo a través de un *bus* o canal⁶. Un *bus* es básicamente un canal de comunicación compartido en el que existen puntos de acceso o lugares en los que un dispositivo puede conectarse para formar parte del bus y así poder comunicarse con el resto de dispositivos conectados al mismo bus.

Los elementos más importantes a tener en cuenta a la hora de diseñar un bus son:

- Líneas del bus: distinguimos entre líneas de datos, de control y de direcciones.
- Anchura, longitud y frecuencia del bus: determinan el *ancho de banda* o cantidad de información que el bus puede transmitir por unidad de tiempo.
- Temporización del bus: pudiendo ser síncrona o asíncrona, e indica la forma en la que se coordinan los eventos en el bus.
- Control de acceso al bus, también conocido como arbitraje del bus.

Antes de pasar a describir cada uno de estos elementos que intervienen en el diseño de un bus, vamos a definir los conceptos de *jerarquía de buses* y *puentes de conexión*.

Jerarquía de buses múltiples

Si se conectan un gran número de dispositivos a un bus, las prestaciones de éste pueden disminuir al aumentar el retardo de propagación debido al tiempo de coordinación de los diferentes dispositivos para el uso del bus y, además, se forma un *cuello de botella* (o congestión) al estar los dispositivos continuamente esperando que sea su turno para usar el bus común. Se puede intentar paliar este problema aumentando el ancho de banda del bus, pero la solución más efectiva consiste en utilizar varios buses, de diferentes velocidades, y organizarlos de *forma jerárquica*.

La ventaja principal de utilizar una organización jerarquizada de los buses es que los dispositivos de E/S más exigentes se ponen más cerca del procesador y se les da prioridad con el fin de evitar los temidos cuellos de botella.

En la figura 6 se puede observar una jerarquía de buses típica que se puede encontrar en un ordenador actual. Se pueden ver el **bus local** y el **bus de memoria** que conectan el procesador con la jerarquía de memoria (caches externas y memoria principal). Éstos son los buses de mayor velocidad ya que el rendimiento de todo el sistema depende en gran medida de la comunicación entre la jerarquía de memoria y el procesador.

En otro nivel de nuestra jerarquía de buses se encuentran los **buses de expansión** tales como el bus PCI (*Peripheral Component Interconnect*) y el bus ISA (*Industry Standard Architecture*), a los que se conectan una gran variedad de dispositivos de E/S, desde tarjetas de sonido hasta impresoras. Con esta configuración podemos aislar el tráfico entre la memoria y el procesador del tráfico correspondiente al resto de dispositivos de E/S.

⁶Los conceptos de bus y de jerarquía de buses ya se introdujeron en Fundamentos de Computadores.

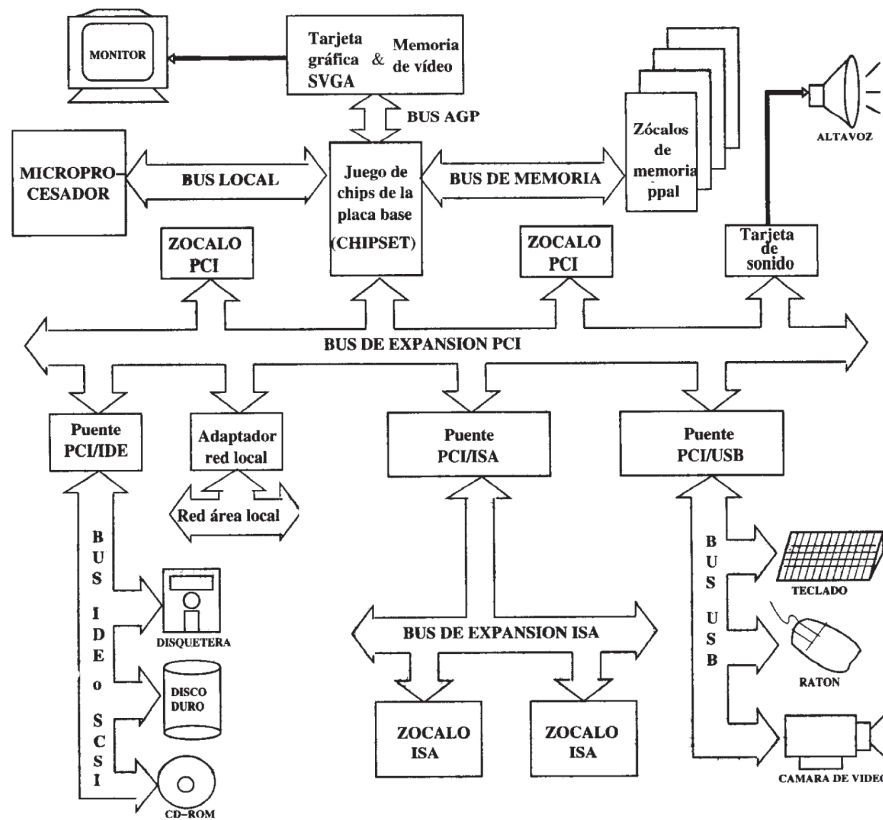


Figura 6: Jerarquía de buses múltiples.

Finalmente, en una jerarquía de buses típica también se encuentran los **buses dedicados** o de propósito específico, tales como los buses IDE y SCSI para dispositivos de almacenamiento (discos duros, CD-ROM, DVD-ROM, disqueteras), el bus AGP (*Accelerated Graphics Port*) para tarjetas gráficas de alto rendimiento y los buses *firewire* y USB para multitud de dispositivos de entrada/salida que podemos encontrar en el mercado (discos duros, lectores de tarjetas flash, cámaras de fotos/vídeo/webcam, teclados, ratones, impresoras, scanners, etc.)

En definitiva, se pretende tener múltiples y diferentes buses jerarquizados, para que dispositivos de diferentes velocidades no interfieran entre sí y se eviten congestiones de tráfico debidas a que los dispositivos más lentos impidan que los más rápidos se comuniquen con el procesador.

Puentes de conexión al bus

Los puentes de conexión al bus permiten la conexión de dos buses diferentes y son capaces de interpretar el contenido de los paquetes que reciben con el fin de realizar una **traducción** de dichos paquetes para poder transferirlos de un bus a otro. En la figura 7 se pueden observar los puentes PCI/IDE, PCI/ISA y PCI/USB. Además de la función de traducción de paquetes, los puentes también se encargan de la **gestión del tráfico**, puesto que son capaces de decidir si los paquetes que les llegan deben ser transmitidos o no al otro extremo en función de la dirección de destino.

7.5.2. Elementos de diseño de un bus

En la sección anterior, ya vimos que un bus es un canal de comunicación compartido que conecta a los diferentes componentes del ordenador. El hecho de ser compartido implica que las comunicaciones son

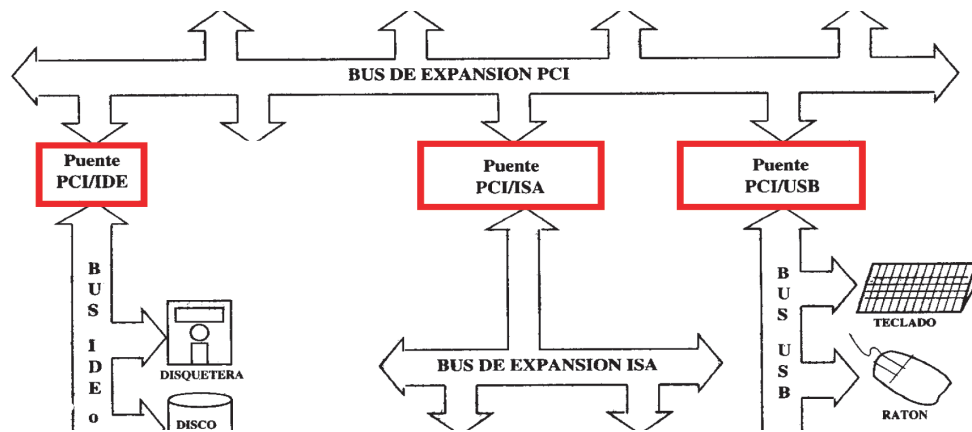


Figura 7: Ejemplos de diferentes puentes de conexión.

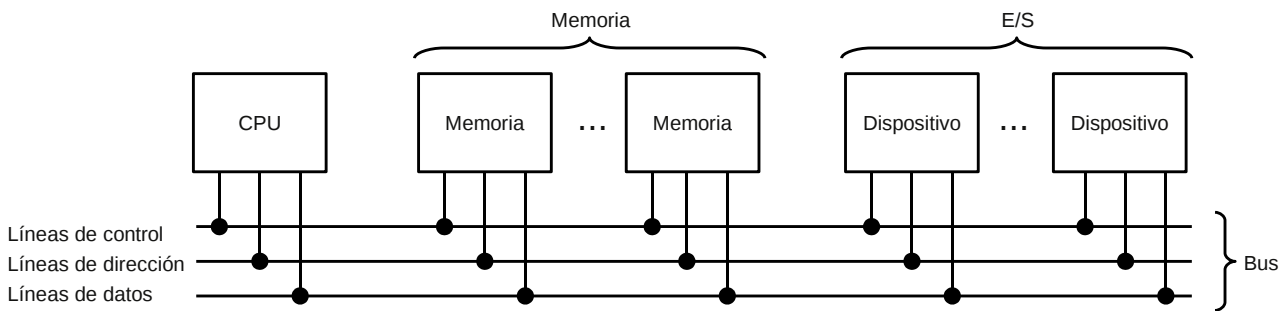


Figura 8: Esquema de un bus de interconexión.

escuchadas por todos los dispositivos conectados a dicho bus y que cualquier señal transmitida por uno de los dispositivos está disponible para que el resto de los dispositivos conectados al bus puedan acceder a ella.

Por este motivo, debemos diferenciar entre un conjunto de **líneas de control** y un conjunto de **líneas de información**, como se aprecia en la figura 8. Cada línea es capaz de transmitir señales binarias representadas por 0 y 1. Las líneas de control se utilizan para señalar peticiones y reconocimientos, y para indicar qué tipo de información se encuentra en las líneas de información en cada momento de la transferencia. Por otro lado, las líneas de información se usan para transmitir tanto direcciones de lectura/escritura como los datos propiamente dichos.

Líneas de control. Se utilizan para gestionar el acceso y uso de las líneas de información. Puesto que las líneas de información son compartidas por todos los componentes, debe existir una forma de controlar su uso. Las líneas de control transmiten tanto información de temporización como órdenes que especifican las operaciones a realizar; también sirven para determinar el tipo de datos que viaja por las líneas de información. Algunas líneas de control típicas se pueden observar en la tabla 1 (es importante tener en cuenta que un bus no tiene por qué tener todas las líneas de control que aparecen en la tabla; dependiendo del bus, éste tendrá unas u otras).

Líneas de información. Dentro de las líneas de información distinguimos entre las líneas de datos y las líneas de dirección:

Líneas de datos: proporcionan un camino para transmitir datos entre los dispositivos del sistema. El conjunto constituido por estas líneas se denomina *bus de datos* y el número de líneas se conoce como *anchura del bus de datos*. La anchura del bus de datos es un factor clave a la hora de determinar las prestaciones del sistema. Por ejemplo, si la anchura del bus es de 8 bits y las

Tabla 1: Ejemplos de líneas de control y efecto que produce su activación.

Línea	Efecto que produce su activación
Escritura en memoria	Hace que el dato del bus se escriba en la posición direccionada
Lectura de memoria	Hace que el dato de la posición direccionada se sitúe en el bus
Escritura de E/S	Hace que el dato del bus se transfiera al puerto de E/S direccionado
Lectura de E/S	Hace que el dato del puerto de E/S direccionado se sitúe en el bus
Petición de bus	Indica que un módulo necesita disponer del control del bus
Concesión de bus	Indica que se cede el control del bus a un módulo que lo solicitó
Petición de interrupción (INTR)	Indica que hay una interrupción pendiente
Interrupción reconocida (INTA)	Indica que la interrupción pendiente se ha reconocido
Reloj	Se utiliza para sincronizar las operaciones

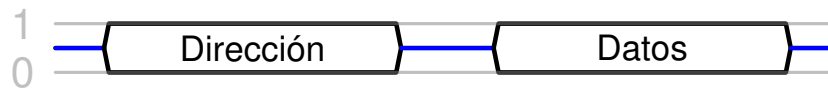


Figura 9: Ejemplo de transferencia usando líneas de información multiplexadas.

instrucciones son de 16 bits, entonces la CPU debe acceder a memoria dos veces cada vez que desee traer una instrucción desde memoria.

Líneas de dirección: se utilizan para determinar la fuente o el destino del dato situado en el bus de datos. El conjunto constituido por estas líneas se denomina *bus de direcciones*. La *anchura del bus de direcciones* determina el número de direcciones disponibles y, por tanto, dependiendo del tipo de bus, la capacidad máxima de memoria en el sistema, el número máximo de puertos direccionables, etc.

Existen dos posibilidades a la hora de diseñar las líneas de información:

- **Líneas de información multiplexadas:** la información para direcciones y datos se transmite a través del mismo conjunto de líneas en instantes distintos (figura 9). Ya que las líneas desempeñan varias funciones, es necesario saber cuándo contienen una dirección y cuándo contienen un dato. Como veremos después, esto depende tanto de la temporización del bus como de su protocolo. Este método es una multiplexación en el tiempo. Tiene como ventajas el ahorro de espacio y costes en el bus y las desventajas de necesitar una circuitería más compleja en cada módulo así como el reducir las prestaciones del bus al compartir los eventos las mismas líneas, impidiendo su realización en paralelo.
- **Líneas de información dedicadas:** cada grupo de líneas tiene una única función específica que realizar; habrá, por tanto, unas líneas de datos y otras de direcciones (figura 10). Con este método se consigue mejorar las prestaciones del bus aunque, por el contrario, es necesario disponer de más líneas por lo que se incrementa el coste del mismo.

7.5.3. Parámetros de los buses

Veamos ahora cuáles son los parámetros más importantes a la hora de diseñar un bus con el fin de poder evaluar posteriormente su rendimiento, que vendrá definido, principalmente, por el *ancho de banda* del bus.

- **Anchura del bus:** se define como el número de líneas de información de las que consta dicho bus. En este sentido, podemos diferenciar entre:

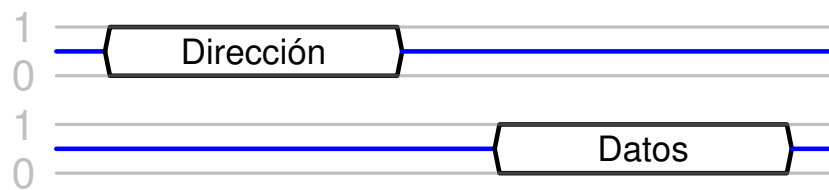


Figura 10: Ejemplo de transferencia usando líneas de información dedicadas.

- **Buses en serie:** con una única línea de información, puesto que únicamente se transmite un bit a la vez (USB, Firewire, Ethernet, etc.)
- **Buses en paralelo:** en los que se transmiten varios bits de forma simultánea (PCI, bus del sistema, etc.). En los buses en paralelo el número de líneas de información se puede incrementar de forma que se pueda transferir una palabra o más en una única transferencia.
- **Longitud del bus:** que puede oscilar desde menos de un metro (bus del sistema, PCI), varios metros (bus SCSI) e incluso llegar hasta cientos de metros (bus Ethernet). Es importante destacar que, a mayor longitud del bus, menor frecuencia de funcionamiento y también menor suele ser la anchura del bus.
- **Frecuencia de funcionamiento:** se define como la frecuencia de la señal de reloj que rige las transferencias realizadas. Éste es un parámetro que sólo aparece en los *buses síncronos* (detallados más adelante).
- **Ancho de banda (*bandwidth*) teórico:** se define como la cantidad de información que puede ser transmitida a través de un bus de comunicación. Se mide en cantidad de información por unidad de tiempo (Kbits/seg, Mbit/seg, Mbytes/seg, etc.). Se calcula con la expresión:

$$AB = \frac{n}{8} \times f, \text{ donde } n \text{ es el número de líneas de datos y } f \text{ es la frecuencia de funcionamiento.}$$

Cabe distinguir el ancho de banda teórico del **ancho de banda efectivo**. Este último se refiere a la cantidad de información que *realmente* se transmite, teniendo en cuenta que para realizar una comunicación entre dos o más dispositivos es necesario dedicar varios ciclos para el protocolo de acceso y para realizar el arbitraje del bus. Más adelante se ilustrará mejor la diferencia entre ancho de banda teórico y efectivo con un ejemplo.

7.5.4. Protocolos de acceso al bus

¿Por qué se necesita un mecanismo para controlar los accesos al bus? Sin ningún control, varios de los dispositivos conectados al bus podrían tratar de acceder simultáneamente a las líneas de control y de información para diferentes transferencias, llegando, por tanto, a una situación caótica.

Por ese motivo, mediante los protocolos de acceso al bus se pretende definir unas políticas de señalamiento y secuenciación que permitan la interacción coordinada y temporizada entre los dispositivos que están tratando de comunicarse. Dicho de otra manera, un protocolo de acceso al bus define qué pasos dar por los dispositivos que se quieren comunicar y en qué instantes se tienen que dar dichos pasos para que la comunicación finalmente tenga éxito.

Para ilustrar la necesidad de un protocolo de acceso al bus, veamos cómo se produciría, a grandes rasgos, la comunicación entre la CPU y la caché de primer nivel cuando se ejecuta una instrucción de carga que necesita acceder a la caché para realizar la lectura de una palabra:

1. El procesador pone la dirección de la palabra a leer en las líneas de dirección del bus y activa la señal de control «leer de memoria».

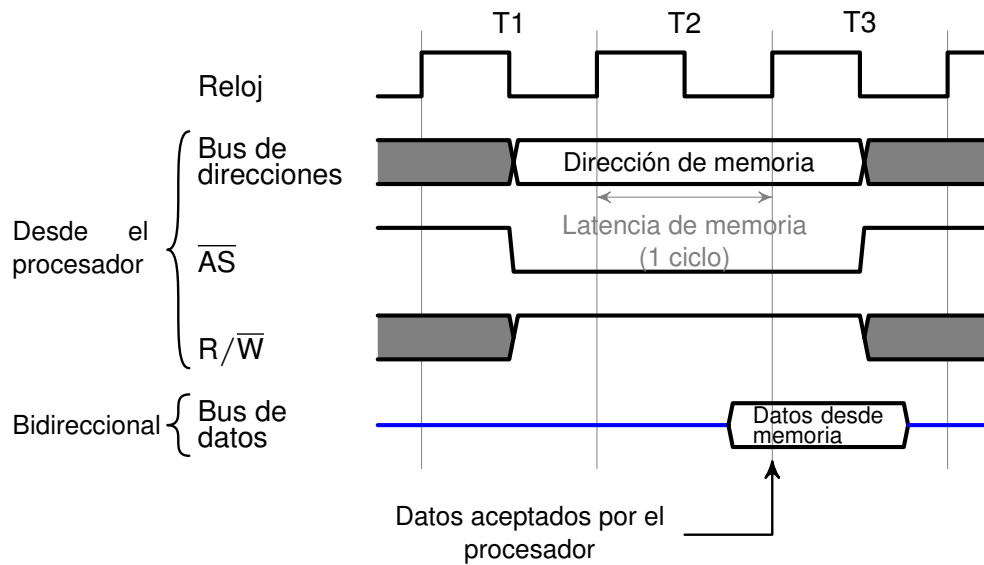


Figura 11: Ejemplo de temporización síncrona de una lectura de memoria.

2. La caché recibe la señal de lectura, realiza el acceso usando dicha dirección, obtiene el dato apropiado pasado un cierto tiempo, para finalmente poner el dato en las líneas de datos del bus.
3. Mientras tanto, el procesador ha quedado *en espera* hasta que el dato aparece en el bus y, en dicho momento, lee la palabra del bus, finalizando la comunicación.

Protocolos con temporización síncrona

El término *temporización* hace referencia a la forma en la que se coordinan los eventos en el bus, de forma que podemos distinguir entre protocolos con *temporización síncrona* y protocolos con *temporización asíncrona*.

En una temporización síncrona se incluye una línea de reloj en el conjunto formado por las líneas de control del bus y se sigue un protocolo estricto cuya ejecución irá marcada por los ciclos o pulsos del reloj. A través de la señal de reloj se transmite una secuencia en la que alternan unos y ceros a intervalos regulares de igual duración. Un único intervalo a uno seguido de otro a cero es lo que se conoce como *tiempo de ciclo de reloj* o ciclo de bus.

La ventaja de los protocolos con temporización síncrona es que tienen un mecanismo de arbitraje sencillo, lo que unido a una interfaz lógica simple da lugar a que estos buses sean muy rápidos. Como ejemplo de bus síncrono podemos citar el bus que comunica el procesador con la memoria.

Por otro lado, los protocolos de temporización síncrona presentan dos inconvenientes:

1. Todos los dispositivos conectados deben funcionar a la misma frecuencia de reloj.
2. Puede aparecer un problema de **sesgo de reloj**, definido como la diferencia de tiempo absoluto desde que dos elementos ven un flanco de reloj. Esta diferencia surge porque la señal de reloj puede que llegue a un dispositivo antes que a otro, produciéndose una desincronización interna del bus, con lo que la señal de reloj debe ser encauzada cuidadosamente para minimizar la diferencia de los tiempos de llegada de esta señal a los dispositivos. El problema del sesgo de reloj se suele acentuar cuanto mayor es la longitud del bus, con lo que es *complicado tener buses síncronos largos y rápidos simultáneamente*.

A continuación, vamos a mostrar algunos ejemplos de lecturas y escrituras en un bus síncrono, tanto con líneas de datos y direcciones separadas como con líneas de información multiplexadas. Conviene recordar

que tanto la temporización como el protocolo de acceso de un bus son específicos de cada bus y vienen determinados por las especificaciones técnicas del mismo. Por ello, el protocolo de acceso que vamos a utilizar en los siguientes ejemplos es específico para nuestro bus síncrono y puede diferir del utilizado por otros buses.

Ejemplo de temporización síncrona: lectura de memoria. En la figura 11 se muestra el diagrama de tiempos de una operación síncrona de lectura de memoria:

1. En el flanco descendente del primer ciclo (T1), la CPU activa una señal de lectura (R/\overline{W}) y sitúa la dirección de memoria en las líneas de dirección. Además, activa una señal de inicio de operación (*Address Strobe*, \overline{AS}) para indicar a la memoria la presencia en el bus de la dirección y de la información de control.
2. Al comienzo del segundo ciclo (T2), en el flanco ascendente de la señal de reloj, la memoria coge la dirección y accede a ella para obtener el dato. Si suponemos que la latencia de acceso a memoria es de sólo un ciclo, al final del segundo ciclo, el dato a leer ya debe estar en el bus de datos.
3. Finalmente, en el flanco descendente del tercer ciclo (T3), el procesador recoge el dato del bus, desactiva las señales \overline{AS} y R/\overline{W} y libera el bus de direcciones. La memoria, por su parte, al ver la desactivación de las líneas de control, libera también el bus de datos.

Ejemplo de temporización síncrona: escritura en memoria. En la figura 12 se muestra el diagrama de tiempos de una operación síncrona de escritura en memoria:

1. En el flanco descendente de reloj del primer ciclo (T1), la CPU, simultáneamente, activa la señal de escritura R/\overline{W} , pone la dirección a escribir en el bus, activa la señal \overline{AS} y, finalmente, pone el dato a escribir en las líneas de datos.
2. Al comienzo del segundo ciclo (T2), en el flanco ascendente de la señal de reloj, la unidad de memoria coge tanto la dirección como el dato y procede a realizar la escritura en memoria. En este ejemplo, también consideramos una latencia de escritura de un ciclo, por lo que al final del segundo ciclo la operación de escritura debe haber finalizado.
3. Finalmente, en el flanco descendente del tercer ciclo (T3), la unidad de memoria le comunica a la CPU que ha terminado de escribir mediante la desactivación de las señales \overline{AS} y R/\overline{W} . La CPU, por su parte, al ver la desactivación de las señales de control, quita del bus tanto la dirección de memoria como el dato.

En la figura 13 se muestra también el cronograma para una operación síncrona de escritura en memoria, pero suponiendo ahora una latencia de acceso a memoria de dos ciclos. La secuencia de pasos es similar al ejemplo anterior, pero terminando un ciclo más tarde.

Ejemplo de temporización síncrona en un bus multiplexado: lectura de memoria. En la figura 14 se muestra el cronograma de una operación síncrona de lectura de memoria, aunque ahora vamos a suponer que utilizamos un bus que multiplexa las líneas de direcciones y datos:

1. En el flanco descendente del primer ciclo (T1), la CPU activa una señal de lectura (R/\overline{W}), sitúa la dirección de memoria en las líneas de información e indica el comienzo de la operación mediante la activación de la señal \overline{AS} .
2. Al comienzo del segundo ciclo (T2), en el flanco ascendente de la señal de reloj, la memoria coge la dirección y accede a dicha posición para obtener el dato. Si suponemos una latencia de un ciclo, al final de este segundo ciclo la memoria debe tener el dato listo para ser enviado al procesador. Mientras

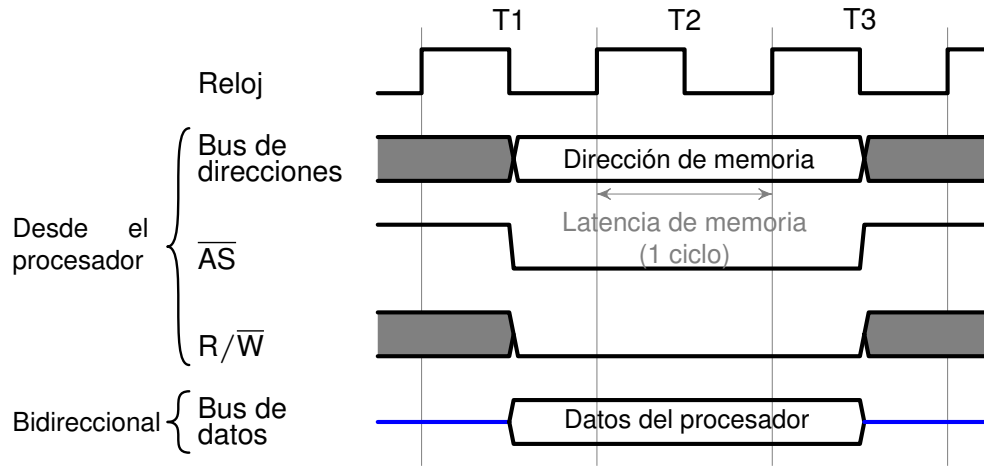


Figura 12: Ejemplo de temporización síncrona de una escritura en memoria.

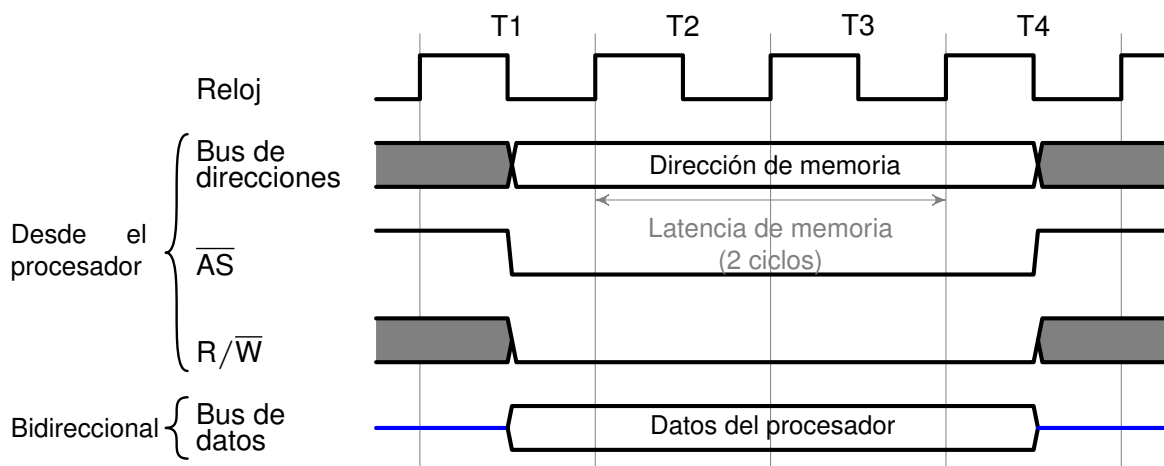


Figura 13: Ejemplo de temporización síncrona de una escritura en memoria, suponiendo una latencia de dos ciclos para el acceso a memoria.

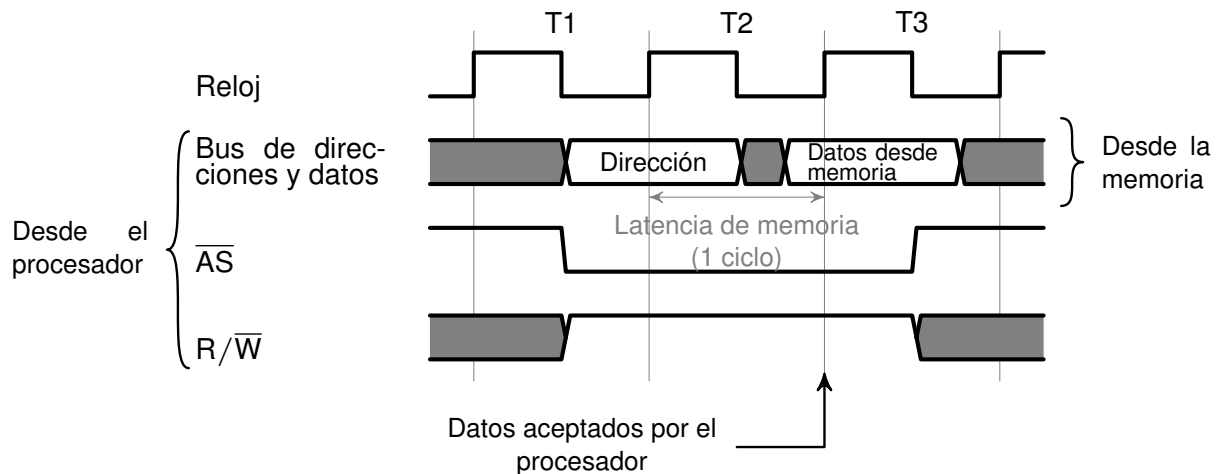


Figura 14: Ejemplo de temporización síncrona de una lectura de memoria en un bus multiplexado.

tanto, en el flanco descendente de la señal de reloj, la CPU libera las líneas de información para que la memoria pueda poner en ellas el dato leído.

- Finalmente, en el flanco descendente del tercer ciclo (T3), el procesador recoge el dato del bus, indicándolo mediante la desactivación de las señales \overline{AS} y R/\overline{W} . La memoria, por su parte, al ver la desactivación de las líneas de control, libera también las líneas de información.

Cabe destacar que, en el caso de las operaciones de lectura, no hay diferencia de ciclos entre un bus multiplexado y otro que no lo es.

Ejemplo de temporización síncrona en un bus multiplexado: escritura en memoria. En la figura 15 se muestra el cronograma de una operación síncrona de escritura en memoria en un bus multiplexado. Si comparamos dicho diagrama de tiempos con el de la figura 12 (que usaba un bus dedicado), se observa que la diferencia proviene de la pérdida de un ciclo en el bus multiplexado puesto que no es posible poner la dirección y el dato en el mismo ciclo, mientras que en el bus dedicado de la figura 12 sí que era posible. Los pasos que se dan son los siguientes:

- En el flanco descendente de reloj del primer ciclo (T1), la CPU, simultáneamente, activa la señal de escritura R/\overline{W} , pone la dirección a escribir en el bus y activa la señal \overline{AS} .
- Al comienzo del segundo ciclo (T2), en el flanco ascendente de la señal de reloj, la unidad de memoria coge la dirección donde se va a realizar la escritura. A continuación, en el flanco descendente, el procesador quita del bus la dirección y coloca en su lugar el dato a escribir.
- Al comienzo del tercer ciclo (T3), en el flanco ascendente de la señal de reloj, la unidad de memoria coge del bus el dato y procede a realizar la escritura. En este ejemplo, también consideramos una latencia de escritura de un ciclo, por lo que al final del tercer ciclo la operación de escritura debe haber finalizado.
- Finalmente, en el flanco descendente del cuarto ciclo (T4), la unidad de memoria le comunica a la CPU que ha terminado de escribir mediante la desactivación de las señales \overline{AS} y R/\overline{W} . La CPU, por su parte, al ver la desactivación de las señales de control, quita del bus el datos, dando por finalizada la operación.

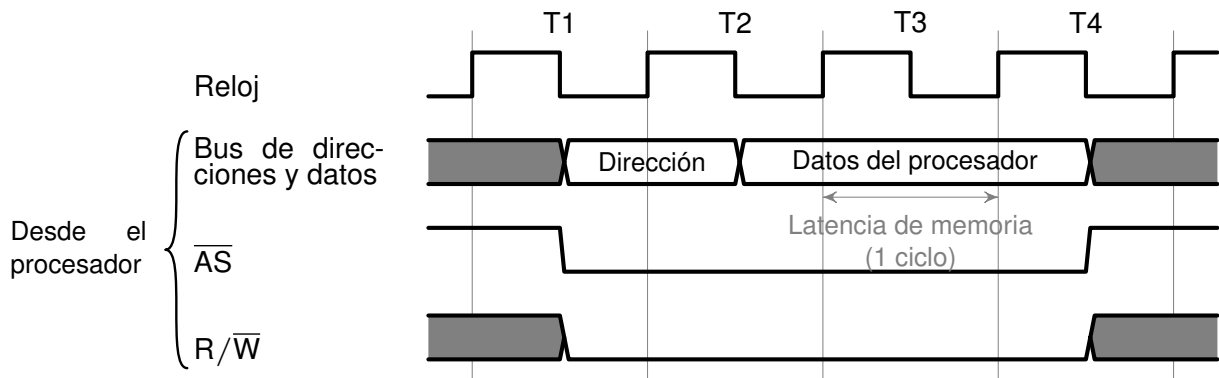


Figura 15: Ejemplo de temporización síncrona de una escritura en memoria en un bus multiplexado.

Ejemplos de cálculo del ancho de banda teórico y efectivo. Sea un bus con líneas separadas para datos de 32 bits y direcciones de 32 bits, con una temporización como la vista en los ejemplos anteriores:

- **Ciclo 1º:** se pone la dirección en el bus (y el dato si es una escritura).
- **Ciclos 2º y 3º:** ciclos de espera (suponemos una latencia de la memoria de dos ciclos).
- **Ciclo 4º:** finaliza la operación de lectura (o escritura). Además, la memoria soporta el modo ráfaga, es decir, se puede seguir leyendo (o escribiendo) hasta un máximo de 4 palabras de 32 bits sin tener que iniciar otra operación.

Veamos tres casos de cálculo de ancho de banda, tanto para lecturas como para escrituras:

1. ¿Cuál es el ancho de banda **teórico** del bus si funciona a una frecuencia de 200 MHz?

En un ciclo se puede transmitir 32 bits de datos (4 bytes). Si $f = 200 \text{ Mhz}$, entonces en un segundo tenemos $200 \times 10^6 \text{ ciclos} \Rightarrow AB \approx 4 \times 200 \text{ MB/s} \approx 800 \text{ MB/s}$.

2. ¿Cuál es el ancho de banda **efectivo** si no consideramos el modo ráfaga?

Si no consideramos el modo ráfaga, hay que iniciar una operación de lectura o escritura nueva por cada palabra a transmitir. Tal y como se vio en la figura 13, cada transferencia requiere un total de cuatro ciclos (estamos suponiendo una latencia de acceso a memoria de dos ciclos). Por lo tanto, transmitimos 4/4 bytes/ciclo. Multiplicando por la frecuencia podemos expresar esta cantidad en MB/s. Es decir, el ancho de banda efectivo es: $AB = \frac{4}{4} \times 200 \times 10^6 \text{ bytes/s} = 200 \text{ MB/s}$.

3. ¿Cuál es el ancho de banda **efectivo** si consideramos el modo ráfaga?

Si consideramos el modo ráfaga, la mejor situación es apurarla al máximo y leer o escribir las 4 palabras de forma consecutiva, tal y como se observa en las figuras 16 y 17. En definitiva, en el caso de las lecturas, una vez que la CPU recibe la primera palabra en el ciclo 4, en lugar de iniciar nuevas operaciones de transferencia, la memoria continua leyendo las otras tres palabras en ciclos sucesivos mientras la CPU las va recibiendo en los ciclos 5, 6 y 7, respectivamente. Por lo tanto, en una única operación que tarda 7 ciclos hemos transmitido 4 palabras (16 bytes). Multiplicando por la frecuencia podemos expresar esta cantidad en MB/s. Es decir, el ancho de banda efectivo del bus es: $AB = \frac{16}{7} \times 200 \times 10^6 \text{ bytes/s} \approx 457,14 \text{ MB/s}$ (no llega al ancho de banda teórico de 800 MB/s pero es más del doble del valor obtenido en el caso anterior).

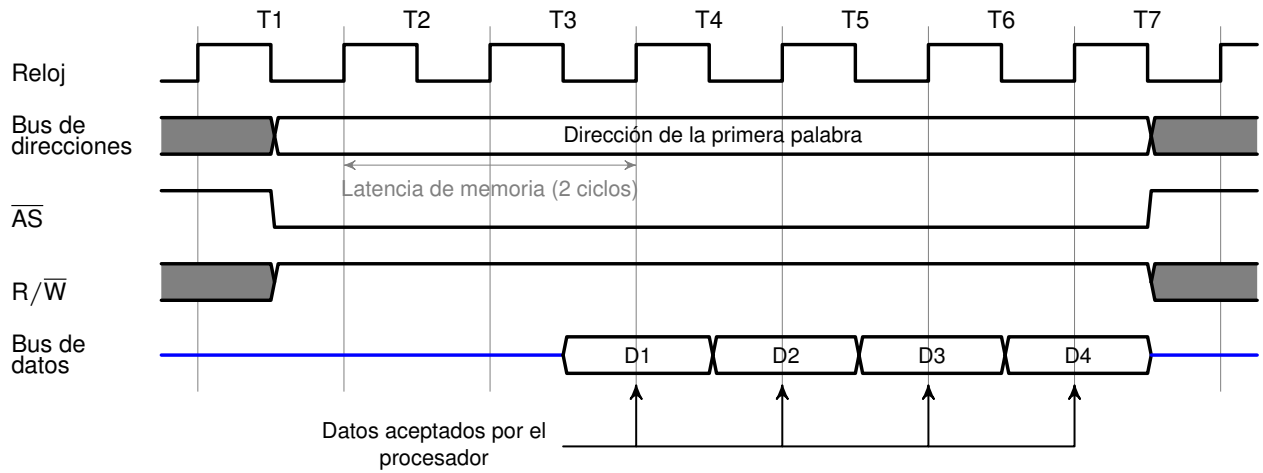


Figura 16: Ejemplo de lectura síncrona en modo ráfaga.

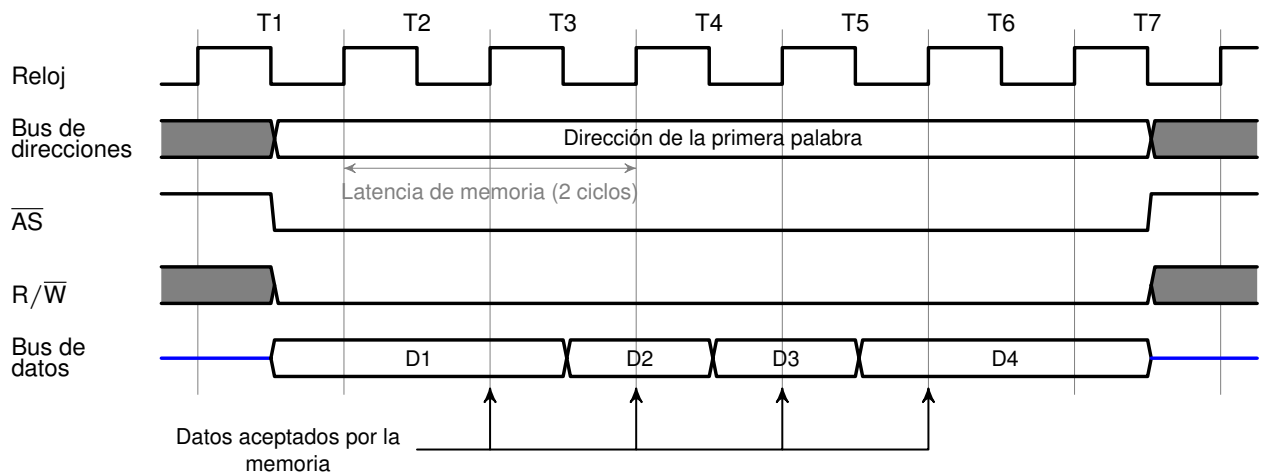


Figura 17: Ejemplo de escritura síncrona en modo ráfaga.

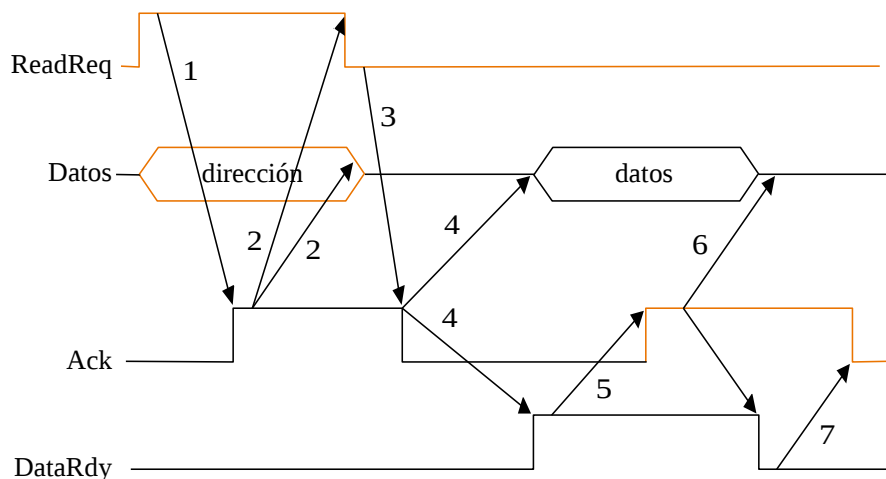


Figura 18: Cronograma de una transferencia asíncrona entre CPU y memoria.

Protocolos con temporización asíncrona

En un bus asíncrono no hay señal de reloj, con lo que el bus puede ser todo lo largo que queramos y podemos conectarle una amplia variedad de dispositivos con diferentes frecuencias de funcionamiento. Por contra, los principales inconvenientes son: la lentitud de los buses asíncronos respecto a los síncronos, la necesidad de introducir líneas de control adicionales y los posibles fallos de sincronización entre los dispositivos. Ejemplos comunes de buses asíncronos de E/S son el bus *firewire* y el USB.

Para coordinar la transmisión de datos en un bus asíncrono, se utiliza un **protocolo de presentación** (*handshaking*) en el que hay una serie de pasos de forma que el emisor y el receptor únicamente proceden al siguiente paso cuando ambos están de acuerdo. Este protocolo precisa las siguientes tres líneas de control adicionales:

1. *ReadReq*: se utiliza para indicar una petición de lectura a memoria. La dirección se pone en las líneas de información al mismo tiempo.
2. *DataRdy*: se utiliza para indicar que la palabra de datos está preparada en las líneas de información. El dato se coloca en las líneas de información al mismo tiempo.
3. *Ack*: se utiliza para reconocer las señales *ReadReq* y *DataRdy* de la otra parte.

Es importante destacar que las señales de control *ReadReq* y *DataRdy* están activas hasta que la otra parte (la memoria o el dispositivo) indica que las líneas de control han sido vistas y que las líneas de información han sido leídas; esta indicación se hace activando la línea *Ack*.

Como ejemplo, vamos a estudiar los pasos necesarios para realizar una comunicación asíncrona entre procesador y memoria para realizar la lectura de una palabra de memoria, tal y como se aprecia en la figura 18.

Los pasos involucrados en la operación de transferencia son los siguientes:

1. La CPU señala una petición de lectura de memoria elevando *ReadReq* y poniendo la dirección en las líneas de información.
2. La memoria ve la línea *ReadReq*, lee la dirección del bus y eleva *Ack* para indicar que la ha visto.
3. La CPU libera la línea *ReadReq* y, en cuanto la memoria se percata de ello, baja la línea *Ack*.
4. Este paso comienza cuando la memoria tiene el dato preparado. Lo coloca en las líneas de datos y eleva *DataRdy*.

5. La CPU ve *DataRdy*, con lo que lee el dato del bus, y señala que tiene el dato mediante la activación de *Ack*.
6. La memoria ve la señal *Ack*, con lo que desactiva *DataRdy* y libera las líneas de información.
7. Finalmente, la CPU al ver que *DataRdy* ha sido desactivada, hace lo propio con la línea *Ack*, que indica que se completa la transmisión.

7.5.5. Mecanismos de control de acceso

Como se comentó al comienzo del apartado 7.5.4, se necesita un mecanismo para controlar los accesos al bus porque, si no fuera así, los dispositivos tratarían de utilizar las líneas de control y datos del bus simultáneamente, lo que nos llevaría a una situación caótica.

Esta situación caótica se evita mediante el uso de algún mecanismo que regule el acceso al bus. Para ello, se introducen uno o más **amos o dueños del bus** (*bus master*) en el sistema. El dispositivo que hace las veces de amo es el que inicia y controla todas las peticiones de uso del bus.

El esquema más sencillo consiste en tener sólo un amo del bus: el procesador. Con este esquema, si un dispositivo A quiere comunicarse con otro dispositivo B, hará la petición al amo (procesador) para poder usar el bus. Cuando el procesador pueda, atenderá esta petición de comunicación y se dedicará a realizarla. Durante toda la comunicación entre los dispositivos A y B, el procesador se encuentra ocupado, no pudiendo, por tanto, realizar ninguna otra función. Así, por ejemplo, la simple lectura de un sector de disco puede requerir que el procesador esté involucrado cientos de miles de veces, impidiéndole realizar otra actividad.

Como se observa, si el procesador debe intervenir en cada una de las peticiones realizadas por cualquiera de los dispositivos de E/S, se perderá mucho tiempo de cálculo. Una solución consiste en tener **múltiples dueños del bus**. En este caso, cada dispositivo conectado al bus dispone de la lógica necesaria para controlar el acceso al mismo, es decir, para ser amo del bus. Esto implica que es necesario un **método de arbitraje** que decida quién va a ser el amo del bus en cada momento. El amo podrá entonces iniciar una transferencia de datos (lectura o escritura) con otro dispositivo que actúa como **esclavo** en este intercambio concreto (ya que responde a las peticiones de lectura y escritura, pero no las genera) y, finalmente, liberará el bus.

Cualquier método de arbitraje debe intentar cumplir estas características:

1. **Respetar las prioridades:** cada dispositivo tiene una prioridad y el de mayor prioridad debe ser atendido primero.
2. **Imparcialidad:** cualquier dispositivo debe tener la oportunidad de acceder al bus.
3. **Reducir el tiempo para arbitrar el bus:** se debe reducir el tiempo de arbitraje del bus y solaparse con las transferencias siempre que sea posible.

La elección del esquema de arbitraje depende de varios factores como son el número de dispositivos de E/S, la longitud del bus, la gestión de prioridades, el grado de imparcialidad necesaria y la velocidad requerida del bus.

Boletines de prácticas

Normas sobre la entrega de prácticas

Será necesario seguir las siguientes reglas para entregar las prácticas de todos los boletines, además de cualquier otra regla específica que se indique en un boletín particular:

- Las prácticas se entregarán únicamente mediante la opción de contenidos del alumno en SUMA.
- Se entregará un único archivo comprimido en formato *.tar.gz* o *.zip* que contendrá la memoria en formato PDF, los circuitos y programas que se hayan generado (código fuente) y cualquier otro fichero que se considere oportuno. El nombre del archivo será *prácticas-DNI-BOLETIN.FORMATO* (por ejemplo: *prácticas-12345678-B2.3.tar.gz*).
- La memoria incluirá, al menos, la siguiente información en un solo documento PDF:
 - Nombre y DNI del autor o autores de la práctica.
 - Descripción de los ficheros y directorios contenidos en el archivo entregado.
 - Contestación a las preguntas planteadas en los boletines. La respuesta a cada pregunta debe ser independiente, y debe estar claramente identificada.
 - Explicación de las pruebas realizadas para comprobar la corrección de la práctica entregada e instrucciones para su reproducción. Cuando sea posible, se incluirán los ficheros utilizados en dichas pruebas.
 - Explicación del trabajo realizado y cualquier aclaración que el alumno considere pertinente.
 - Lista de bibliografía y otras fuentes de información consultadas.

No se corregirá ninguna práctica que no se ciña estrictamente a los formatos especificados anteriormente.

B7.1. Programación de la E/S en MIPS

B7.1.1. Objetivos

Este boletín de prácticas persigue un doble objetivo. Por un lado, el boletín ilustra la forma más sencilla, pero también generalmente más ineficiente, de controlar la transferencia de datos entre la memoria y los dispositivos de entrada/salida en MIPS: el sondeo (o *polling*). Para ello, vamos a describir cómo funciona la E/S mapeada en memoria del dispositivo «terminal» (teclado+pantalla) simulado por MARS.

Por otro lado, el boletín describe el funcionamiento de la E/S dirigida por interrupciones en MIPS. Ya que las interrupciones y las excepciones se manejan de forma muy parecida en MIPS, primero veremos qué registros del procesador intervienen en el manejo de dichos eventos para, después, pasar a comentar un programa de ejemplo que hace uso de interrupciones para la lectura de datos desde la terminal simulada por MARS.

B7.1.2. Prerequisitos

- Lectura de los apuntes de teoría.

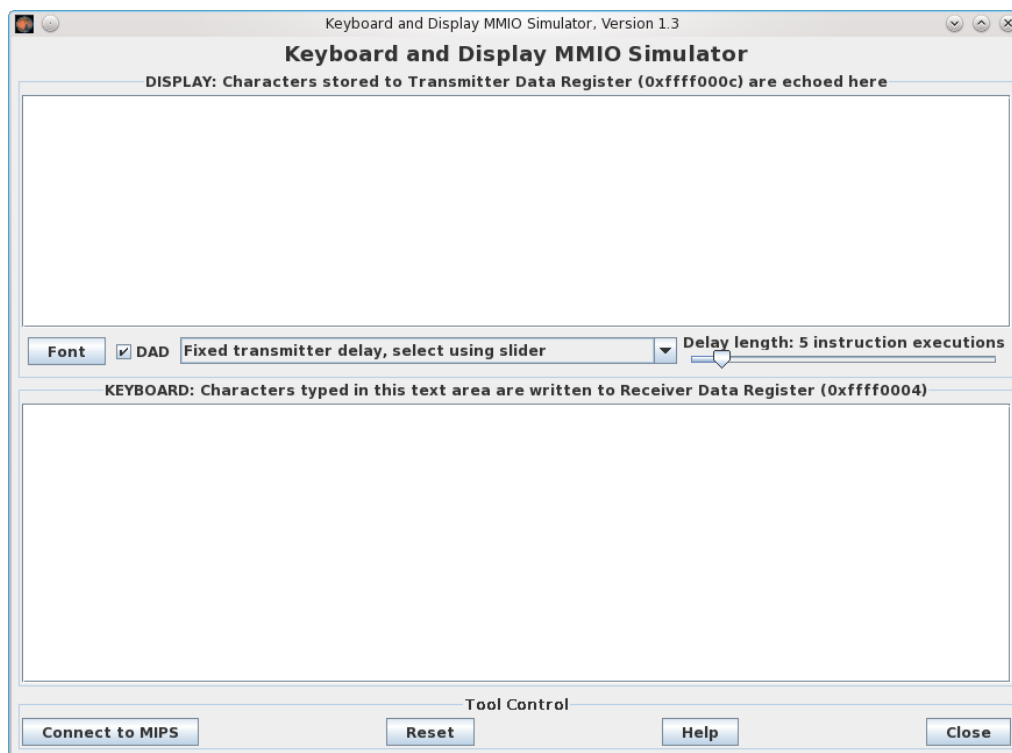


Figura B7.1: Simulador de terminal de texto en MARS

B7.1.3. Plan de trabajo

El plan de trabajo de esta sesión será el siguiente:

1. Lectura por parte del alumno de las secciones de la [B7.1.4](#) a la [B7.1.8](#).
2. Realización, de forma individual, de los ejercicios propuestos en el boletín (con supervisión del profesor).

B7.1.4. Dispositivos mapeados en memoria.

Entre las distintas herramientas que pone MARS a nuestra disposición, encontramos un simulador de una terminal de texto formada por un teclado y una pantalla. A este simulador se accede a través de las opciones de menú `Tools` → `Keyboard and Display Simulator` que, al seleccionarla, nos mostrarán una ventana como la que aparece en la figura [B7.1](#) (observa el botón `Connect to MIPS` que habrá que pulsar para que el simulador de la terminal se conecte al simulador general de MIPS). Los programas en ensamblador del MIPS ejecutándose sobre MARS pueden leer los caracteres que se teclean en este terminal. Además, si el programa MIPS escribe caracteres en el terminal, aparecen también en dicha ventana.

Recordemos que podemos realizar la E/S de nuestro programa utilizando todos los servicios del sistema operativo descritos en la práctica 3 (llamadas `syscall`). Estos servicios se comunican con el teclado y la pantalla a bajo nivel, evitando al usuario el tener que conocer los detalles concretos de dicha comunicación. En esta práctica, sin embargo, estamos justamente interesados en esos detalles. Para poder estudiar la comunicación entre la CPU y los dispositivos a bajo nivel, vamos a hacer uso, en primer lugar, de la opción de E/S mapeada a memoria que incluye MARS.

El dispositivo terminal consta de dos unidades independientes: un receptor y un transmisor. El receptor

lee caracteres del teclado, mientras que el transmisor escribe caracteres en pantalla¹. Existen cuatro registros de correspondencia directa con la memoria que afectan a la E/S con el terminal en MARS:

1. El registro **control del receptor** (*Receiver Control*) está en la posición de memoria `0xffff0000`. Sólo usa los dos bits de menor peso de sus 32 bits:
 - El bit 0 (*preparado*) es de sólo lectura y, si vale 1, significa que ha llegado un carácter desde el teclado. Este bit cambia de 0 a 1 cuando se teclea un carácter en el teclado y cambia de 1 a 0 cuando el carácter es leído desde el registro *datos del receptor*.
 - El bit 1 es de habilitación de interrupción del teclado, y está relacionado con el manejo de la E/S a través de interrupciones. Si, por programa, este bit se pone a 1, el teclado genera una interrupción cada vez que se pulsa una tecla, indicando así al programador que hay un carácter disponible en el registro *datos del receptor*.
2. El registro **dato del receptor** (*Receiver Data*) está en la posición `0xffff0004`. Los ocho bits de menor peso de este registro contienen el último carácter tecleado. El resto de bits están a 0. Este registro es sólo de lectura y cambia únicamente cuando se teclea un nuevo carácter.
3. El tercer registro es el **control del transmisor** (*Transmitter Control*), en la dirección `0xffff0008`. De nuevo, de él se usan sólo los dos bits de menor peso:
 - El bit 0 (*preparado*) es de sólo lectura y, si tiene el valor 1, significa que el transmisor de datos está preparado para aceptar un nuevo carácter. Si es 0, el transmisor de datos aún está ocupado escribiendo el carácter anterior (es decir, se simula el retardo real que suele haber al enviar información a los dispositivos de salida, ya que la CPU es normalmente mucho más rápida que ellos).
 - De modo análogo a lo que ocurría con el *control del receptor*, el bit 1 está también relacionado con el manejo de la E/S a través de interrupciones. Si se pone a 1 por programa, cada vez que la consola esté lista de nuevo para imprimir un carácter (después de haber impreso otro), emitirá una interrupción para avisar de dicho evento.
4. El último registro es el registro **dato del transmisor** (*Transmitter Data*), en la dirección `0xffff000c`. Cuando se escribe un valor en esta posición, sus ocho bits de menor peso se envían a la consola. En ese momento el bit de preparado del registro *control del transmisor* se pone automáticamente a 0, lo cual indica que está ocupado mostrando un carácter. Dicho bit permanecerá a 0 hasta que pase el tiempo suficiente para enviar el carácter a la pantalla, momento en el cual se pondrá de nuevo a 1.

B7.1.5. Programación de la E/S por sondeo

Un primer método (relativamente ingenuo) para realizar la entrada/salida de caracteres sería el mostrado en el programa de la figura B7.2. Como se puede observar, el programa se dedica simplemente a leer una cadena de diez caracteres del teclado (sin eco en el terminal conforme se van pulsando las teclas) para posteriormente imprimirla de golpe en la terminal, utilizando las direcciones de memoria comentadas en el apartado anterior para comunicarse con ambos dispositivos.

Sobre dicho programa merece la pena resaltar los dos siguientes aspectos:

- Con respecto a la lectura de teclado, obsérvese que se va cargando el registro *control del receptor* para comprobar continuamente su bit menos significativo, hasta que éste valga 1 (espera activa), momento en el que se sabe que ha llegado un carácter. Entonces, simplemente se lee del *dato del receptor*, se almacena en la cadena y se repite el proceso para el resto de caracteres a leer.

¹Esto significa que los caracteres tecleados no tienen “eco” automáticamente en la pantalla. En su lugar, el programa debe construir explícitamente el “eco” escribiendo en el transmisor cada carácter que lee del receptor, si así lo desea.

```

1 # Pequeño programa de prueba de E/S por sondeo en MARS
2
3 ##### Segmento de datos #####
4     .data
5 cadena: .asciiz "-----"
6 ##### Segmento de texto #####
7     .text
8 main:
9 # Lectura por teclado de una cadena de diez caracteres.
10     la     $t0,cadena          #
11     addi   $t1,$t0,10         # Apunta al fin de la cadena (10 caracteres).
12 ready0: lb     $s0,0xffff0000  # Carga registro control del receptor
13     andi   $s1,$s0,0x00000001  # ¿Hay algo?
14     beqz   $s1,ready0        # Si aún no, bucle de espera activa.
15     lb     $s2,0xffff0004     # Lee carácter del dato del receptor.
16     sb     $s2,0($t0)         # Guardar carácter leído en cadena.
17     addi   $t0,$t0,1         # Siguiente byte.
18     bne    $t0,$t1,ready0     # Repetimos hasta llenar la cadena.
19
20 # Impresión en pantalla de la cadena.
21     la     $t0,cadena
22 otrol: lb     $s2,0($t0)      # Leemos carácter de la cadena.
23     sb     $s2,0xffff000c     # Lo mandamos a la pantalla.
24 ready1: lb     $s0,0xffff0008  # Carga registro control del transmisor.
25     andi   $s1,$s0,0x00000001  # ¿Ha terminado de imprimirse?
26     beqz   $s1,ready1        # Mientras no, bucle de espera activa.
27     addi   $t0,$t0,1         # Siguiente byte.
28     bne    $t0,$t1,otrol     # Repetimos hasta imprimir la cadena.
29
30 # Finalización del programa.
31     li     $v0, 10
32     syscall                    # Termina el programa.

```

Figura B7.2: sondeo.s

- En lo que se refiere a la impresión en pantalla, el proceso es similar, aunque, de alguna forma, inverso: primero se envía el carácter al *dato del transmisor*, para después realizar la espera activa en el *control del transmisor* hasta comprobar que ya puede enviarse el siguiente carácter. Sólo así se asegura que ningún carácter se pierda sin haber sido impreso.

B7.1.6. Excepciones e interrupciones en MIPS

Las excepciones e interrupciones tienen en común que ambas son eventos inesperados que provocan un cambio en el flujo normal de ejecución de un programa. La diferencia radica únicamente en el origen de dicho evento: mientras que en las *excepciones* dicho cambio lo provoca un evento generado internamente por el procesador (por ejemplo, un desbordamiento aritmético en una operación de suma, el intento de ejecución de una instrucción de código de operación desconocido o el intento de carga de una palabra de dirección no alineada, entre otros), las *interrupciones* vienen desde el exterior de la CPU (en la mayoría de las ocasiones generadas por un dispositivo de E/S, que necesita de una atención inmediata por parte del procesador).

A pesar de las diferencias, en ambos casos la CPU debe comportarse de la misma manera: debe suspender la ejecución del programa en la instrucción que se esté ejecutando en ese momento, para pasar a atender de modo inmediato la situación especial que se acaba de producir. La rutina que atiende a las excepciones e interrupciones en MIPS se encuentra siempre situada a partir de la dirección `0x80000180` y se denomina *rutina de servicio de la excepción/interrupción (RSI)*.

B7.1.7. Registros del coprocesador 0

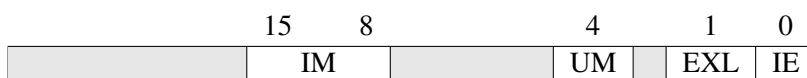
La parte del procesador de MIPS que contiene la información necesaria para realizar adecuadamente este tratamiento es el *coprocesador 0*. Aunque dicho coprocesador contiene una gran cantidad de registros en las implementaciones de MIPS reales, MARS sólo simula cuatro de ellos, que serán suficientes para realizar esta práctica:

Registro EPC (*Exception Program Counter*). Es el registro \$14 del coprocesador 0. Contiene el valor que tenía el contador de programa en el momento de producirse el evento inesperado. A este respecto, se nos pueden presentar dos casos:

- Si dicho evento fue una excepción, entonces el EPC tomará el valor de la dirección de la instrucción que la provocó (el *add* que originó el desbordamiento, el *lw* no alineado o la instrucción errónea que sea).
- Para el caso de las interrupciones, puesto que se producen de forma *asíncrona* con respecto a la ejecución del programa, el EPC apuntará a la instrucción a ejecutar tras la interrupción. Si se trata del teclado, por ejemplo, un usuario puede pulsar una tecla en cualquier momento, independientemente de la instrucción que se esté ejecutando (piénsese, por ejemplo, en un programa numérico que va calculando un determinado resultado con precisión cada vez mayor hasta que pulsemos una tecla, momento en el que queremos que se pare). Para el caso de una transferencia de disco, como ejemplo alternativo, nos interesará que nuestro programa (u otros que se ejecutan concurrentemente con el nuestro) se sigan ejecutando mientras aquella termina, momento en que el controlador del disco avisará a la CPU mediante otra interrupción. De nuevo, en este caso, no podremos saber con antelación el momento en el que se producirá el final de la transferencia ni, por tanto, la dirección de la instrucción interrumpida.

Tanto en el primer caso como en el segundo la importancia del EPC queda de manifiesto: es la única información de que disponemos para poder reanudar la ejecución de nuestro programa tras el tratamiento de la excepción/interrupción (en el caso de que así nos interese).

Registro Status. Es el registro \$12 del coprocesador 0. Se trata de un registro que sirve para poder indicar a la CPU cómo queremos que se traten los distintos tipos de interrupciones que se produzcan (esto es, si deseamos atenderlas mediante la rutina de servicio o simplemente ignorarlas). De nuevo, de sus 32 bits sólo nos interesarán los siguientes:



- Bits 15–8 (*Interrupt Mask*, IM): se trata de una máscara de 8 bits que sirve para habilitar selectivamente los distintos tipos de interrupciones disponibles. Pueden emplearse para establecer prioridades entre las distintas interrupciones. En esta práctica utilizaremos solamente el bit 8 (el de menos peso), correspondiente a la interrupción provocada por la pulsación de una tecla.
- Bit 4 (*User Mode*, UM): si se pone a 1, la CPU se sitúa en modo usuario (para ejecutar programas normales). Si se pone a 0, la CPU se sitúa en modo núcleo (para ejecutar código del S.O.).
- Bit 1 (*Exception Level*, EXL): se activa automáticamente al producirse una excepción/interrupción/trap. Como efectos colaterales, se inhiben globalmente las interrupciones y se pasa a modo núcleo (valgan lo que valgan IE y UM; EXL tiene prioridad sobre ellos).
- Bit 0 (*Interrupt Enable*, IE): si se pone a 1, quedarán habilitadas todas las interrupciones indicadas en el campo IM. Si se pone a 0, se ignorarán todas. Sirve para evitar/posibilitar que la propia RSI se vea interrumpida por una segunda interrupción.

Registro Cause. Es el registro \$13 del coprocesador 0. Se trata de un registro que sirve para indicar la causa de la excepción o interrupción. De los 32 bits sólo nos interesarán los siguientes:



- Bits 6–2 (campo *Exception Code*, ExCod): contienen un código de excepción de 5 bits, que indica el tipo de excepción o interrupción que se produjo según la siguiente tabla (se muestran sólo algunos de los valores más importantes):

Nº	Nombre	Descripción
0	INT	Se trata de una interrupción externa (no una excepción).
2	TLB_L	Fallo de TLB en Lectura.
3	TLB_E	Fallo de TLB en Escritura.
4	ADDRL	Excepción por intento de carga de dato o instrucción desde una dirección no alineada.
5	ADDRS	Excepción por intento de almacenamiento en una dirección no alineada.
6	IBUS	Excepción por valor de PC erróneo (fuera del ámbito del programa; p.e., PC=0x00000000).
7	DBUS	Excepción similar a la anterior, pero para datos (p.e., por intento de cargar o almacenar en la dirección 0x00000000).
10	INV	Código de instrucción inválido.
12	OVF	Excepción por desbordamiento aritmético (p.e., al sumar 0x7ffffff y 0x7ffffff).

- Bits 15–8 (campo *Interrupt Pending*s, IP): se trata de 8 *flags* que indican distintos tipos de posibles interrupciones pendientes (sin atender). Este campo ocupa las mismas posiciones que el campo IM del registro *Status* para poder hacer rápidamente una máscara y determinar cuál atender. En esta práctica, usaremos sólo el bit 8, cuya activación indicará que hay una interrupción pendiente para el teclado.

Registro BadVAddr. Es el registro \$8 del coprocesador 0. Contiene la dirección errónea que provocó la excepción (sólo para el caso de las excepciones 4, 5, 6 y 7 indicadas en la tabla anterior, donde dicho valor tiene algún sentido)².

B7.1.8. Programación de la E/S por interrupciones.

El programa de la figura B7.2 era claramente ineficiente, puesto que mantenía a la CPU completamente ocupada (y por tanto sin poder hacer otra cosa útil) en cada espera activa a que el correspondiente dispositivo de E/S estuviera listo para enviar/recibir un carácter. Una alternativa mucho más interesante consiste en utilizar el mecanismo de interrupciones para comunicarse con los periféricos. En esta otra aproximación, la CPU puede simplemente estar haciendo otro trabajo mientras espera que algún dispositivo de E/S la interrumpa para demandar su atención.

El programa de la figuras B7.3 y B7.4 muestra un código de ejemplo que utiliza interrupciones para capturar una pulsación de teclado. Como podemos ver, el programa principal es un simple bucle (líneas 17–34) que va imprimiendo “TIC”, “TAC”, alternativamente, con un pequeño retardo entre cada uno³. Antes de

²Ojo, no confundir BadVAddr con el EPC: éste último contiene la dirección de la instrucción que provocó la excepción (p.e., donde está el lw erróneo), mientras que aquél contiene la dirección en sí (p.e., el valor no alineado 0x10010003 desde donde se intentó cargar una palabra).

³Este retardo se introduce simplemente para facilitar su visualización al usuario, ya que, de otro modo, iría demasiado rápido como para poder seguirlo de forma cómoda.

```

1 # Pequeño programa para probar la E/S por interrupciones en MARS
2     .data
3 Tic:  .ascii "\nTIC"
4 Tac:  .ascii "\nTAC"
5 pause: .word 10000
6
7     .text
8 main:
9 # Habilitamos las interrupciones de teclado.
10     mfc0    $t0, $12          # Leemos el registro Status.
11     ori     $t0, $t0, 0x00000101
12     mtc0    $t0, $12          # Habilitamos interrupciones (registro Status).
13     la     $t0, 0xFFFF0000    # Preparamos el registro "control del receptor"
14     li     $t1, 2              # para que nos avise (interrumpa) cuando
15     sw     $t1, 0($t0)        # se pulse una tecla (poniendo su bit1=1).
16
17 # Cuerpo principal de la función main.
18 loop:
19     lw     $t0, pause
20 loop1:
21     sub    $t0, $t0, 1
22     bgez  $t0, loop1          # Bucle de espera antes de "TIC".
23     la    $a0, Tic
24     li    $v0, 4
25     syscall                # Imprime "TIC".
26
27     lw     $t0, pause
28 loop2:
29     sub    $t0, $t0, 1
30     bgez  $t0, loop2          # Bucle de espera antes de "TAC".
31     la    $a0, Tac
32     li    $v0, 4
33     syscall                # Imprime "TAC".
34
35     j     loop                # Vuelta al inicio de bucle principal.
36

```

Figura B7.3: interrupciones.s, función main.

entrar en dicho bucle infinito, el programa habilita las interrupciones de teclado (líneas 9–15) de manera que, cuando se pulse una tecla, pase a ejecutarse la rutina de servicio correspondiente.

La rutina de servicio, por su parte, lo que se hace es recoger el carácter leído de teclado y guardarlo en un buffer (líneas 55–65). Además, si este carácter es un retorno de carro, entonces muestra el contenido almacenado en el buffer hasta el momento y procede a limpiarlo (líneas 67–72). Después se vuelve al programa principal que continúa con su bucle. Obsérvese que la rutina de servicio se ubica a partir de la dirección 0x80000180 en el espacio del núcleo, haciendo uso de la directiva `.ktext 0x80000180`.

Como se puede comprobar, la rutina de servicio no es tan distinta de una rutina normal, en el sentido de que posee un código de entrada que guarda la información de los posibles registros modificados por el manejador, para recuperarlos después de realizar el trabajo y antes de volver al programa principal. Más en detalle, la estructura de la rutina es la siguiente:

1. En primer lugar, se salvan los registros `$at`, `$v0` y `$a0`, ya que van a ser utilizados posteriormente por el manejador. Obsérvese, en particular, que es necesario salvar el registro temporal del ensamblador (`$at=$1`), ya que la interrupción/excepción pudo ocurrir perfectamente en mitad de una pseudoinstrucción que estuviese utilizando dicho registro de modo auxiliar.

```

37 ##### Manejador de interrupción (código/datos del núcleo) #####
38
39 # Segmento de datos del núcleo.
40     .kdata
41 __s01_: .word 0           # espacio para salvar registros
42 __s02_: .word 0           # espacio para salvar registros
43 __buffer: .byte 0
44         .space 255       # Espacio para guardar caracteres tecleados
45 __ptr:   .word __buffer  # Puntero al inicio del espacio de almacenamiento
46
47 # Segmento de código del núcleo.
48
49     .ktext 0x80000180
50
51 # En primer lugar, salvamos los registros que vamos a sobrescribir dentro del núcleo.
52     move    $k1, $at      # Salvamos $at.
53     sw     $v0, __s01_    # Salvamos $v0.
54     sw     $a0, __s02_    # Salvamos $a0.
55
56 # Cuerpo de la rutina de servicio de interrupción.
57     li     $a0, 0xFFFF0000
58     lw     $v0, 4($a0)    # Leemos caracter tecleado.
59     li     $a0, '\n'     # Retorno de carro (RC).
60     beq    $v0, $a0, __flush # Si es RC: salta
61     lw     $a0, __ptr     # Leemos el puntero al buffer.
62     sb     $v0, 0($a0)    # Guardamos carácter en buffer.
63     addi   $a0, $a0, 1    # Incrementamos el puntero.
64     sb     $0, 0($a0)     # Guardamos un 0 en el buffer.
65     sw     $a0, __ptr     # Guardamos el valor del puntero.
66     j     __fin          # Vamos al fin de rutina de servicio.
67
68 __flush:
69     la     $a0, __buffer  # Cargamos la dirección del buffer.
70     li     $v0, 4        # Llamada para imprimir cadena.
71     syscall
72     sw     $a0, __ptr     # Inicializamos puntero a buffer.
73     sb     $0, 0($a0)    # Inicializamos buffer (buffer[0]='\0').
74
75 # Recuperamos lo salvado y volvemos a (EPC).
76 __fin:
77     mtc0   $0, $13       # Limpiamos el registro Cause.
78
79     mfc0   $k0, $12      # Leemos el registro Status.
80     ori    $k0, 0x1      # Habilitamos las interrupciones.
81     mtc0   $k0, $12      # Escribimos el registro Status.
82     lw     $v0, __s01_    # Recuperamos valor registros $v0, $a0 y $at.
83     lw     $a0, __s02_
84     move   $at, $k1
85     eret
86                                     # Pone EXL=0 (en registro Status) y
87                                     # vuelve al programa de usuario.

```

Figura B7.4: interrupciones.s, rutina de servicio.

Como vemos, el salvado de registros no puede hacerse en la pila, ya que ésta podría no ser segura (la rutina de servicio podría estar ejecutándose como consecuencia de una excepción provocada por un mal valor de \$sp). Por ello, se utiliza una zona de datos del núcleo⁴. Aunque el registro \$at lo podríamos

⁴Esto provoca que el código no sea *reentrante*, es decir, que no se puede atender una interrupción mientras se está atendiendo otra

haber guardado también en memoria, en este ejemplo se guarda en el registro `$k1`. Recordemos que el manejador puede utilizar los registros `$k0` y `$k1` sin problema, puesto que su uso está reservado al núcleo y se supone que no deben ser empleados por los programas de usuario.

2. Tras guardar los registros que se van a modificar, se ejecuta el cuerpo principal de la rutina de servicio que, como hemos dicho, guarda el carácter que lee de teclado en un buffer y, si dicho carácter es un retorno de carro, imprime por pantalla todo el contenido del buffer y lo inicializa para recibir la siguiente línea de entrada.
3. Finalmente, para volver del manejador al programa principal, simplemente tenemos que limpiar el registro *Cause* (`mtc0 $0, $13`), habilitar las interrupciones poniendo de nuevo a 1 el bit *IE* del registro *Status*, recuperar los valores de `$at`, `$v0` y `$a0` previamente guardados y ejecutar la instrucción `eret` para regresar a la instrucción apuntada por EPC (`eret`, además, volverá a poner a 0 el campo EXL del registro *Status*).

Naturalmente, el comportamiento del manejador de interrupciones anterior es quizá demasiado sencillo. Sin embargo, es suficiente para ilustrar la mejora significativa que se produce en la eficiencia de la E/S con respecto a la aproximación seguida en el código de la figura **B7.2**: mientras que antes la CPU estaba el 100 % del tiempo interrogando al registro de control correspondiente, a la espera de que el usuario pulsara una tecla, ahora puede continuar haciendo trabajo útil.

B7.1.9. Ejercicios

Para cada ejercicio, el portafolios deberá incluir una explicación de cómo se ha resuelto y, en caso de que se realice o modifique algún programa, el código realizado y al menos un ejemplo de su ejecución.

1. Modifica la rutina de servicio que aparece en la figura **B7.4** para que sólo se ejecute el cuerpo de la misma si lo que se ha producido realmente es una interrupción y no una excepción. En el caso de que se produzca una excepción, el programa deberá mostrar un mensaje indicando el hecho y finalizará la ejecución del programa.

Recuerda añadir un ejemplo de ejecución del nuevo programa. Para probar que funciona correctamente, puedes añadir en la función `main` del programa (figura **B7.3**) alguna instrucción (o varias, si es necesario) que produzca una excepción, como leer una palabra de una dirección de memoria que no sea múltiplo de 4, saltar a la dirección 0 de memoria, producir un desbordamiento en una operación aritmética, etc.

2. Modifica el programa de la figura **B7.2** para que el sondeo continuo se convierta en un sondeo “periódico”. Para ello, antes de las dos instrucciones `beqz` que hay (y que implementan la espera activa) llama a un procedimiento de nombre `tictac` que muestre, mediante la llamada al sistema correspondiente, las cadenas “TIC” y “TAC” alternativamente en cada invocación (es decir, si en la invocación anterior mostró “TIC” ahora debe mostrar “TAC” y viceversa). Al igual que en el programa de la figura **B7.3**, añade un bucle de espera antes de mostrar cada cadena para evitar que éstas aparezcan excesivamente deprisa por pantalla.

Al ejecutar el programa modificado, ¿hay alguna diferencia “visible” con respecto al programa original?

excepción o interrupción anterior. De todas formas, esto no es problema puesto que durante la ejecución de la rutina de servicio el resto de interrupciones quedan temporalmente inhabilitadas, mediante la puesta, automáticamente, a 1 del campo EXL del registro *Status*.

Ejercicios

E7.1. Gestión de la E/S

1. Supóngase que tenemos un sistema de memoria que utiliza un reloj de 50 MHz. La memoria transmite peticiones de 8 palabras (de 32 bits cada una) a la velocidad de 1 palabra por ciclo. Para las lecturas de memoria, los accesos se presentan como sigue: 1 ciclo para aceptar la dirección, 2 ciclos de latencia y 8 ciclos para transmitir 8 palabras. Para las escrituras en memoria, los accesos se realizan como se indica: 1 ciclo para aceptar la dirección, 2 ciclos de latencia, 8 ciclos para transmitir las 8 palabras y 3 ciclos para escribir el código de corrección de errores.

Calcule la máxima anchura de banda efectiva en MB/seg para un patrón de acceso que conste de:

- a) Sólo lecturas de memoria.
 - b) Sólo escrituras en memoria.
 - c) Mezcla de 65 % de lecturas de memoria y 35 % de escrituras en memoria.
2. El sistema de memoria y de buses del ejercicio anterior se diseñó originalmente para soportar un procesador con bloques de caché de 8 palabras. Se diseñó un nuevo procesador con bloques de caché de 16 palabras. Hay dos organizaciones alternativas para la memoria y el bus.
 - a) Utilizar una organización de 8 palabras como la descrita en el ejercicio 1, y realizar dos accesos separados de 8 palabras para cada fallo.
 - b) Convertir el sistema de memoria para que suministre 16 palabras iniciando dos accesos separados de 8 palabras. Para lecturas, el sistema transmite las 8 primeras palabras mientras busca las 8 segundas. Para escrituras, la transmisión de las 8 segundas palabras puede comenzar inmediatamente después de recibir las 8 primeras palabras.

Utilizando los datos del ejercicio precedente, encontrar la máxima anchura de banda sostenible para cada uno de estos mecanismos suponiendo que las lecturas y escrituras se presentan con igual frecuencia.

3. Considerar un procesador de 50 MHz que utiliza el sistema de memoria del ejercicio 1 para tratar fallos de la caché. Suponer que el procesador tiene una caché de postescritura y que se han realizado las siguientes medidas: la tasa de fallos de la caché es de 0,05 fallos por instrucción; el 40 % de los fallos requiere una operación de postescritura, mientras que el otro 60 % requiere sólo una lectura.

Suponiendo que el procesador se detiene mientras dura un fallo (incluyendo el tiempo de postescritura si ésta es necesaria), encontrar el número de ciclos por instrucción empleados para tratar los fallos de caché.

4. Considere dos sistemas de buses diferentes:
 - a) El bus A es un bus de direcciones y datos multiplexados de 64 bits. El transmitir una dirección o un elemento de datos de 64 bits emplea un ciclo de bus. Las lecturas o escrituras en memoria incurrir en una latencia de tres ciclos. Comenzando con el cuarto ciclo, el sistema de memoria puede leer o escribir hasta 8 palabras a la velocidad de 2 palabras cada ciclo de bus.
 - b) El bus B es un bus con separación de datos de 32 bits y direcciones de 32 bits. Cada transmisión necesita un ciclo de bus. Una lectura en memoria incurre en una latencia de tres ciclos; entonces, comenzando con el cuarto ciclo, el sistema de memoria puede suministrar hasta 8 palabras a la

velocidad de 1 palabra cada ciclo de bus. Para una escritura, la primera palabra se transmite con la dirección; después de una latencia de tres ciclos se pueden transmitir hasta 7 palabras adicionales a la velocidad de 1 palabra cada ciclo de bus.

Determine también la máxima anchura de banda efectiva, en palabras por ciclo de bus, que cada sistema de bus y de memoria puede proporcionar. Evalúe también estos buses suponiendo peticiones de 1 palabra, donde el 60 % de las peticiones son lecturas y el 40 % escrituras.

5. Tenemos una cámara de vídeo conectada a nuestro ordenador a través de un bus de entrada/salida. Si la resolución de la imagen es de 320×200 pixels con 65536 colores por pixel, se pide:
- Calcular el mínimo ancho de banda necesario para poder visualizar 25 imágenes por segundo.
 - Si cada vez que se recibe un pixel se produce una interrupción, cuyo tratamiento consume 100 ciclos, calcular el porcentaje de tiempo que el procesador dedica a tratar las interrupciones.
 - Si para transferir cada imagen se programa el controlador DMA, calcular el porcentaje de tiempo que el procesador permanece ocupado sabiendo que se consumen 50 ciclos en inicializarlo y 100 ciclos en el tratamiento de la interrupción que marca el final de la transferencia.

Nota: la frecuencia del procesador es de 400MHz.

6. Tenemos un disco duro que transfiere datos en bloques de 16 palabras a una velocidad de 4 Mbytes por segundo. La frecuencia del procesador es de 600 MHz. Determine el porcentaje del tiempo de CPU que se consume en cada uno de los casos siguientes:
- Utilizando E/S por interrupciones, sabiendo que la sobrecarga para cada transferencia de un bloque, incluida la interrupción, es de 1000 ciclos de reloj.
 - Utilizando DMA, sabiendo que iniciar cada transferencia de 4 Kbytes nos cuesta 2000 ciclos de reloj y el tratamiento de la interrupción al terminar nos cuesta otros 1000 ciclos. Ignorar cualquier impacto de la conexión del bus entre el procesador y el controlador de DMA.
 - Compara las dos opciones anteriores y justifica cual es la mejor.

Nota: las palabras son de 32 bits.

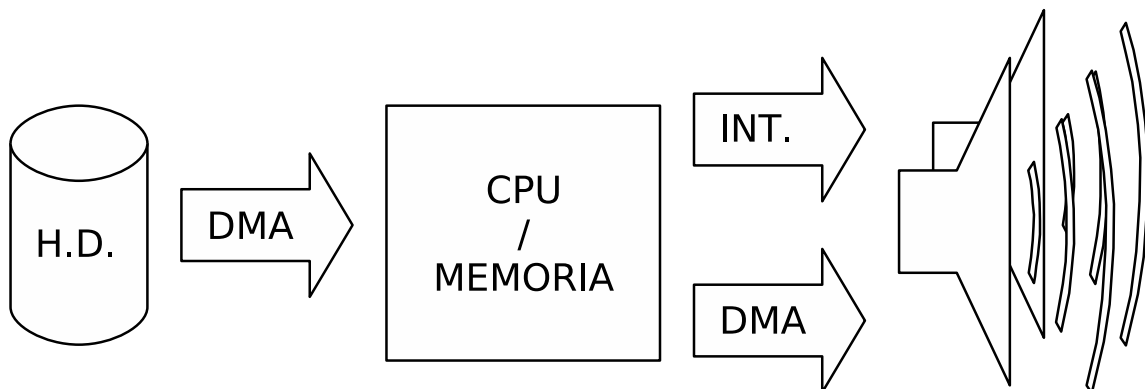
7. Una tarjeta de vídeo se encuentra conectada a un ordenador a través de un bus de entrada/salida dedicado que posee 64 líneas para direcciones y 64 líneas para datos. Transmitir una dirección o un dato de 128 bits consume un ciclo de bus. Leer o escribir en memoria supone una latencia de 4 ciclos. A partir del sexto ciclo el sistema de memoria puede leer o escribir hasta 8 palabras a razón de dos palabras por ciclo de bus. La frecuencia de funcionamiento del sistema es de 200 MHz.
- Determinar el máximo ancho de banda en bytes por segundo, si el 75 % de las peticiones son lecturas y 25 % son escrituras, para bloques de 8 palabras.
 - Si cada vez que se reciben 32 bytes se produce una interrupción, cuyo tratamiento consume 2 ciclos, calcular el porcentaje de tiempo que el procesador dedica a tratar las interrupciones.
 - Si se programa el controlador DMA, para transferencias de 12 Kbytes, calcular el porcentaje de tiempo que el procesador permanece ocupado sabiendo que se consumen 150 ciclos en inicializarlo y 300 ciclos en el tratamiento de la interrupción que marca el final de la transferencia.

Nota: las palabras son de 32 bits.

8. Un sistema tiene una tarjeta de sonido capaz de generar sonido con calidad de CD, es decir, 44.100 muestras por segundo, donde en cada muestra se envían 32 bits (16 bits por cada uno de los dos canales de sonido que forman una comunicación en estéreo). Disponemos de un Pentium I a 133 Mhz. Se pretende usar este sistema para reproducir ficheros de sonido en formato WAV. Para ello, se leen muestras de sonido del disco duro y se envían a la tarjeta de sonido:

- ETAPA 1: Se leen muestras de sonido en bloques de 4KB del disco duro. Para ello, se usa un controlador DMA que requiere 2000 ciclos de inicialización y 1000 ciclos al terminar la transferencia.
- ETAPA 2: Procesamiento de los datos en la CPU. Para el formato WAV el procesamiento requiere 0 ciclos de CPU.
- ETAPA 3: Para reproducir el sonido, el procedimiento para enviar las muestras a la tarjeta de sonido puede ser por unos de estos dos métodos diferentes:
 - Por interrupciones (44.100 interrupciones por segundo), donde en cada interrupción, que requiere 100 ciclos de CPU, se envía a la tarjeta una muestra de 32 bits.
 - Por DMA, donde cada transferencia de 16KB requiere un procesamiento de 2000 ciclos de inicialización más 1000 ciclos al terminar.

- a) Calcular el % de CPU consumido en esta tarea (las 3 etapas), usando interrupciones en la ETAPA 3.
- b) Calcular el % de CPU consumido en esta tarea (las 3 etapas), usando DMA en la ETAPA 3.
- c) Calcular el % de CPU consumido en esta tarea (las 3 etapas), si se reproducen ficheros MP3 (formato que comprime WAV a un 10 % del tamaño original), sabiendo que el procesamiento para descomprimir los datos en la CPU de un bloque de 4KB de MP3 requiere 20.000.000 ciclos en la etapa 2.



9. Una empresa necesita escanear documentos de tamaño A4 a una resolución de 300ppp (puntos por pulgada lineal) en escala de grises de 16 bits (niveles de gris de 0 a 65535). Esto es, cada punto del papel produce 16 bits. El documento A4 tiene un tamaño de 8,25 pulgadas de ancho y 11,75 de largo. Se barajan dos posibilidades para comprar el escáner, que estará conectado a un Pentium II a 300MHz:

- i. Por el puerto paralelo, que tiene una velocidad de 360.360 bytes por segundo.
- ii. Por el puerto USB 1.1, con una velocidad de 1,5 Mbytes por segundo.

Teniendo en cuenta estos datos, se quiere saber:

- a) ¿Cuánto tiempo tardará en escanear una página un escáner conectado por cada puerto i) y ii)?
- b) En el caso i) el puerto paralelo se programa por interrupciones en las que se procesa un byte. Cada interrupción requiere de 100 ciclos para ser procesada. ¿Qué porcentaje de CPU se dedica a la E/S en este caso?
- c) En el caso ii) se utiliza DMA. El controlador USB tiene un buffer de 128 bytes, que se utiliza para hacer transferencias DMA. El tiempo de programar el DMA para cada transferencia es de 500 ciclos, y el de tratar la interrupción de finalización es de 200 ciclos. ¿Qué porcentaje de CPU se dedica a la E/S en este caso?
- d) Finalmente se baraja otra estrategia: sondeo. Cada cierto tiempo, la CPU pregunta al escáner si tiene un byte disponible. Cada una de estas preguntas le llevan al procesador Pentium II 50 ciclos. Independientemente de la conexión, se quiere que mientras se escanea un documento, el procesador no pase más de un 10 % sirviendo el proceso de sondeo de la E/S. ¿Cuánto tardará entonces en escanearse una página A4?
10. Suponga un sistema de memoria formado por una caché con un tamaño de 64 KB y bloques de 8 palabras (1 palabra = 32 bits). El bus que conecta la caché y la memoria principal (MP) utiliza un reloj de 500 MHz, está multiplexado y tiene un ancho de 32 bits. La transmisión de una dirección requiere un ciclo de bus, mientras que el acceso a la MP tiene una latencia de 4 ciclos. El bus dispone de un modo ráfaga capaz de transmitir hasta un máximo de 4 palabras a la velocidad de una palabra cada dos ciclos.
- a) Indique los ciclos que requiere una operación de lectura y una operación de escritura de un bloque (8 palabras) en MP teniendo en cuenta que se realizan dos accesos consecutivos al bus para transferir un bloque completo. Además, en el caso de las escrituras también se envía por el bus un código de CRC (1 ciclo adicional), de forma que las escrituras se realizan como sigue: envía dirección, envía 4 palabras en modo ráfaga, espera la latencia y envía el CRC.
- b) Si la caché usa una política de post-escritura y hay un 40 % de bloques modificados, ¿cuántos ciclos de bus requiere una instrucción de carga que falla en caché? ¿Y una instrucción de almacenamiento que falla en la caché?
- c) Sabemos que el tráfico que circula por el bus proviene de un 70 % de instrucciones de carga y un 30 % de almacenamientos, ¿cuál será el máximo ancho de banda efectivo?
- d) Si usáramos una política de escritura directa, ¿cuál será el máximo ancho de banda efectivo para la misma distribución de cargas/almacenamientos que el apartado c)?
- e) Supongamos que la MP está paginada en páginas de 8 KB y que definimos una operación `flush p` que saca de la MP al proceso `p` pero que antes transfiere aquellas páginas físicas con el bit de modificado activo al disco duro. Si el proceso `p` ocupa un total de 20 páginas físicas estando el 75 % de ellas modificadas, ¿cuántos ciclos tardaremos en realizar un `flush p` si usamos un mecanismo por interrupciones que transfiere los datos en bloques de 16 palabras con una sobrecarga de 1000 ciclos por interrupción? ¿y por DMA si cada transferencia es de 1 página completa y requiere 500 ciclos para inicializar el DMA y 1500 ciclos para la interrupción de terminación?
11. Disponemos de una cámara digital de 10 megapíxeles donde cada píxel está representado por 24 bits (8 bits por canal de color). Las fotos de la cámara se pueden descargar conectándola a un PC a través del bus USB 2.0 y transferirlos a la memoria del procesador con un ancho de banda efectivo de 15 MBytes/s de datos a través del bus, el cual tiene un ancho de banda total de 480 Mbits/s. Si el procesador utiliza memoria virtual paginada con un tamaño de página de 4 KB y opera a una frecuencia de 3 GHz se pide:
- a) El % de ocupación de la CPU para la E/S al descargar imágenes de la cámara de fotos si la transferencia se realiza mediante interrupciones y se necesitan 400 ciclos de reloj para transferir 4 bytes de datos.

- b) El % de ocupación de la CPU si la transferencia se realiza mediante DMA y en cada operación DMA se transfiere una página (4 KB) completa a memoria principal. La preparación de la operación de DMA necesita 4000 ciclos de reloj y el tratamiento de la interrupción de terminación es de 8000 ciclos.
- c) Supongamos que la descarga de imágenes no se pudiese hacer mediante interrupciones ni DMA, y que sólo se puede acceder a la cámara mediante polling. Si suponemos que cada palabra transferida (4 bytes) necesita una operación de encuesta, la cual consume 1000 ciclos de CPU, ¿cuál debería ser la frecuencia de la CPU para poder soportar la técnica de polling?
- d) ¿Cuál sería el tiempo total en segundos, incluyendo los ciclos requeridos de preparación y terminación, necesario para transmitir una imagen completa por el método basado en DMA?

Nota: considerar $1 \text{ MB} = 2^x$ bytes; $1 \text{ KB} = 2^y$ bytes y $1 \text{ GHz} = 10^z$ hercios.

12. Considerar un bus con líneas de datos y direcciones multiplexadas de 32 bits de ancho y un procesador de 500MHz. La memoria tiene una latencia de 3 ciclos. Transmitir una dirección por el bus o una palabra de 32 bits requiere 1 ciclo. Además, el sistema de memoria soporta un modo ráfaga que a partir del 4º ciclo le permite transmitir una palabra por ciclo hasta un máximo de 4. Finalmente, las escrituras requieren un último ciclo para mandar un código de corrección de errores (CRC). Se pide:
- a) Calcular el ancho de banda máximo (en MB/seg) en los casos de que solo haya lecturas, solo haya escrituras, y que haya una mezcla de 65 % de lecturas y 35 % de escrituras.
 - b) Repetir el apartado anterior, pero suponiendo que ahora el bus usa líneas dedicadas para direcciones y datos (32 líneas en cada caso), de manera que al realizar una escritura, puede ponerse una palabra para escribir a la vez que la dirección. Además, el modo ráfaga permite escribir 3 palabras más a 1 por ciclo.
 - c) Supongamos que ahora se conecta al bus un disco duro externo capaz de transmitir 120 MB/seg. Si las transferencias tienen que hacerse mediante sondeo, teniendo en cuenta que una operación de sondeo para transmitir 4 palabras requiere 300 ciclos. ¿Cual debería ser la frecuencia de la CPU para que la transferencia solo requiera una dedicación de la CPU del 15 %?
 - d) Si en lugar de sondeo, se hace uso de interrupciones para avisar a la CPU cada vez que se transmiten 4 palabras, cuyo tratamiento implica 50 ciclos. ¿Que porcentaje del tiempo de procesamiento de la CPU se dedicaría a esta tarea?
 - e) Finalmente, suponer que en lugar de sondeo o interrupciones, se hace uso de una controladora de DMA para controlar la transferencia de los datos. La controladora trabaja con bloques de 4KB, su inicialización requiere 50 ciclos, y el tratamiento de la interrupción al terminar cada transferencia otros 100 ciclos. En este caso, ¿cual será el porcentaje de tiempo que la CPU dedica a controlar la transferencia de los datos?

Nota: tened en cuenta que el prefijo M expresa una potencia en base 10 (10^x) cuando se refiere a la frecuencia del procesador, mientras que en el resto de casos, los prefijos M y K expresan potencias en base 2 (2^x).

13. Tenemos una cámara digital de 5 megapíxeles, con 65536 colores por píxel, conectada al bus USB 2.0, el cual tiene un ancho de banda total de 480 Mbits/s. Si nuestro procesador opera a una frecuencia de 200 MHz, determinar lo siguiente:
- a) Calcular el mínimo ancho de banda necesario para poder transferir 5 imágenes por segundo.
 - b) El porcentaje de ocupación de la CPU si la transferencia se realiza mediante DMA y en cada operación DMA se transfiere 1 KByte. Se sabe que la preparación de la operación de DMA necesita 100 ciclos de reloj y el tratamiento de la interrupción de terminación es de 50 ciclos.

- c) El porcentaje de ocupación de la CPU si la transferencia se realiza mediante interrupciones y se necesitan 50 ciclos de reloj para transferir un bloque de 4 palabras.
- d) Supongamos que el sistema no soportase interrupciones ni DMA, y que sólo pudiese accederse a ella mediante sondeo. Si suponemos que cada bloque de 4 palabras transferidas necesita una operación de encuesta, la cual consume 300 ciclos de CPU, ¿cuál debería ser la frecuencia de la CPU para poder soportar la técnica de sondeo?
- e) ¿Cuál debería ser el tamaño del bloque a transferir, mediante interrupciones, si queremos que el procesador no dedique más de un 39 % a la realización de la transferencia? (Se tomará como frecuencia del procesador la del enunciado; también se considerará un consumo por interrupción de 50 ciclos).

Notas: 1 palabra = 4 bytes. Considerar $1 \text{ MB} = 2^x$ bytes; $1 \text{ KB} = 2^y$ bytes; $1 \text{ GHz} = 10^z$ hercios y $1 \text{ MP} = 10^t$ píxeles

14. Un sistema dispone de un procesador Core 2 Duo a 2,66 Ghz y de una tarjeta de sonido capaz de generar sonido con calidad de CD, es decir, 44.100 muestras por segundo, donde en cada muestra se envían 32 bits (16 bits por cada uno de los dos canales de sonido que forman una comunicación en estéreo). Se pretende usar este sistema para reproducir ficheros de sonido en formato WAV. Para ello, se leen muestras de sonido del disco duro y se envían a la tarjeta de sonido siguiendo las siguientes 3 etapas:

- Etapa 1: se leen muestras de sonido en bloques de 4 KB del disco duro. Para ello, se usa un controlador DMA que requiere 4000 ciclos de inicialización y 2000 ciclos al terminar la transferencia.
- Etapa 2: procesamiento de los datos en la CPU. Para el formato WAV supondremos que el procesamiento requiere 0 ciclos de CPU ya que la tarjeta de sonido es capaz de reproducir directamente datos en dicho formato.
- Etapa 3: para reproducir el sonido, el procedimiento para enviar las muestras a la tarjeta de sonido puede ser por unos de estos dos métodos diferentes:
 - Por interrupciones (44.100 interrupciones por segundo), donde en cada interrupción, que requiere 200 ciclos de CPU, se envía a la tarjeta una muestra de 32 bits.
 - Por DMA, donde cada transferencia de 8 KB requiere un procesamiento de 4000 ciclos de inicialización más 2000 ciclos al terminar.

Se pide:

- a) Calcular el % de CPU consumido en esta tarea (las 3 etapas), usando interrupciones en la etapa 3.
- b) Calcular el % de CPU consumido en esta tarea (las 3 etapas), usando DMA en la etapa 3.
- c) Calcular el % de CPU consumido en esta tarea (las 3 etapas), si se reproducen ficheros MP3 (formato que comprime WAV a un 10 % del tamaño original), sabiendo que el procesamiento para descomprimir los datos en la CPU de un bloque de 4 KB de MP3 requiere 40 millones de ciclos en la etapa 2. Supondremos que en la etapa 3 se sigue usando DMA.
- d) Supongamos que en la etapa 3 los datos se envían a la tarjeta de sonido a través de un bus con líneas dedicadas para datos y direcciones, con 32 bits para datos y otros 32 bits para direcciones. Si enviar una dirección requiere un ciclo de bus, la tarjeta de sonido tiene una latencia de 2 ciclos y el bus soporta un modo ráfaga capaz de enviar 32 bytes por ráfaga, a razón de 4 bytes por ciclo, ¿con qué frecuencia mínima debe funcionar el bus para reproducir el sonido correctamente?

Nota: considera que $1 \text{ MB} = 2^x$ bytes, $1 \text{ KB} = 2^y$ bytes y $1 \text{ GHz} = 10^z$ hercios.

15. Un disco duro se encuentra conectado a un ordenador a través de un bus de entrada/salida síncrono que posee 32 líneas para direcciones y 32 líneas para datos. Transmitir una dirección o un dato consume un ciclo de bus. Leer o escribir en memoria supone una latencia de 3 ciclos. A partir del quinto ciclo el sistema puede leer o escribir hasta 4 palabras de 32 bits a razón de una palabra por ciclo de bus. Además para las escrituras se necesitan dos ciclos adicionales para escribir el código de corrección de errores. Si sabemos que tanto el bus como el procesador tienen una frecuencia de funcionamiento de 500 MHz y que el 70 % de operaciones son de lectura y el 30 % de escritura.

Se pide:

- a) Porcentaje de uso de la CPU si se usan interrupciones en la gestión de la entrada/salida. Sabemos que en cada interrupción se transfieren 8 palabras y supone un gasto de 10 ciclos.
- b) Porcentaje de uso de la CPU si se usa DMA en la gestión de la entrada/salida. Sabemos que inicializar la DMA consume 200 ciclos y 100 ciclos el tratamiento de la interrupción. En cada operación se transfiere un bloque de 10 KB.
- c) Otra opción para gestionar la entrada/salida es mediante sondeo. Se establece como requisito que en esta tarea nunca se utilice más de un 49 % del tiempo del procesador. Además sabemos que para realizar cada sondeo se necesitan un total de 20 ciclos. Sabiendo esto, ¿qué número mínimo de palabras se necesitará transmitir en cada sondeo para que se cumpla el requisito anteriormente dado?
- d) Queremos transferir a memoria un archivo de 30 GB que se encuentra en disco mediante el método de DMA. ¿Qué tiempo se necesitará para realizar esta tarea? Nota: tener en cuenta que mientras se programa y se trata la interrupción de la DMA no se transfiere ningún dato.

E7.2. Solución a ejercicios seleccionados

11. a) Como generamos una interrupción cada 4 bytes transmitidos, el número de interrupciones/seg = $15 \times 2^{20} \text{ bytes/seg} / 4 \text{ bytes/inter} = 3,932,160 \text{ interrupciones/seg}$.
Ciclos consumidos/seg = $3,932,160 \text{ interrupciones/seg} \times 400 \text{ ciclos/interrupción} = 1,572,864,000 \text{ ciclos/seg}$.
% ocupación CPU = $1,572,864,000 / 3 \times 10^9 = 52,4 \%$.
- b) Número de operaciones DMA/seg = $15 \times 2^{20} \text{ bytes/seg} / 4 \times 2^{10} \text{ bytes/op.DMA} = 3,75 \times 2^{10} = 3840 \text{ op.DMA/seg}$.
Ciclos consumidos/seg = $3840 \text{ op.DMA/seg} \times 12000 \text{ ciclos/op.DMA} = 46,080,000 \text{ ciclos/seg}$.
% ocupación CPU = $46,080,000 / 3 \times 10^9 = 1,5 \%$.
- c) Para soportar una transferencia de 15 MBytes/seg mediante sondeo necesitamos:
 $15 \times 2^{20} \text{ bytes/seg} / 4 \text{ bytes/encuesta} = 3,932,160 \text{ encuestas/seg}$.
Como cada encuesta consume 1000 ciclos tenemos que:
 $3,932,160 \text{ encuestas/seg} \times 1000 \text{ ciclos/encuesta} = 3,932,160,000 \text{ ciclos/seg} = 3,93 \text{ GHz}$.
Ésa es la frecuencia mínima (3,93 GHz) y tendríamos un 100 % de ocupación de la CPU.
- d) Tamaño 1 imagen = $10 \times 10^6 \text{ píxeles} \times 3 \text{ bytes/píxel} = 30 \times 10^6 \text{ bytes}$.
Tiempo transmisión datos = $30 \times 10^6 \text{ bytes} / 15 \times 2^{20} \text{ bytes/seg} = 1,90 \text{ segundos}$.
Cuando usamos DMA, el tiempo total debe incluir los ciclos de preparación y terminación. El número de operaciones DMA para transmitir $30 \times 10^6 \text{ bytes}$, a razón de 4KB por operación DMA es: $30 \times 10^6 / 4096 = 7324,2 \text{ op.DMA} \rightarrow$ redondeando al siguiente entero: 7325 op.DMA.
Tiempo preparación y terminación = $12000 \text{ ciclos/op.DMA} \times 7325 \text{ op.DMA} / 3 \times 10^9 \text{ ciclos/seg} = 0,0293 \text{ segundos}$.
Tiempo total = $1,90 \text{ segundos} + 0,0293 \text{ segundos} = 1,9293 \text{ segundos}$.
12. a) Una operación de lectura haciendo uso del modo ráfaga requiere: 1 ciclo para mandar la dirección + 3 ciclos de latencia de memoria + 4 ciclos para mandar 4 palabras = 8 ciclos.
Una operación de escritura haciendo uso del modo ráfaga requiere: 1 ciclo para mandar la dirección + 3 ciclos de latencia de memoria + 4 ciclos para mandar 4 palabras + 1 ciclo para el CRC + 1 ciclo para finalizar la operación = 10 ciclos.
Patrón de solo lecturas: una operación de lectura transmite 4 palabras (de 4 bytes cada una) en 8 ciclos:

$$\frac{4 \text{ palabras} \times 4 \text{ bytes}}{8 \text{ ciclos}} = \frac{16 \text{ bytes}}{8 \text{ ciclos}} = 2 \text{ bytes/ciclo}$$

$$2 \text{ bytes/ciclo} \times 500 \text{ Mhz} = 2 \text{ bytes/ciclo} \times 500 \times 10^6 \text{ ciclos/segundo} = 1000000000 \text{ bytes/segundo} = 953,67 \text{ MB/seg}$$

Patrón de solo escrituras: una operación de escritura transmite 4 palabras (de 4 bytes cada una) en 10 ciclos:

$$\frac{4 \text{ palabras} \times 4 \text{ bytes}}{10 \text{ ciclos}} = \frac{16 \text{ bytes}}{10 \text{ ciclos}} = 1,6 \text{ bytes/ciclo}$$

$$1,6 \text{ bytes/ciclo} \times 500 \text{ Mhz} = 1,6 \text{ bytes/ciclo} \times 500 \times 10^6 \text{ ciclos/segundo} = 800000000 \text{ bytes/segundo} = 762,94 \text{ MB/seg}$$

Para un patrón con 65 % de lecturas y 35 % de escrituras, tenemos que los ciclos promedio por operación son:

$$0,65 \times 8 + 0,35 \times 10 = 8,7 \text{ ciclos}$$

Luego, en una operación de este tipo, se transmiten 4 palabras (de 4 bytes cada una) en 8,7 ciclos:

$$\frac{4 \text{ palabras} \times 4 \text{ bytes}}{8,7 \text{ ciclos}} = \frac{16 \text{ bytes}}{8,7 \text{ ciclos}} = 1,84 \text{ bytes/ciclo}$$

$$1,84 \text{ bytes/ciclo} \times 500 \text{ Mhz} = 1,84 \text{ bytes/ciclo} \times 500 \times 10^6 \text{ ciclos/segundo} = \\ 920000000 \text{ bytes/segundo} = 877,38 \text{ MB/seg}$$

- b) Ahora, según la nueva organización, tenemos que una operación de lectura haciendo uso del modo ráfaga requiere: 1 ciclo para mandar la dirección + 3 ciclos de latencia de memoria + 4 ciclos para mandar 4 palabras = 8 ciclos.

Una operación de escritura haciendo uso del modo ráfaga requiere: 1 ciclo para mandar la dirección y la primera palabra + 3 ciclos de latencia de memoria + 3 ciclos para mandar 3 palabras + 1 ciclo para el CRC + 1 ciclo para finalizar la operación = 9 ciclos.

Patrón de solo lecturas: al ser el mismo número de ciclos que antes, el ancho de banda es el mismo.

Patrón de solo escrituras: ahora, una operación de escritura transmite 4 palabras (de 4 bytes cada una) en 9 ciclos:

$$\frac{4 \text{ palabras} \times 4 \text{ bytes}}{9 \text{ ciclos}} = \frac{16 \text{ bytes}}{9 \text{ ciclos}} = 1,78 \text{ bytes/ciclo}$$

$$1,78 \text{ bytes/ciclo} \times 500 \text{ Mhz} = 1,78 \text{ bytes/ciclo} \times 500 \times 10^6 \text{ ciclos/segundo} = \\ 890000000 \text{ bytes/segundo} = 848,77 \text{ MB/seg}$$

Para un patrón con 65 % de lecturas y 35 % de escrituras, tenemos que los ciclos promedio por operación son:

$$0,65 \times 8 + 0,35 \times 9 = 8,35 \text{ ciclos}$$

Luego, en una operación de este tipo, se transmiten 4 palabras (de 4 bytes cada una) en 8,35 ciclos:

$$\frac{4 \text{ palabras} \times 4 \text{ bytes}}{8,35 \text{ ciclos}} = \frac{16 \text{ bytes}}{8,35 \text{ ciclos}} = 1,92 \text{ bytes/ciclo}$$

$$1,92 \text{ bytes/ciclo} \times 500 \text{ Mhz} = 1,92 \text{ bytes/ciclo} \times 500 \times 10^6 \text{ ciclos/segundo} = \\ 960000000 \text{ bytes/segundo} = 915,53 \text{ MB/seg}$$

- c) Lo primero que hay que observar es que el disco duro tiene un ancho de banda muy inferior al del bus, por lo que será el disco duro el que marque la velocidad de las operaciones a realizar.

Velocidad del disco duro: 120 MB/segundo.

Cada transferencia de 4 palabras (16 bytes) se realiza mediante una operación de polling que consume 300 ciclos.

$$\frac{120 \text{ MB/seg.}}{16 \text{ bytes/transf.}} = \frac{125829120 \text{ bytes/seg.}}{16 \text{ bytes/transf.}} = 7864320 \text{ transf./seg.}$$

Cada transferencia son 300 ciclos, luego:

$$7864320 \text{ transf./seg.} \times 300 \text{ ciclos/transf.} = 2359296000 \text{ ciclos/seg.}$$

Para que estos 2359296000 ciclos solo sean el 15 % de la CPU, tenemos:

$$\frac{2359296000}{x} = 0,15 \Rightarrow x = \frac{2359296000}{0,15} = 15728640000 \text{ ciclos/seg.} = 15,73 \text{ Ghz}$$

d)

$$\frac{120 \text{ MB/seg.}}{16 \text{ bytes/transf.}} = \frac{125829120 \text{ bytes/seg.}}{16 \text{ bytes/transf.}} = 7864320 \text{ transf./seg.}$$

Cada transferencia son 50 ciclos, luego:

$$7864320 \text{ transf./seg.} \times 50 \text{ ciclos/transf.} = 393216000 \text{ ciclos/seg.}$$

Teniendo en cuenta que el procesador trabaja a 500 Mhz, tenemos que:

$$\frac{393216000 \text{ ciclos/seg.}}{500 \times 10^6 \text{ ciclos/seg.}} = 0,7864 \Rightarrow 78,64 \% \text{ de uso de CPU}$$

e)

$$\frac{120 \text{ MB/seg.}}{4 \text{ KB/transf.}} = \frac{122880 \text{ KB/seg.}}{4 \text{ KB/transf.}} = 30720 \text{ transf./seg.}$$

Cada transferencia son 150 ciclos, luego:

$$30720 \text{ transf./seg.} \times 150 \text{ ciclos/transf.} = 4608000 \text{ ciclos/seg.}$$

Teniendo en cuenta que el procesador trabaja a 500 Mhz, tenemos que:

$$\frac{4608000 \text{ ciclos/seg.}}{500 \times 10^6 \text{ ciclos/seg.}} = 0,0092 \Rightarrow 0,92 \% \text{ de uso de CPU}$$

13. a) Cada imagen necesita $5 \times 10^6 \times 2$ bytes para ser codificada. Por tanto, para poder transferir 5 imágenes cada segundo se necesitará un ancho de banda de $5 \times 10^6 \times 2 \times 5 = 50 \times 10^6$ bytes por segundo.

b) Se necesitarán realizar $\frac{50 \times 10^6}{1024}$ transferencias cada segundo.

Cada transferencia requiere $100 + 50$ ciclos de CPU, por lo que cada segundo la CPU estará ocupada durante $\frac{50 \times 10^6}{1024} \times (100 + 50)$ ciclos.

Por tanto, teniendo en cuenta que la frecuencia de la CPU es de 200 MHz, la CPU el porcentaje de uso de la CPU será:

$$\frac{\frac{50 \times 10^6}{1024} \times (100 + 50)}{200 \times 10^6} = 3,6621 \%$$

c) En este caso, se recibirán $\frac{50 \times 10^6}{4 \times 4}$ interrupciones cada segundo, y cada una de ellas requerirá 50 ciclos de tiempo de CPU, por lo que el tiempo de uso de la CPU será:

$$\frac{\frac{50 \times 10^6}{4 \times 4} \times 50}{200 \times 10^6} = 78,125 \%$$

- d) Sería necesario realizar, al menos, $\frac{50 \times 10^6}{4 \times 4}$ operaciones de sondeo por segundo. Cada operación consumiría 300 ciclos de la CPU, por lo que la frecuencia mínima de la CPU debería ser:

$$\frac{50 \times 10^6}{4 \times 4} \times 300 = 937,5 \text{ MHz}$$

En este caso, el uso de la CPU sería del 100 %.

- e) Sea t el tamaño del bloque. Siguiendo los mismos razonamientos que en el apartado 3, se deberá cumplir que:

$$\frac{\frac{50 \times 10^6}{t} \times 50}{200 \times 10^6} = 39 \%$$

Por tanto:

$$t = \frac{50 \times 10^6 \times 50}{0,39 \times 200 \times 10^6} = 32,0512 \text{ bytes}$$

Es decir, se necesitará un tamaño de 33 bytes para obtener una ocupación de la CPU ligeramente menor al 39 %.

14. a) Para reproducir el sonido con la calidad deseada, necesitamos enviar 44100 muestras por segundo y 32 bits por muestra, lo que nos da una tasa de transferencia de:

$$44100 \text{ muestras/seg} \cdot 4 \text{ bytes/muestra} = 176400 \text{ bytes/segundo}$$

Con esta tasa de transferencia, vamos a ver qué porcentaje de CPU necesitamos en cada etapa:

- Etapa 1: necesitamos realizar $176400 / 4096 = 43,07$ transferencias de DMA por segundo. Puesto que cada transferencia requiere $(4000 + 2000) = 6000$ ciclos de CPU, en total necesitamos $43,07 * 6000 = 258398,4$ ciclos/segundo.
- Etapa 2: cuando se reproducen ficheros WAV, esta etapa no consume ciclos de CPU.
- Etapa 3: se producen 44100 interrupciones por segundo, donde cada una necesita 200 ciclos de CPU. Por tanto, en esta etapa necesitamos $44100 * 200 = 8820000$ ciclos por segundo.

Luego, el porcentaje de CPU que necesitamos en las tres etapas es:

$$\%CPU = \frac{258398,4 + 0 + 8820000}{2,66 \cdot 10^9} \times 100 = 0,34 \%$$

- b) Este caso es similar al anterior, la diferencia está en que en la tercera etapa usamos DMA. Puesto que necesitamos seguir enviando 176400 bytes/segundo a la tarjeta de sonido, tendremos que realizar:

$$\frac{176400}{8192} = 21,53 \text{ transferencias de DMA por segundo}$$

Puesto que cada transferencia requiere $(4000 + 2000) = 6000$ ciclos de CPU, en total necesitamos $21,53 * 6000 = 129199,2$ ciclos/segundo.

Luego, el porcentaje de CPU que necesitamos en las tres etapas ahora es:

$$\%CPU = \frac{258398,4 + 0 + 129199,2}{2,66 \cdot 10^9} \times 100 = 0,015 \%$$

c) El utilizar ficheros MP3 cambia las etapas 1 y 2, pero no la etapa 3, ya que los datos se tienen que seguir enviando a la tarjeta de sonido a la misma velocidad y en el mismo formato.

En la etapa 1, al leer ficheros MP3, es decir, al leer la información comprimida, necesitamos leer a una velocidad 10 veces menor, lo que supone 10 veces menos transferencias de DMA y, por tanto, 10 veces menos ciclos de CPU por segundo. Ahora necesitaremos $258398,4 / 10 = 25839,84$ ciclos/segundo.

En la etapa 2 necesitamos consumir 40 millones de ciclos de CPU para descomprimir un bloque MP3 de 4 KB. Ya hemos dicho que al leer la información comprimida, necesitamos una tasa de transferencia 10 veces menor. Por tanto, la tasa de transferencia ahora es $176400 / 10 = 17640$ bytes/segundo, lo que supone descomprimir $17640 / 4096 = 4,31$ bloques de MP3 por segundo, que en total necesitan $4,31 * 40 * 10^6 = 172,3 * 10^6$ ciclos/segundo.

Luego, el porcentaje de CPU que necesitamos en las tres etapas ahora es:

$$\%CPU = \frac{25839,84 + 172,3 \cdot 10^6 + 129199,2}{2,66 \cdot 10^9} \times 100 = 6,5\%$$

d) El bus debe tener un ancho de banda efectivo de 176400 bytes/segundo. Para transferir 32 bytes, el bus necesita $1 + 2 + 8 = 11$ ciclos. La fórmula que obtenemos es:

$$\frac{32}{11} \cdot f = 176400$$

De donde podemos despejar f que será igual a 60637,5 ciclos/segundo.

15. a) Lo primero que tenemos que hacer es calcular el ancho de banda. Para ello utilizamos la siguiente fórmula:

$$BW = \frac{\text{bytes}}{\text{ciclos}} \times \text{frecuencia}$$

Para las lecturas tenemos la secuencia de ciclos que podemos ver en la figura E7.1, mientras que para las escrituras la secuencia de ciclos se muestra en la figura E7.2.

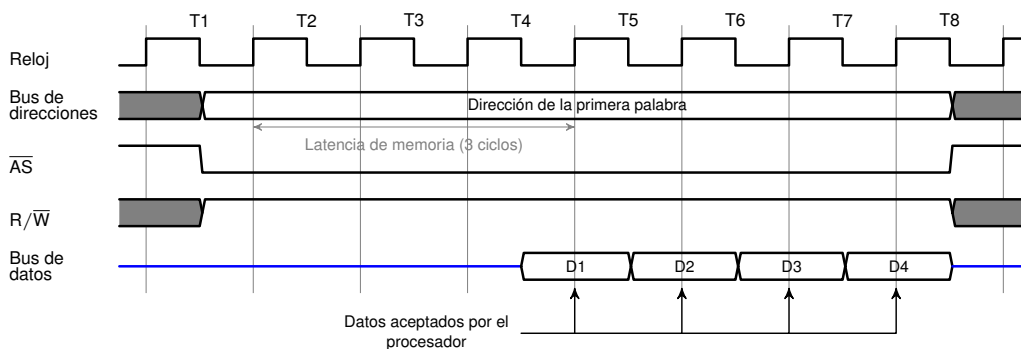


Figura E7.1: Lectura síncrona de 4 palabras en modo ráfaga.

$$BW = \frac{4 \text{ palabras} \cdot 4 \text{ bytes/palabra}}{(10 \text{ ciclos} \cdot 0'3 + 8 \text{ ciclos} \cdot 0'7)} \cdot 500 \cdot 10^6 = 930'23 \text{ MB/s} \approx 930 \text{ MB/s}$$

Una vez que hemos calculado el ancho de banda, podemos pasar a calcular el porcentaje de uso de la CPU. En primer lugar, tenemos que calcular el número de interrupciones que se producirán en un segundo:

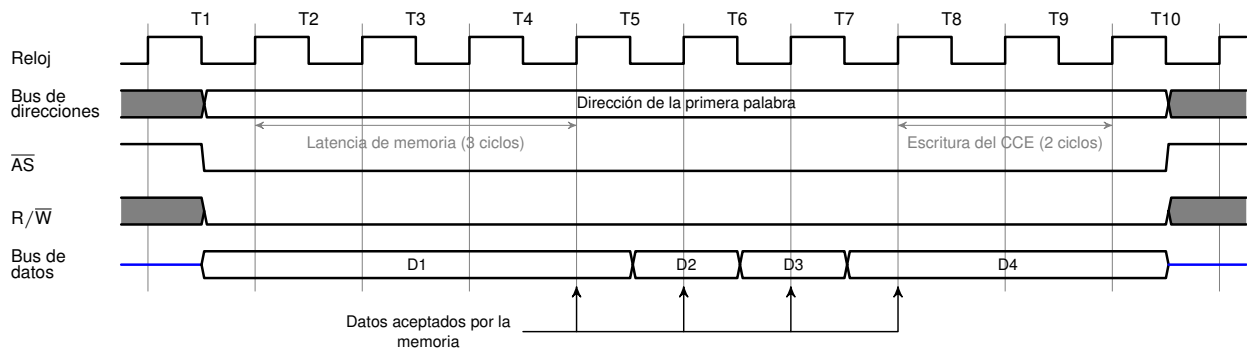


Figura E7.2: Escritura síncrona de 4 palabras en modo ráfaga con código de corrección de errores final.

$$\text{N}^\circ \text{ interrupciones} = \frac{930 \text{ MB/s}}{8 \text{ palabras} \cdot 4 \text{ bytes}} = \frac{930 \cdot 10^6 \text{ bytes/s}}{32 \text{ bytes}} = 29062500 \text{ interrupciones/s}$$

Ahora calculamos el total de ciclos que se consumirán en la realización de las interrupciones:

$$\text{Ciclos totales en interrupciones por segundo} = 29062500 \text{ int/s} \cdot 10 \text{ ciclos} = 290625000 \text{ ciclos/s}$$

Por último, calculamos el porcentaje de uso de la CPU:

$$\% \text{ CPU} = \frac{290625000}{500 \cdot 10^6} \cdot 100 = 58'13 \%$$

- b) Para este apartado se seguirán pasos parecidos a los dados en el apartado a). En primer lugar vamos, a calcular el número de operaciones de DMA que se realizarán en un segundo:

$$\text{N}^\circ \text{ op. DMA} = \frac{930 \text{ MB/s}}{10 \text{ KB}} = \frac{930 \cdot 10^3 \text{ KB/s}}{10 \text{ KB}} = 93000 \text{ operaciones/s}$$

Ahora calculamos el total de ciclos que se consumirán en el uso del DMA:

$$\text{Ciclos totales en DMA por segundo} = 93000 \text{ op. DMA/s} \cdot (200 + 100) \text{ ciclos} = 27900000 \text{ ciclos/s}$$

Por último, calculamos el porcentaje de uso de la CPU:

$$\% \text{ CPU} = \frac{27900000}{500 \cdot 10^6} \cdot 100 = 5'58 \%$$

- c) El primer paso es averiguar qué número de ciclos podemos dedicar como máximo a gestionar la entrada/salida mediante sondeo. Si en un segundo el procesador es capaz de ejecutar $500 \cdot 10^6$ ciclos, y para sondeo sólo podemos dedicar un máximo del 49 % del tiempo del procesador, entonces se podrán dedicar un total de:

$$500 \cdot 10^6 \cdot 49 \% = 245 \cdot 10^6 \text{ ciclos/segundo}$$

Ahora vamos a calcular el número de sondeos que se pueden realizar en un segundo:

$$\text{N}^\circ \text{ de sondeos} = \frac{245 \cdot 10^6 \text{ ciclos/segundo}}{20 \text{ ciclos}} = 12250000 \text{ sondeos/segundo}$$

Una vez que sabemos el número de sondeos, sólo nos queda calcular el número de palabras que hay que enviar en cada sondeo:

$$\text{N}^{\circ} \text{ de palabras} = \frac{\frac{930 \cdot 10^6 \text{ bytes/s}}{4 \text{ bytes/pal.}}}{12250000 \text{ sondeos/s}} = 18'98 \rightarrow 19 \text{ palabras/sondeo}$$

- d) En primer lugar vamos a ver lo que se tardaría en transmitir los 30GB a memoria sin tener en cuenta el tiempo usado en la programación del DMA:

$$\text{Tiempo} = \frac{30 \cdot 10^3 \text{ MB}}{930 \text{ MB/s}} = 32'26 \text{ segundos}$$

Ahora vamos a calcular el tiempo usado en la programación del DMA. Para ello vamos a calcular cuantas veces se usa el DMA en la transferencia del fichero:

$$\text{N}^{\circ} \text{ op. DMA} = \frac{30 \cdot 10^6 \text{ KB}}{10 \text{ KB}} = 3 \cdot 10^6 \text{ operaciones}$$

El siguiente paso es calcular el número de ciclos gastados en realizar todas estas operaciones:

$$\text{Ciclos totales en DMA} = 3 \cdot 10^6 \text{ op. DMA} \cdot (200 + 100) \text{ ciclos} = 900 \cdot 10^6 \text{ ciclos}$$

El último paso es calcular cuánto tiempo dedica el procesador en ejecutar todos esos ciclos:

$$\text{Tiempo} = \frac{900 \cdot 10^6 \text{ ciclos}}{500 \cdot 10^6 \text{ ciclos/s}} = 1'8 \text{ segundos}$$

El tiempo total para la transferencia del fichero será 32'26 seg + 1'8 seg = 34'06 segundos.