

# TEMA 5: ENTRADA / SALIDA

- Una de las principales tareas del SO es **controlar todos los dispositivos de E/S de la computadora**.
- Para poder hacer esto, el SO debe ser capaz de:
  - Emitir **comandos** para los dispositivos.
  - Captar las **interrupciones** que produzcan los dispositivos.
  - Manejar los **errores** que puedan provocar los dispositivos.
- Proporcionar una abstracción al usuario en forma de **interfaz simple entre los dispositivos y el resto del sistema**.
  - ↳ Hasta donde sea posible, esta interfaz debe ser igual para todos los dispositivos.
- El **software** de E/S forma una parte significativa del código del SO. Está estructurado en **niveles**, cada uno de los cuales tiene una **tarea bien definida**.
- Sobre el **hardware** de E/S nos centraremos únicamente en su **interfaz** con el software.

## HARDWARE DE E/S

### TIPOS DE DISPOSITIVOS DE E/S

- Dispositivos de **BLOQUE** → almacenan información en bloques de tamaño fijo (512B – 32KB).
  - Cada bloque tiene su propia **dirección**.
  - Las **lecturas y escrituras** se realizan en unidades de **uno o más bloques enteros consecutivos**, no se puede leer o escribir parte de un bloque.
  - Los bloques se leen o escriben de manera **independiente** a los demás.
  - ▷ Discos duros, CDs, memorias USB, etc.
- Dispositivos de **CARÁCTER** → envían o aceptan un flujo de caracteres sin estructura de bloque.
  - Los caracteres **no son direccionables** → **no** tienen operaciones de **búsqueda**.
  - ▷ Impresoras, ratones, interfaces de red, etc.
- Esta **clasificación no es perfecta**, pues algunos dispositivos de E/S no son ni de bloque ni de carácter (por ejemplo, el reloj).
  - De todas formas, es una clasificación lo bastante general como para poder usarla para independizar la parte del SO dedicada a la E/S.

Dispositivo	Velocidad de transferencia de datos
Teclado	10 bytes/seg
Ratón	100 bytes/seg
Módem de 56K	7 KB/seg
Escáner	400 KB/seg
Cámara de video digital	3.5 MB/seg
802.11g inalámbrico	6.75 MB/seg
CD-ROM de 52X	7.8 MB/seg
Fast Ethernet	12.5 MB/seg
Tarjeta Compact Flash	40 MB/seg
FireWire (IEEE 1394)	50 MB/seg
USB 2.0	60 MB/seg
Red SONET OC-12	78 MB/seg
Disco SCSI Ultra 2	80 MB/seg
Gigabit Ethernet	125 MB/seg
Unidad de disco SATA	300 MB/seg
Cinta de Ultrium	320 MB/seg
Bus PCI	528 MB/seg

### CONTROLADORAS DE DISPOSITIVOS

- Los dispositivos de E/S tienen un componente mecánico y otro electrónico denominado **CONTROLADORA** o controlador de dispositivo.
  - ↳ Normalmente la controladora será un **chip** en la **tarjeta principal** o una **tarjeta de circuito integrado** que se inserta en una ranura de expansión (PCI).
- La controladora tendrá un conector para un cable que lleva al dispositivo en sí. Muchas veces, a una sola tarjeta controladora se le pueden conectar **varios dispositivos**.
- Si la **interfaz** entre la controladora y el dispositivo es **estándar**, las empresas fabricarán controladoras y dispositivos que se adapten a ella.

### COMUNICACIÓN CONTROLADORA - CPU

- Cada controladora tiene unos cuantos **registros** que usa para comunicarse con la CPU.
  - El SO puede **enviar órdenes** al dispositivo **escribiendo** en los registros.
  - El SO puede **conocer el estado** del dispositivo **leyendo** en los registros.
- Además de los registros, muchos dispositivos tienen un **búfer de datos** en el que el SO puede leer o escribir.
- La **CPU** puede **comunicarse** con los registros y búferes de los dispositivos mediante un **puerto de E/S** o una **dirección de memoria**.

### PUERTOS DE E/S

- Consiste en asignarle a cada registro de control un **número de puerto de E/S** (entero de 8 o 16 bits).
- El conjunto de todos los puertos de E/S forma el **espacio de puertos de E/S**.
  - **Sólo el SO** puede acceder a él.
  - Es **independiente del espacio de direcciones de la memoria**: una dirección  $x$  en el espacio de direcciones de memoria es diferente a una dirección  $x$  en el de puertos.
- Las operaciones de lectura y escritura en puertos no se pueden realizar con las instrucciones para memoria  $lw$  y  $sw$ , se usan **instrucciones especiales**  $IN$  y  $OUT$ .

### ASIGNACIÓN DE MEMORIA

- Consiste en asignar los registros de control al **espacio de direcciones de memoria**.
- A cada registro de control se le asigna una dirección de memoria **única** para la que **no hay memoria asignada**.

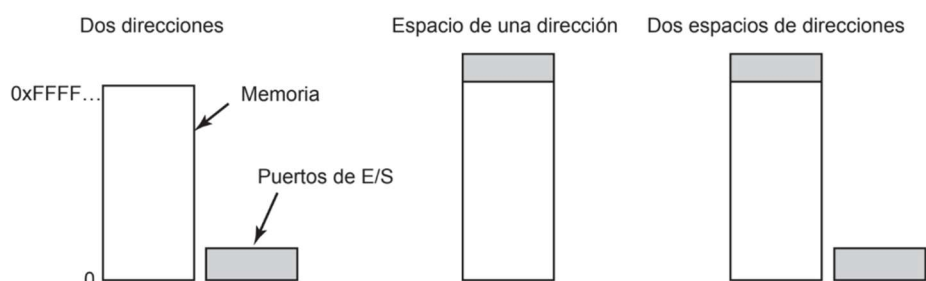
### ESQUEMA HÍBRIDO

- Consiste en asignar los **búferes de datos** al **espacio de direcciones de memoria** y los **registros de control** en un **espacio de puertos de E/S** separado.

### FUNCIONAMIENTO DE ESTOS ESQUEMAS

Cuando la CPU desea una palabra (ya sea de memoria o de un puerto de E/S):

1. La CPU coloca la dirección que necesita (ya sea de memoria o de un puerto de E/S) en el bus de direcciones.
2. La CPU activa la señal *read* en el bus de control.
3. En función del esquema usado:
  - A. Si se usan puertos de E/S:
    1. Se añade un bit en el bus de control para indicar si se necesita una palabra de E/S o de memoria.
    2. Responde la memoria o el dispositivo de E/S.
  - B. Si se usa asignación de memoria:
    1. Todos los módulos de memoria y los dispositivos de E/S comprueban si la dirección está en su rango.
    2. Responde el módulo de memoria o dispositivo de E/S que la tenga en su rango.



## VENTAJAS DE LA E/S POR ASIGNACIÓN DE MEMORIA

- ✓ No se necesitan **instrucciones especiales** para leer y escribir en los registros de control.
  - Como los registros de control se direccionan como cualquier otra variable en memoria, **los drivers de la controladora pueden estar completamente en C**.
  - ↳ Las instrucciones especiales que se usan para leer y escribir en los puertos E/S sólo se pueden ejecutar en ensamblador, no se pueden ejecutar en C o C++.
- ✓ No se necesita un mecanismo de **protección especial** para evitar que los procesos realicen operaciones de E/S.
 

Simplemente, el **SO no coloca** la porción del espacio de direcciones que contiene los **registros de control** en los **espacios de direcciones virtuales de los procesos**.

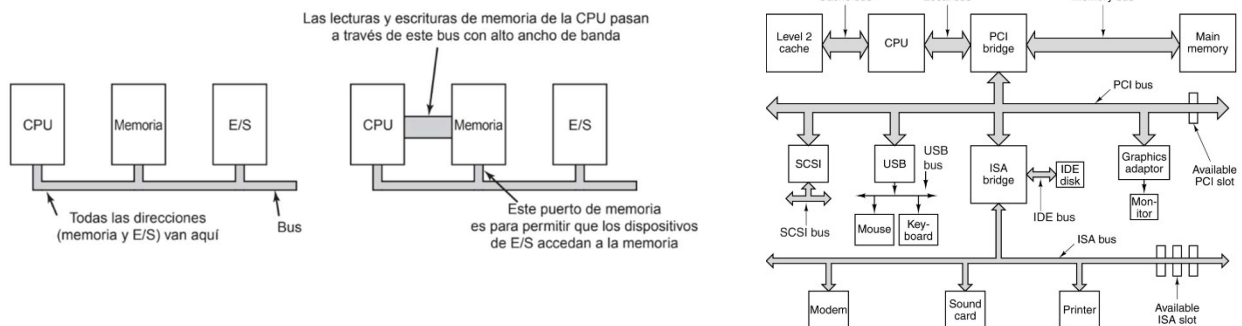
  - Si cada dispositivo tiene sus registros de control en una página distinta del espacio de direcciones, **se le puede permitir a un proceso operar con un determinado dispositivo** incluyendo la página deseada en su espacio de direcciones y omitiendo el resto de páginas asociadas a otros dispositivos.

## DESVENTAJAS DE LA E/S POR ASIGNACIÓN DE MEMORIA

- ✗ **Llevar a caché un registro de control** de un dispositivo puede ser **desastroso** → al intentar acceder al estado del dispositivo habría **problemas de coherencia de caché**.
  - Para evitar esto, las páginas del espacio de direcciones asignadas a los registros de control deben marcarse como **no cacheables**.
    - ↳ Esto añade complejidad adicional al hardware y al SO.
- ✗ **Los módulos de memoria y los dispositivos de E/S** deben examinar **todas las referencias** a memoria para saber a cuáles tienen que responder.
 

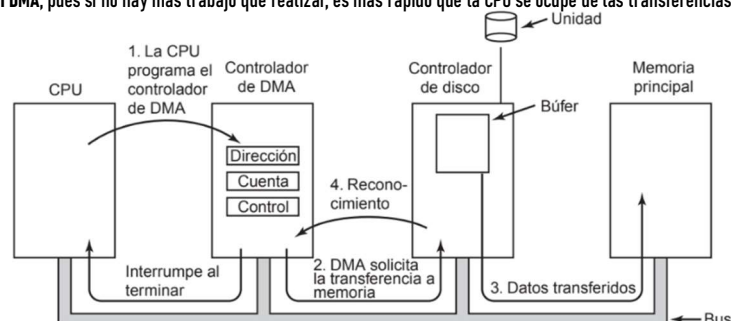
Esto es sencillo si la computadora tiene un solo bus, pero normalmente existirá un **bus separado** de alta velocidad dedicado únicamente a la **memoria**. Como consecuencia, **los dispositivos de E/S no tienen manera de ver las direcciones de memoria que van por el bus de memoria**, así que no las pueden responder.

  - Hay dos posibles soluciones a esto:
    - Se envían todas las referencias a la memoria principal, y si esta no responde la CPU prueba con otros buses.
      - ↳ Requiere hardware adicional.
    - Se filtran las direcciones que salen de la CPU con un **punto PCI** que contiene registros de rango que se cargan durante el arranque del sistema, de manera que las direcciones ubicadas dentro de un rango que no pertenece a la memoria se envían al bus PCI en lugar de al bus de memoria.



## ACCESO DIRECTO A MEMORIA

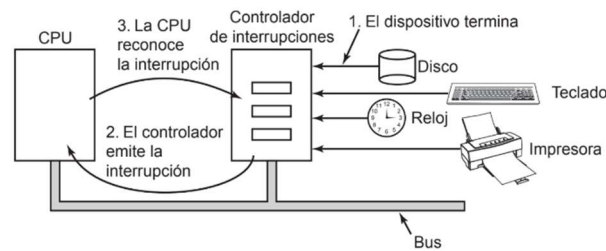
- El **acceso directo a memoria (DMA)** permite liberar a la CPU de labores poco sofisticadas relacionadas con la transferencia de datos de operaciones de E/S.
- Para usar este esquema, el hardware debe tener alguna **controladora de DMA**, que accede al bus de forma independiente de la CPU.
  - Normalmente, hay una **única controladora de DMA** que maneja varios dispositivos de E/S.
  - Contiene varios **registros** en los que la CPU puede escribir y leer.
    - Registro de **dirección de memoria**.
    - Registro **contador de bytes**.
    - Registros de **control**, que especifican
      - el puerto E/S a utilizar.
      - el tipo de transferencia (R/W).
      - unidad de transferencia (byte o palabra).
      - número de bytes a transferir por **ráfaga**.
  - Puede gestionar **varias transferencias** a la vez, de manera que cada una de ellas usa una **controladora de E/S distinta** y tiene un **canal con sus propios registros**.
  - Usa **direcciones físicas** para realizar las transferencias, por lo que las direcciones virtuales se tendrán que traducir antes de pasárselas a la controladora DMA.
- Las **lecturas de disco sin DMA** siguen el siguiente esquema:
  1. La controladora del disco lee el bloque solicitado, lo coloca bit a bit en su búfer interno y comprueba que no hay errores en él.
  2. La controladora del disco produce una interrupción.
  3. La CPU lee el bloque del búfer palabra a palabra (pues para poder acceder al buffer se tienen que cargar sus datos en un registro) y los va almacenando en memoria.
- Las **lecturas de disco con DMA** siguen el siguiente esquema:
  1. El SO programa la controladora de DMA escribiendo en sus registros y le indica al disco que debe leer datos en su búfer y verificarlos.
  2. Cuando ya hay datos válidos en el búfer del disco, la controladora de DMA envía una petición de lectura al controlador del disco que incluirá la dirección de memoria principal donde se escribirán los datos.
  3. La controladora del disco va escribiendo los datos de su búfer en memoria principal en ciclos de bus estándar, es decir, palabra por palabra.
  4. Cuando acaba una escritura, le envía un reconocimiento a la controladora de DMA. Esta incrementa la dirección de memoria a escribir y disminuye la cuenta de bytes.
  5. Cuando la cuenta de bytes llega a 0, la controladora de DMA envía una señal a la CPU para informarle de que ya acabó la transferencia de datos.
- Las controladoras DMA pueden operar en el bus de dos maneras:
  - En **modo palabra** → la controladora DMA compite con la CPU por el acceso al bus, **robando ciclos** de bus a la CPU para hacer las transferencias de datos.
  - En **modo bloque** → la controladora DMA indica al dispositivo que debe adquirir el bus, realizar una **ráfaga** de transferencias y después liberarlo.
  - El **modo de ráfaga** es mucho **más eficiente** que el robo de ciclo, pero **puede bloquear la CPU** si las ráfagas son muy largas.
- **No todos los sistemas usan DMA**, pues si no hay más trabajo que realizar, es más rápido que la CPU se ocupe de las transferencias.



## INTERRUPCIONES Y E/S

**A nivel de hardware, las interrupciones funcionan de la siguiente manera:**

1. Cuando un dispositivo de E/S (o DMA) termina su trabajo, produce una interrupción imponiendo una señal en la línea de bus que tenga asignada.
2. Esta señal es detectada por la controladora de interrupciones, que está dentro de la tarjeta principal.
3. La controladora de interrupciones emite una señal para interrumpir a la CPU, pasándole un número que indica qué dispositivo de E/S (o DMA) provocó la interrupción.
4. La CPU deja lo que está haciendo y comienza a ejecutar la rutina de manejo de la interrupción, cuya dirección se encuentra en la entrada del vector de interrupciones correspondiente al número del dispositivo que provocó la interrupción.
5. La CPU ejecuta la rutina de interrupción, que siempre comenzará por guardar la información de retorno en la pila.



## HARDWARE DE E/S

## CONCEPTOS CLAVE

- Existen 2 tipos de transferencias E/S:
  - Transferencias **síncronas** → implican el bloqueo del proceso involucrado.
  - Transferencias **asíncronas** → son controladas por interrupciones, la CPU inicia la transferencia y hace otras cosas hasta que llega la interrupción.
    - La mayoría de operaciones de E/S son asíncronas, el So es el responsable de hacer que en los programas de usuario parezcan de bloqueo.
- Todas las transferencias usan un **búfer en la zona de kernel de la memoria principal** como almacenamiento temporal hasta que se llevan los datos a su destino final, que en ocasiones no se conoce hasta que se examina el contenido del búfer.
  - Permite realizar un chequeo de errores sobre los datos y desacoplar el llenado del vaciado del búfer.
  - Su uso implica una cantidad considerable de copiado, lo que afecta al rendimiento de las operaciones de E/S.
- Hay 3 maneras distintas para llevar a cabo la E/S: programada, controlada por interrupciones o mediante DMA.

E/S PROGRAMADA

- La CPU **sondea constantemente el dispositivo de E/S** para ver si está listo para continuar con la transferencia.
- ✓ Es la forma más **simple** de realizar operaciones de E/S.
- ✗ Se **ocupa la CPU** hasta que se finaliza la operación.
- ▷ Ejemplo: `syscall` para imprimir "ABCDEFGH" en una impresora.
  1. El SO copia el búfer del espacio de usuario en un arreglo en el espacio de kernel para usarlo con más facilidad.
  2. El SO comprueba si la impresora está disponible. Si no lo está, espera hasta que lo esté.
  3. Tan pronto como la impresora está disponible, el SO copia un carácter al registro de datos de la impresora.
  4. Se repiten los pasos 2 y 3 hasta finalizar la cadena.

```
copiar_del_usuario(buter, p, cuenta);          /* p es el búfer del kernel */
for (i=0; i<cuenta; i++) {                    /* itera en cada carácter */
    while (*reg_estado_impresora != READY);   /* itera hasta que esté lista */
    *registro_datos_impresora = p[i];         /* imprime un carácter */
}
regresar_al_usuario();
```

## E/S CONTROLADA POR INTERRUPCIONES

- La CPU comienza la transferencia y **se planifica otro proceso**, de manera que el que comenzó la transferencia queda **bloqueado** hasta que esta finalice.
- ✖ El dispositivo de E/S generará **muchas interrupciones** para poder realizar la operación, que tendrá que resolver la CPU, **desperdiciando tiempo de CPU**.
- ▷ Ejemplo: syscall y manejador de interrupciones para imprimir "ABCDEFGH" en una impresora.
  1. El SO copia el búfer del espacio de usuario en un arreglo en el espacio de kernel para usarlo con más facilidad.
  2. Tan pronto como la impresora esté disponible se envía el primer carácter.
  3. La CPU llama al planificador y se bloquea el proceso.
  4. Cuando la impresora ha impreso el carácter y puede recibir otro, genera una interrupción.
  5. La interrupción bloquea el proceso actual y ejecuta el procedimiento de manejo de interrupciones de la impresora.
  6. Si no hay más caracteres en la cadena, se desbloquea el proceso.
  7. Si hay más caracteres por imprimir, se le envía a la impresora el siguiente y la CPU vuelve a hacer lo que estaba haciendo antes de la interrupción.

```

copiar_del_usuario(bufer, p, cuenta);
habilitar_interrupciones();
while (*reg_estado_impresora != READY);
*registro_datos_impresora = p[0];
planificador();

if (cuenta==0) {
    desbloquear_usuario();
} else {
    *registro_datos_impresora = p[i];
    cuenta=cuenta - 1;
    i = i + 1;
}
reconocer_interrupcion();
regresar de interrupcion();

```

## E/S CON DMA

- La controladora DMA realiza la transferencia sin intervención de la CPU.
- Es como una E/S programada pero el trabajo lo realiza la DMA en lugar de la CPU.
- ✗ Libera la CPU durante la operación de E/S para realizar otro trabajo.
- ✓ Requiere hardware especial.
- ▷ Ejemplo: syscall y manejador de interrupciones para imprimir "ABCDEFGH" en una impresora.

```
copiar_del_usuario(bufer, p, cuenta);
establecer_controlador_DMA();
planificador();
reconocer_interrupcion();
desbloquear_usuario();
regresar de interrupcion();
```