Introducción a los Sistemas Operativos

Entrada / Salida

I.S.O.

- ☑ Versión: Mayo 2013
- ☑ Palabras Claves: Entrada , Salida, Dispositivos, Interrupciones, DMA, driver

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.

Variedad en los dispositivos de I/O

- Legible por el Hombre
  - Usados para comunicarse con el usuario
    - Impresoras, Terminales: Pantalla, Teclado, Mouse
- Legible por la Máquina
  - Utilizados para comunicarse con los componentes electrónicos
    - Discos, Cintas, Sensores, etc.
- Comunicación
  - Usados para comunicarse con dispositivos remotos
    - Líneas Digitales, Modems, Etc.

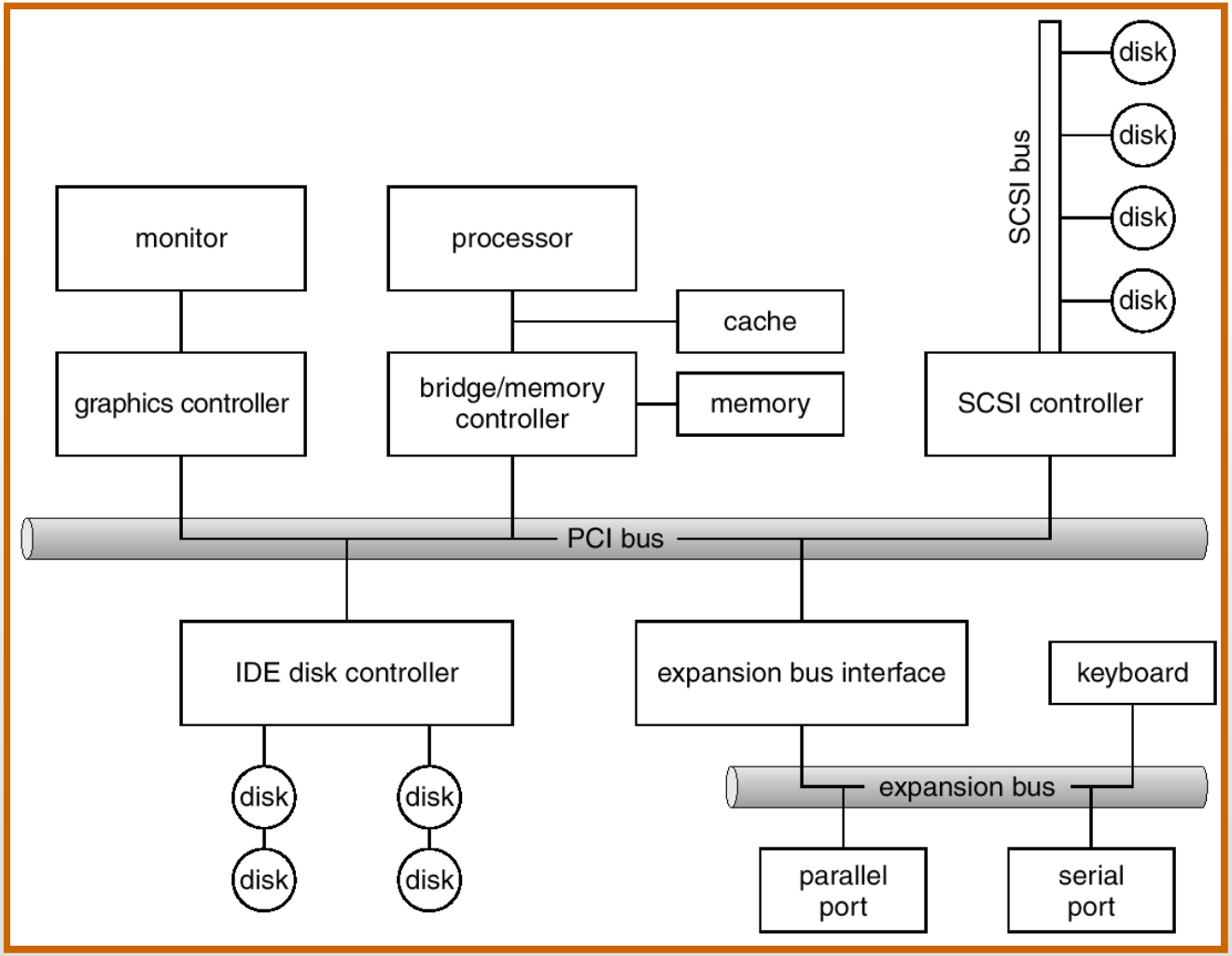
Problemas que surgen

- Amplia Variedad
  - Manejan diferentes cantidad de datos
  - En Velocidades Diferentes
  - En Formatos Diferentes
- La gran mayoría de los dispositivos de E/S son más lentos que la CPU y la RAM

Hardware y software involucrado

- ☑ Buses
- ☑ Controladores
- ☑ Dispositivos
- ☑ Puertos de E/S – Registros
- ☑ Drivers
- ☑ Comunicación con controlador del dispositivo: I/O Programada, Interrupciones, DMA

Estructura de Bus de una PC



Comunicación: CPU - Controladora

- ¿Cómo puede la CPU ejecutar comandos o enviar/recibir datos de una controladora de un dispositivo?
  - La controladora tiene uno o mas registros:
    - Registros para señales de control
    - Registros para datos
- La CPU se comunica con la controladora escribiendo y leyendo en dichos registros

Comandos de I/O

- CPU emite direcciones
  - Para identificar el dispositivo
- CPU emite comandos
  - Control – Que hacer?
    - Ej. Girar el disco
  - Test – Controlar el estado
    - Ej. power? Error?
  - Read/Write
    - Transferir información desde/hacia el dispositivo

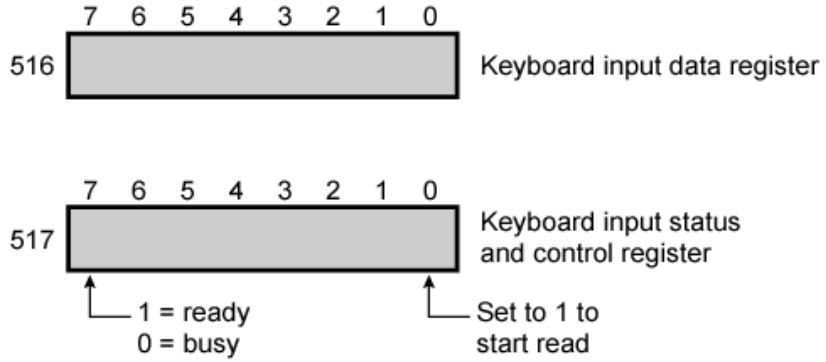
Mapeo de la E/S (I/O Mapping)

- Correspondencia en memoria (Memory mapped I/O)
  - Dispositivos y memoria comparten el espacio de direcciones.
  - I/O es como escribir/leer en la memoria.
  - No hay instrucciones especiales para I/O
    - Ya se dispone de muchas instrucciones para la memoria
- Isolated I/O (Aislada, uso de Puertos de E/S)
  - Espacio separado de direcciones
  - Se necesitan líneas de I/O. Puertos de E/S
  - Instrucciones especiales
    - Conjunto Limitado

Memory Mapped and Isolated I/O

ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load AC	"1"	Load accumulator
	Store AC	517	Initiate keyboard read
202	Load AC	517	Get status byte
	Branch if Sign = 0	202	Loop until ready
	Load AC	516	Load data byte

(a) Memory-mapped I/O

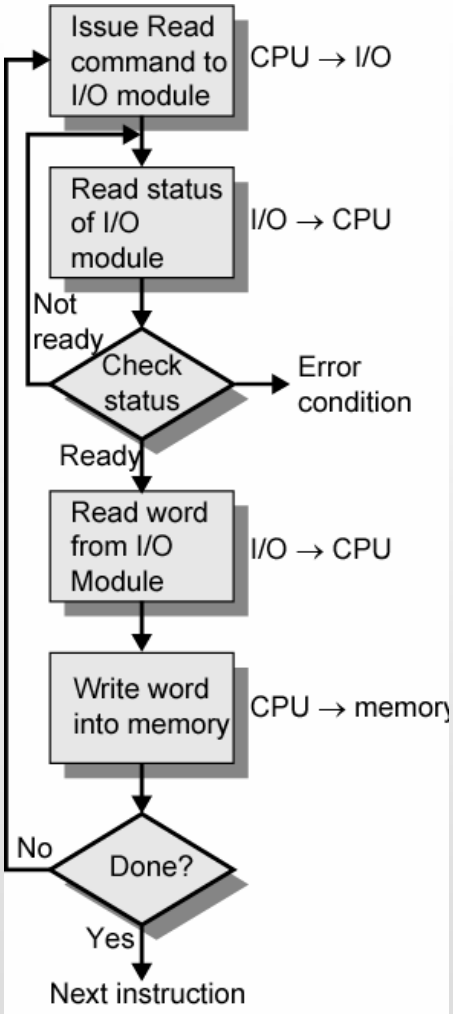


ADDRESS	INSTRUCTION	OPERAND	COMMENT
200	Load I/O	5	Initiate keyboard read
201	Test I/O	5	Check for completion
	Branch Not Ready	201	Loop until complete
	In	5	Load data byte

(b) Isolated I/O

Técnicas de I/O - Programada

- CPU tiene control directo sobre la I/O
  - Controla el estado
  - Comandos para leer y escribir
  - Transfiere los datos
- CPU espera que el componente de I/O complete la operación
- Se desperdician ciclos de CPU

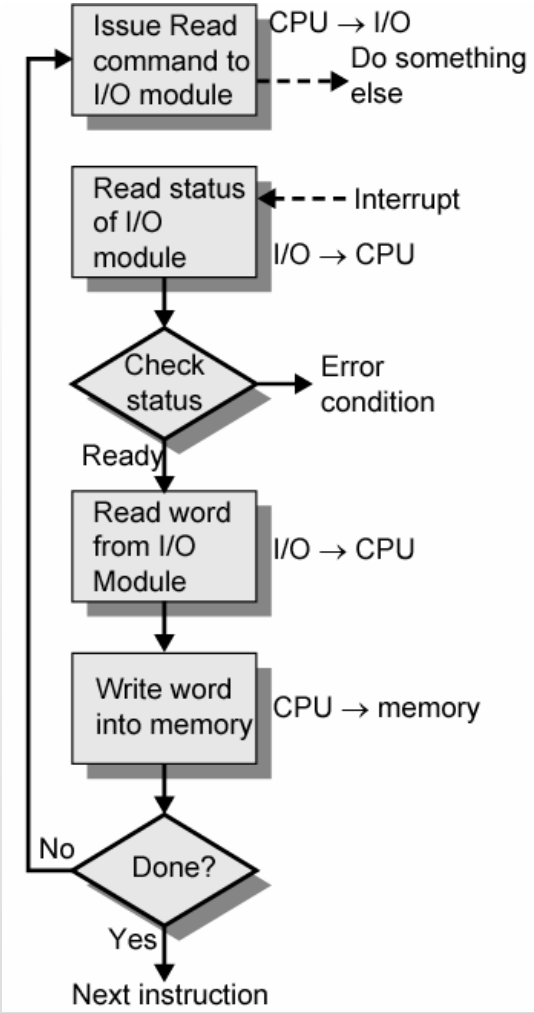


Polling

- En la I/O Programada, es necesario hacer polling del dispositivo para determinar el estado del mismo
  - Listo para recibir comandos
  - Ocupado
  - Error
- Ciclo de “Busy-wait” para realizar la I/O
- Puede ser muy costoso si la espera es muy larga

Técnicas de I/O - Manejada por Interrupciones

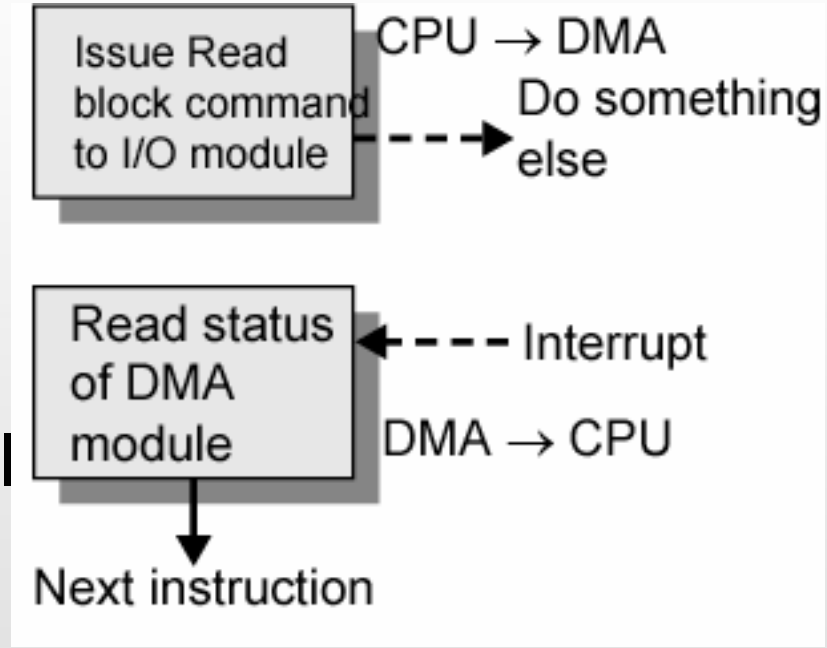
- Soluciona el problema de la espera de la CPU
- La CPU no repite el chequeo sobre el dispositivo
- El procesador continúa la ejecución de instrucciones
- El componente de I/O envía una interrupción cuando termina



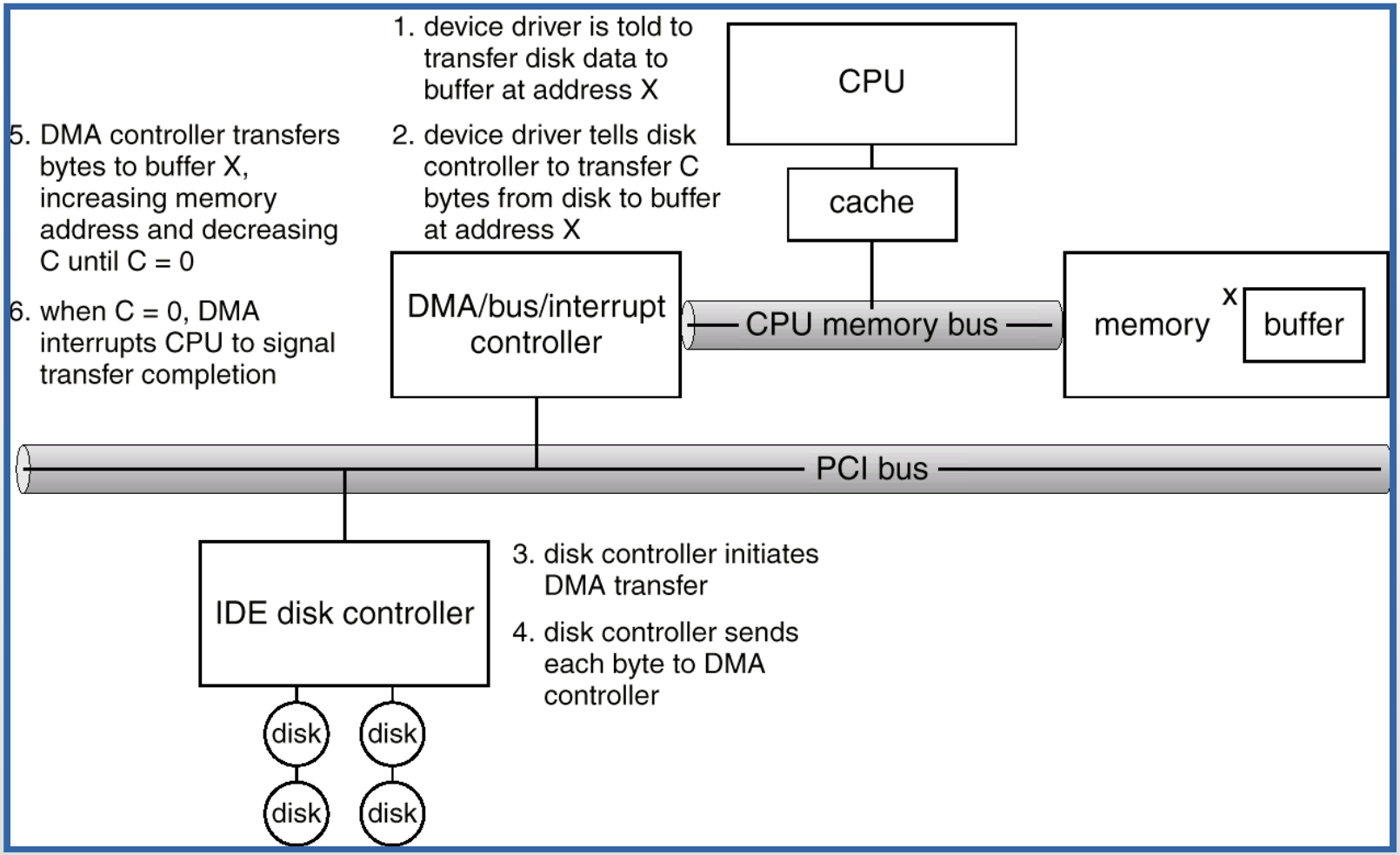
Técnicas de I/O - DMA

DMA (Direct Memory Access)

- ☑ Un componente de DMA controla el intercambio de datos entre la memoria principal y el dispositivo
- ☑ El procesador es interrumpido luego de que el bloque entero fue transferido.



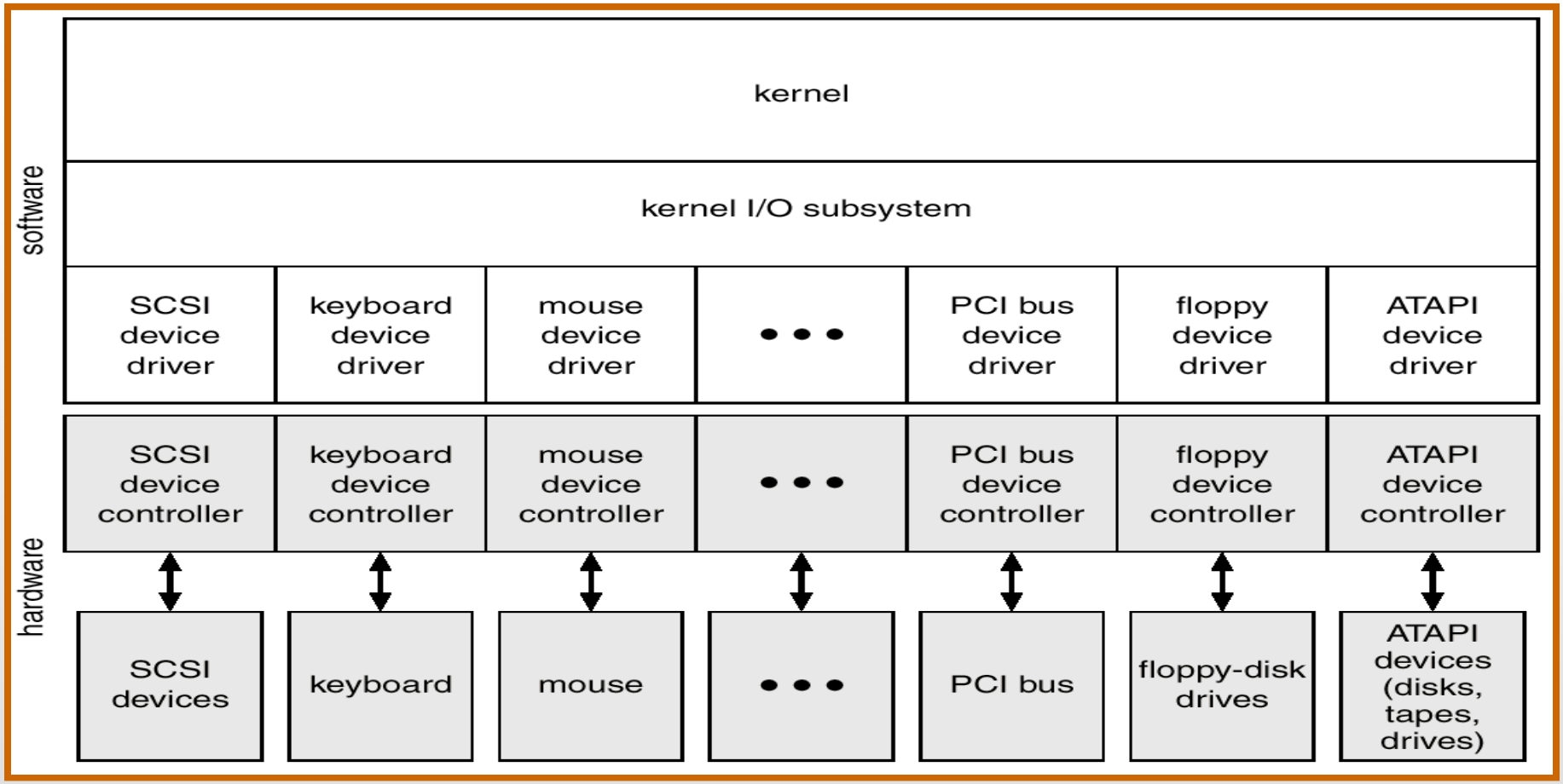
Pasos para una transferencia DMA



Interfaz de I/O - Metas

- ☑ Es deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada.
- ☑ Ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” para que los procesos vean a los dispositivos, en términos de operaciones comunes como: read, write, open, close, lock, unlock

Subsistema de I/O



Interfaz de I/O - Metas

- Respecto a la EFICIENCIA
  - Los dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria
  - El uso de la multiprogramación permite que los procesos esperen por la finalización de la I/O mientras que otro se ejecuta
  - I/O no puede alcanzar la velocidad de la CPU

Aspectos de los dispositivos de I/O

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only readÐwrite	CD-ROM graphics controller disk

Aspectos de los dispositivos de I/O (cont)

- Unidad de Transferencia
  - Dispositivos por bloques (discos):
    - Operaciones: `Read, Write, Seek`
  - Dispositivos por Caracter (keyboards, mouse, serial ports)
    - Operaciones: `get, put`
- Formas de Acceso
  - Secuencial o Aleatorio

Aspectos de los dispositivos de I/O (cont)

- ☑ Velocidad

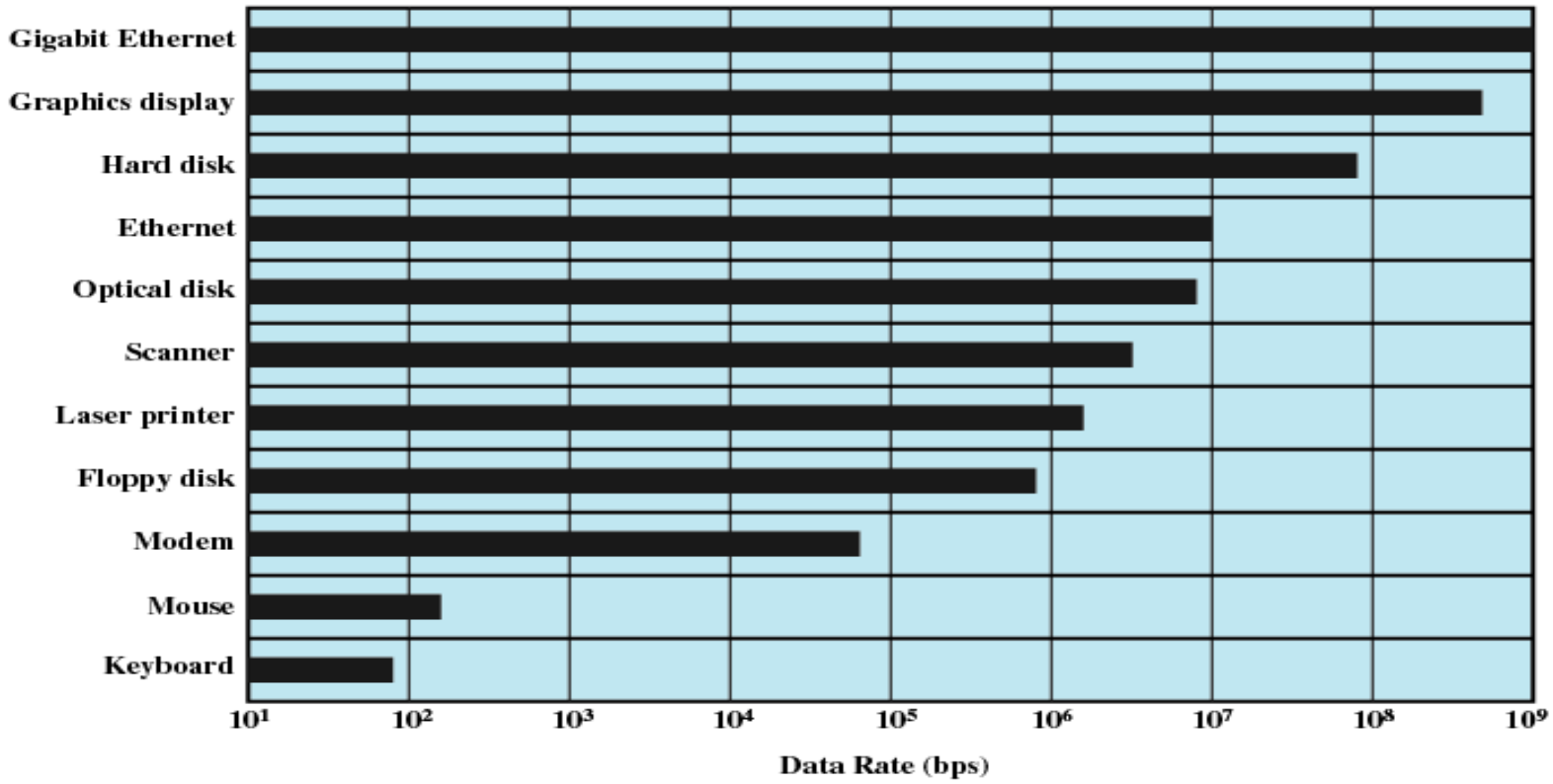


Figure 11.1 Typical I/O Device Data Rates

Aspectos de los dispositivos de I/O (cont)

- Tipo de acceso
  - Acceso Compartido: Disco Rígido
  - Acceso Exclusivo: Impresora

- Tipo de acceso:
  - Read only: CDROM
  - Write only: Pantalla
  - Read/Write: Disco

Subsistema de I/O - Servicios

- Planificación
  - Organización de los requerimientos a los dispositivos
  - Ej: Planificación de requerimientos a disco para minimizar movimientos
- Buffering – Almacenamiento de los datos en memoria mientras se transfieren
  - Solucionar problemas de velocidad entre los dispositivos
  - Solucionar problemas de tamaño y/o forma de los datos entre los dispositivos

Subsistema de I/O - Servicios (cont.)

- Caching – Mantener en memoria copia de los datos de reciente acceso para mejorar performance

- Spooling – Administrar la cola de requerimientos de un dispositivo
  - Algunos dispositivos de acceso exclusivo, no pueden atender distintos requerimientos al mismo tiempo: Por ej. Impresora
  - Spooling es un mecanismo para coordinar el acceso concurrente al dispositivo

Subsistema de I/O - Servicios (cont.)

- Reserva de Dispositivos: Acceso exclusivo
- Manejo de Errores:
  - El S.O. debe administrar errores ocurridos (lectura de un disco, dispositivo no disponible, errores de escritura)
  - La mayoría retorna un número de error o código cuando la I/O falla.
  - Logs de errores

Subsistema de I/O - Servicios (cont.)

- Formas de realizar I/O
  - Bloqueante: El proceso se suspende hasta que el requerimiento de I/O se completa
    - Fácil de usar y entender
    - No es suficiente bajo algunas necesidades
  - No Bloqueante: El requerimiento de I/O retorna en cuanto es posible
    - Se implementa usando multi-threading
    - Ejemplo: interfaz de usuario que recibe input desde el teclado/mouse y se muestra en el screen.

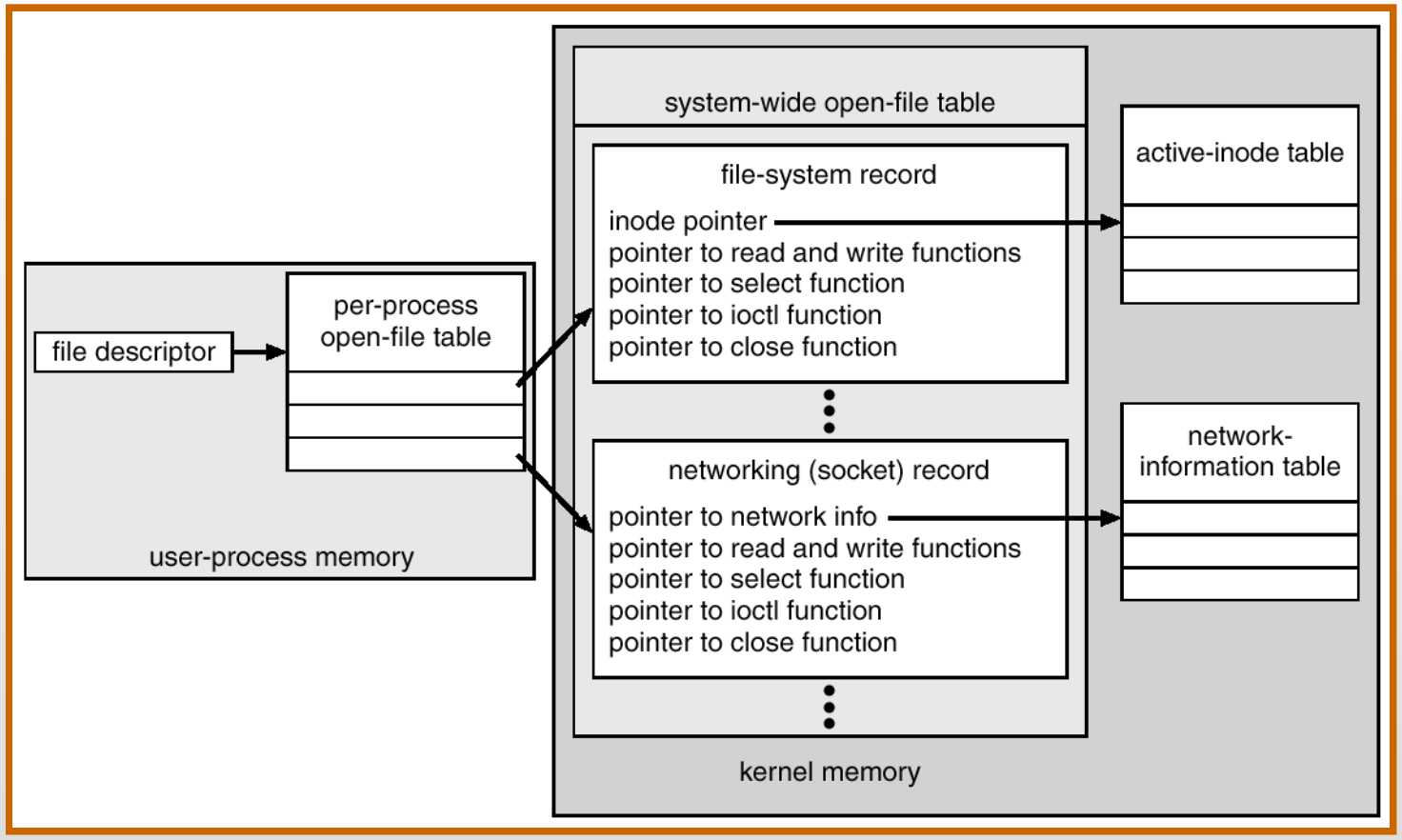
    Aplicación de video que lee frames desde un archivo mientras va mostrandolo en pantalla.

Subsistema de I/O - Estructuras de Datos

- El Kernel mantiene la información de estado de cada dispositivo o componente
  - Archivos abiertos
  - Conexiones de red
  - Etc.
- Hay varias estructuras complejas que representan buffers, utilización de la memoria, disco, etc.

Subsistema de I/O - Estructura de Datos

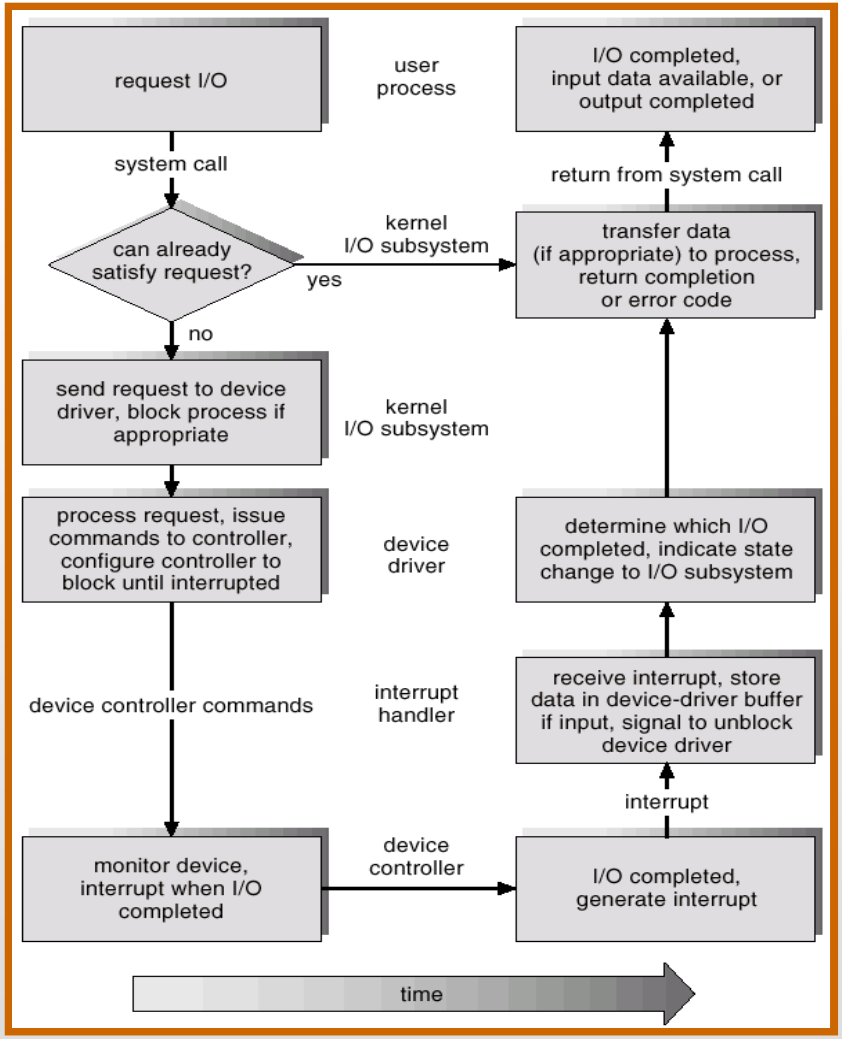
UNIX I/O Kernel Structure



Desde el Requerimiento de I/O hasta el Hardware

- Consideremos la lectura sobre un archivo en un disco:
  - Determinar el dispositivo que almacena los datos
  - Traducir el nombre del archivo en la representación del dispositivo.
  - Lectura física de los datos en un buffer de memoria
  - Marcar los datos como disponibles al proceso que realizo el requerimiento
  - Retornar el control al proceso

Ciclo de vida de un requerimiento de I/O



Subsistema de I/O - Drivers

- Contienen el código dependiente del dispositivo
- Manejan un tipo dispositivo
- Traducen los requerimientos abstractos en los comandos para el dispositivo
  - Escribe sobre los registros del controlador
  - Acceso a la memoria mapeada
  - Encola requerimientos
- Comúnmente las interrupciones de los dispositivos están asociadas a una función del driver

Subsistema de I/O - Drivers

- Interfaz entre el SO y el HARD
- Forman parte del espacio de memoria del Kernel
  - En general se cargan como módulos
- Los fabricantes de HW implementan el driver en función de una API especificada por el SO
  - open(), close(), read(), write(), etc
- Para agregar nuevo HW sólo basta indicar el driver correspondiente sin necesidad de cambios en el Kernel

Driver - Ejemplo en Linux

- Linux distingue 3 tipos de dispositivos
  - Carácter: I/O programa o por interrupciones
  - Bloque: DMA
  - Red: Ports de comunicaciones
- Los Drivers se implementan como módulos
  - Se cargan dinámicamente
- Debe tener al menos estas operaciones:
  - init_module: Para instalarlo
  - cleanup_module: Para desinstalarlo.

Driver - Ejemplo en Linux (cont.)

- Operaciones que debe contener para I/O
  - open: abre el dispositivo
  - release: cerrar el dispositivo
  - read: leer bytes del dispositivo
  - write: escribir bytes en el dispositivo
  - ioctl: orden de control sobre el dispositivo

Driver - Ejemplo en Linux (cont.)

- Otras operaciones menos comunes
  - llseek: posicionar el puntero de lectura/escritura
  - flush: volcar los búferes al dispositivo
  - poll: preguntar si se puede leer o escribir
  - mmap: mapear el dispositivo en memoria
  - fsync: sincronizar el dispositivo
  - fasync: notificación de operación asíncrona
  - lock: reservar el dispositivo
  - .....

Driver - Ejemplo en Linux (cont.)

- Por convención, los nombres de las operaciones comienzan con el nombre del dispositivo
- Por ejemplo, para /dev/ptr

```
int ptr_open (struct inode *inode, struct file *filp)

void ptr_release (struct inode *inode, struct file *filp)

ssize_t ptr_read (struct file *flip, char *buff,
                  size_t count, loff_t *offp)

ssize_t ptr_write (struct file *filp, const char *buff,
                   size_t count, loff_t *offp)

int ptr_ioctl (struct inode *inode, struct file *filp,
               unsigned int cmd, unsigned long arg)
```

Driver - Ejemplo en Linux (cont.)

- Acceso al hardware
  - Funciones para acceso a los puertos de I/O <asm/io.h>

```
unsigned char inb (unsigned short int port)
void outb (unsigned char value, unsigned short int port)
```

- Leen o Escriben un byte en el puerto indicado

Performance

- I/O es uno de los factores que mas afectan a la performance del sistema:
  - Utiliza CPU para executar los drivers y el codigo del subsistema de I/O
  - Context switches ante las interrupciones y bloqueos de los procesos
  - Copia de datos:
    - Aplicaciones (espacio usuario) – Kernel
    - Kernel (memoria fisica) - Controladora

Mejorar la Performance

- ☑ Reducir el número de context switches
- ☑ Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación
- ☑ Reducir la frecuencia de las interrupciones, utilizando:
  - Transferencias de gran cantidad de datos
  - Controladoras mas inteligentes
  - Polling, si se minimiza la espera activa.
- ☑ Utilizar DMA