

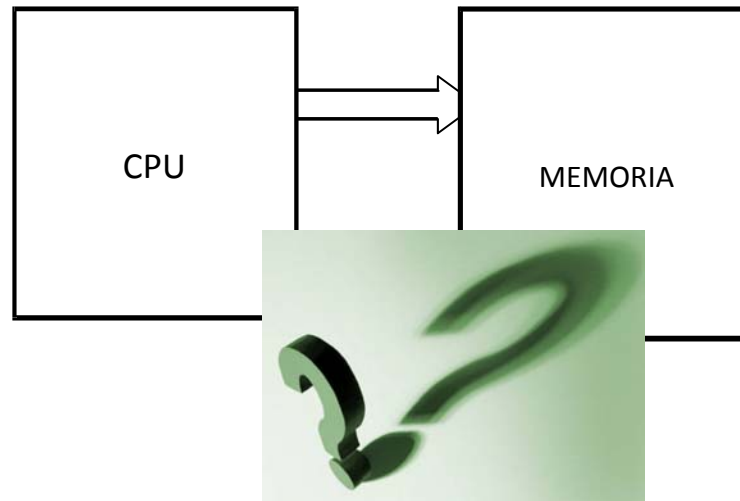


Fundamentos de Computadores

2º Cuatrimestre
2012-2013

**MÓDULO 12: Introducción al Subsistema de
Entrada/Salida**

¿Basta con CPU y Memoria?

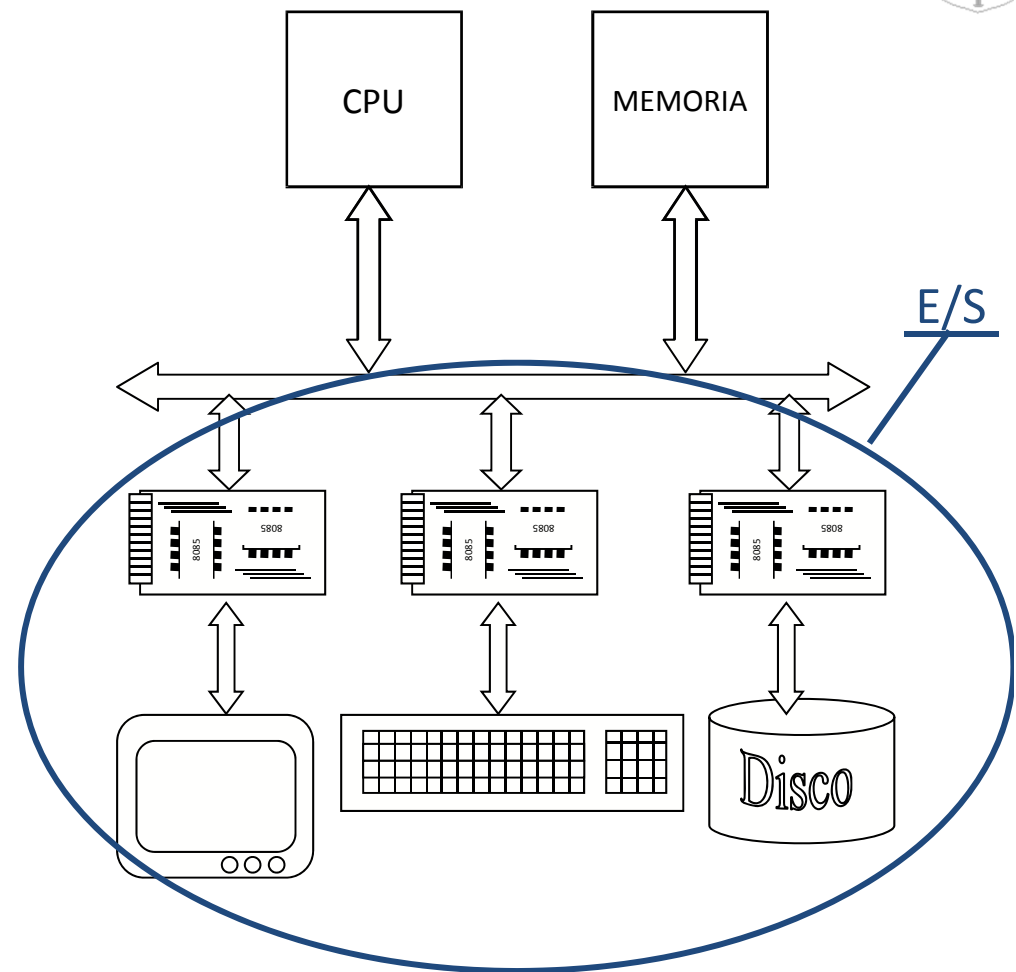


- El código/datos de una aplicación debe estar en memoria pero... ¿cómo llega allí? ¿Dónde está almacenado el fichero binario?
- ¿Cómo se produce la interacción con el usuario?

Necesidad del subsistema de E/S



- ¿De dónde proceden los datos que llegan a memoria? ¿A dónde van?
- ¿Cómo interacciona la CPU con el mundo exterior?
- Esta unidad presenta los aspectos más relevantes de la E/S
 - En muchas ocasiones, el **rendimiento del subsistema de E/S** determina el rendimiento global del sistema





Tipos de periféricos

- Dispositivos de **presentación de datos**.
 - Interaccionan con los usuarios, transportando datos entre éstos y la máquina
 - Ratón, teclado, pantalla, impresora, etc.
- Dispositivos de **almacenamiento de datos**.
 - Forman parte de la jerarquía de memoria: Interactúan de forma autónoma con la máquina
 - Discos magnéticos y cintas magnéticas...
- Dispositivos de **comunicación** con otros procesadores.
 - Permiten la comunicación con procesadores remotos a través de redes
 - Tarjeta de red, módem...
- Dispositivos de **adquisición de datos**.
 - Permiten la comunicación con sensores y actuadores que operan de forma autónoma.
 - Se utilizan en sistemas de control automático de procesos por computador
 - Suelen incorporar conversores de señales A/D y D/A.

Latencia y ancho de banda



- **Ancho de banda:** cantidad de datos que se pueden transferir por unidad de tiempo (medido en Mbit/s, MB/s...)
 - Los dispositivos de almacenamiento (discos SATA) y red (Gigabit Ethernet) proporcionan un gran ancho de banda
- **Latencia:** tiempo requerido para la transferencia mínima (medido en segundos)
 - La latencia de todos los dispositivos de E/S es muy alta en comparación con la velocidad del procesador

Dispositivos	Ancho de banda
Sensores	1 B/s – 1 KB/s
Teclado	10 B/s
Pantalla (CRT)	2 KB/s
Impresora de línea	1 – 5 KB/s
Cinta (cartridge)	0.5 – 2 MB/s
Cinta	3-6 MB/s
Disco	70 – 600 MB/s
Tarjeta de red	128MB/s– 12.5 GB/s

⊠ Problema: ¿cómo diseñar un único sistema que trate dispositivos tan diferentes?

Componentes del subsistema de E/S



- **Dispositivo periférico**

- Dispositivo en sí, de carácter mecánico, magnético.... (teclado, platos del disco..)

- **Controlador del dispositivo (hardware)**

- Interfaz hardware a través del cual el dispositivo se comunica con el procesador
- Se *entiende* con el canal de comunicación y conoce las características físicas del dispositivo

- **Controlador software (*driver*)**

- Código encargado de programar el controlador HW del dispositivo concreto
- Forma parte del sistema operativo (suele proporcionarlo el fabricante)

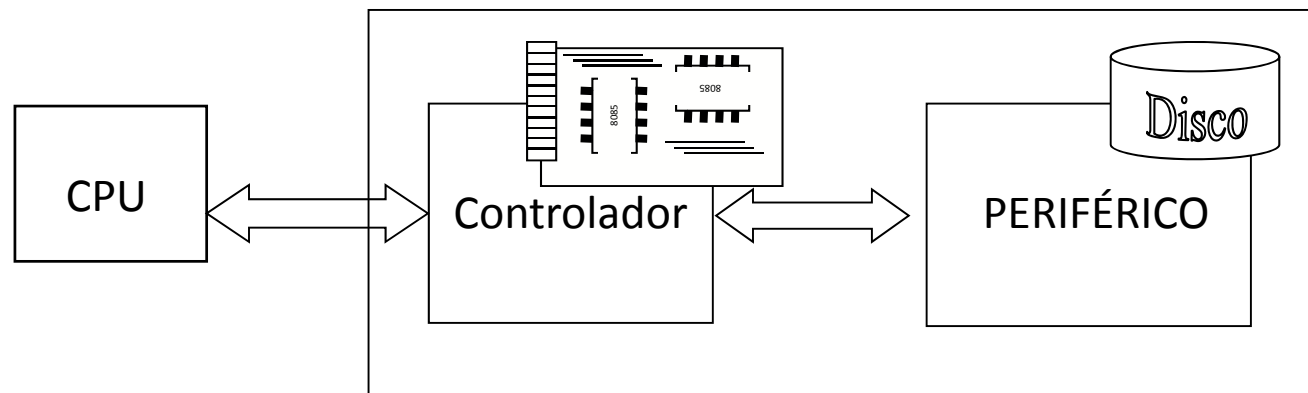
- **Canal de comunicación (p. ej. bus)**

- Interconexión entre la CPU, Memoria y el controlador HW del dispositivo
- Puede ser compartido por varios dispositivos
 - Necesidad de arbitraje

Controlador del dispositivo



- Los dispositivos periféricos se conectan al computador a través de un controlador de E/S
- Es un dispositivo HW que traduce peticiones de la CPU al periférico
 - Ej. CPU solicita el bloque 33. El controlador pide al disco que mueva el brazo a la pista 12 y lea el sector 313



Controlador del dispositivo



- **Registro de datos de entrada/salida**
 - Usados para el intercambio de datos entre CPU y periférico
- **Registro de estado**
 - Contiene el estado del dispositivo: preparado, situación de error, ocupado...
- **Registro de control**
 - Codifica la orden que la CPU da al dispositivo (imprime un carácter, lee un bloque...)
- **CUIDADO: NO son registros de la CPU. ¡¡¡Están (generalmente) en otro chip!!!**

Operación de E/S



- Entonces, ¿en qué consistirá una operación de E/S?
 - La CPU leerá/escribirá de/a los registros del controlador del periférico deseado
 - Pero, ¿cómo sabe *dónde* escribir?
 - De hecho.... ¿cómo se lee/escribe en esos registros?
 - ¿Cómo saber cuándo ha terminado la operación?

Funciones básicas del subsistema de E/S



1. **Identificación** única del dispositivo por parte de la CPU
 - ¿Cómo indicar el dispositivo de E/S deseado?
2. Capacidad de **envío y recepción de datos**
 - Tipos de transferencia
 - Lectura: computador ← periférico
 - Escritura: computador → periférico
 - ¿Cómo transferir datos? ¿Cómo comunicarnos con el periférico?
3. **Sincronización** de la transmisión
 - Exigida por la diferencia de velocidad de los dispositivos de E/S con la CPU
 - ¿Cómo sabemos si el dispositivo ya está preparado para otra transferencia?

Identificación del dispositivo: direccionamiento



- ¿Cómo indica la CPU a qué dispositivo quiere acceder?
 - Debe hacerlo a través de alguna instrucción de su repertorio (¡¡¡ejecutar instrucciones es lo único que sabe hacer la CPU!!!)
- Entonces ¿qué instrucciones se emplearán para programar el controlador HW del dispositivo elegido?

Direccionamiento



■ E/S aislada

- Espacio de direcciones **diferente** al de la memoria principal (existen dos direcciones 0x00000000)
- Existen instrucciones específicas de E/S
 - IN dir_E/S, Ri (CPU ← Periférico)
 - OUT Ri, dir_E/S (Periférico ← CPU)

■ E/S localizada en memoria

- La memoria y los dispositivos de E/S **comparten** el espacio de direcciones
- Podemos usar instrucciones tipo load/store (lw/sw)
 - LOAD Ri, dir_E/S (CPU ← Periférico)
 - STORE Ri, dir_E/S (Periférico ← CPU)

Direccionamiento: E/S Aislada



Direcciones

0x0000

0x0004

Memoria
principal
(SDRAM –
DDR)

0xFFFF

Direcciones

0x0000

0x0004

R. datos

R. control

R. estado

R. datos

R. control

R. estado

0xFFFF



- load R1, 0x0000 → Lee de memoria principal
- in R1, 0x0000 → Lee un dato del disco
- Ejemplos: Intel, AMD...

Direccionamiento: E/S localizada en memoria



Direcciones

0x0000
0x0004

Memoria
principal
(SDRAM –
DDR)

0xEFFF

Direcciones

0xF000
0xF004

R. datos

R. control

R. estado

R. datos

R. control

R. estado

0xFFFF



- load R1, 0x0000 → Lee de memoria principal
- load R1, 0xF000 → Lee un dato del disco
- Ejemplos: ARM, Motorola

Transmisión de datos



- Ya sabemos identificar el dispositivo... ¿cómo leemos/escribimos datos?
 - Indicar al dispositivo la operación que queremos hacer escribiendo en su *registro de control*
 - Leer/escribir de su *registro de datos*
- El código que interacciona a tan bajo nivel con el dispositivo es el conocido como **controlador SW (driver)**
 - Lo proporciona el fabricante del dispositivo



Sincronización

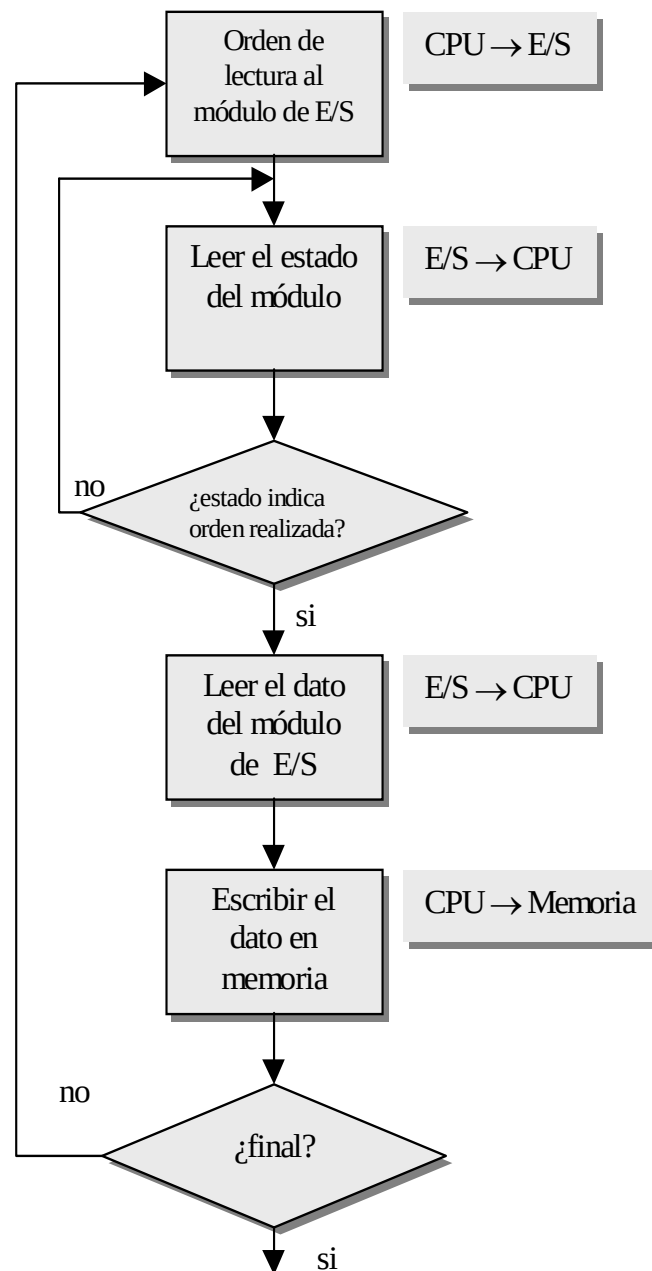
- Antes de enviar/recibir datos a/desde un periférico hay que asegurarse de que el dispositivo está preparado para realizar la transferencia: sincronización entre la CPU y el dispositivo de E/S
 - ¿Cómo sabe la CPU cuándo ha terminado la transferencia?
 - ¿Cómo sabe si todo ha ido bien?
- Existen dos mecanismos básicos de sincronización de la E/S
 - E/S programada con espera de respuesta (polling)
 - E/S por interrupciones

E/S programada (polling)



- Cada vez que la CPU quiere realizar una transferencia entra en un **bucle** en el que lee una y otra vez el **registro de estado** del periférico hasta que esté preparado para realizar la transferencia
- Problemas
 - La CPU no hace trabajo útil durante el bucle de espera
 - Con dispositivos lentos el bucle podría repetirse miles/millones de veces
 - La dinámica del programa se detiene durante la operación de E/S
 - Ejemplo: en un vídeo-juego no se puede detener la dinámica del juego a espera que el usuario puse una tecla
 - Dificultades para atender a varios periféricos
 - Mientras se espera a que un periférico esté listo para transmitir, no se puede atender a otro

E/S Programada





Ejercicio

- Determinar el porcentaje de tiempo dedicado por la CPU, usando E/S programada para atender a
 - Un ratón
 - Un disco
- Datos:
 - Número de ciclos que requiere la operación completa de E/S = 400 (duración total de bucle de muestreo)
 - Frecuencia de procesador = 500 MHz.
 - El **ratón** debe leerse 30 veces/s. para que no se pierda ningún movimiento.
 - El **disco** transfiere datos en bloques de 4 palabras, y puede transferir a una velocidad de 4MB/seg. No debe perderse ningún dato.

¿Hay margen de mejora?



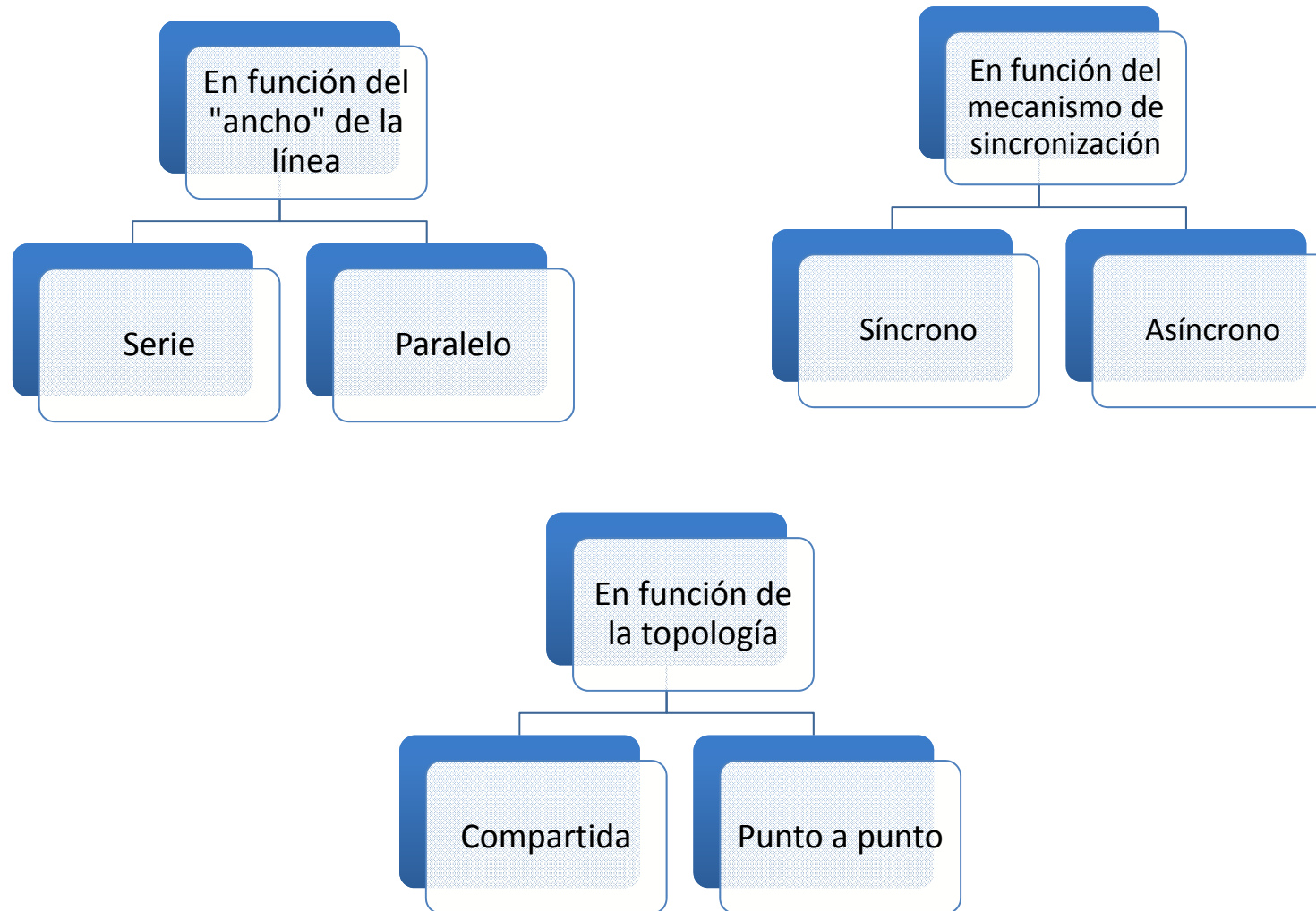
- ¿Y si el dispositivo avisase a la CPU cuando haya terminado su operación?
 - La CPU podría hacer trabajo útil mientras el periférico hace la operación solicitada
 - Sólo tendría que retomar el control de la operación cuando el periférico hubiese terminado
- Este mecanismo se denomina E/S por interrupciones
 - Pero eso es ya otra historia...

¿Cómo se comunican los distintos módulos?



- ¿Cómo llegan las órdenes desde la CPU hasta los controladores?
- ¿Cómo llegan los datos de los periféricos a la CPU (y viceversa)?
- Aspectos básicos de los canales de comunicación
 - La información se transmite por *cables*
 - Es necesario sincronizar emisor y receptor
 - El canal de comunicación puede ser compartido

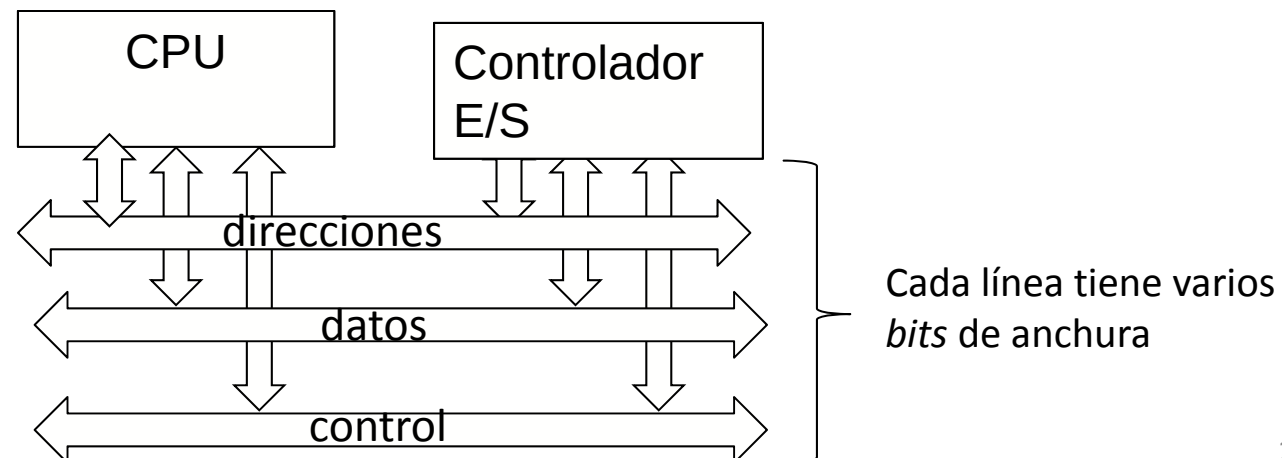
Taxonomía de líneas de comunicación





Transmisión paralela

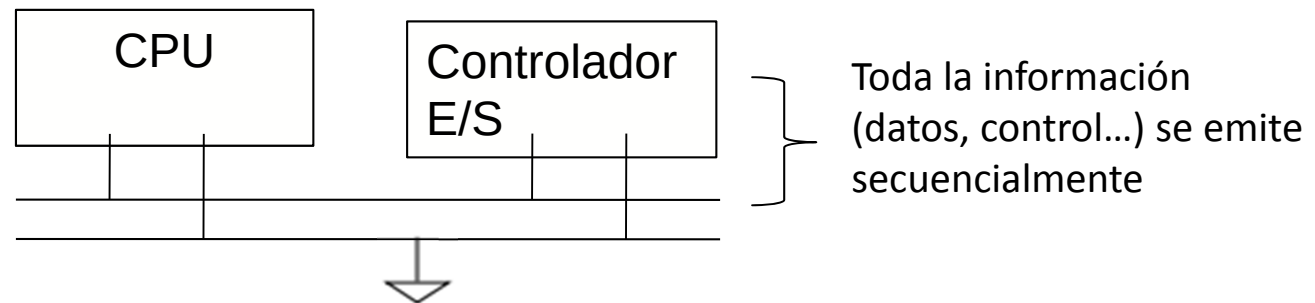
- Utiliza varias líneas de comunicación (cables) a través de las cual se se envía varios **bits de información de forma simultánea**
 - ✓ Ancho de banda elevado (en teoría...)
 - ✗ Para conectar dispositivos a distancias medias o largas resulta muy costosa
 - ✗ Frecuencia de funcionamiento limitada por factores físicos
 - ✗ Los dispositivos de baja velocidad no aprovechan el potencial de la transmisión paralela (ratón, módem...)





Transmisión serie

- Utiliza una única línea de comunicación (2 cables: dato y tierra) a través de la cual se envían los bits de información de **forma secuencial**
 - ✓ Es menos costosa que la E/S paralela
 - ✓ Permite frecuencias de funcionamiento mayores
 - ✗ A igualdad de frecuencia que el caso paralelo, el ancho de banda es menor



- Es posible aumentar el ancho de banda entre dos dispositivos usando varias conexiones serie

Comunicación síncrona vs asíncrona



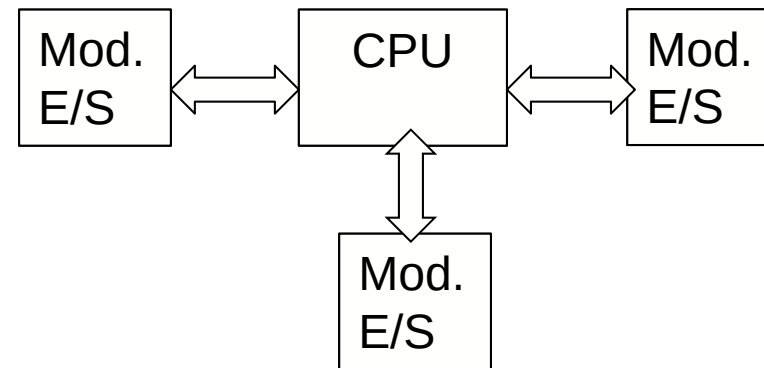
- ¿Cuándo empieza/acaba el envío de un dato?
- Asíncrona
 - La señal de reloj NO se envía por la línea de comunicación
 - Emisor y receptor utilizan sus propias señales de reloj.
 - Es necesario establecer un **protocolo** para la sincronización entre ambos
 - ✓ No es imprescindible que emisor y receptor funcionen a la misma frecuencia
- Síncrona
 - La señal de reloj viaja con el resto de señales
 - ✓ Más sencillo y, en general, más eficiente
 - ✗ Exige que emisor y receptor funcionen a la misma frecuencia

Punto a punto vs. Compartida



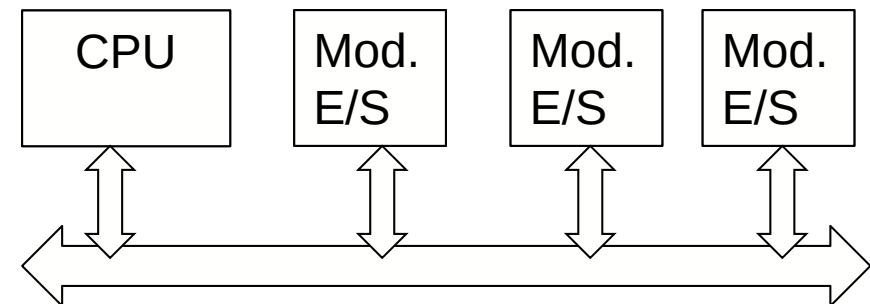
■ Comunicación Punto a punto

- ✓ Gran rendimiento
- ✗ Exige un gran número de interfaces



■ Compartida

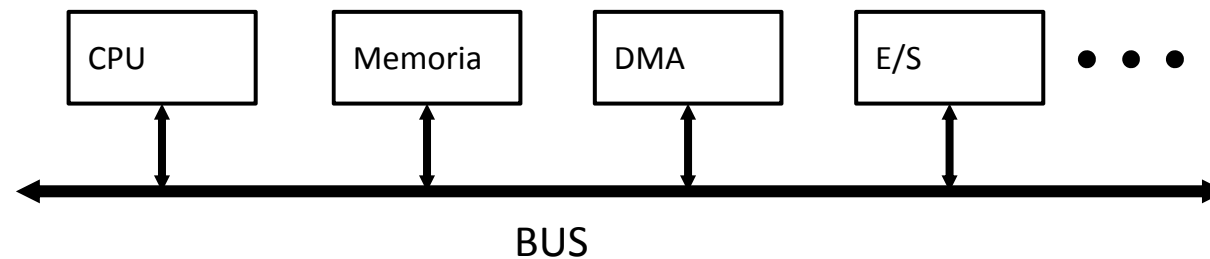
- ✓ Más versátil y barato
- ✗ Cuello de botella en la comunicación
- ✗ Exige sincronización entre dispositivos para evitar escrituras simultáneas



Comunicación tradicional en el computador: buses



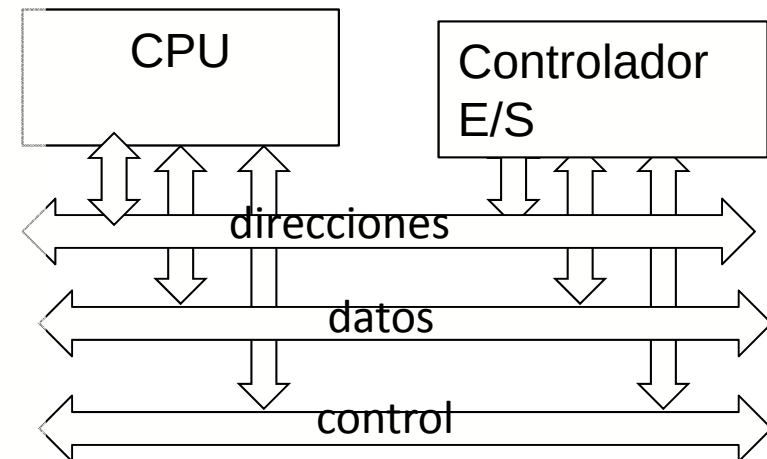
- Conjunto de *cables* que conectan múltiples componentes:
 - Paralelo
 - Compartido
 - Es necesario **arbitrar** el uso del bus
 - Hay versiones síncronas y asíncronas



Organización general de un Bus



- Líneas de control
 - Utilizadas para implementar el protocolo de comunicación entre componentes
- Líneas de datos
 - Transmite la información entre emisor y receptor
- Líneas de dirección
 - Contienen la información de direccionamiento
 - Habitualmente, coinciden físicamente con las de datos





Master versus Slave

- Una operación de entrada/salida se divide en:
 - *Petición*: enviar el comando (y dirección).
 - *Envío*: transferencia de los datos
- El maestro (*Master*) es quien inicia la transacción
 - Realiza la petición por la línea de control
 - La CPU siempre es *Master*
- El esclavo (*Slave*) responde a la petición
 - Comprueba que la dirección le corresponda a él
 - Envía/recibe datos del maestro
 - En general, cualquier dispositivo de E/S se comporta únicamente como *Slave*

(Más) conceptos básicos de buses



■ Tipos básicos de transferencia:

- Escritura: Master $\xrightarrow{\text{DATO}}$ Slave
- Lectura: Slave $\xrightarrow{\text{DATO}}$ Master

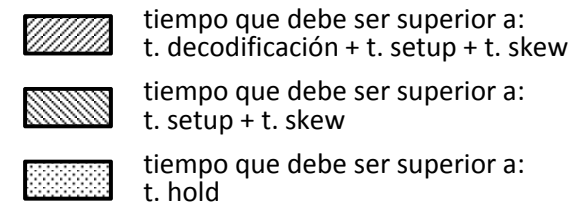
■ Fases de una transferencia

- 1. Direccionamiento (identificación) del *slave*
- 2. Especificación del tipo de operación (lectura o escritura)
- 3. Transferencia del dato
- 4. Finalización del ciclo de bus

■ Control de la transferencia:

- Sincronización: determinar el inicio y el final de cada transferencia
- Arbitraje: controlar del acceso al bus en caso de varios masters

-
- Diagrama de un bus de datos de 4 canales. Se muestran dos dispositivos, Master y Slave, conectados a cuatro líneas de bus: Dirección, Datos, R/W* y Reloj. Las flechas indican la dirección del flujo de información en cada línea.



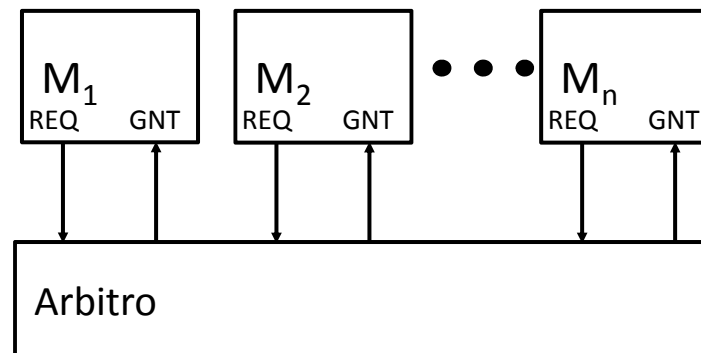
Conflictos en el acceso: arbitraje



- Si puede haber varios emisores (*Master*), es necesario que el acceso al bus sea *libre de conflictos*
 - Se debe evitar que dos módulos escriban simultáneamente en el bus
 - Cada dispositivo solicita permiso para poder tomar el control del bus

Ejemplo de arbitraje

- Cada master se conecta al árbitro mediante dos líneas individuales:
 - BUS REQUEST (REQ): línea de petición del bus
 - BUS GRANT (GNT): línea de concesión del bus
- Varias peticiones de bus pendientes: el árbitro puede aplicar distintos algoritmos de decisión
 - FIFO
 - Prioridad fija
 - Prioridad variable

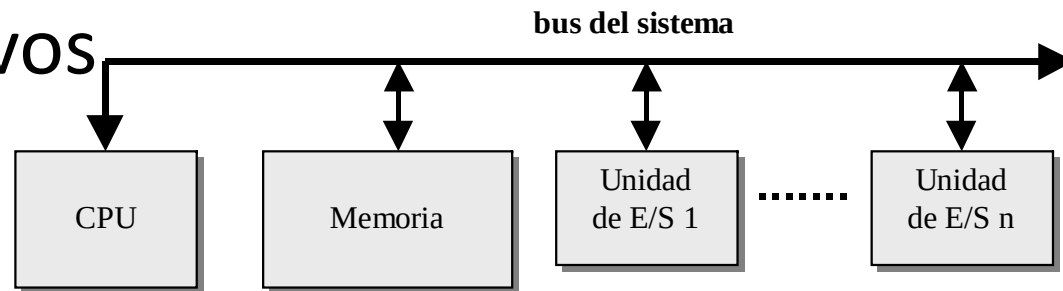


Limitaciones de la aproximación bus

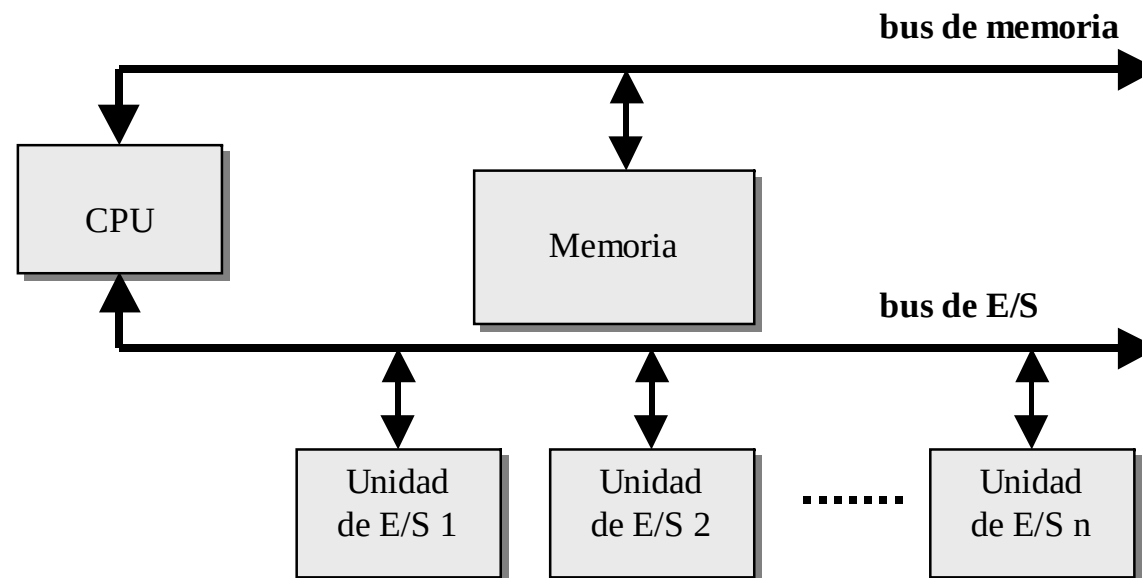


- Alternativa 1: un solo bus para memoria y todos dispositivos

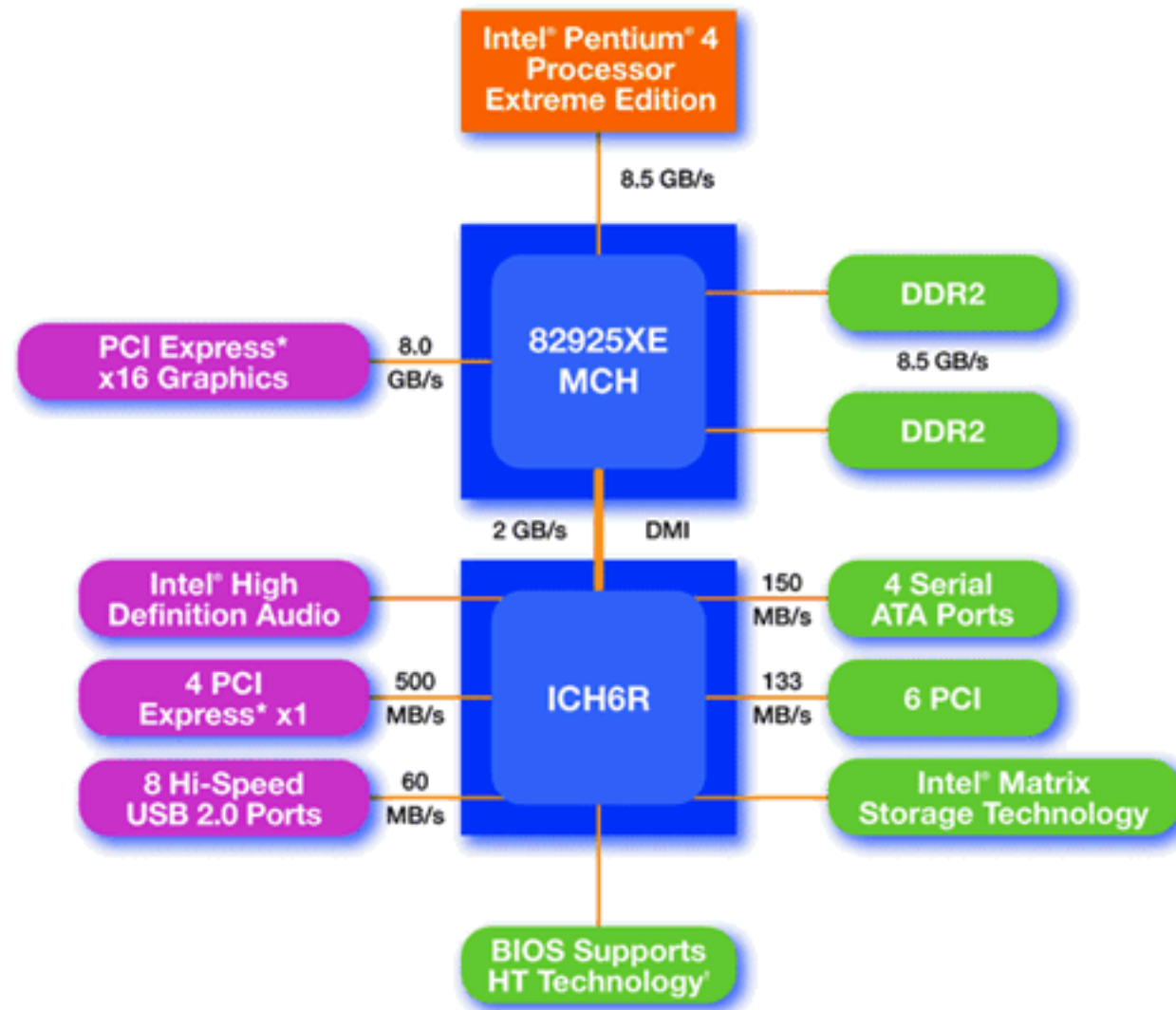
✗ ¿Problemas?



- Alternativa 2: jerarquía de buses



Chipsets de Intel. 2004





¿Y ahora qué? ¡¡¡E/S Serie!!!

- Los buses paralelos presentan limitaciones físicas:
 - ✗ A alta frecuencia, cada línea del bus tendrá diferencias de retardo significativas
 - ✗ La frecuencia de funcionamiento está muy limitada
 - ✗ El arbitraje introduce más latencia
 - ✗ Alto consumo energético
 - Ejemplos: PCI, SCSI, IDE, VMEbus
- Con la tecnología actual, resulta más ventajoso construir canales más estrechos pero de mayor velocidad
 - **Comunicación serie, síncrona y punto a punto**
- El cambio de paradigma proporciona más flexibilidad:
 - ✓ Alto rendimiento en conexiones punto a punto
 - ✓ Permite múltiples transferencias en paralelo
 - ✓ Permite sintonizar el ancho de banda necesario para cada dispositivo con varios canales serie en paralelo
 - ✓ Bajo consumo energético
 - PCI Express, Infiniband, Serial ATA (SATA), USB

Chipsets de Intel. 2008

