

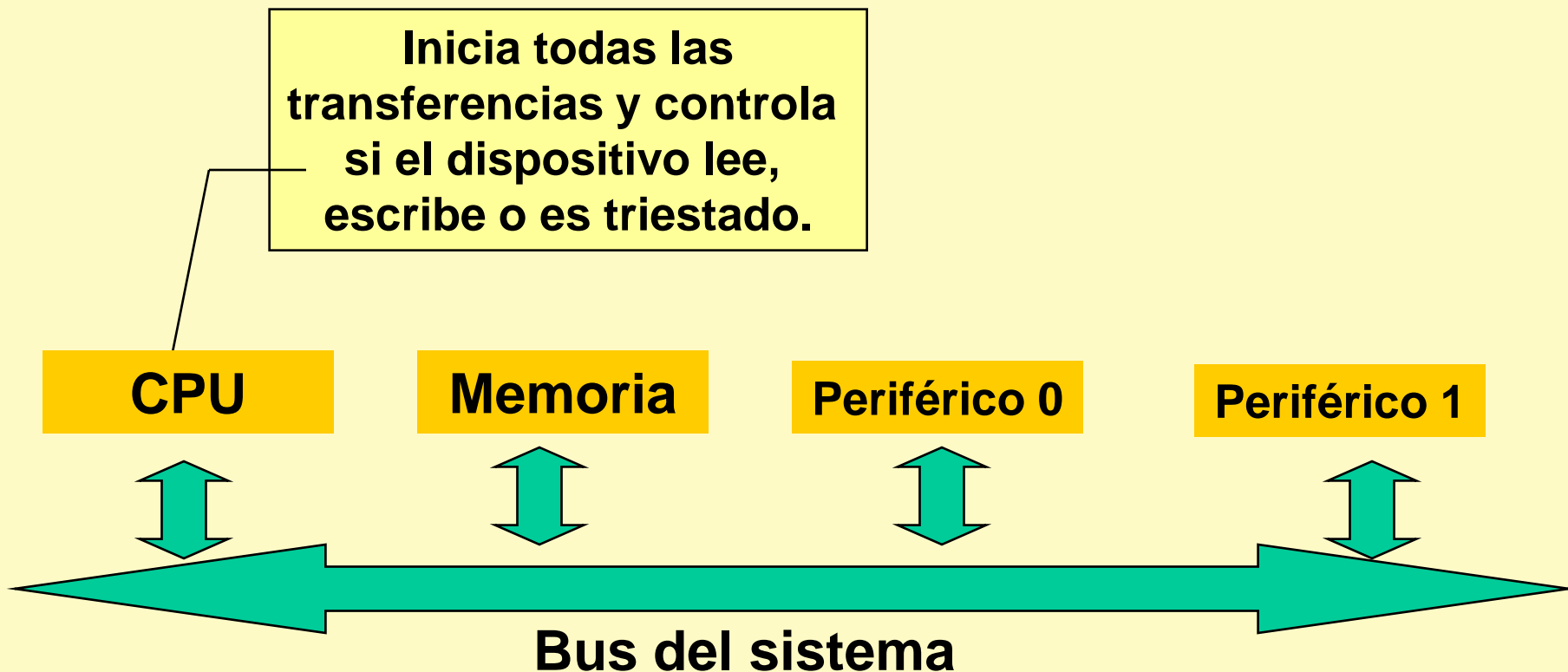
TEMA : BUSES

- 1. Estructuras del Bus**
- 2. Tipos de Bus**
- 3. Especificación de un Bus**
- 4. Paralelismo y multiplexación**
- 5. Arbitraje**
- 6. Tipos de transferencia**
- 7. Temporización y direccionamiento**
- 8. Ejemplos de Buses. PCI**



→ Estructuras del Bus

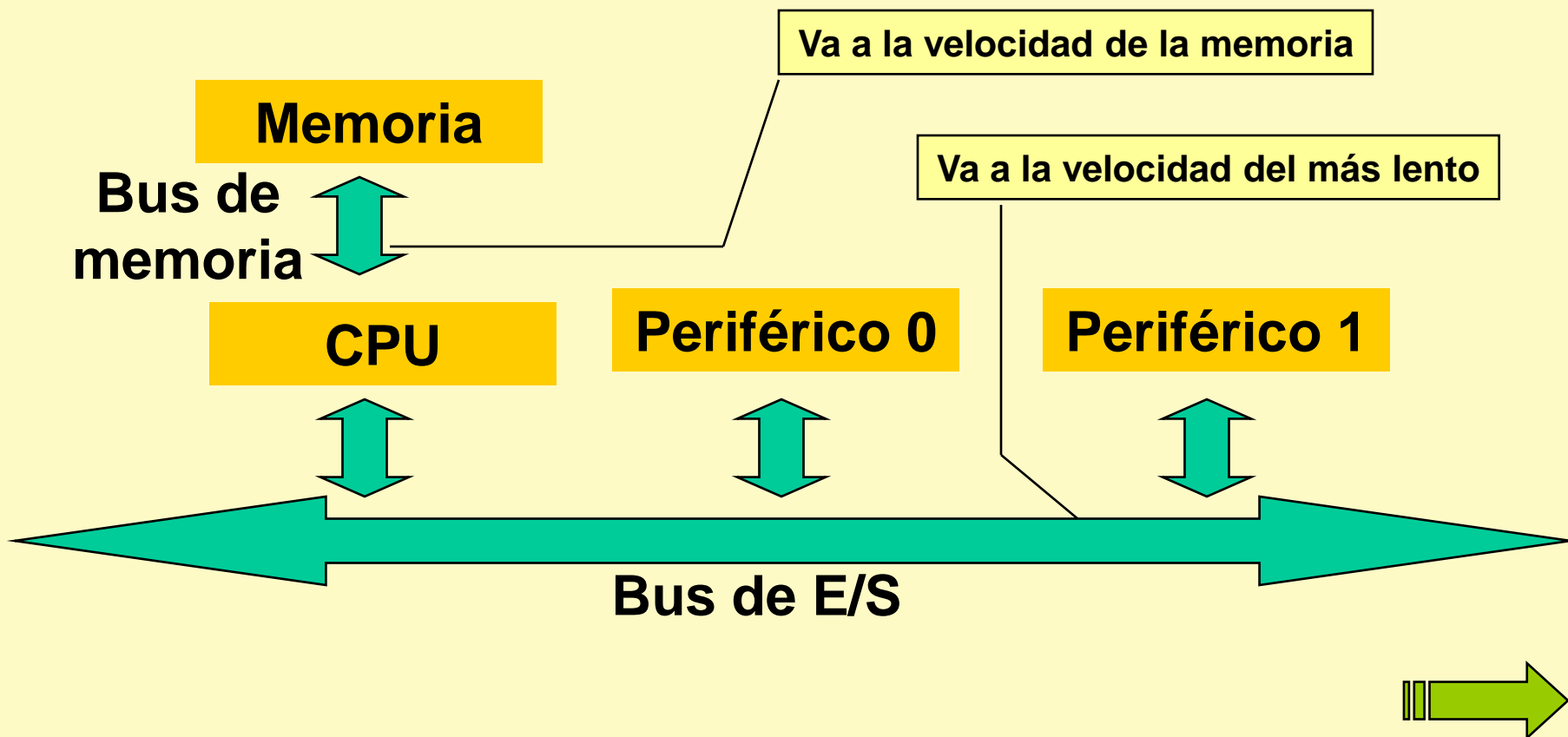
1. Bus único.
2. Buses separados.
3. Bus único avanzado.
4. Buses separados avanzados.
5. Estructura en estrella.
6. Estructura en anillo.
7. Bus único multiplexado en tiempo.



Problema: La velocidad de transferencia de la CPU y de la memoria es muy superior a la de los periféricos. Por ello si el bus es síncrono todo ira regido por la velocidad del más lento

Buses separados:

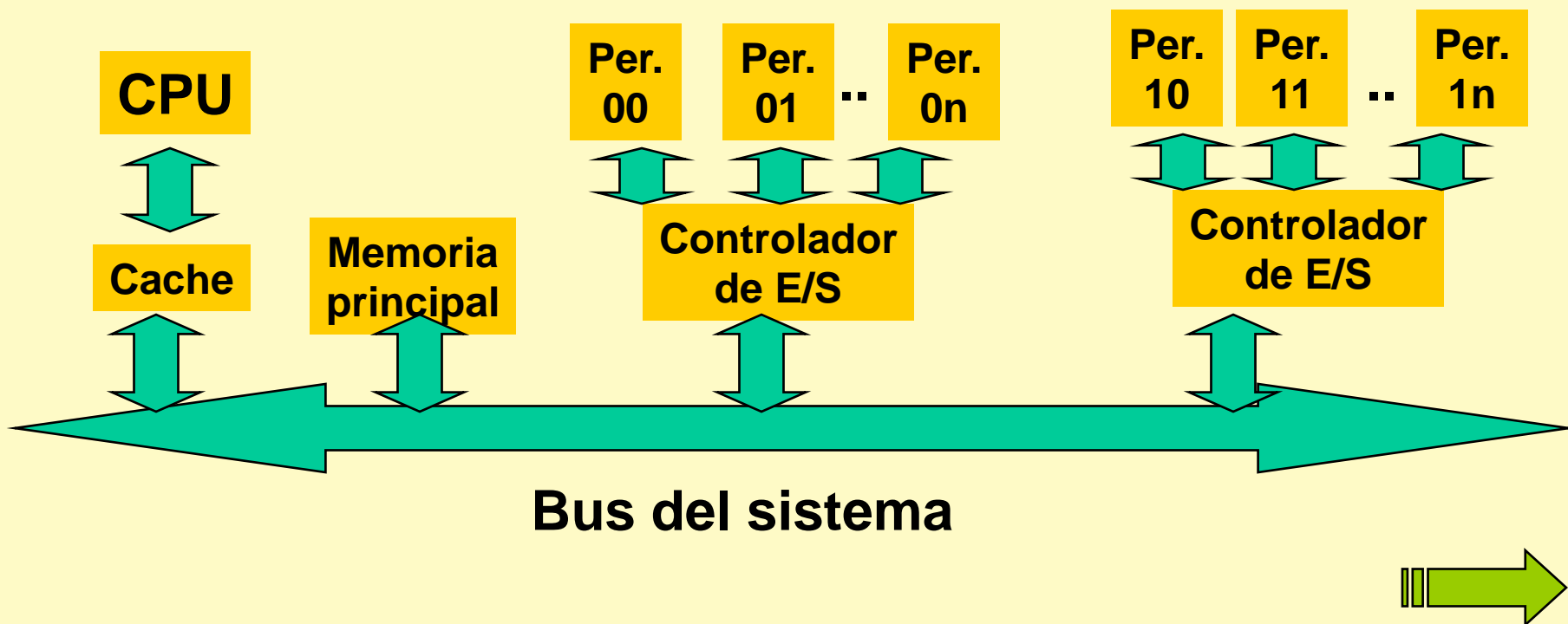
- Uno rápido (memoria) y otro lento (E/S).







Bus único avanzado:

-Objetivo: que la CPU pueda seguir trabajando con memoria mientras que los periféricos terminan su operación.



Controlador E/S:

- CPU **programa** el controlador indicando 
 - tipo de operación (R/W)
 - periférico (0..n)
 - tamaño del bloque dado
- CPU **sigue trabajando con memoria.**
- El **controlador interrumpe** a la CPU si 
 - periférico no está preparado
 - pide (W) el primer dato
 - proporciona (R) el primer dato
- CPU lee/escrive el dato.
- CPU sigue trabajando **hasta la próxima interrupción.**



Buffers:

- El controlador puede disponer de un **pequeño bloque de memoria** propia (1KB).
- La CPU puede escribir/leer datos de E/S de 1KB en 1KB por lo que se interrumpe menos frecuentemente (1024 veces menos).

Es más rápida transfiriendo 1Kb que 1024 veces 1B.

- Los **periféricos pueden tener buffer propio**; esto ocasiona que el controlador quede libre para manejar otro periférico mientras tanto.
- La CPU también puede tener Buffer propio (Caché).



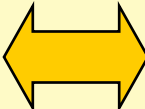


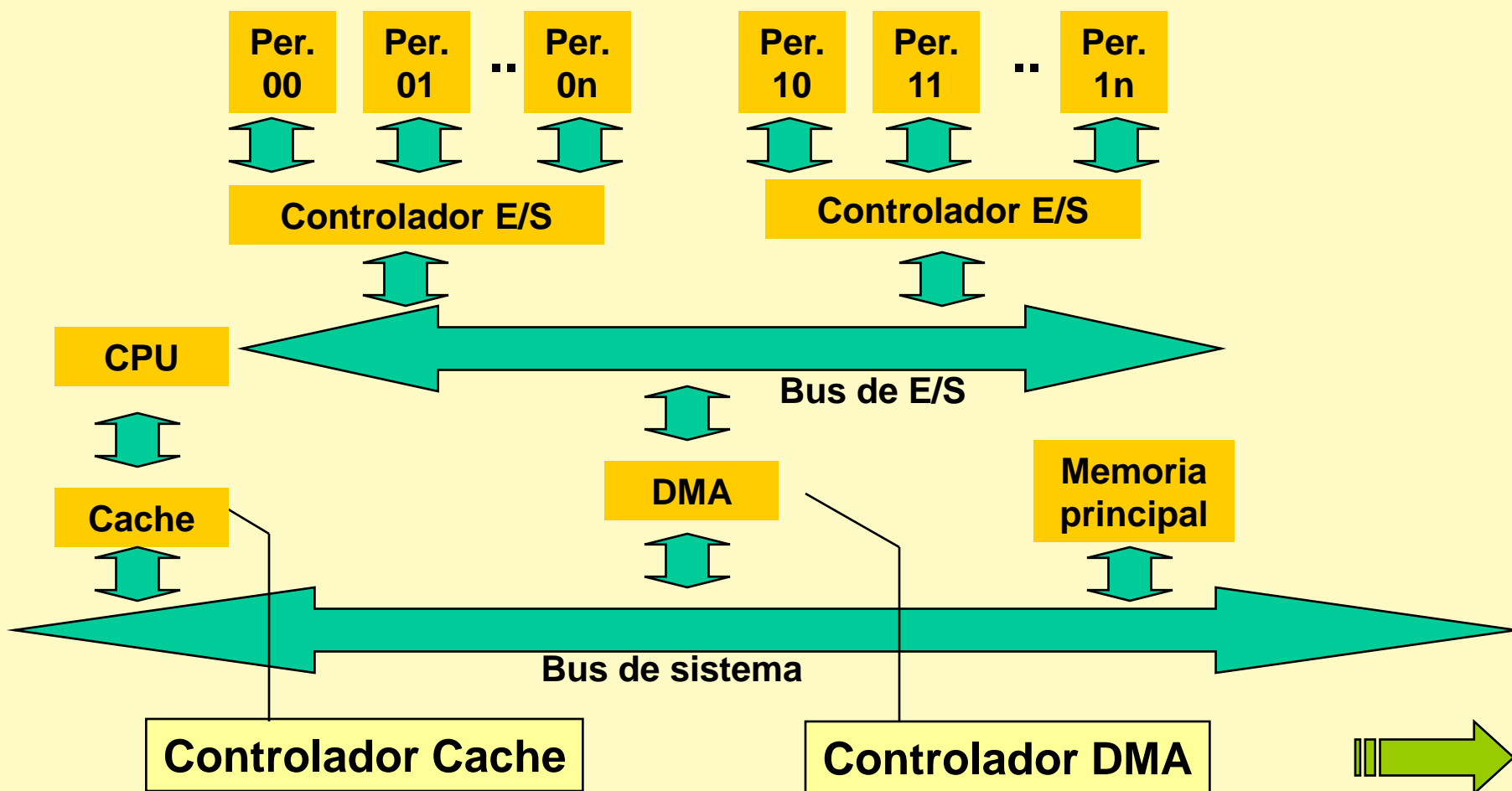
Interrupciones:

- Son un mecanismo (hardware, línea INTR) por el cual la CPU:

- **Memoriza el estado actual o CONTEXTO:** contador de programa, flags de estado, otros registros...
- **Ejecuta la Rutina de Servicio de Interrupción ISR** la cual debe:
 - 1.- **Identificar el dispositivo:**
 - IRQ vectorizadas: el Controlador pone un vector en el bus que identifica el periférico y el ISR
 - consulta (polling)
 - ISR comprueba estado de los periféricos hasta localizar el causante de la IRQ
 - 2.- **Atenderlo (R/W).**
 - 3.- **Atender el Controlador, ¿(re)(des)programar?**
- **Retorna al contexto anterior IRET**

Buses Separados:

Objetivo: liberar a la CPU de tráfico E/S  MEM.





Controlador DMA (Direct Memory Access):

1.- CPU **programa** al controlador indicándole:

Tipo de operación R/W

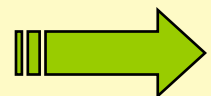
Tamaño de bloque de memoria

Dirección inicial de memoria

2.- El Controlador DMA aprovecha cuando el **Bus de Sistema queda libre**:

- Lee de memoria y escribe el dato en E/S o escribe en memoria el dato leído en el BUS E/S.
- Incrementa la dirección y así sucesivamente.

3.- El Controlador E/S debe haber sido programado para actuar conjuntamente.



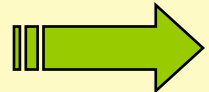


Controlador Cache:

- **Principio de localidad de las referencias de memoria:**
 - **Temporal:** tendencia de la CPU a referenciar dentro de poco datos recientemente referenciados.
 - **Espacial:** tendencia de la CPU a referenciar datos cercanos (dirección) al último referenciado.
 - **Secuencial:** tendencia de la CPU a referenciar el siguiente dato (dirección+1) al último referenciado.
- **Controlador intenta que la CPU encuentre en Caché lo que necesita.**
- **Transfiere Memoria ↔ Caché en pequeños bloques.**
- **CPU / Cache=E/S / Buffer**
- **La Cache es más rápida, cara y pequeña que la memoria normal.**

Otras necesidades de velocidad:

- Deseo de reducir el tiempo de ejecución y el coste de los procesadores. Se toman estas medidas:
 - Varios procesadores iguales: repartir tiempo, tolerancia de fallos...
 - Coprocesadores específicos: gráficos, matemáticos...
 - Controladores Cache, DMA, E/S...
 - Todo esto trabajando simultáneamente en paralelo.
- Es viable sólo si:
 - Cada uno puede trabajar bastante tiempo aisladamente.
 - Acceden ocasionalmente al bus para transferir ráfagas de datos.



Dos estructuras típicas



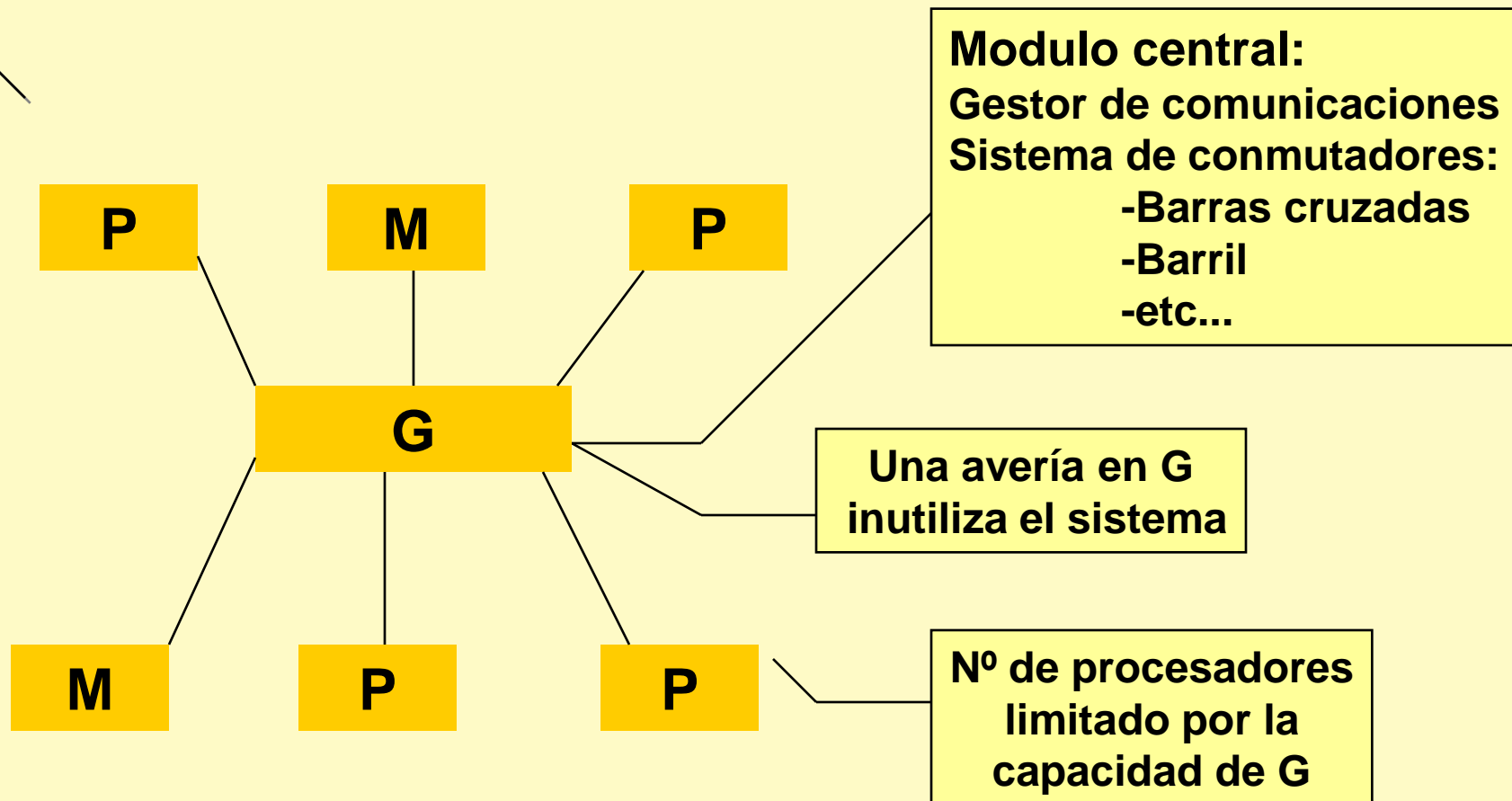
Estructura en estrella

Estructura en anillo



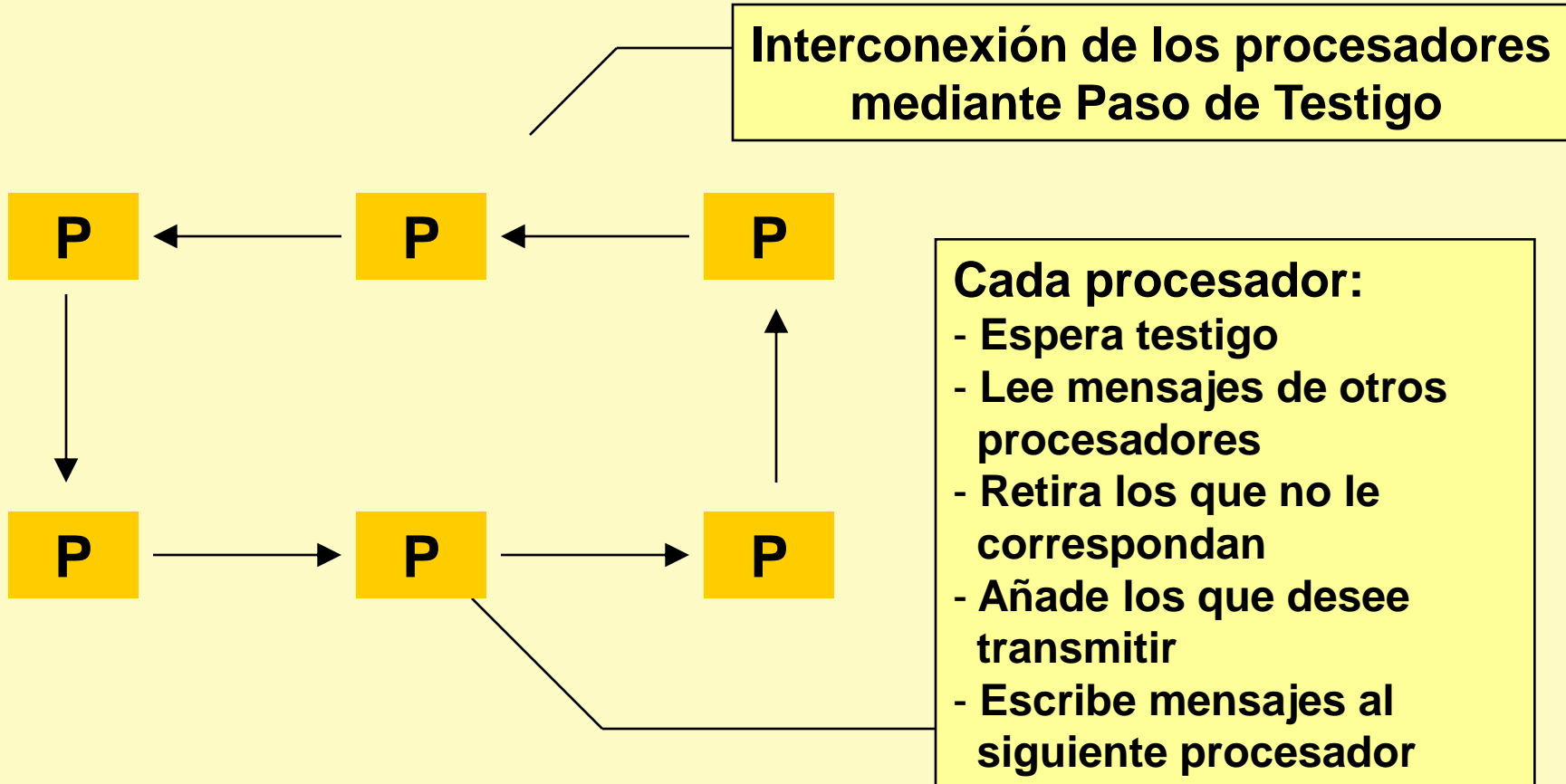
Estructura en estrella:

Interconexión Procesadores-Memoria compartida





Estructura en anillo:



Ventajas: +Barato (2 hilos, coaxial, altas velocidades)

+Sencillo, ampliable (insertar otro procesador en cadena)

Desventajas: La avería de un procesador inutiliza el sistema.



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus

→ 2. Tipos de Bus

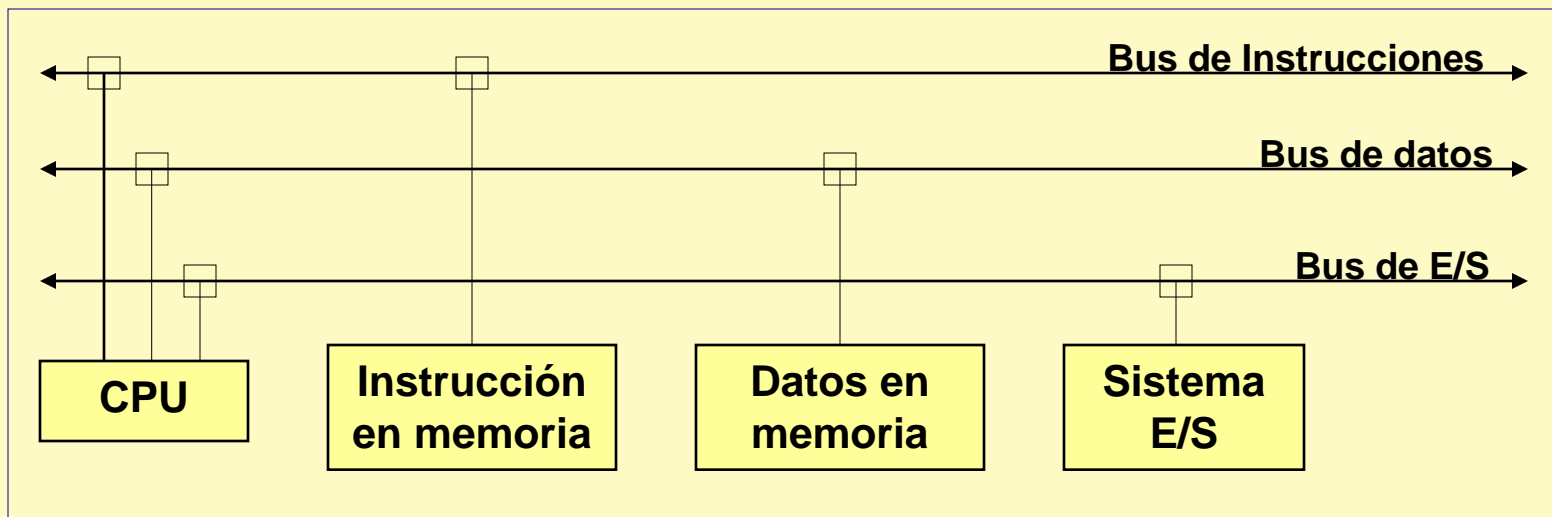
2.1 Según dedicación (dedicados/no dedic.)

2.2. Según particionamiento

2.3. Según nivel de jerarquía

Buses dedicados

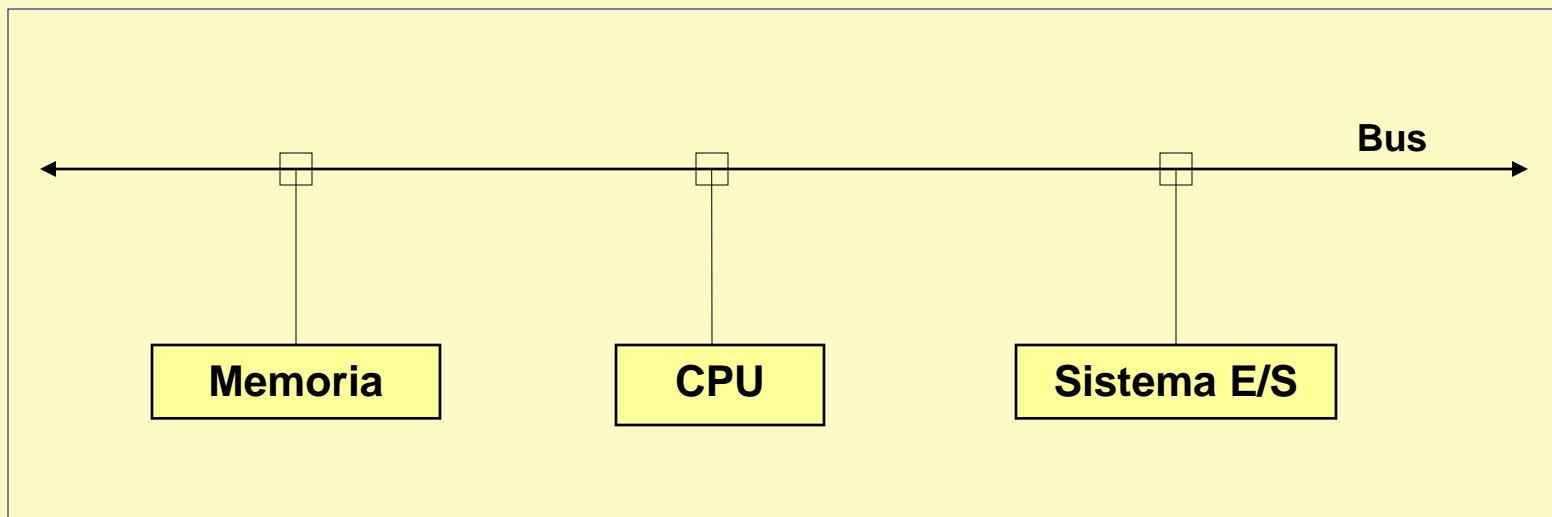
- Pueden establecer **más comunicaciones** simultáneamente.
- Pueden realizar **más funciones** en paralelo (E/S, Datos, ...)
- Pero el **coste** es **mucho mayor** (nº contactos y cables)
- Sólo interesa si hay muchos elementos (CPU, Mem) muy rápidos y muy caros (más que el bus).



Estructura típica de un **bus dedicado**

Buses no dedicados

- Son más baratos y más lentos:
 - Sólo hay 1 comunicación simultáneamente.
- Requieren mecanismo de arbitraje:
 - Conceder el bus a quien lo necesite pero no a más de 1.
- Si hay muchos elementos, **bus = cuello de botella**.



Estructura típica de un **bus no dedicado**



Particionamiento:

- POR RECURSOS
- POR FUNCIÓN

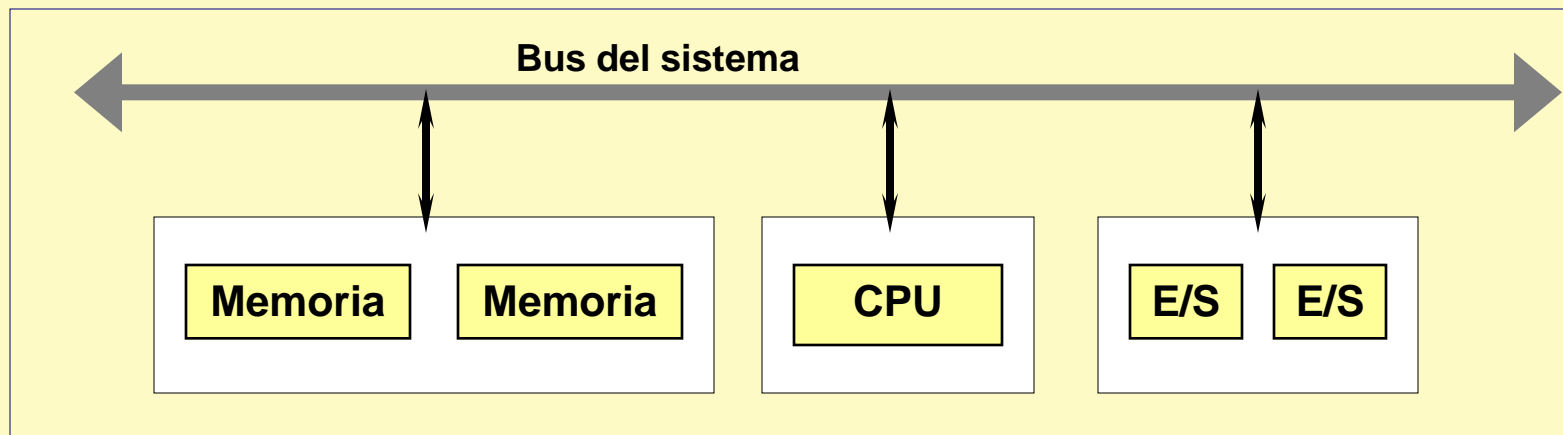
Por recursos: (Ej. VME):

Recursos del mismo tipo se agrupan en tarjetas.

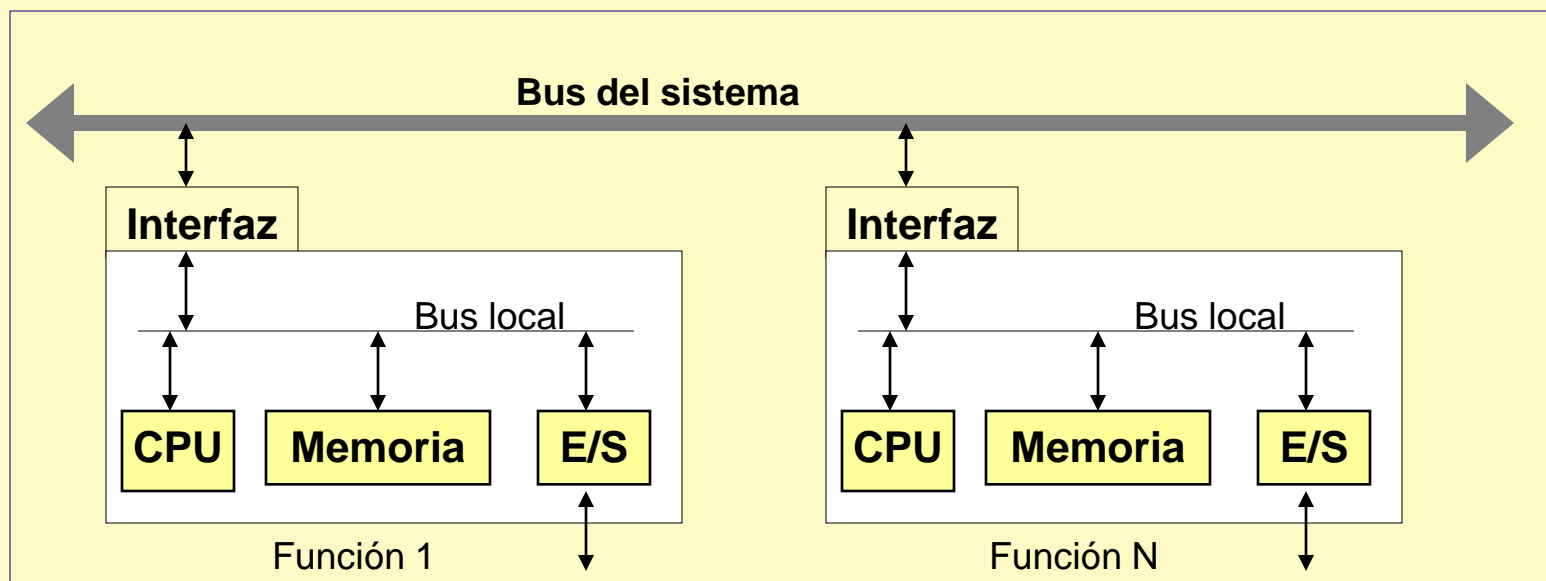
Particionamiento por función

1. Cada tarjeta es un sistema (casi) independiente.
2. Sólo requieren ocasionalmente pasarse datos o sincronizarse.

Ej. Multibus II y Futurebus.



Bus particionado **por recursos**



Bus particionado **por función**



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus

2. Tipos de Bus

→ **3. Especificación de un Bus**

4. Paralelismo y multiplexación

5. Arbitraje

6. Tipos de transferencia

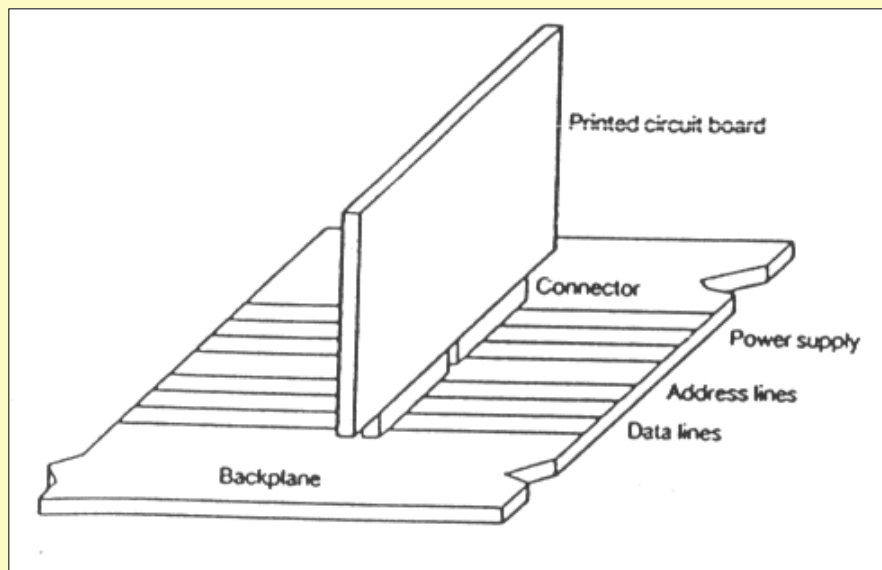
7. Temporización y direccionamiento

8. Ejemplos de Buses. PCI

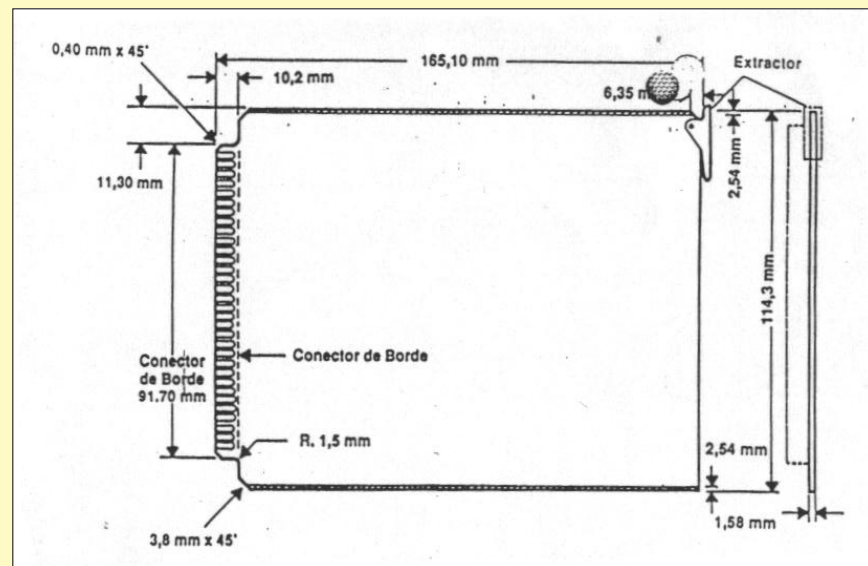
Debe incluir toda la información necesaria para saber conectar un dispositivo al bus. Hay varios niveles:

- **Mecánico:** Define el soporte (rack, PCB), n° líneas, tipo conector, dimensiones tarjetas,...
- **Eléctrico:** Tipo transceivers (tensión/corriente), alimentación, impedancias, nivel señal,...
- **Lógico:** Define n° señales y su significado, activas alta/baja, ...
- **Temporización:** Tipos de ciclo (R/W, IO/M, Tw) y sus cronogramas.
- **Transferencia simple:** Define protocolos arbitraje, transmisión, ciclo partido, detección errores.
- **Transferencia doble:** Posibilidad transferencia bloque, mecanismos de reintento/recuperación bloque.

Nivel Mecánico:



Backplane Bus

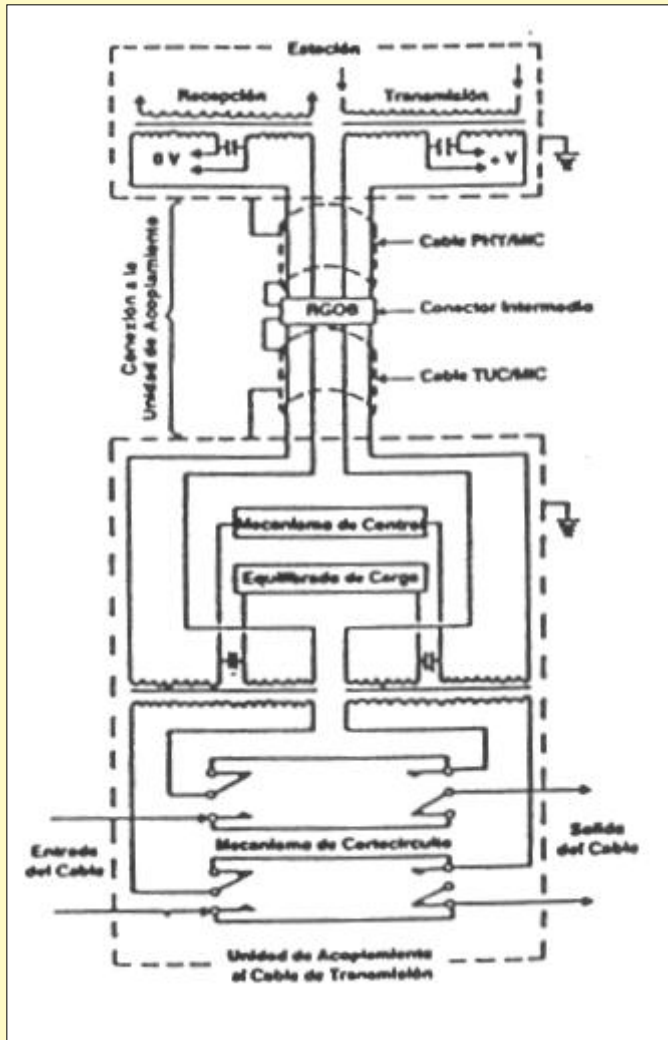


Placa STD

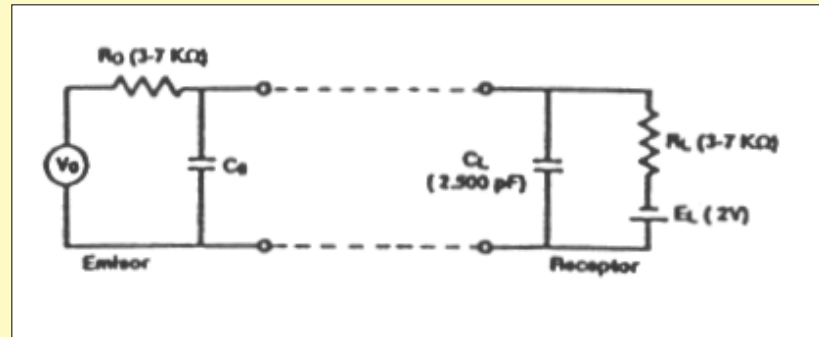
A veces el estándar de fábrica (Ej. RS-232-C) no especifica conector (DB-25 o DB-9)



Nivel Eléctrico:



Circuito equivalente RS-232-C



Estrechamente relacionado con nivel mecánico.

Ambos niveles influyen en:

Distancia máxima conexión / n° tarjetas.

Inmunidad al ruido

Velocidad Máxima de transferencia.

Paralelismo / Multiplexación

Coste (cables, conectores, ...)



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus
2. Tipos de Bus
3. Especificación de un Bus
- ➔ **4. Paralelismo y multiplexación**
5. Arbitraje
6. Tipos de transferencia
7. Temporización y direccionamiento
8. Ejemplos de Buses. PCI



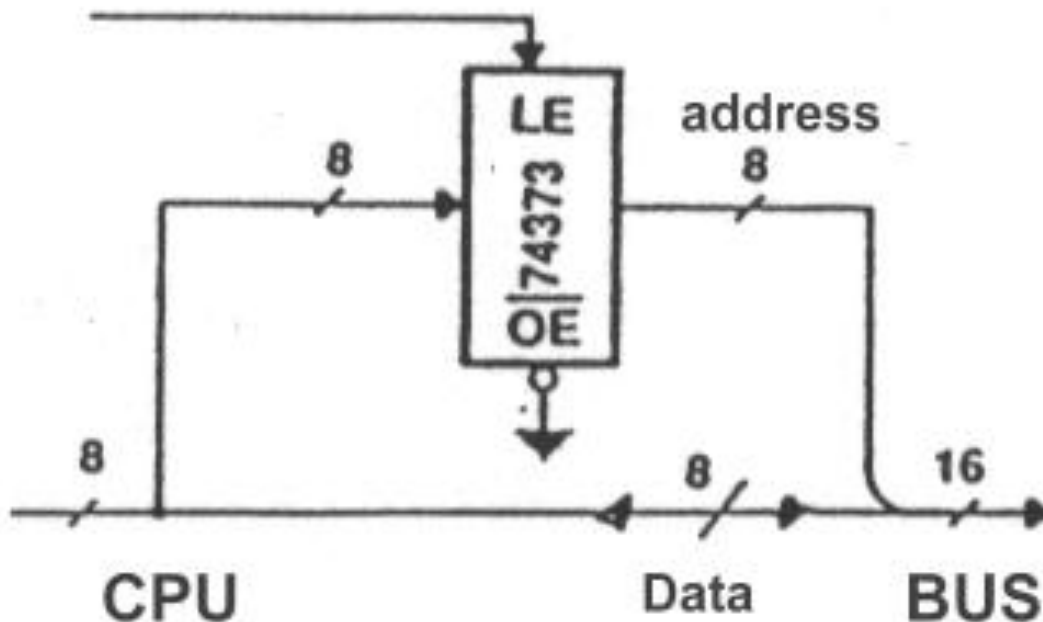
- Idealmente, todos los dispositivos debieran tener el mismo ancho palabra del bus.
- **Paralelismo total:** Bus datos y Bus direcciones separados.
Aumenta la velocidad de transferencia.
Aumenta el coste.
- Sin embargo interesa disminuir el n^0 de pines en dispositivos, complejidad conectores, tamaño tarjetas, n^0 hilos, coste, etc.



Multiplexación Datos/Direcciones:

- El bus lleva datos/dir separados. Dispositivos cargan dir en “latch”

Demultiplexación



No es tan grave la pérdida de velocidad, ya que en una lectura cpu-mem la cpu debe esperar tiempo de acceso.

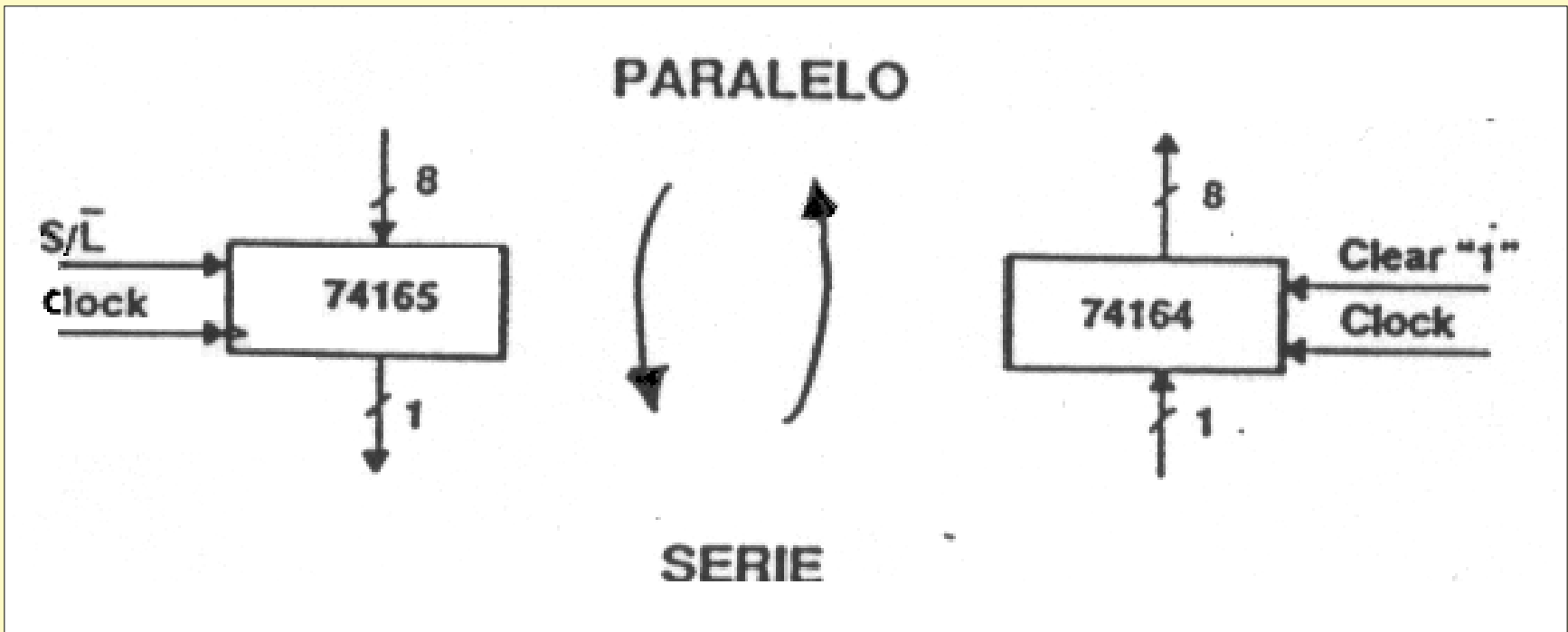


Multiplexación



Bus Serie:

- Caso extremo de multiplexación.
- Paralelismo Nulo. Toda la información (Datos, Dir, Control) va por 2 hilos.
- Muy apropiado para grandes distancias (bajo coste).



Elementos necesarios para bus serie.



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus
2. Tipos de Bus
3. Especificación de un Bus
4. Paralelismo y multiplexación

→ 5. Arbitraje

6. Tipos de transferencia
7. Temporización y direccionamiento
8. Ejemplos de Buses. PCI



Tipos de dispositivo (o Tarjetas):

- **Activos:** Eventualmente pueden requerir el uso del bus para iniciar una transferencia.
- **Pasivos:** Sólo pueden responder a una transferencia, nunca iniciarla.
- **Master:** En cada instante el bus está siendo usado por él.
- **Slave:** Dispositivo(s) que responde(n) al master.

Pasivo → Esclavo siempre

Activo ← Master ocasionalmente

Con Bus Único { Situación de Contención del bus.
Activos deben acordar quién usa el bus, si no → cortocircuito.

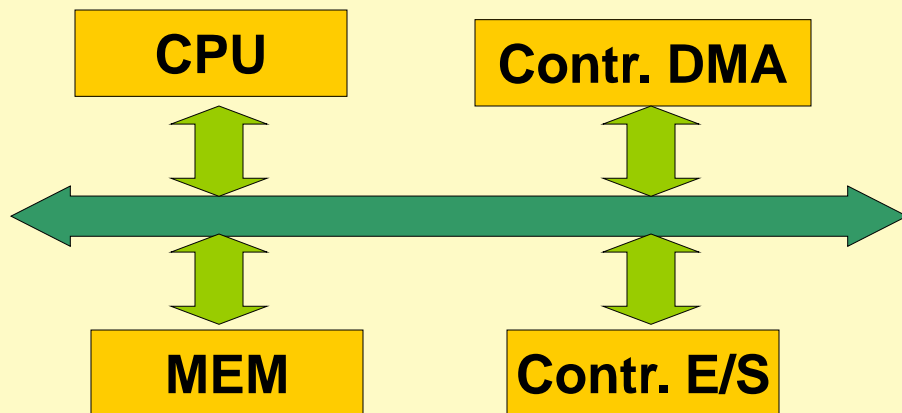


Tipos de bus único:

- **Maestro único:** (CPU) Controla bus siempre. No hay contención.
- **Por robo de ciclo:** (DMA)
 - Hay un elemento activo que usualmente es maestro (CPU).
 - Hay otro activo (DMA) que ocasionalmente pide el bus por muy poco tiempo (HOLD-HLDA).
 - No hay arbitraje: CPU siempre cede el bus y lo recupera pronto.
- **Arbitraje:**
 - Varios **elementos activos** (Ej. Multiproceso simétrico).
 - Requiere **protocolo justo** (no dejar a ninguno esperando eternamente).



Ej. Bus único, robo ciclo DMA, CPU, Mem, E/S.



Master: DMA	Fuente: E/S
Slave: MEM E/S	Destino: MEM

CPU recupera bus y lee de memoria.

Master: CPU	Fuente: MEM
Slave: MEM	Destino: CPU

Activos: CPU, DMA

Pasivos: Mem, E/S (E/S solo puede interrumpir)

Master: Usualmente CPU

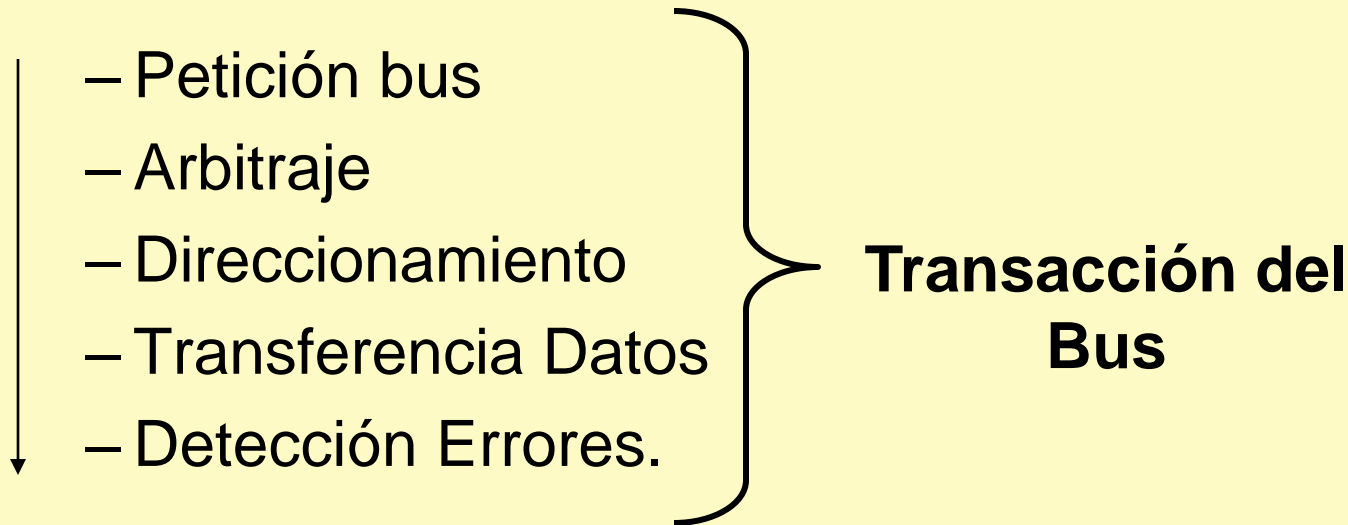
Slave: Usualmente MEM

Controlador E/S dispone de 1 buffer de datos para CPU; interrumpe CPU

Controlador DMA roba bus (HOLD) y transfiere buffer E/S a MEM.



En el caso más general (arbitraje) la secuencia de operaciones se llama **transacción**.





Detección de Errores:

Error: Se transmiten datos incorrectos o se pierden datos.

Tipos Comunes (deben detectarse para poder reintentar la transferencia)

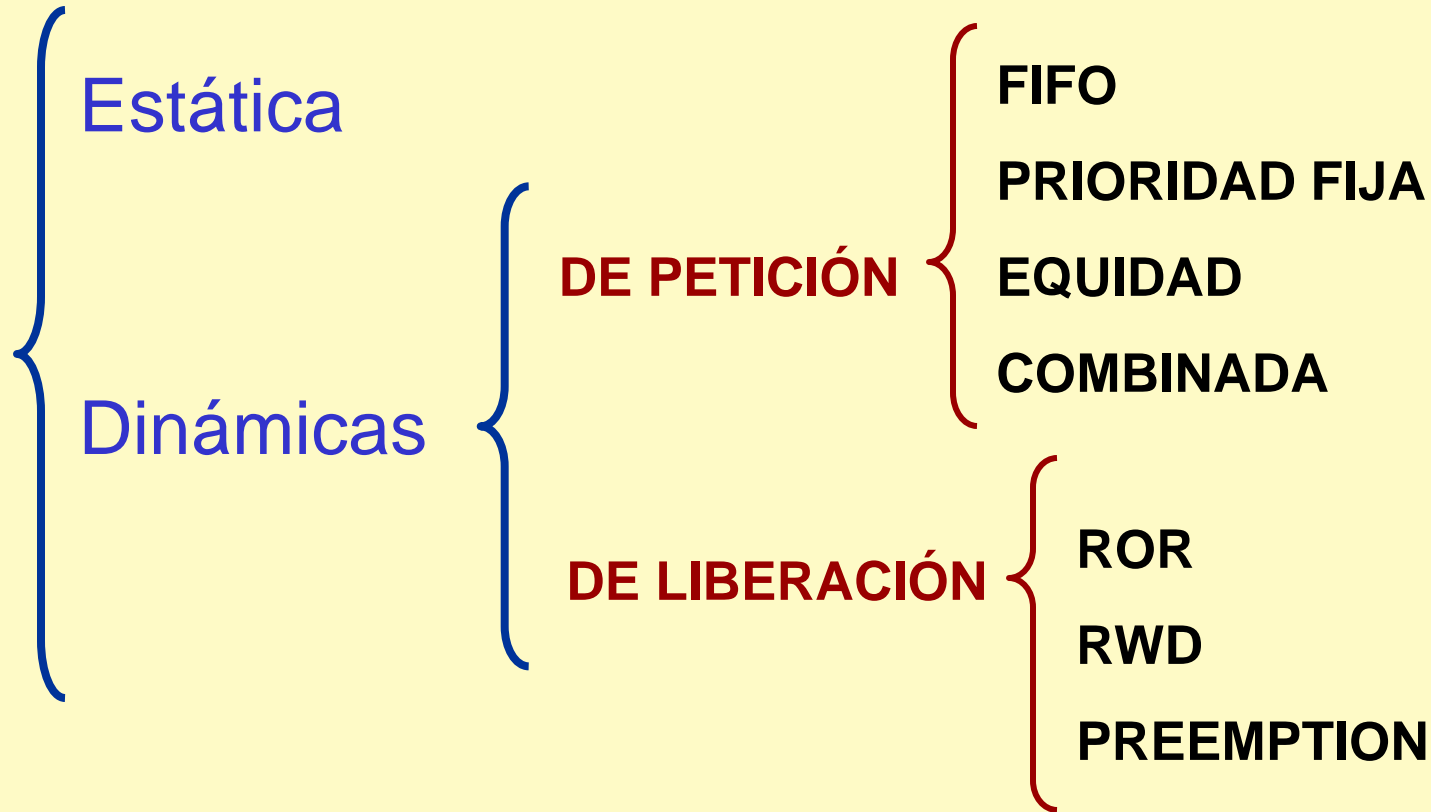
- **Direccionamiento:** Intentar acceder a direcciones inexistentes (memoria o dispositivos) o direcciones protegidas (ej. Código del SO).
 - **Soluciones:** Si bus asíncrono (señal RDY) dispositivo inexistente no responde. Modulo temporizador: Espera Time-Out y fuerza repetición ciclo.
Si memoria local a 1 CPU, cpu suele detectar/ proteger.



- **Datos:** Crosstalk, ruido, fluctuación alimentación
 - **Soluciones:** Códigos detectores (Paridad), correctores (Hamming), CRC, Checksum.
Al detectar, reintentar transferencia.
- **Arbitraje:** Se concede el bus a varios dispositivos (corrupción) o a ninguno (deadlock).
 - **Soluciones:** Bus datos/dir cortocircuitable (se detectará error paridad).
Módulo temporizador:
Deadlock → Time-Out → Reintentar.

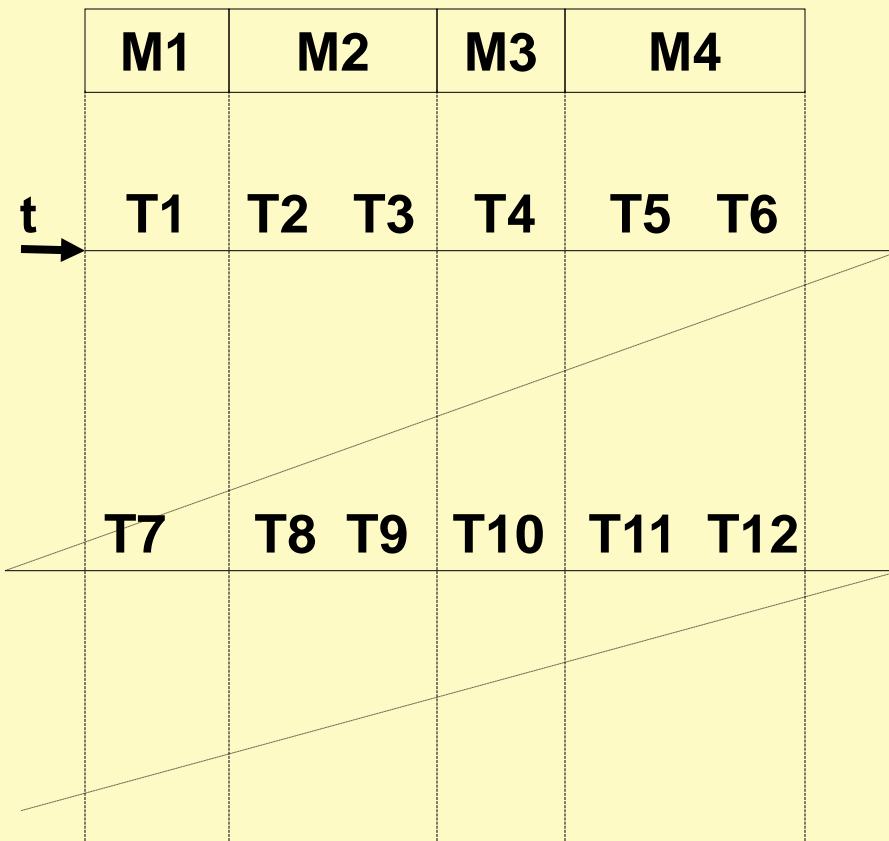


Políticas de Arbitrio



Estática: Repetir turnos (transaction slots) entre masters de una manera prefijada.

- Ej. 4 tarjetas activas, potenciales maestros.



**El reparto se denomina
Frame (marco)**

**Típico en sistemas
síncronos (duración T fija).**

**Esquema simple, sencillo de
implementar.**

**Desperdicia slots (ancho de
banda del bus)**



DINÁMICAS: Permiten cambiar MASTER según la situación actual. Requieren HARDWARE.

- **PETICIÓN:** Cómo decidir, de entre tarjetas activas que solicitan el bus, el próximo MASTER.
 - **FIFO:** First-in, First Out. Bus se concede a quien lleva más tiempo pidiéndolo.
 - **PRIORIDAD:** Cada tarjeta tiene una prioridad. La petición de mayor prioridad gana.
 - **EQUIDAD:** Garantizar que no se concede al mismo 2 veces habiendo otra petición pendiente. Ej. **Round-Robin** → Ir concediendo en ronda (entre los que lo soliciten). Evita **STARVATION**: si máxima prioridad solicita mucho, restantes prioridades mueren de hambre.
 - **COMBINADA:** Típico en sistemas (multiprocesador | E/S) y multiproceso asimétrico. Usar Política Prioridad para tareas ocasionales y de duración corta. Equidad para tareas normales. Ej. E/S se atienden inmediatamente (si hay varias, en orden de prioridad). Procesadores atienden cíclicamente mientras no haya E/S.

DINÁMICAS:

- **LIBERACIÓN:** Cómo decidir cuando deja libre el bus el Master actual.
 - **ROR:** Release on Request (liberar cuando haya otra petición). Típico en sistemas monoprocesador. { CPU posee el bus casi siempre
Sólo cede si DMA, robo ciclo, etc.
Ahorra tiempo arbitraje: CPU accede frecuentemente sin tener que competir.
 - **RWD:** Release When Done (liberar al acabar). Típico multiprocesador. Master sólo usa bus durante 1 transacción, si quiere más, debe competir.
 - **PREEMPTION:** Una transferencia en curso puede ser interrumpida (Master libera bus) por una petición de mayor prioridad.



- **GESTIÓN CENTRALIZADA**
- **DAISY – CHAIN**
- **HÍBRIDA (COMBINADA)**
- **GESTIÓN DISTRIBUIDA**

Cada política de arbitrio puede realizarse (en hardware) de diversas maneras:

EXTREMOS:

- **Módulo Árbitro:** Programable para implementar diversas políticas.
- **Lógica Distribuida:** Cada tarjeta activa contiene circuitería de arbitrio.



Cada tarjeta Activa dispone de líneas de arbitrio:

- Br_i (PET_i): Petición del Bus al árbitro.
- BG_i (CON_i): Árbitro concede Bus a Tarjeta i

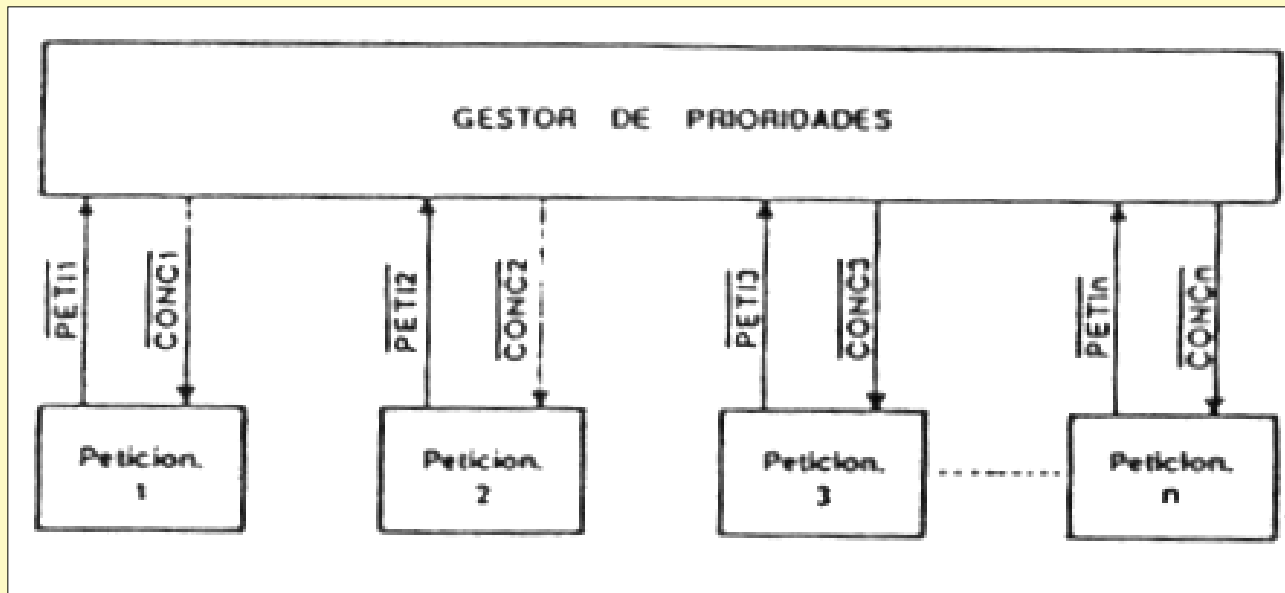
Tarjeta Arbitro programable para cualquier política.

Ej: $N=16$, 8 tarjetas E/S, 8 tarjetas procesador

Tratar $N=1..8$ E/S por prioridad decreciente.

$N=9..16$ Procesadores, prior. Round-Robin

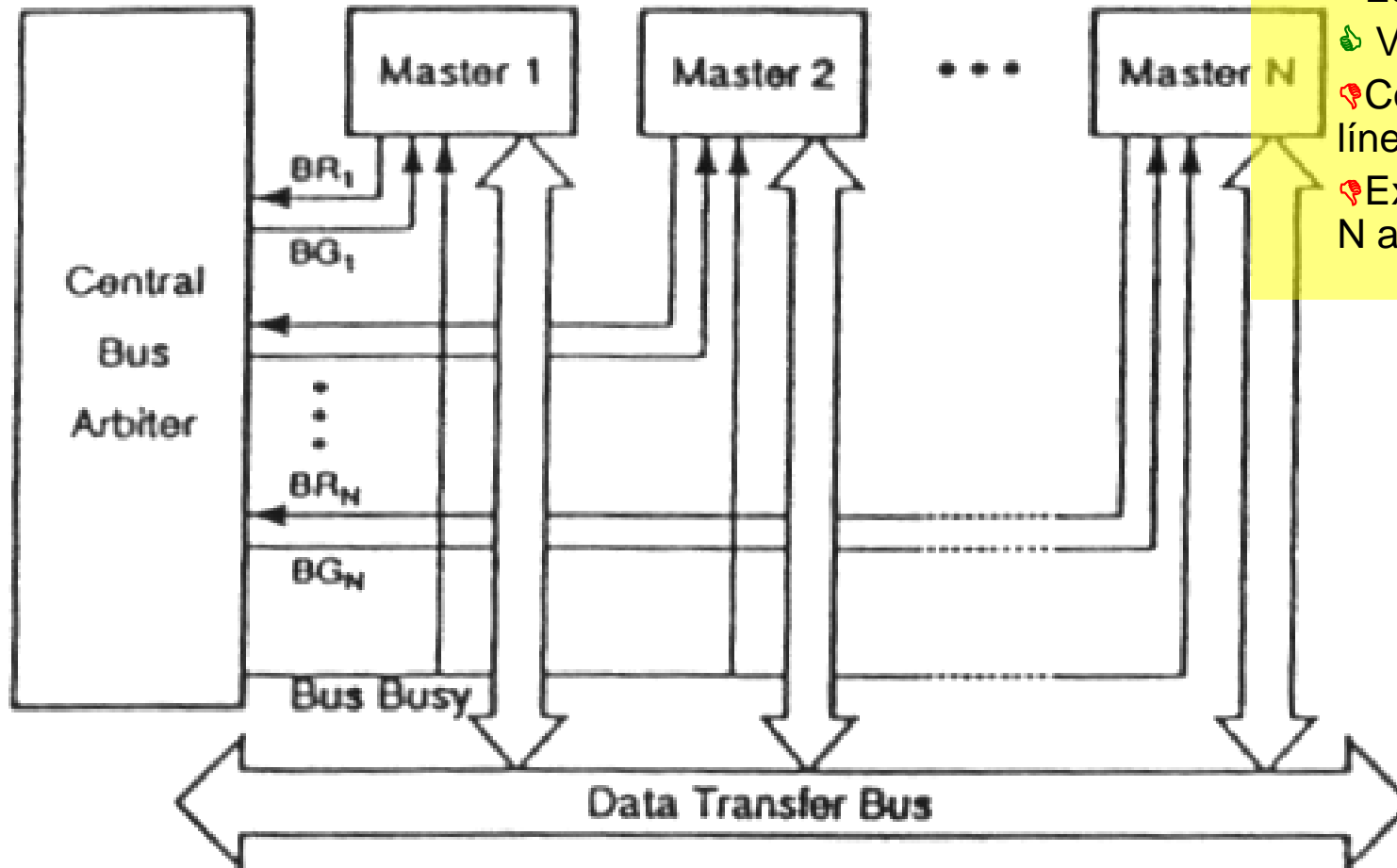
**GESTIÓN
CENTRALIZADA**
DAISY – CHAIN
HÍBRIDA
(COMBINADA)
GESTIÓN
DISTRIBUIDA



**Gestión de
Prioridades
Centralizada**



Demandas Independientes con árbitro central



Legends: BR_i (Bus request from master i) BG_i (Bus grant to master i)

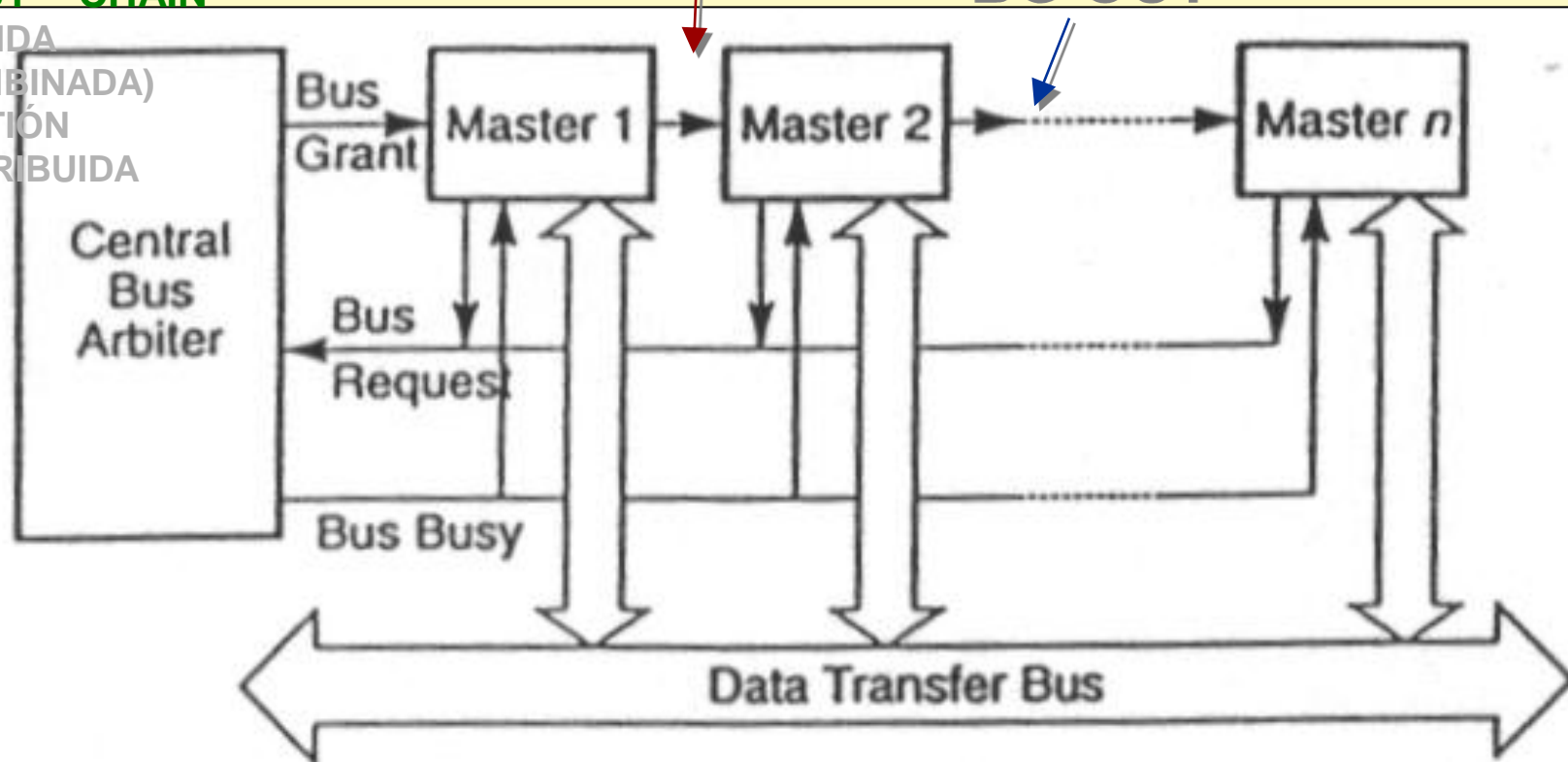
- 👍 Esquema flexible
- 👍 Velocidad arbitraje
- 👎 Coste: Árbitro, líneas BR/BG
- 👎 Expansibilidad: máx N activas.



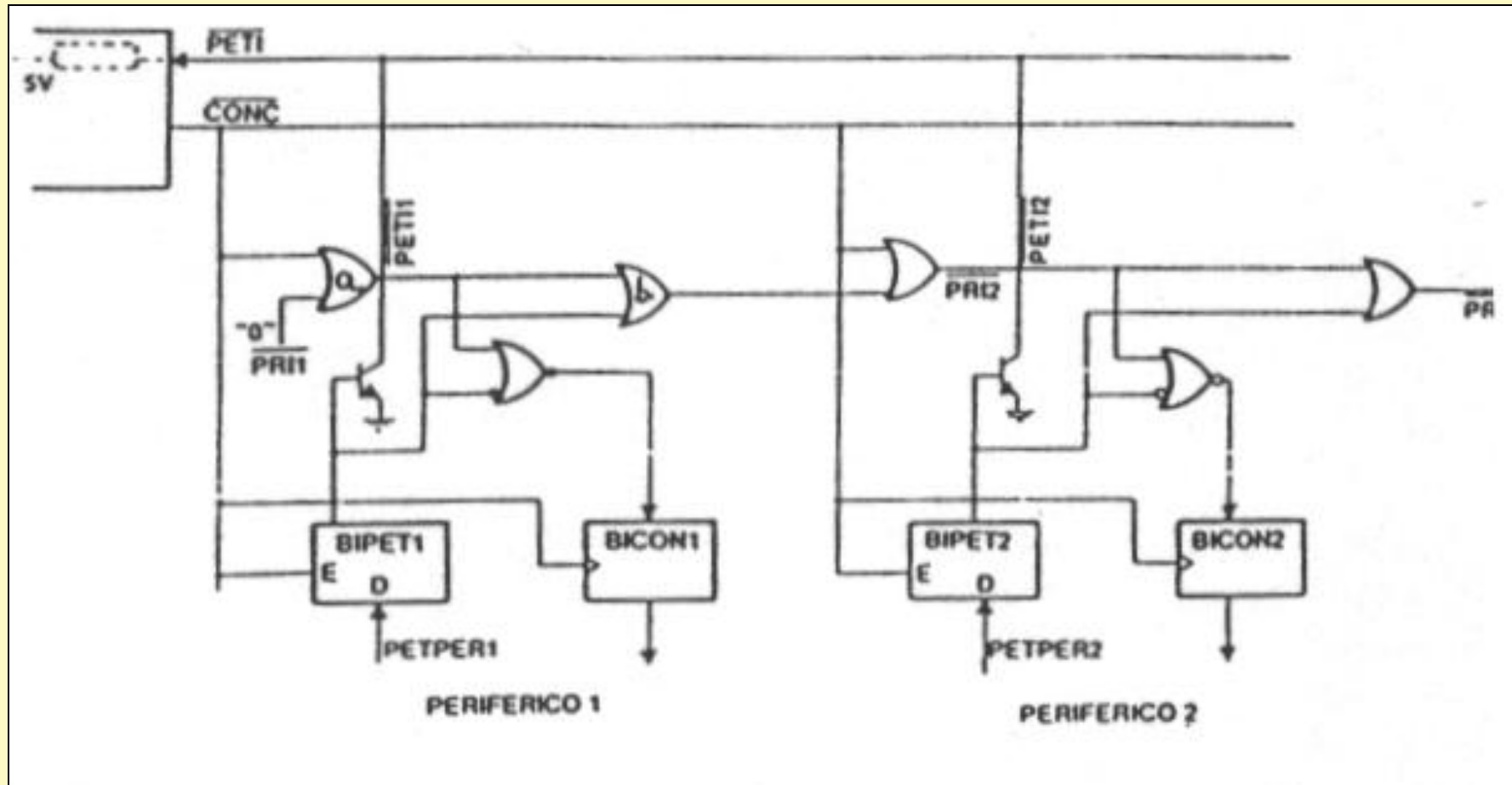
GESTIÓN
CENTRALIZADA
DAISY – CHAIN
HÍBRIDA
(COMBINADA)
GESTIÓN
DISTRIBUIDA

BG IN

BG OUT



Algunos autores consideran Daisy-Chain una variante de Centralizada (hay árbitro). “BR compartida y BG daisy-chain”.



También se puede considerar Daisy-Chain una variante Distribuida:
Cada tarjeta tiene circuitería propia

La tarea del árbitro es casi trivial (no programable)

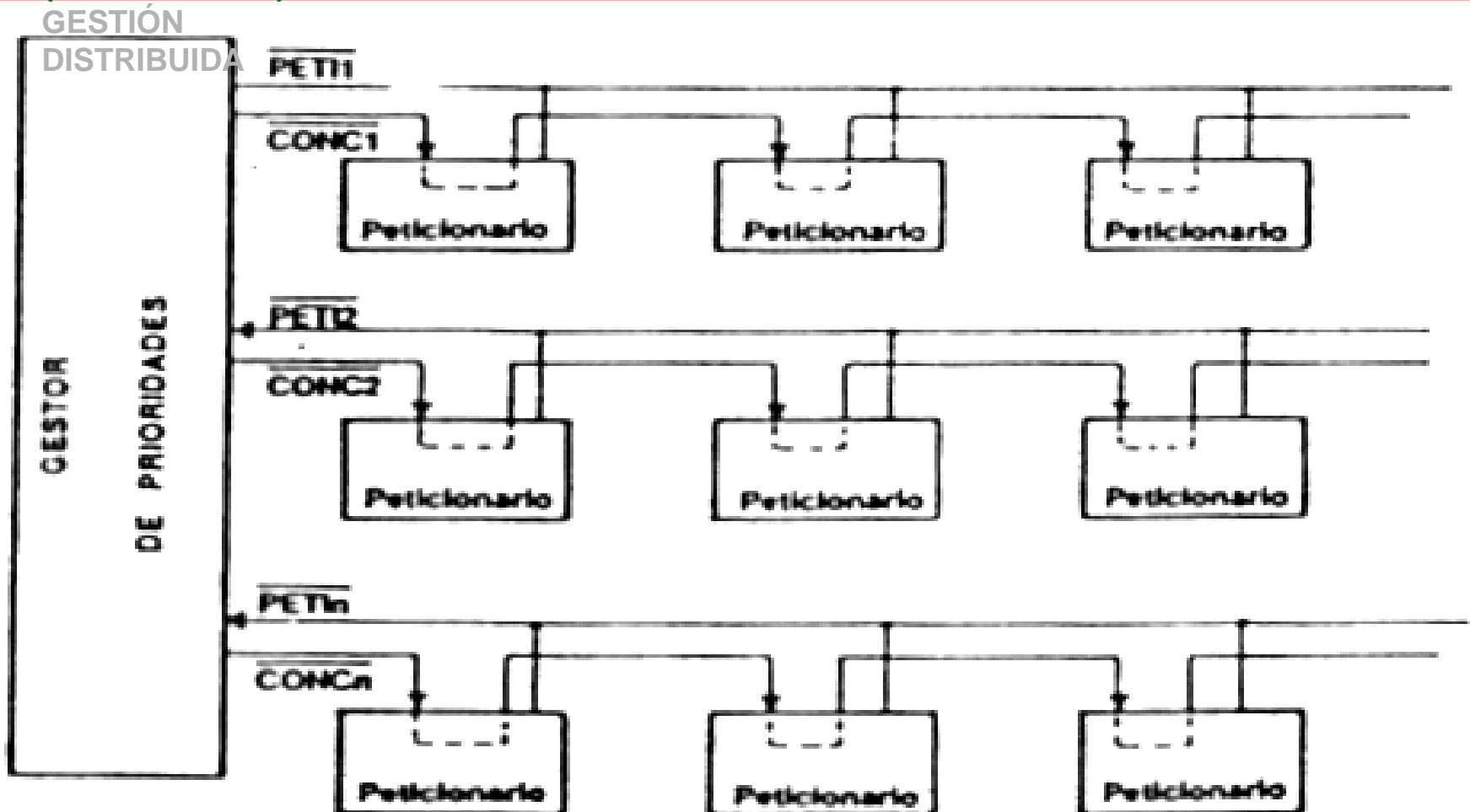


- 👍 **Simplicidad lógica de control:** poca circuitería, pocas líneas,
- 👍 **Expansibilidad:** n° maestros limitado sólo por tiempo propagación... ilimitado.
- 👎 **Inequidad:** Prioridad fija por posición en cadena (posible STARVATION)
- 👎 **Lentitud:** Ciclo arbitraje suficientemente largo como para garantizar propagación.
- 👎 **Estático:** Cambiar prioridades implica cambiar orden FISICO de las tarjetas en daisy-chain.
Ruptura puerta OR de tarjeta activa implica ruptura de la cadena (no tolerancia a fallos).



GESTIÓN
CENTRALIZADA
DAISY – CHAIN
HÍBRIDA
(COMBINADA)

- Un daisy-chain a cada línea del árbitro central.
- Política Programable entre cadenas.
- Prioridad fija dentro de cada cadena.





1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus
2. Tipos de Bus
3. Especificación de un Bus
4. Paralelismo y multiplexación
5. Arbitraje

→ 6. Tipos de transferencia

7. Temporización y direccionamiento
8. Ejemplos de Buses. PCI



Tipos de transferencia:

Según contenido:

- Lectura.
- Escritura.
- DMA (THREE-PARTY)

Según procedimiento:


- Conectada (Ciclo completo) (Circuit Switching)
- Partida (Ciclo Partido) (Packet Switching)

Según tamaño:

- Vacía (DUMMY)
- Palabra (Single)
- Bloque (Burst)



Escritura: Master proporciona datos a Slave.

- Si bus multiplexado: Direcciones/Datos 

{	1º Ciclo Direcciones
	2º Ciclo Datos
- Si no, Master puede dar Dir/Dato SIMULTANEAMENTE. De todas formas SLAVE lee el dato cuando se active (Tiempo de selección por direccionamiento).

Lectura: Master requiere dato de Slave. Slave escribe el dato cuando se active (Tiempo de acceso)

No es tan grave multiplexar el Bus

Modificación: Master lee, INMEDIATAMENTE escribe.
INDIVISIBLE.

Ej: operación TEST&SET para semáforos.

Comprobación: Master escribe. INMEDIATAMENTE lee.
INDIVISIBLE. Útil para chequeo.



Conmutación de circuitos:

- Se establece la conexión (**circuito**) y se transmiten los datos.
- Es el tipo de comunicación usado en las transferencias:
 - Normales
 - Conectadas
 - De ciclo completo

Conmutación de Paquetes:

- La transferencia se **divide en varios “Trozos” (Paquetes)**.
- Cada “trozo” se transmite independientemente.
- Puede seguir un camino (enrutamiento) distinto.
- Puede llegar en distinto orden secuencial.



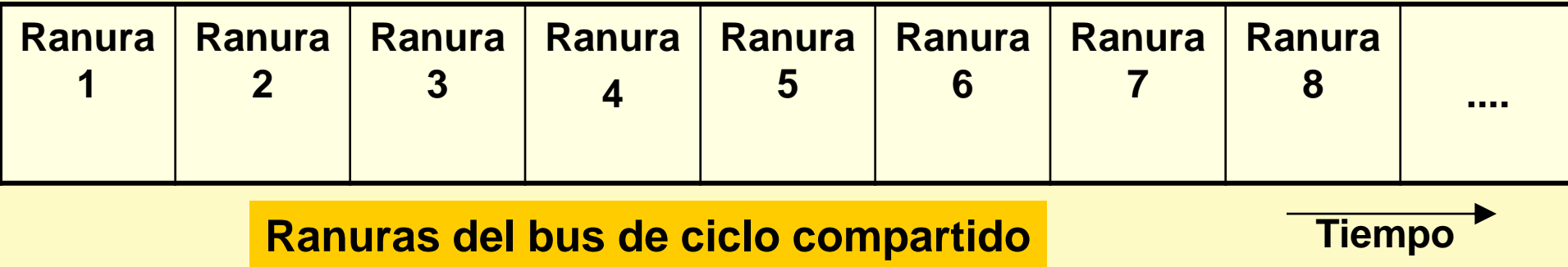
Ejemplo: ETHERNET

Dispone de dos protocolos a nivel de red:

- **TCP: Transmission Control Protocol: Implementa Circuit-Switching.**
 - Se establece un circuito (tal vez a través de varios gateways-routers).
 - Todos los mensajes siguen el mismo camino, llegan en el mismo orden. Ej: rlogin, telnet.
- **UDP: User Datagram Protocol: Implementa Packet-Switching.**
 - Los datos se trocean en Datagramas.
 - Para cada Datagrama se establece una conexión y se envía.
 - En el destino se reordenean en orden secuencial. Ej: ftp



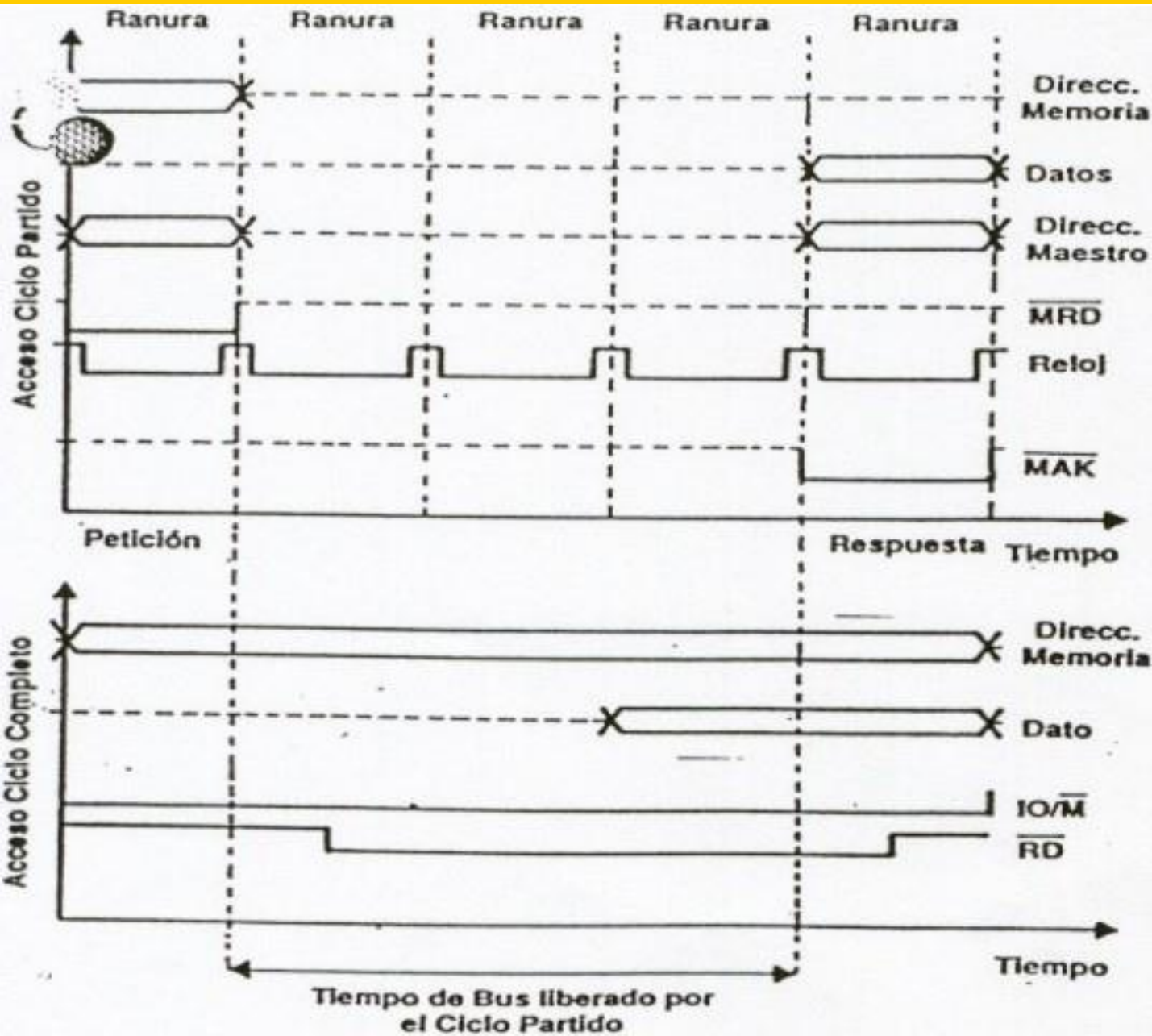
Es la filosofía implícita en las transferencias **CICLO PARTIDO**.



En vez de dividir el mensaje en datagramas dividimos la TRANSFERENCIA: → Dirección → Dato

Ejemplo: Lectura en ciclo partido:

- MASTER direcciona a SLAVE
- MASTER libera bus !!! (ciclo PARTIDO) otros MASTER pueden usar bus.
- SLAVE es muy lento. Pasa tiempo acceso.
- SLAVE direcciona maestro!!!
- SLAVE escribe datos.
- SLAVE reconoce fin de ciclo (MAK)



Lectura en memoria mediante bus de ciclo compartido



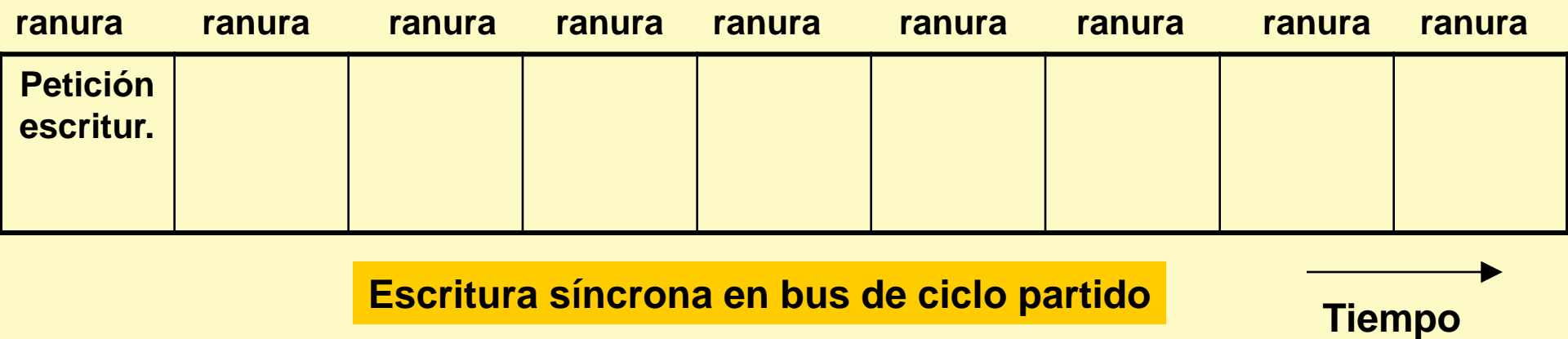
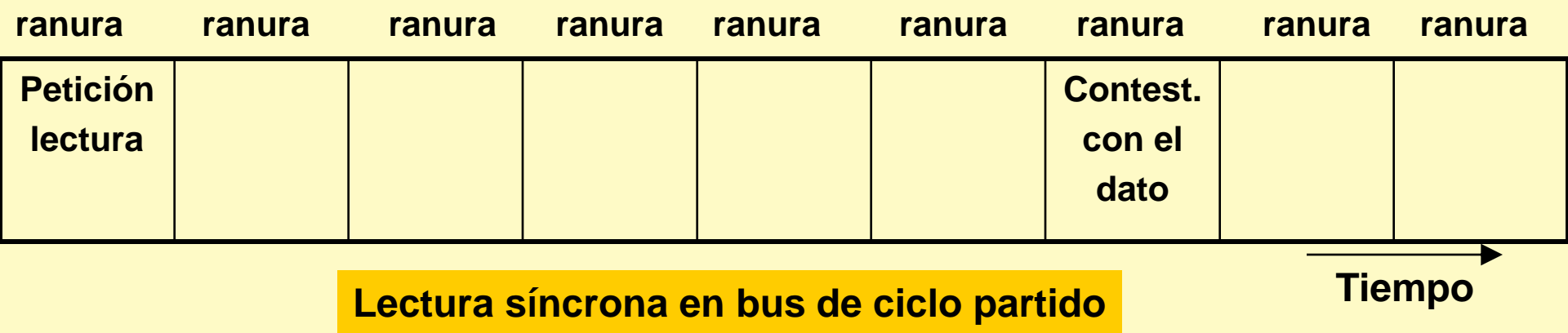
Ciclo partido:

- La secuencia petición / transferencia realizada está partida.
- El bus está libre durante el tiempo invertido.
- **LECTURA:**
 - El dispositivo (lento) hace “latch” de la dirección.
 - Responde al cabo del tiempo de acceso.
 - Debe direccionar al maestro (maestro debe leer dato).
- **ESCRITURA:**
 - El dispositivo hace “latch” de dir/dato.
 - No tiene mucho sentido en bus síncrono (no hay confirmación).
 - Sí tendría sentido COMPROBACION síncrona partida (write-read).

Asíncronismo: Temporización para conectar dispositivos lentos a buses rápidos



El concepto de ciclo partido (tipo de transferencia) es independiente del sincronismo (modo de temporización).





ranura	ranura	ranura	ranura	ranura	ranura	ranura	ranura	ranura
Petición lectura	Contest. Direc. Correct.					Contest. con el dato		

Lectura asíncrona en bus de ciclo partido

→
Tiempo

ranura	ranura	ranura	ranura	ranura	ranura	ranura	ranura	ranura
Petición Escritur.						Confirm. Escritur.		

Lectura semisíncrona en bus de ciclo partido

→
Tiempo



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

TEMA 5: BUSES

1. Estructuras del Bus
2. Tipos de Bus
3. Especificación de un Bus
4. Paralelismo y multiplexación
5. Arbitraje
6. Tipos de transferencia
- ➔ **7. Temporización y direccionamiento**
8. Ejemplos de Buses. PCI



TEMPORIZACION:

SINCRONA:

CPU no tiene forma de garantizar que transferencia OK

- Típico en buses locales (TIPO 1) o monoprocesador (1 MASTER)
- Master proporciona dir (T1)
 - y datos (T2) si espera escritura
 - lee datos (T3) si operación lectura
 - asume que datos leídos si operación escritura
- No hay mecanismo por el cual CPU pueda recibir reconocimiento.
- CPU asume que todos los dispositivos funcionan correctamente y a la misma velocidad que la CPU.
- Todas las transferencias suceden en un ciclo/flanco predeterminado de reloj.
- Hay una señal CLK común a todos los dispositivos.
- Ventajas: Circuitería de control mínima -> menor coste.
- Desventajas: El Bus debe ir a la velocidad del dispositivo más lento (o el dispositivo no reaccionará a tiempo)



TEMPORIZACION:

SEMISINCRONA:

- Añadir línea de control adicional RDY (ó Wait)
 - MASTER inserta ESTADOS DE ESPERA hasta que Slave reacciona.
 - Ventajas:
 - Circuitería sencilla
 - Inmunidad a ruido
- Relativa en comparación con asíncrona
- Desventajas: Tiempo de uso del Bus incrementado en múltiplos ciclos bus.



TEMPORIZACION:

ASINCRONA:

- No existe reloj común CLK.
- Conversación Master/Slave mediante HANDSHAKE.
- “Handshake”: Estrechar la mano (ofrece 1º, ofrece 2º, agitar, retira 1º, retira 2º).
- Ventajas: Permite conexión de periféricos de cualquier velocidad.
- Desventajas: Circuitería más compleja, por lo que se incrementa el coste.



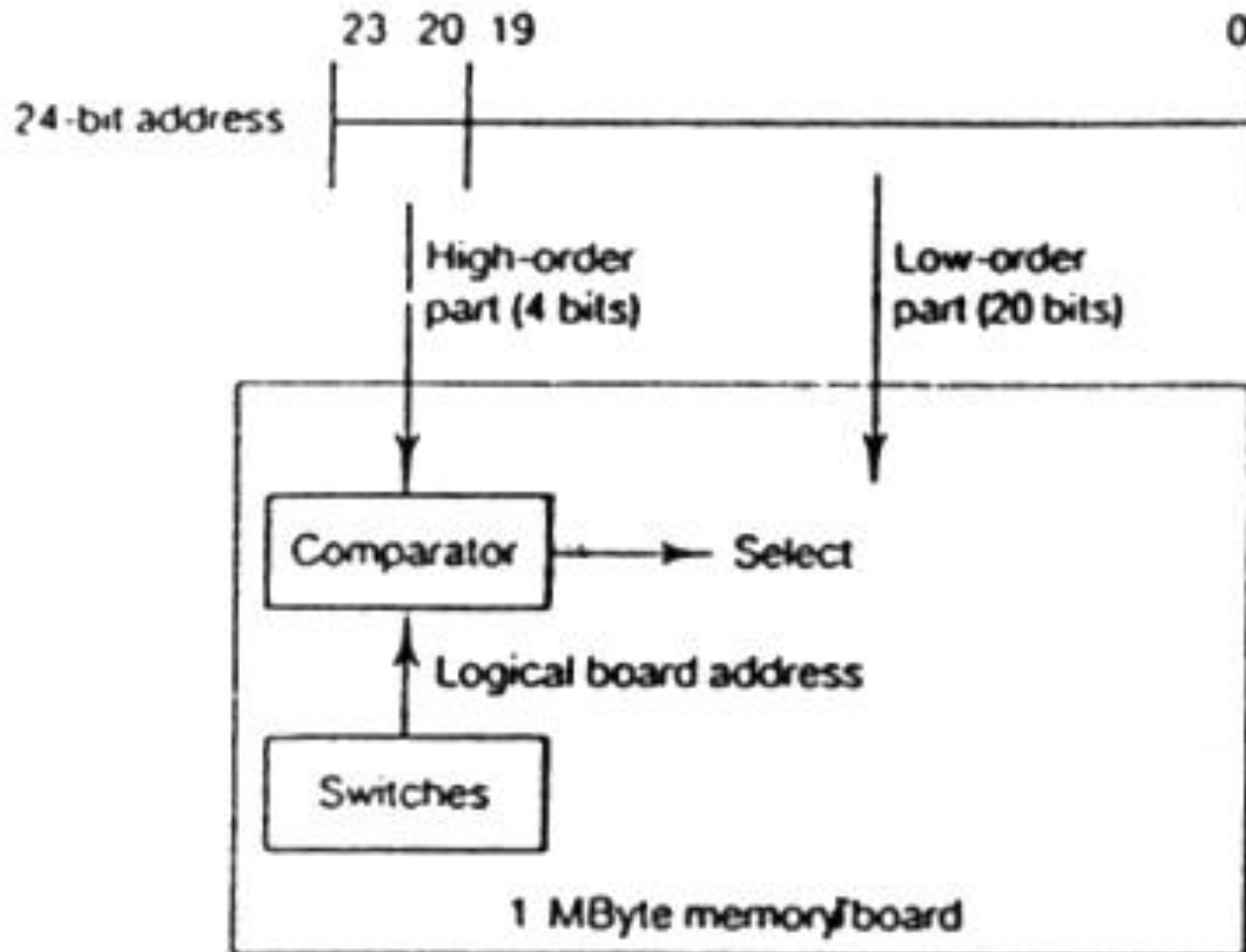
Direccionamiento

Una vez en posesión del bus, **MASTER debe**

- Establecer comunicación con SLAVE
- Seleccionarlo
- Ponerlo en funcionamiento (CS)
- Direccionarlo
- Después, transferencia de datos.

Cada tarjeta tiene

- Una Dirección Base (única, que la distingue del resto de tarjetas)
- Un Rango de Direcciones por el que se reparten sus elementos direccionables, memoria, registros control/estado, registros E/S...

**SEGÚN MÉTODO DE SELECCIÓN:****Direcccionamiento Lógico**



Direccionamiento Lógico:

Las tarjetas tienen {
Conmutadores (manuales)
Comparadores (switch/address MsBits)

El operador {
Diseña el Mapa de Direcciones (qué tarjeta
va en qué dirección Base)
Configura los conmutadores

Ej: Bus direcciones 24 bits, todas las tarjetas 4 conmutadores.

Hasta 16 tarjetas. Rango: $2^{20} = 1$ Mbyte / tarjeta

Direcciones Base: 000000, 100000, 200000 ...

... E00000, F00000

No todas las tarjetas tienen que tener el mismo nº de switches.



Direccionamiento Geográfico: (Fastbus, MultibusII, Nubus)

- El rack (bastidor) tiene conectores-ranura para tarjetas (slots).
- Los slots llevan Cableado un número, en orden secuencial (slot number).
- El número de ranura (posición de tarjeta en el rack) determina Base y Rango Direccionamiento.
- Usualmente, buses con direccionamiento geográfico, pueden conmutar a direcc. Lógico mediante una línea de control (GEO/LOG).
- D. Geográfico se usa durante inicialización sistema, S.O. Chequea todas las tarjetas/excluye las que fallen.
- Las tarjetas tienen un registro Base (equivalente a switches dir. Lógico) Programable.
- S.O. Reparte Mapa de Direcciones entre tarjetas que superan chequeo y conmuta a direccionamiento lógico.
- S.O. Puede volver a dir. Geográfico y reconfigurar sistema de nuevo.



SEGÚN NÚMERO DE SLAVES:

Direccionamiento Normal: Sólo 1 slave responde al master.

Direccionamiento Extendido: (Broad) Varios slaves (incluso todas las tarjetas).

- **BROADCAST** : Escritura simultánea a varias tarjetas (memoria, mensajes a CPU's)
- **BROADCAST** : Lectura simultánea de varias tarjetas. Usualmente restringido a OR-cableado.

Ej. { Arbitraje: AP# podían escribirse simultáneamente IRQ's A cada interrupción puede asignarse una posición. Broadcast implicando a 32 tarjetas



Implementación.

Dirección Reservada: $\left\{ \begin{array}{l} 000...0h \\ FFF...Fh \end{array} \right\}$ Leer dir. Res. Provoca broadcast
Escribir dir. Res. Provoca broadcast

Broadcast/call Universal, se dirige a Todas las tarjetas.

Ej. Ethernet: $\left\{ \begin{array}{l} \text{Comando "rusers" UNIX hace broadcast} \\ \text{Otros ordenadores van respondiendo} \\ \text{Acabar comando con ^C} \end{array} \right.$

Utilidad:

Coherencia Cache:

- Sistema multiprocesador, cada procesador con caché propia, datos compartidos.
- Procesador modifica su caché: Debe actualizar memoria compartida
Debe avisar a otros procesadores.

Solución: BROADCAST: Escribir simultáneamente

 $\left\{ \begin{array}{l} \text{Caché propia} \\ \text{Memoria compartida} \\ \text{Caché de los otros} \end{array} \right.$



1. Evolución y prestaciones de los computadores
2. Unidades funcionales de un computador.
3. Nivel de lenguaje máquina.
4. Desarrollo de programas en ensamblador.
5. Buses del sistema. Bus PCI.
6. Sistema de memoria interna.
7. Sistema de memoria externa.
8. Sistema de entrada/salida.

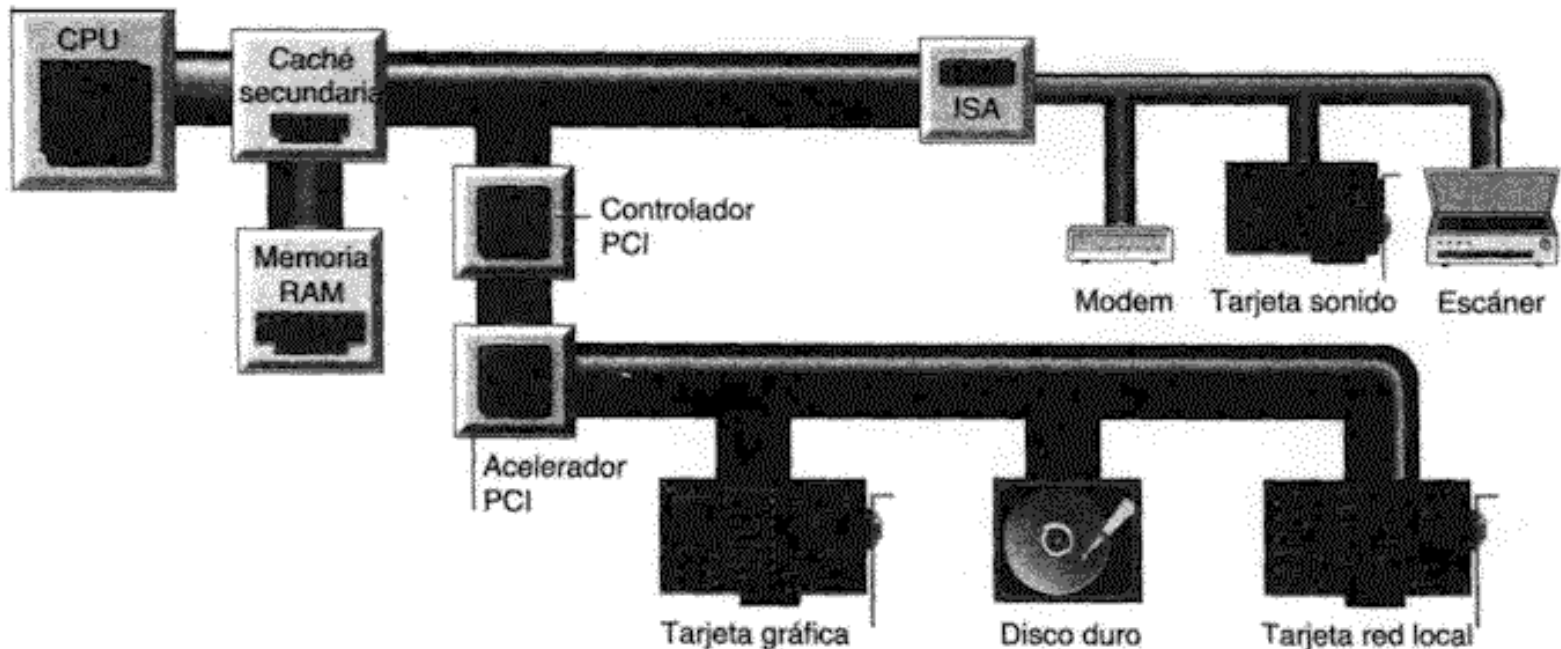
TEMA 5: BUSES

1. Estructuras del Bus
2. Tipos de Bus
3. Especificación de un Bus
4. Paralelismo y multiplexación
5. Arbitraje
6. Tipos de transferencia
7. Temporización y direccionamiento

➔ **8. Ejemplos de Buses. PCI**

PCI significa: interconexión de los componentes periféricos (Peripheral Component Interconnect). Fue desarrollado por Intel.

El bus PCI es *independiente de la CPU*, ya que entre la CPU y el bus PCI se instalará siempre un controlador de bus PCI, lo que facilita en gran medida el trabajo de los diseñadores de placas.





1. El bus PCI **no depende del reloj de la CPU**, porque está separado de ella por el controlador del bus.
2. El límite práctico en **la cantidad de conectores para buses PCI es de tres/seis**; más conectores aumentarían la capacitancia del bus y las operaciones a máxima velocidad resultarían menos fiables.
3. A pesar de presentar un rendimiento similar al de un bus local conectado directamente, en realidad PCI no es más que la eliminación de un paso en el microprocesador. En lugar de disponer de su propio reloj, un bus PCI se adapta al empleado por el microprocesador y su circuitería, por tanto los componentes del PCI están sincronizados con el procesador. El actual estándar PCI trabaja con un **ancho de datos de 32/ 64 bits** y a una **velocidad de 132 MB/s o 264 MB/s**



SCSI: Small Computer System Interface.

Es un estándar universal para la conexión paralela de periféricos

- Podríamos definir SCSI como un subsistema de E/S inteligente, completa y bidireccional. Un solo adaptador host SCSI puede controlar desde 7 dispositivos inteligentes SCSI conectados a él (SCSI-1) hasta 127 (SCSI-3).
- Una ventaja del bus SCSI frente a otros interfaces es que los dispositivos del bus se direccionan lógicamente en vez de físicamente. Esto sirve para **2 propósitos**:
 - Elimina cualquier limitación que el PC-Bios imponga a las unidades de disco.
 - El direccionamiento lógico elimina la sobrecarga que el host podría tener en manejar los aspectos físicos del dispositivo como la tabla de pistas dañadas..
- Ideado para entornos UNIX y Macintosh, permite un ancho de datos de 8 bits para SCSI-1 hasta 16 bits para SCSI-3, con velocidades de transferencia de 4MB/s a 40 MB/s respectivamente.



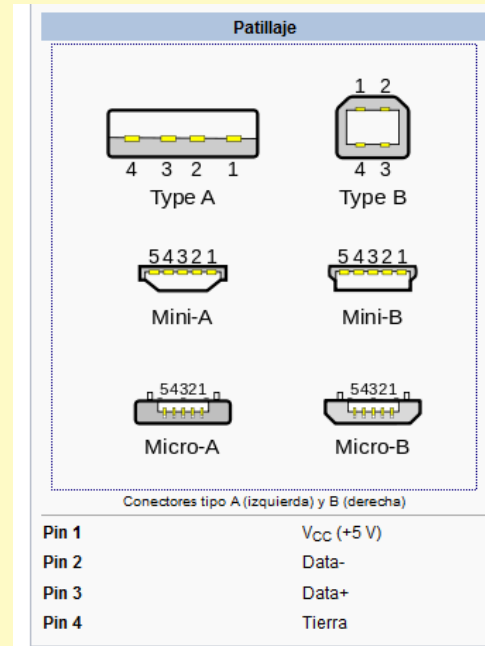
USB: (Universal Serial Bus).

Baja velocidad (1.0): Tasa de transferencia de hasta 1,5 Mbit/s (188 kB/s). Utilizado en su mayor parte por dispositivos de interfaz humana (*Human Interface Device*, en inglés) como los teclados, los ratones (mouse), las cámaras web, etc.

Velocidad completa (1.1): Tasa de transferencia de hasta 12 Mbit/s (1,5 MB/s).

Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbit/s (60 MB/s) pero por lo general de hasta 125 Mbit/s (15,6 MB/s). El cable USB 2.0 dispone de cuatro líneas, un par para datos, y otro par de alimentación.

Súper alta velocidad (3.0): Tiene una tasa de transferencia de hasta 4,8 Gbit/s (600 MB/s). La velocidad del bus es diez veces más rápida que la del USB 2.0, debido a que han incluido 5 conectores adicionales. De esta manera, dos líneas se utilizan para enviar, otras dos para recibir, y una quinta se encarga de suministrar la corriente. Así, el tráfico es bidireccional





FireWire (IEEE 1394) o bus serie de altas prestaciones. Se pueden conectar hasta 63 dispositivos y se logran altas velocidades de 400 Mbps. Una de sus líneas es de suministro de energía. Debido a su bajo costo y sencillez se utiliza en computadoras embebidas, tales como televisores de altas prestaciones, cámaras de video digitales

Futurebus + (IEEE 896.1) Es una normalización proyectada para equipos de muy altas prestaciones, que puede considerarse como una evolución de las normas Multibus II y VME. Permite la construcción de sistemas multiprocesador (de hasta 32 procesadores) compartiendo memoria. Ancho de datos: 64,128,256 bits y vel. Transf: 95.2MB/s, 3.2 GB/s