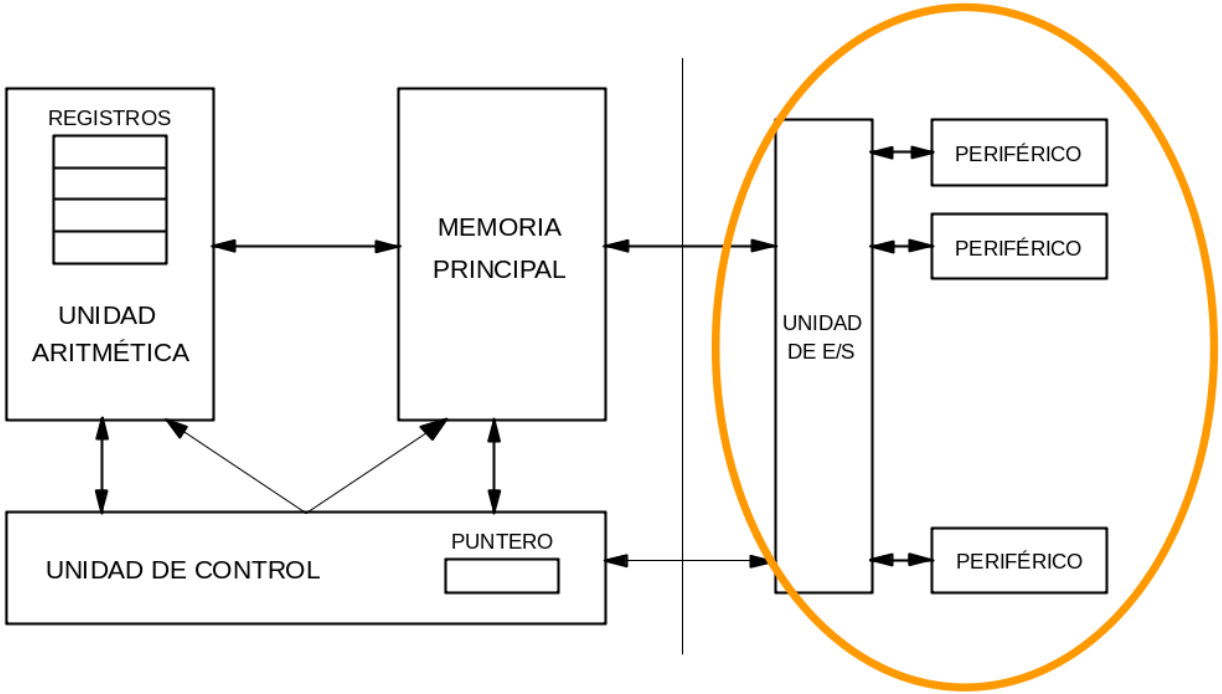
Entrada/salida
Índice

- **Implementa la comunicación con el exterior**

- **Algunas máquinas tienen un bloque de E/S grande**
  - computadores de sobremesa, consolas de juegos, equipos multimedia

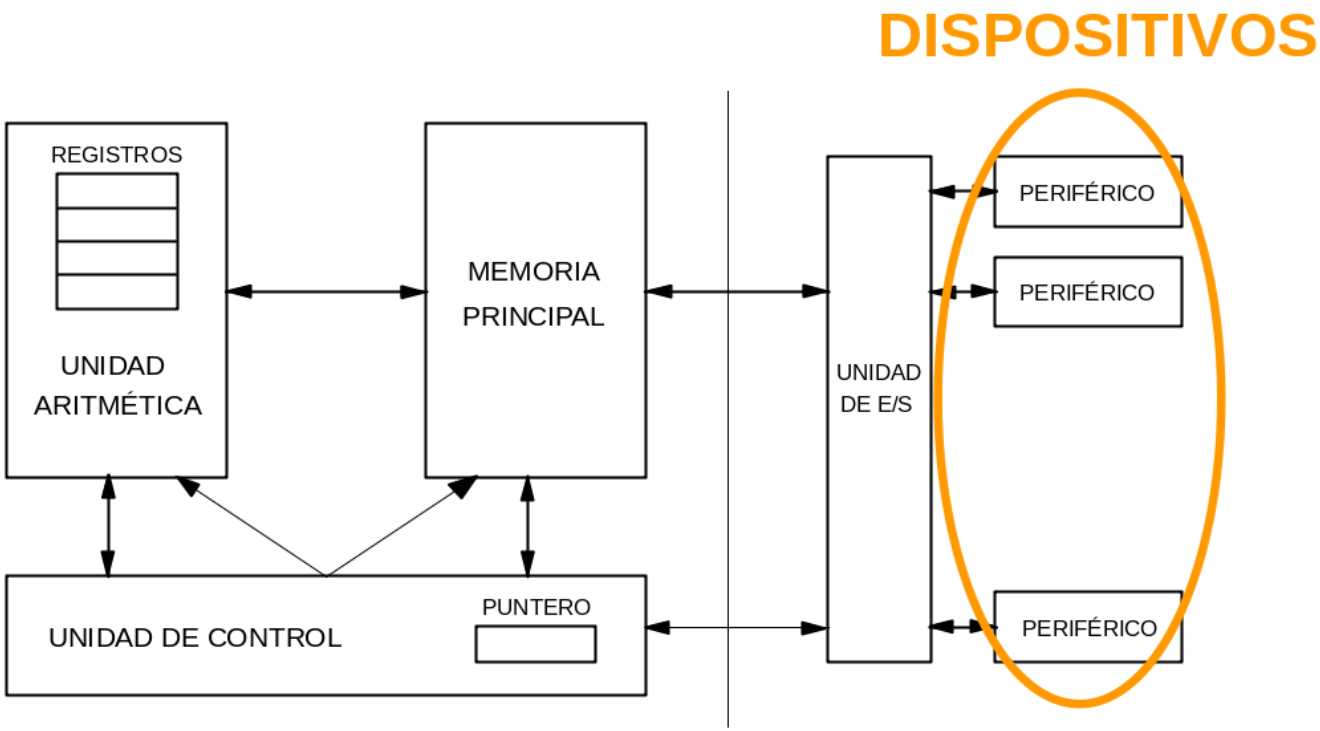
- **… mientras que otras lo tienen muy reducido**
  - supercomputadores dedicados a cálculo intensivo

- **Distinguimos:**
  - **Periféricos → dispositivo que implementa un modo específico de comunicar datos**
  - **Unidad o módulo de E/S → bloque conectado a los buses del computador y a los periféricos**

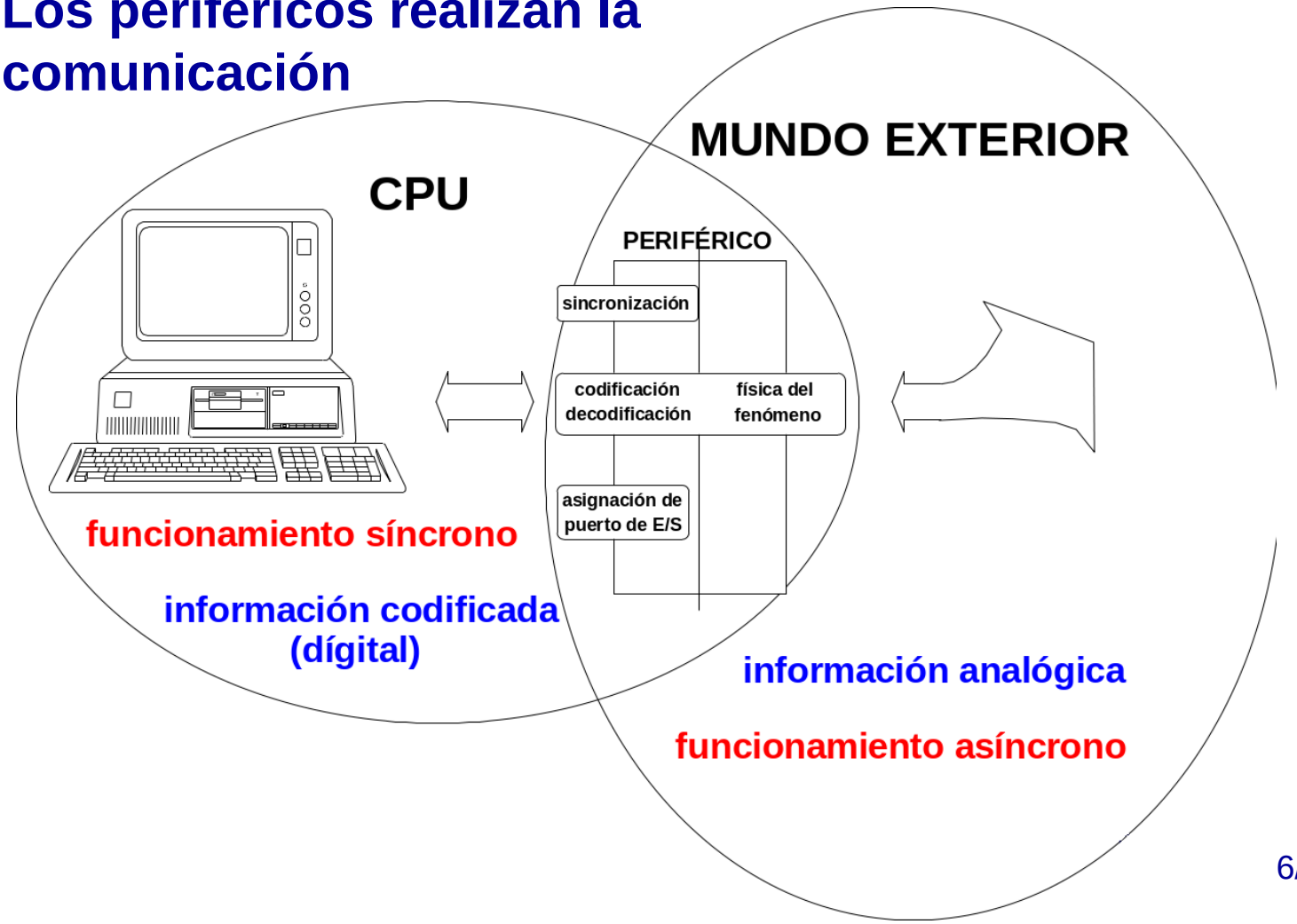


- **¿Por qué los periféricos no se conectan directamente a los buses?**
  - ➔ **Gran diversidad de periféricos**
  - ➔ **La velocidad de transferencia de los periféricos es muy diferente a la del procesador**
    - ➔ Generalmente mucho menor
  - ➔ **Los formatos de los datos de los dispositivos externos son muy diferentes a los del computador**

- **Dispositivos periféricos en la máquina de Von Neumann**



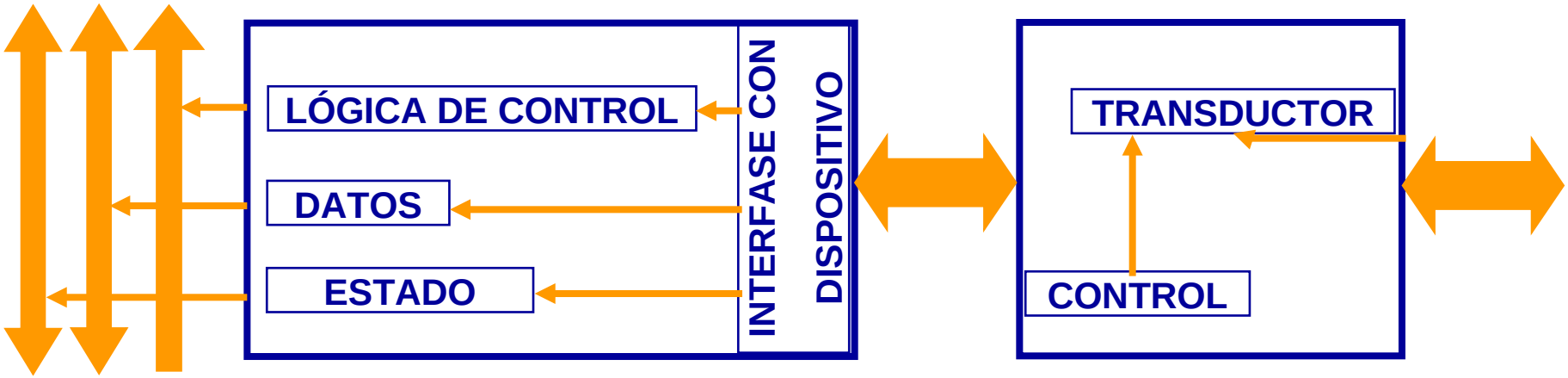
- **Los periféricos realizan la comunicación**



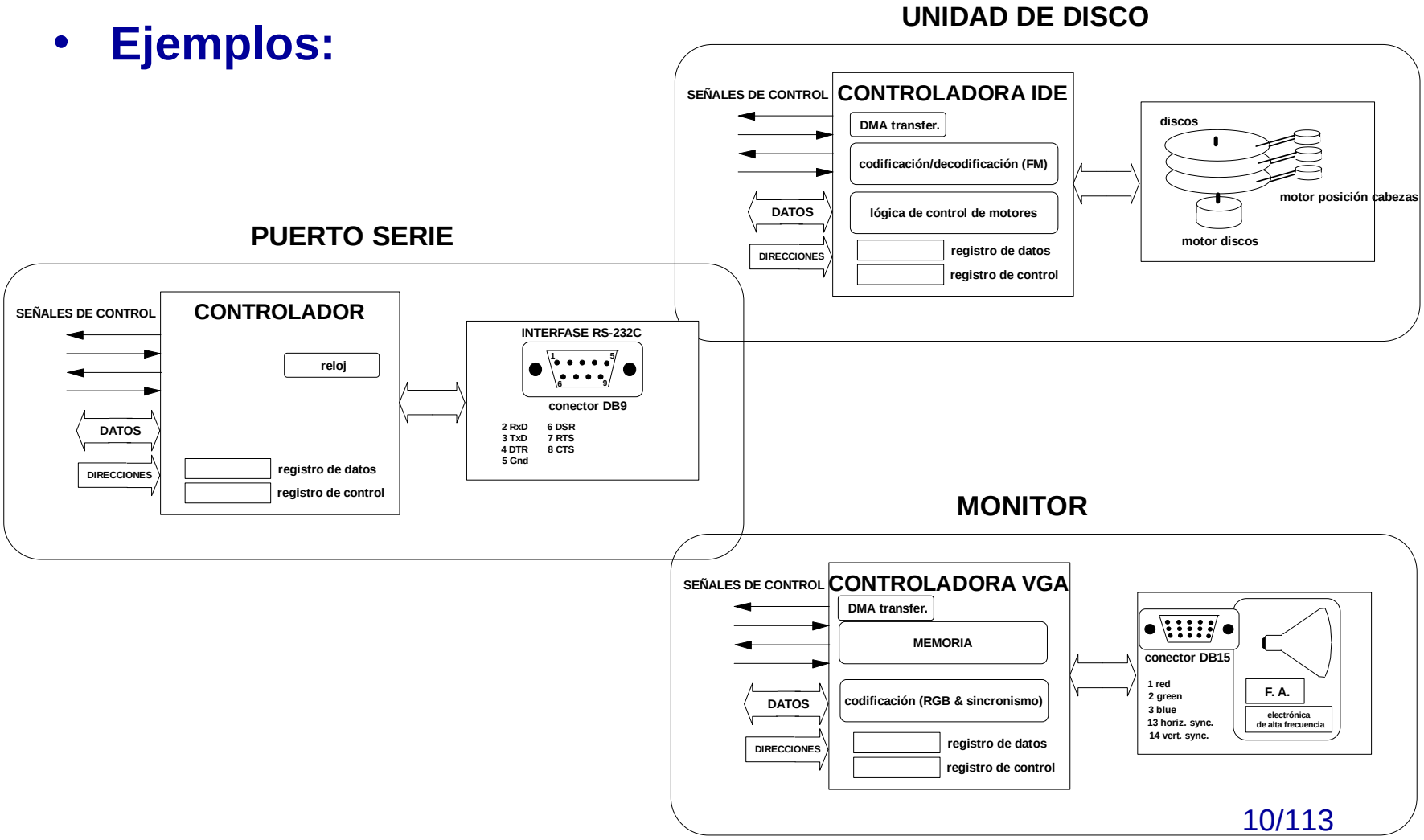
- **Tipos de periféricos:**
  - **Comunicación hombre-máquina**
  - **Comunicación máquina-máquina**
    - Telemática
    - Control de procesos
  - **Almacenamiento masivo**

- **Partes de un periférico :**
  - **Controlador**
    - Se relaciona con el procesador o CPU
    - Su misión es doble:
      - Protocolo de transferencia
      - Transferencia propiamente dicha
  - **Dispositivo**
    - Se relaciona con el exterior
    - Basado en un fenómeno físico o químico transforma señales exteriores a señales eléctricas

- **Esquema básico controlador-dispositivo:**



- **Ejemplos:**

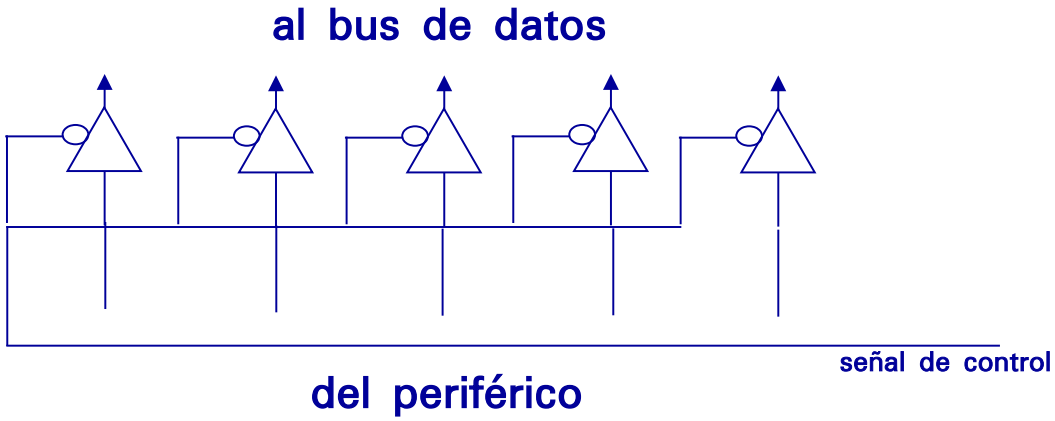


- **Operaciones de lectura de E/S (I)**

  - ➔ **Requieren un *buffer* triestado para no cargar el bus de datos; solamente se activa cuando el dato está listo**

  - ➔ **Además pueden tener un registro interno o una memoria *buffer***
    - ➔ En dispositivos que leen a mucha velocidad se instala un *buffer* circular
      - Un puntero recorre el *buffer* almacenando información
      - Otro puntero indica que mitad del *buffer* está lista para ser leída en bloque

- **Operaciones de lectura de E/S (II)**
  - ➔ **Esquema ejemplo**

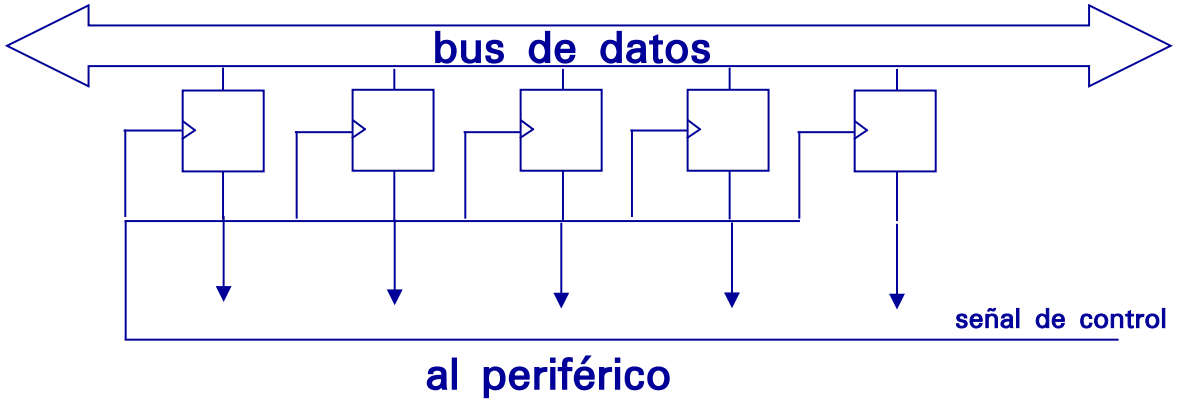


- **Operaciones de escritura de E/S (I)**

  - ➔ **Requieren de un registro donde almacenar los datos temporalmente hasta que el dispositivo pueda darles curso ya que el periférico es mucho más lento que el procesador**

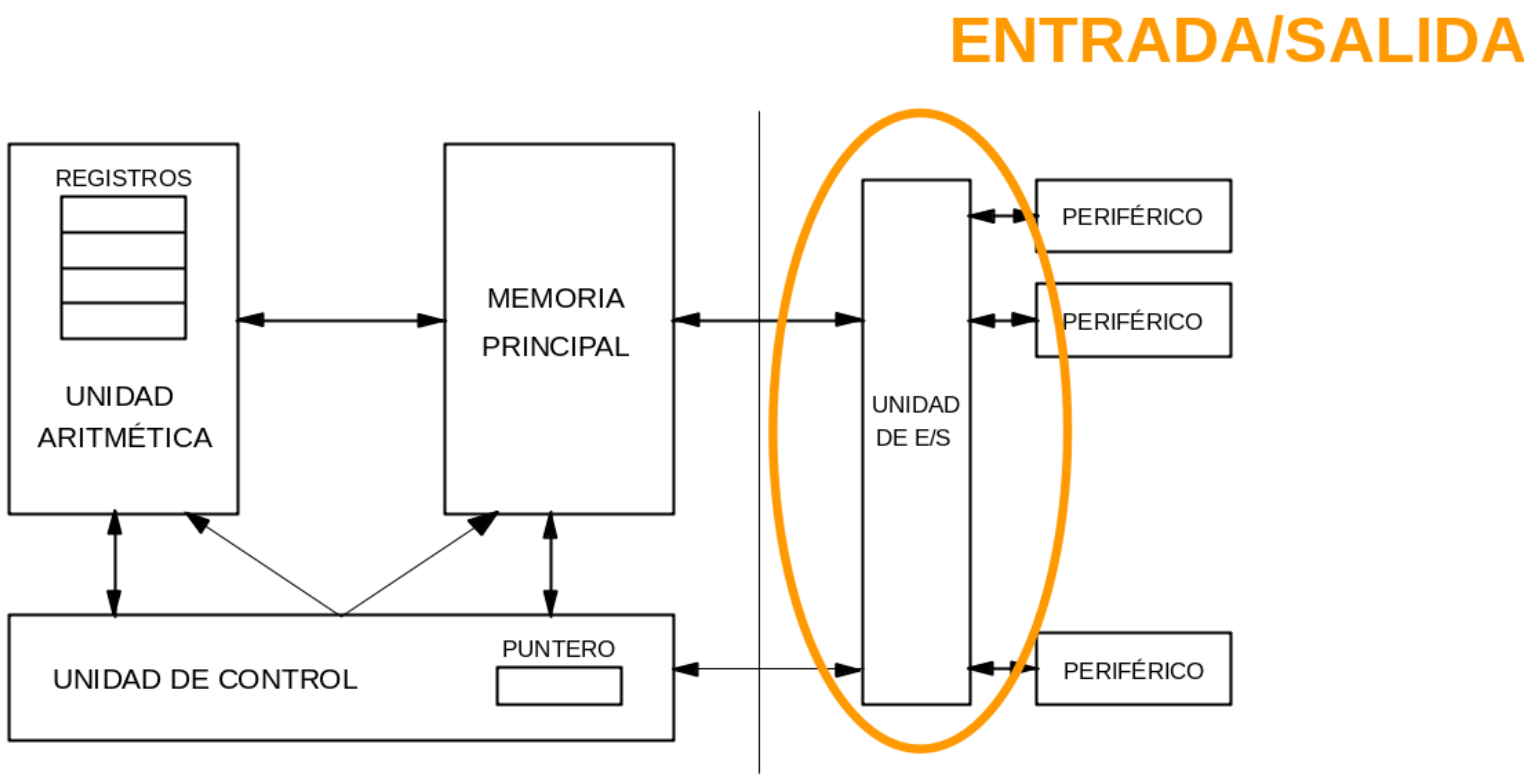
  - ➔ **Este registro puede ser una memoria *buffer* organizada como una cola**

- **Operaciones de escritura de E/S (II)**
  - ➔ **Esquema ejemplo:**



- **Protocolos de transferencia**
  - **La temporización de las transferencias se ajusta a dos tipos de protocolos:**
    - Síncronos
      - Procesador y periférico comparten reloj
    - Asíncronos
      - No comparten reloj; para indicar la disponibilidad necesitan implementar un conjunto de señales y unas reglas de comunicación

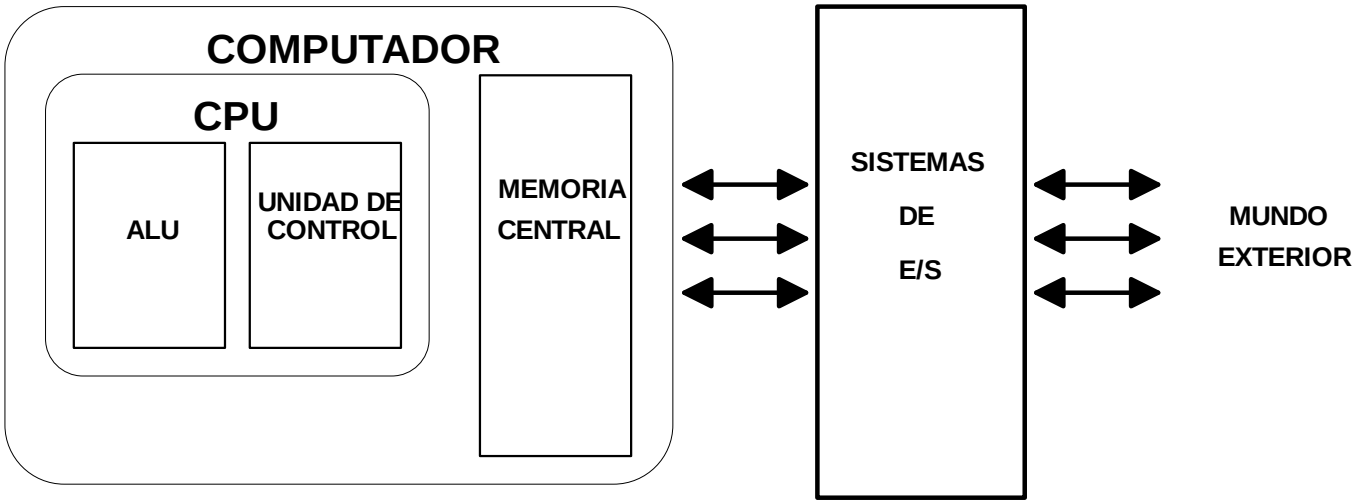
- **Módulo o bloque de entrada/salida (I)**
  - ➔ **Es el bloque encargado de comunicar el computador con los periféricos**



- **Módulo o bloque de entrada/salida (II)**

  ➔ **Asigna el mapa de entrada/salida a cada periférico**

  ➔ **Implementa la sincronización**



- **De manera análoga al mapa de memoria (conjunto de direcciones donde guardamos datos), el mapa de E/S es el conjunto de puertos que dan acceso a dispositivos periféricos**

- **Dos soluciones:**
  - ➔ **Mapas de memoria y E/S separados o disjuntos**
    - ➔ INTEL
  - ➔ **Mapas de memoria y E/S comunes o E/S "mapeada" en memoria**
    - ➔ MOTOROLA, VAX de Digital

- **La forma de direccionar puertos de E/S es similar a la de posiciones de memoria**
  - ➔ **Si los mapas son comunes, se suelen agrupar los puertos en un rango de direcciones con objeto de no "particionar" excesivamente la memoria**
  - ➔ **Si los mapas son disjuntos necesitamos:**
    - ➔ Alguna señal *hardware* que indique a cual de los mapas estamos emitiendo una dirección
      - En el caso INTEL esa señal es IO/M
    - ➔ Instrucciones especiales de transferencia de E/S (que manejen esas señales)

- El mapa de E/S es “doble” o está desdoblado
- No siempre hay una correspondencia unívoca entre el elemento accedido en escritura y lectura como sucede en memoria
- Los accesos a un puerto de E/S en escritura no tienen por qué ser sobre el mismo elemento físico que en lectura, incluso pueden ser más de 2

- El elemento físico efectivamente accedido en cada puerto depende de la combinación de varios factores y es específico de cada periférico

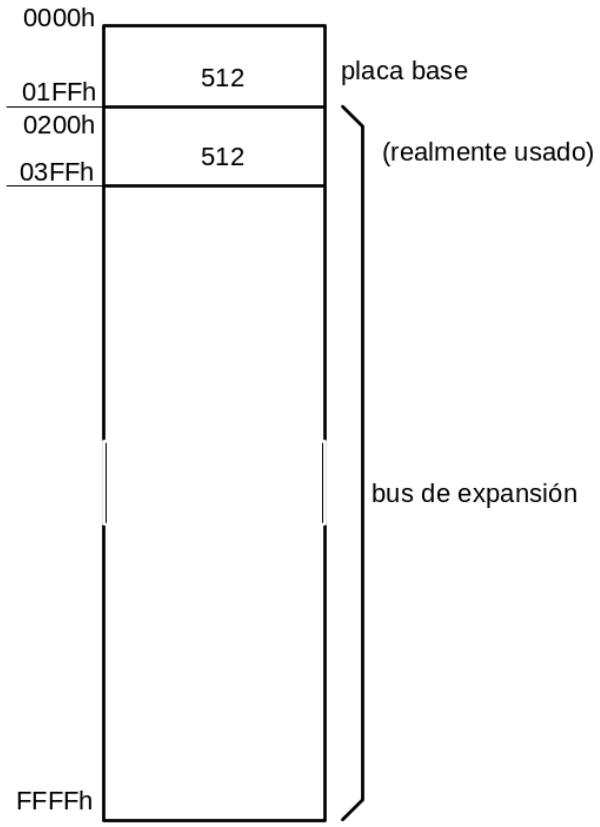
- Esos factores pueden ser:
  - ➔ Tipo de acceso (lectura o escritura) y señales de control involucradas (RD o WR)
  - ➔ Combinación con el valor de un determinado registro de control
  - ➔ Secuencia de acceso (por ejemplo, el primer acceso es a un registro, el segundo a otro y así sucesivamente)

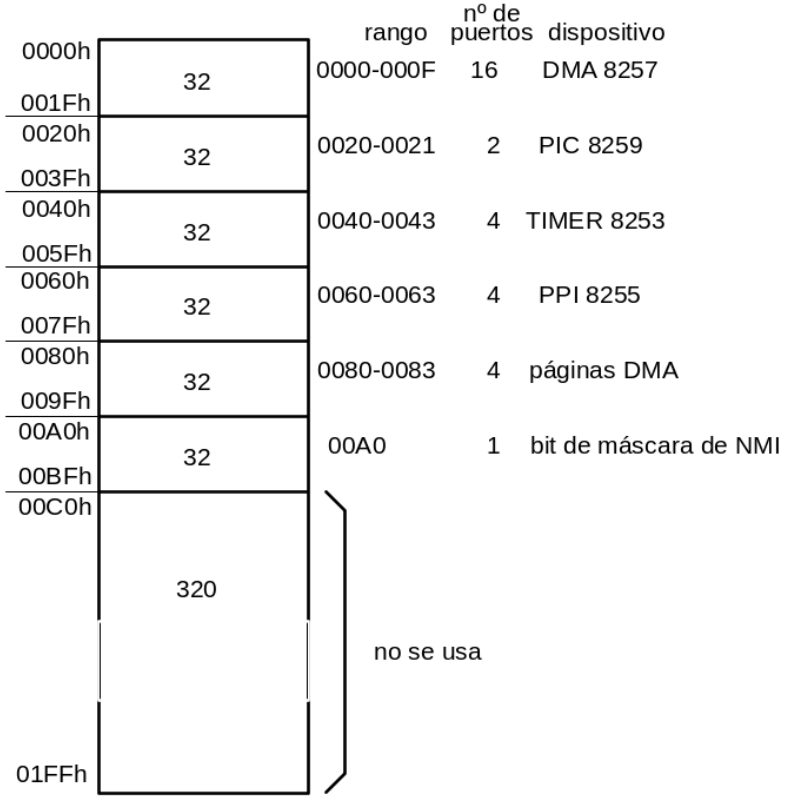
- **Cuando los mapas son comunes, las instrucciones que acceden a los puertos son las mismas que las de acceso a memoria (por ejemplo, MOVE)**

- **Cuando los mapas son disjuntos necesitamos instrucciones especiales que manejen el *hardware* adecuadamente**
  - **IN o INPUT -> entrada de datos de E/S**
  - **OUT o OUTPUT -> salida de datos de E/S**

Ejemplo: mapa de E/S en el 80x86

➔ Mapa disjunto de 64K puertos





- **La sincronización viene a resolver el problema de:**
  - **Lentitud de los periféricos respecto al procesador**
  - **Asincronía del mundo exterior**

  - **Hay que tener en cuenta dos aspectos:**
    - **E/S programada**: la CPU tiene todo el protagonismo ya que inicia y lleva a cabo la transferencia

    - **E/S por interrupción**: la CPU ejecuta la transferencia pero el inicio es pedido por el periférico que indica así su disponibilidad

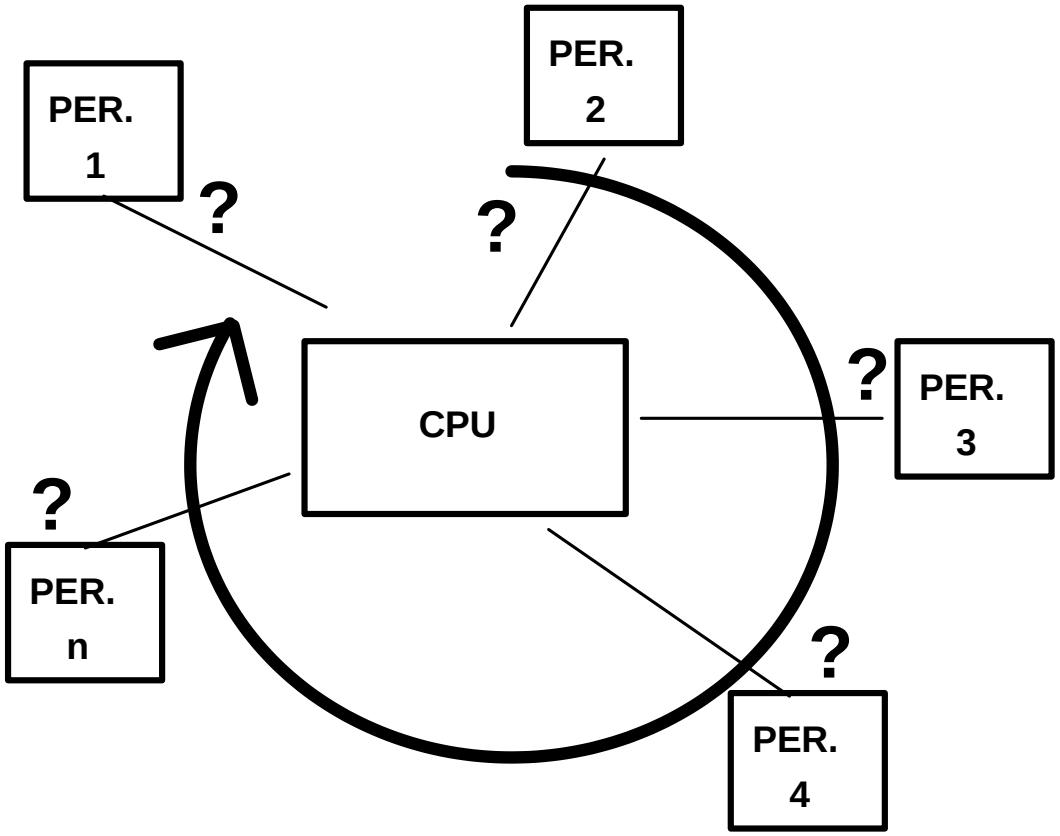
    - **Acceso directo a memoria (DMA)**: el inicio es solicitado por el periférico y la transferencia es realizada por un controlador especializado

Entrada/salida programada

- La CPU inicia la transferencia y la lleva a cabo
- La transferencia puede ser:
  - ➔ incondicional: la CPU no comprueba si el periférico está disponible. De esta forma se pueden hacer lecturas o escrituras erróneas
  - ➔ condicional: la CPU comprueba si el periférico está disponible antes de iniciar una transferencia de información. Este método se conoce como sondeo o *polling*

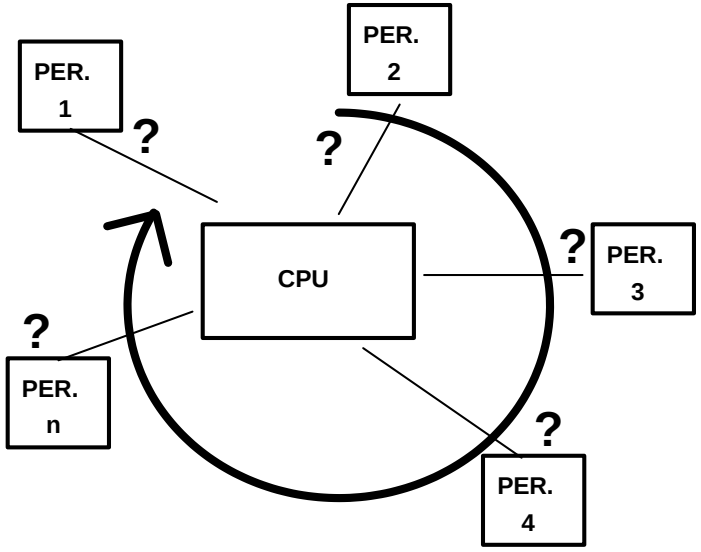
- Esta forma de realizar la E/S tiene el inconveniente de que malgasta el tiempo de proceso pero es sencilla de programar y el *hardware* es simple

- **Sondeo: la CPU consulta si los periféricos están disponibles para realizar una transferencia**



POLLING

- Como observamos en la figura, la ronda de consultas consume un tiempo fijo que crece con el número de dispositivos
- Al tiempo de consulta se le ha de sumar el tiempo de transferencia siempre que ésta se realice: si ningún dispositivo pide transferencia dicho tiempo es nulo; si la piden todos es máximo

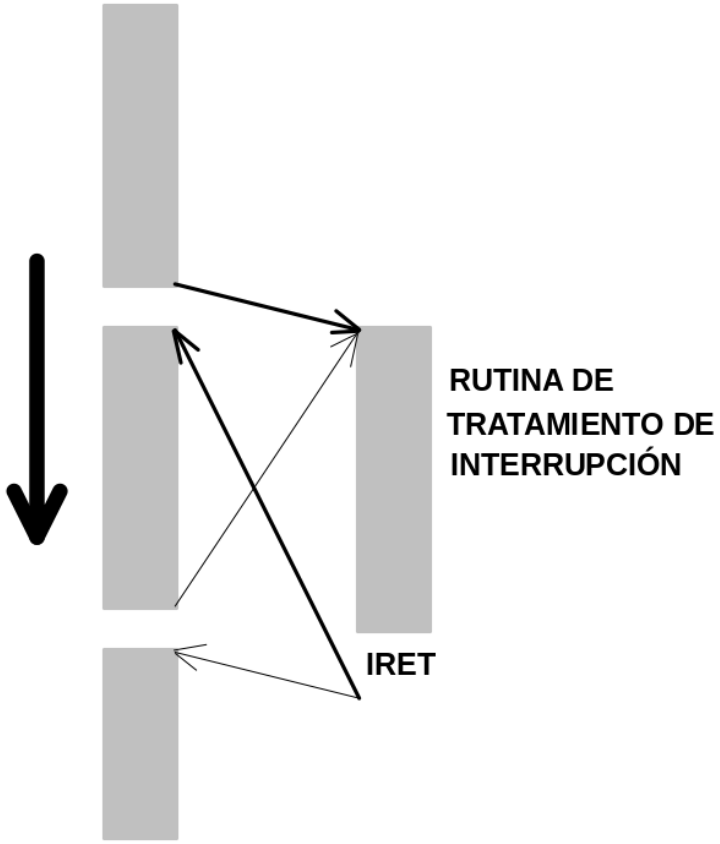


POLLING

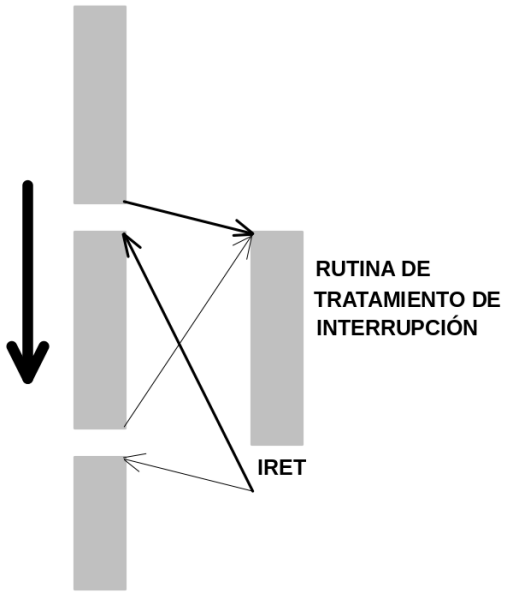
Entrada/salida por interrupción

- En las transferencias de E/S por interrupción el periférico realiza la petición de servicio
- La petición se realiza mediante una señal de control específica: se requiere soporte *hardware*
- También se conoce como interrupción *hardware* en contraposición a las interrupciones *software* de la BIOS o del sistema operativo

- **Si la interrupción es aceptada por la CPU, ésta abandona momentáneamente el programa principal para ejecutar la rutina de tratamiento de la interrupción**



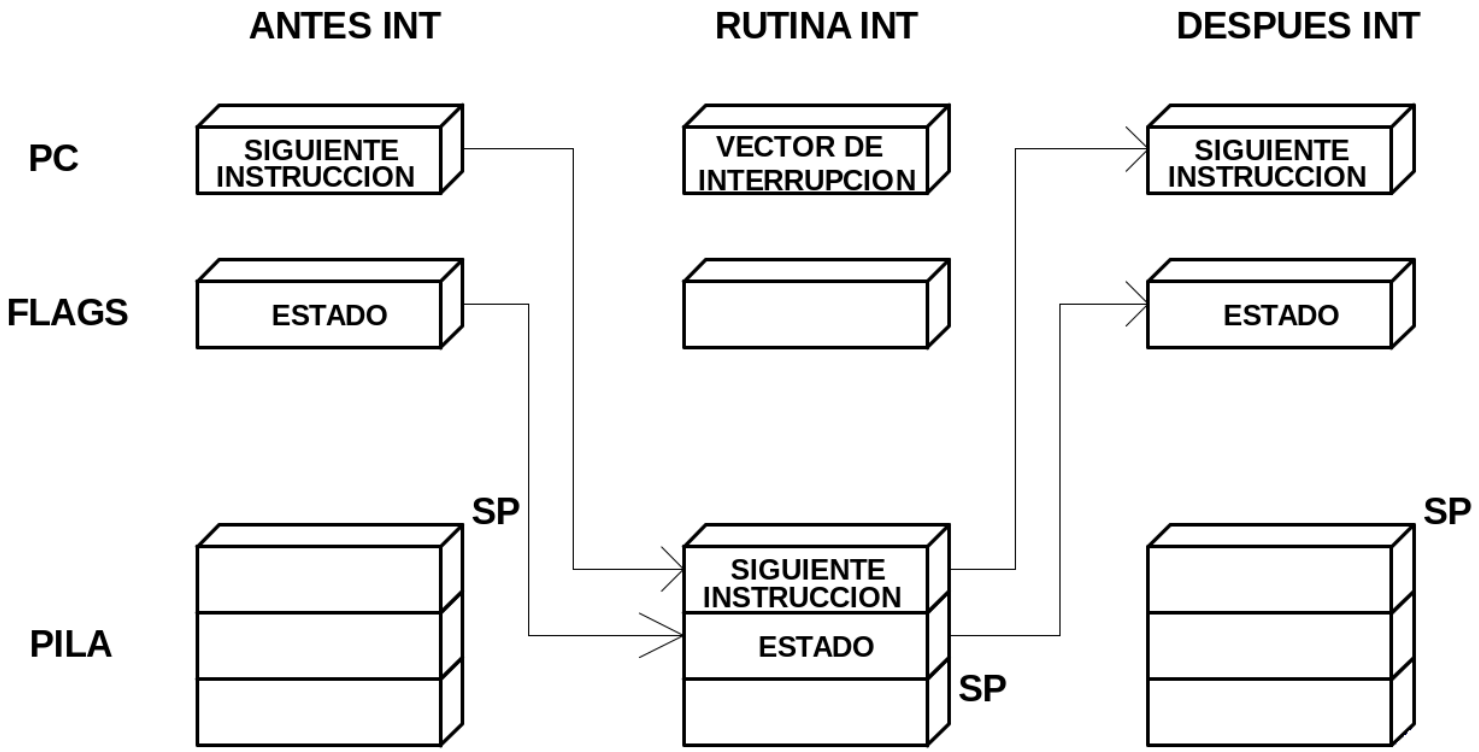
- La sincronización mediante interrupción solamente consume tiempo si se necesita hacer la transferencia. En caso contrario, el procesador dedica todo el tiempo a la tarea principal
- No obstante, la transferencia se realiza mediante la ejecución de código



- **Secuencia:**
  1) **El periférico envía una señal *hardware* de solicitud al procesador**
  2) **El procesador termina la ejecución en curso (o alcanza un punto dado del cauce) antes de responder a las solicitudes de interrupción, es decir, pasa un tiempo**
     - ➔ Se toma un promedio estadístico conocido como tiempo de latencia de atención a interrupciones
  3) **El procesador comprueba si hay interrupciones; selecciona aquella que va a atender en caso de tener varias; emite la señal *hardware* de reconocimiento**
  4) **Se salva el contexto (estado, dirección de la siguiente instrucción, etc.)**

5) **Se transfiere el control a la rutina de servicio**
   - ➔ Puede haber:
     - Un único programa
     - Uno por cada línea de interrupción
     - Uno por cada dispositivo
     - Uno por cada tipo de interrupción
   - ➔ El procesador debe disponer de la información necesaria para encontrar la rutina
6) **La rutina de servicio puede salvar en la pila los registros del procesador, deshabilitar la atención a interrupciones, etc. y puede reprogramar el periférico; todo esto consume un tiempo de programación**
7) **Efectúa la transferencia (el tiempo de transferencia dependerá de la cantidad de datos)**
8) **La rutina finaliza recuperando la información salvada y devolviendo el control al proceso en curso**

- **Antes de comenzar la rutina de tratamiento de la interrupción la CPU debe salvar la dirección de retorno y el registro de estado**



- **Como vemos, el procesamiento de la interrupción tiene aspectos *hardware* y *software*:**

  - ***Hardware*** → **requiere soporte físico**

  - ***Software*** → **ejecución de instrucciones, lo cual es intrínsecamente lento**

- **Para poder manejar la E/S por interrupción hemos de diseñar mecanismos *hardware* que den soporte a las situaciones siguientes:**
  - **Aceptación o no de la petición de interrupción**
  - **Ubicación de la rutina de servicio**
  - **Identificación de la rutina de servicio**
  - **Conexión de varios periféricos con capacidad de interrumpir y gestión de prioridades en caso de peticiones simultáneas**

- **Aceptación de interrupciones:**
  - **Las interrupciones pueden ser:**
    - **enmascarables**: se pueden dejar de atender por procedimiento *software*
    - **no enmascarables**: siempre son atendidas

- **En un computador PC el microprocesador cuenta con dos líneas diferentes de petición de interrupción -INT y NMI- donde la primera atiende peticiones enmascarables con el *flag* de interrupción IF y la segunda no puede dejar de atender las peticiones de interrupción**

- **Ubicación de la rutina de servicio:**

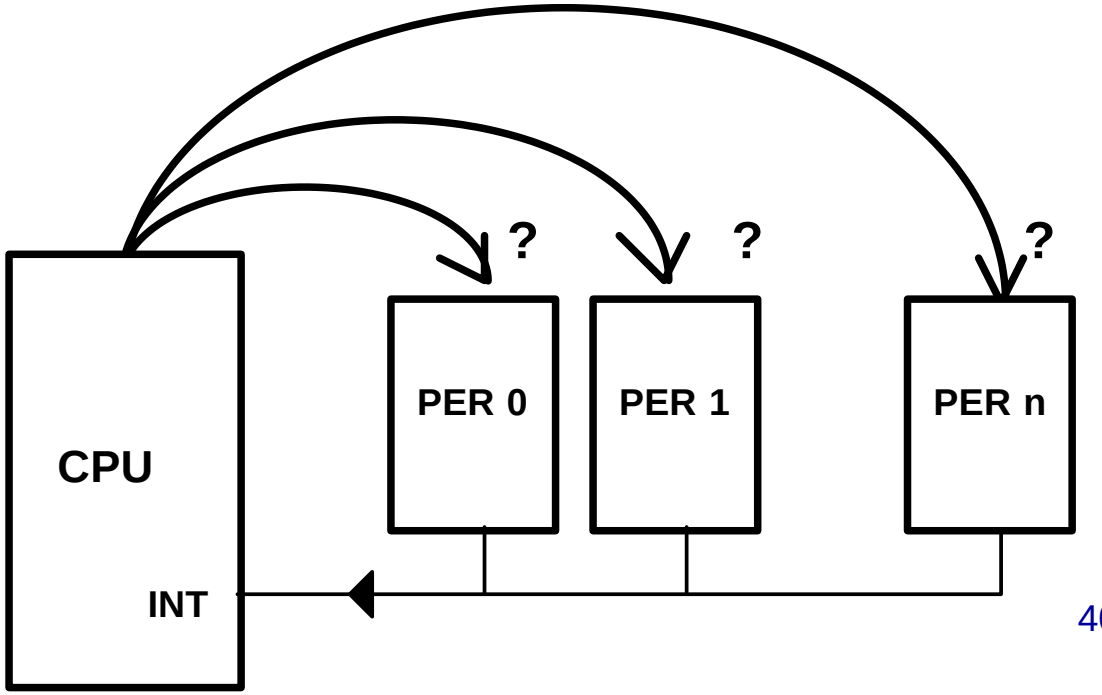
  - **La rutina de servicio es el programa que realiza la transferencia de datos**

  - **Puede ser:**
    - Un procedimiento de una aplicación en ejecución actualmente
    - Un programa TSR (*Terminate and Stay Resident*) que se deja instalado en memoria esperando que sea llamado
      - Los *drivers* de los periféricos, entre otras cosas, dejan instalados este tipo de programas

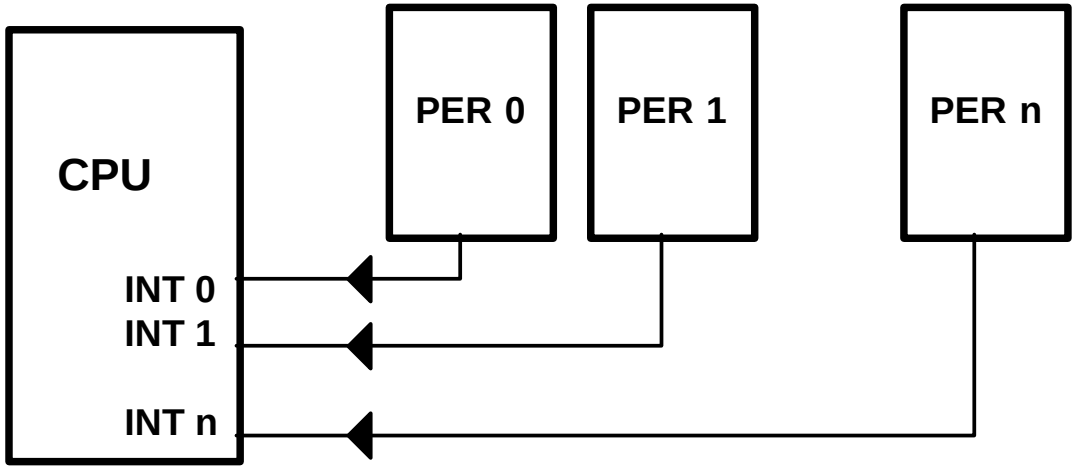
- **Identificación de la rutina de servicio:**
  - **La ubicación de la rutina de servicio se puede hacer:**
    - Dando su dirección de inicio
    - Dando un código que servirá de entrada en una tabla
      - Este método es más flexible
      - Se suele hablar de niveles de interrupción (códigos) y de vectores (direcciones)
  - **La rutina de servicio la puede pedir:**
    - El procesador
      - Fija
      - Cuando tiene varias entradas en función de la utilizada
    - El propio periférico
    - Un controlador interpuesto entre procesador y periférico

- **Conexión de varios periféricos:**
  - **Ya que un sistema de E/S cuenta con diferentes periféricos capaces de generar interrupciones, hemos de ver como conectarlos con la CPU**
  - **También hemos de resolver la situación en la que varios de estos dispositivos realizan una petición de interrupción al mismo tiempo**
    - Es necesario establecer prioridades para decidir a cual de ellos se atiende en primer lugar
  - **Finalmente, hemos de determinar en cada caso la ubicación de la rutina de servicio**
    - Diversos métodos que pueden mezclarse

- **CPU con una sola línea de interrupción:**
  - ➔ **Todos los periféricos hacen la petición en la misma línea**
  - ➔ **Una sóla rutina de servicio**
  - ➔ **La rutina de interrupción identifica mediante sondeo el periférico que interrumpió a la CPU y desactiva la petición**
  - ➔ **La prioridad se determina por el orden en que se realiza el sondeo**

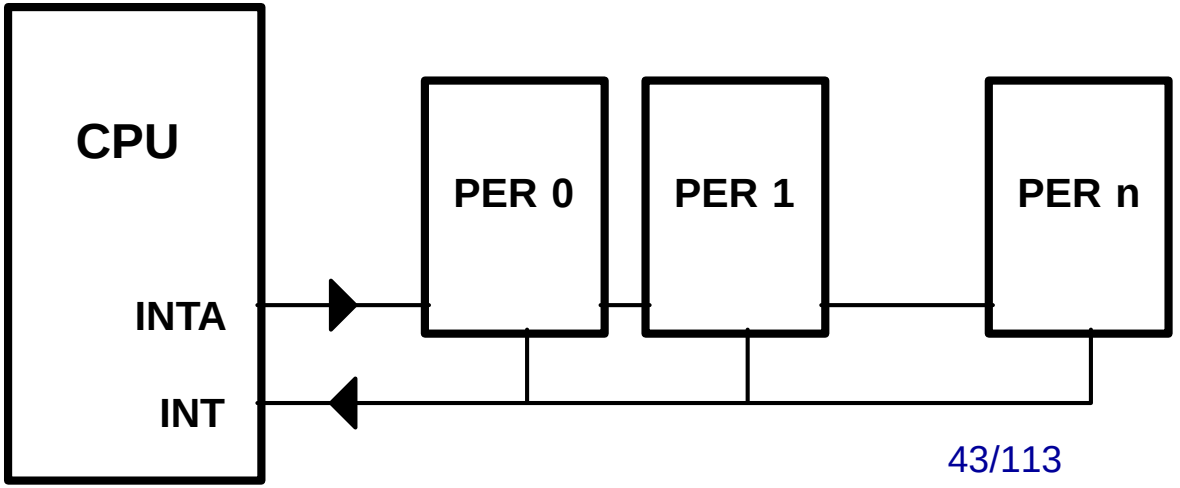


- **CPU con varias líneas de interrupción:**
  - ➔ **Cada periférico hace la petición por una línea distinta**
  - ➔ **Cada línea tiene una rutina de servicio distinta**
  - ➔ **La prioridad la determina internamente la CPU para cada línea**

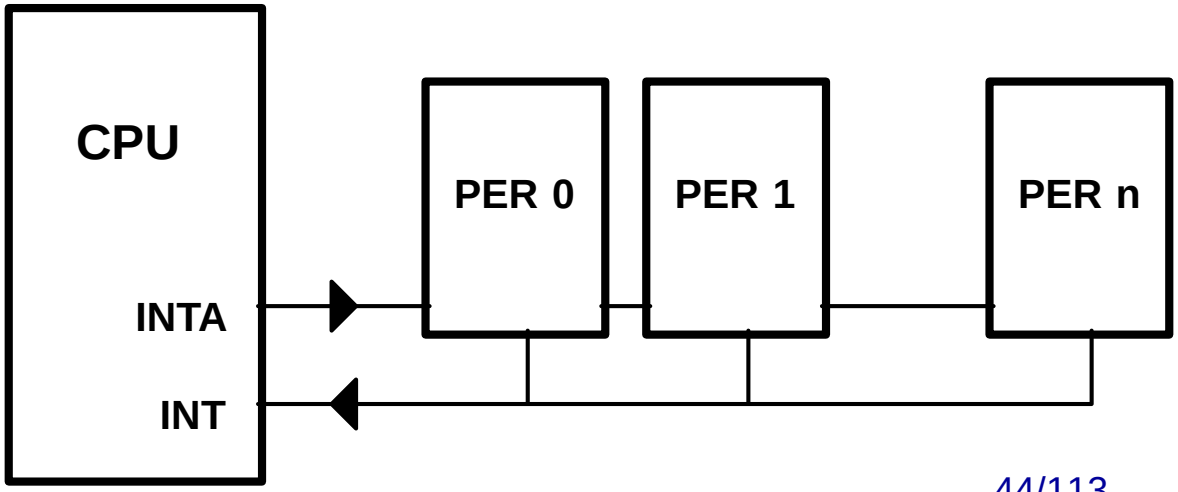


- CPU con líneas de petición de interrupción y de aceptación:
  - en este caso, la CPU cuenta con una línea de entrada INT para las peticiones de interrupción y con una línea de salida INTA para dar los reconocimientos de interrupción
  - existen varias soluciones:
    - encadenamiento de periféricos o *daisy-chain*
    - interrupciones vectorizadas
    - gestión centralizada de interrupciones mediante controlador

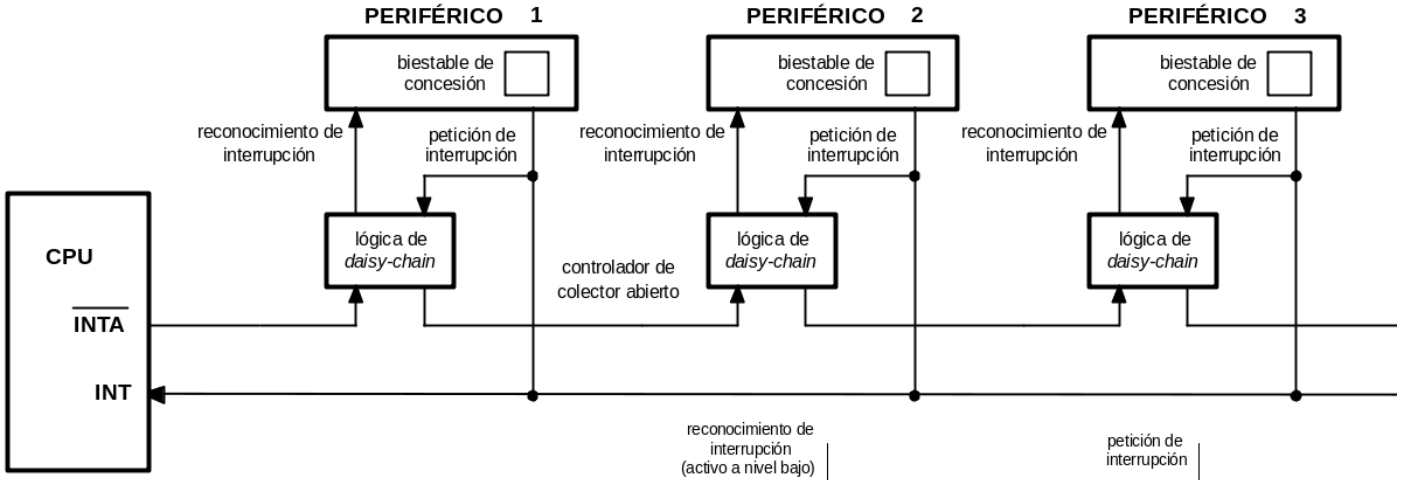
- **Encadenamiento de periféricos**
  - ➔ Todos los periféricos piden servicio por la misma línea
  - ➔ Al aceptar la interrupción (INTA), el periférico de mayor prioridad desactiva la petición y activa un biestable de concesión (habrá que ver cómo se asignan prioridades)
  - ➔ A continuación, se identifica por sondeo el periférico que ha interrumpido observando el biestable de concesión y borrándolo a la vez

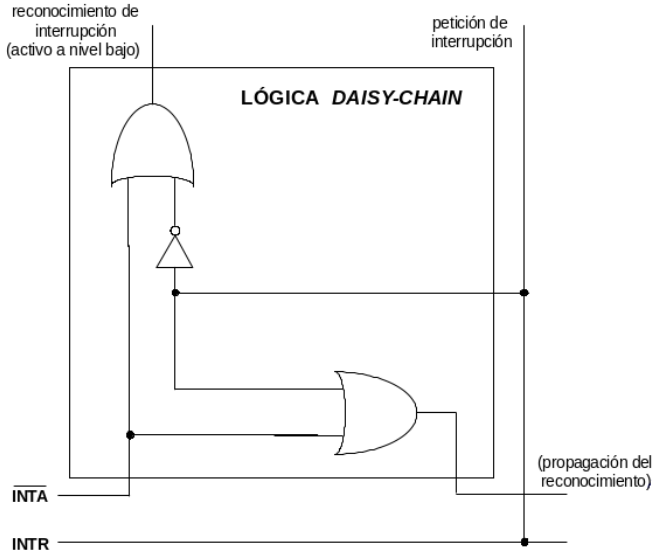


- **Encadenamiento *daisy-chain* (I)**
  - ➔ **La señal de aceptación (INTA) se propaga de periférico en periférico hasta que llega al primero que ha interrumpido; la señal INTA deja de propagarse y activa un biestable de concesión**
  - ➔ **Rutina de servicio única y resolución de prioridades por *daisy-chain***



- **Encadenamiento *daisy-chain* (II)**



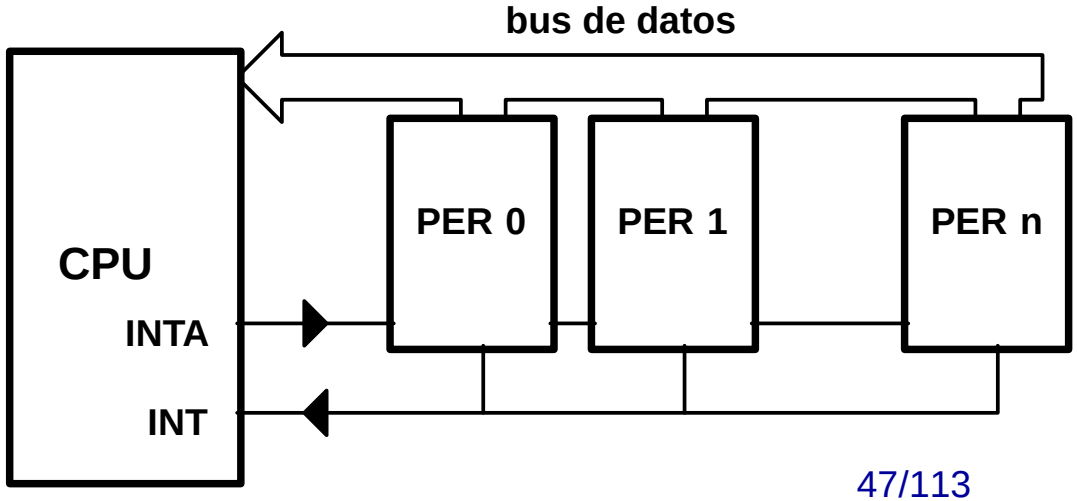


La lógica daisy-chain captura la señal INTA (activa a nivel bajo) si el periférico ha hecho una petición de interrupción (activo a nivel alto) impidiendo que se propague al siguiente periférico.

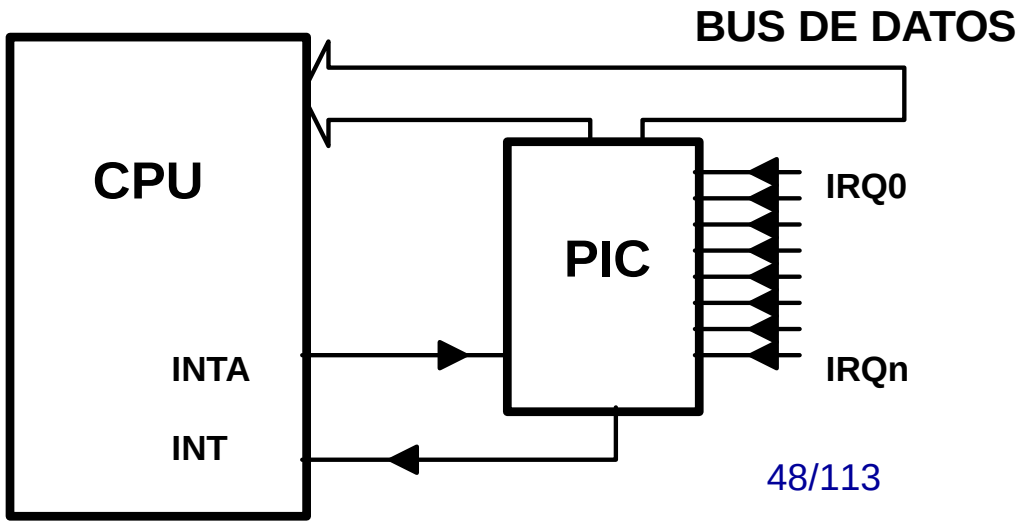
- **Encadenamiento *daisy-chain* (III)**

  ➔ **Es importante el orden en el que se colocan los periféricos en la cadena de reconocimiento; si ponemos periféricos muy "avariciosos" cerca del procesador van a impedir que se atiendan las solicitudes de E/S de los que están más alejados**

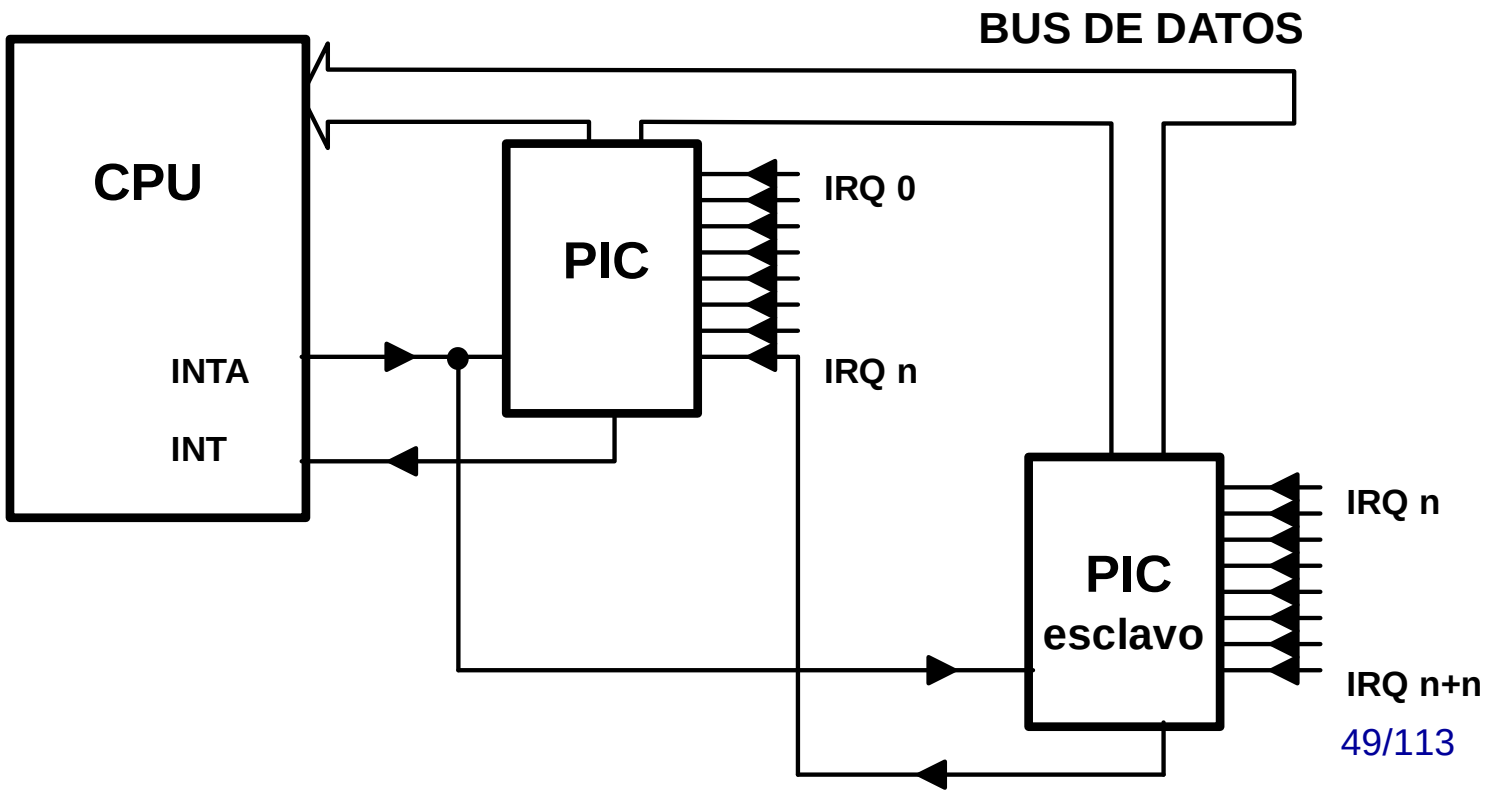
- **Interrupciones vectorizadas**
  - ➔ Todos los periféricos piden servicio por la misma línea
  - ➔ Cuando la CPU reconoce la interrupción, el periférico que ha interrumpido se identifica poniendo en el bus de datos el vector de interrupción (dirección) o una codificación del mismo (nivel)
  - ➔ La resolución de prioridades se resuelve por algún mecanismo programable



- **Gestión de interrupciones mediante controlador PIC**
  - **Permite la conexión de distintos periféricos a la línea de petición de la CPU expandiéndola en varios niveles**
  - **Resuelve el tema de las prioridades de forma flexible gracias a la programación de distintos mecanismos:**
    - prioridad fija (por orden)
    - línea prioritaria la que lleva más tiempo sin atenderse
  - **Permite enmascarar líneas momentáneamente gracias a un registro de máscara programable**

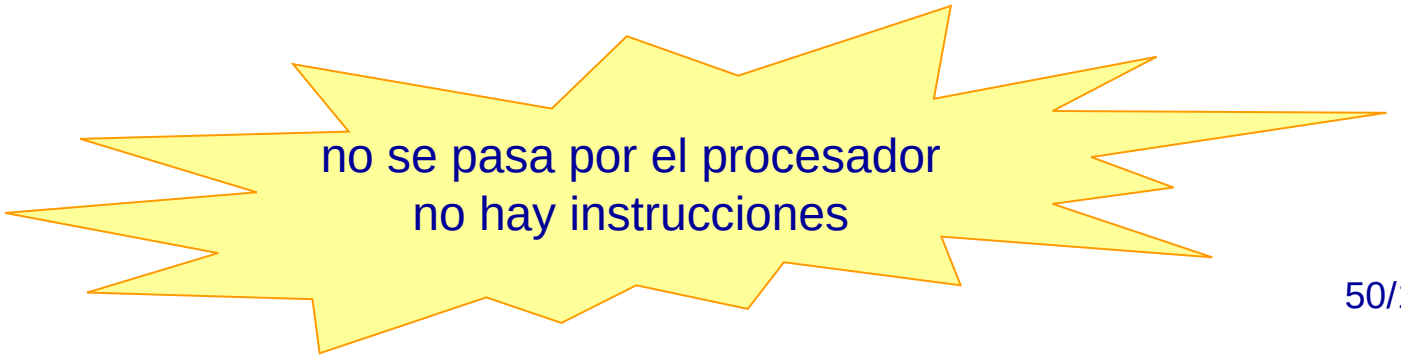


- **Características del PIC (controlador programable de interrupciones)**
  - ➔ **Admite controladores "esclavos" con lo que se pueden expandir aún mas los niveles de interrupción**



Acceso directo a memoria

- En el acceso directo a memoria o DMA (*Direct Memory Access*) el controlador del periférico se comunica directamente con la memoria principal del computador
- La CPU no realiza ninguna tarea, tanto la inicialización como la transferencia propiamente dicha son gobernadas por el periférico
- La transferencia de E/S, por tanto, no se lleva a cabo mediante instrucciones sino por la activación de las señales de control de acceso a memoria



- **Este tipo de transferencia de E/S tiene sentido solamente cuando se mueven bloques de datos ya que conlleva una sobrecarga temporal asociada al controlador DMA**

- **Requiere una programación previa del controlador de DMA que consume un tiempo**
  - Cantidad de datos a transferir
  - Posición de memoria
  - Puerto de E/S
  - Modo de transferencia

- **Secuencia:**
  1) **El periférico envía una señal *hardware* de solicitud al procesador**
  2) **El procesador termina un ciclo de bus (no la ejecución de una instrucción) antes de responder a las solicitudes de DMA**
     - ➔ Latencia de atención a DMA
  3) **El procesador emite la señal *hardware* de reconocimiento y cede los buses al controlador de DMA**
     - ➔ NO se salva el contexto

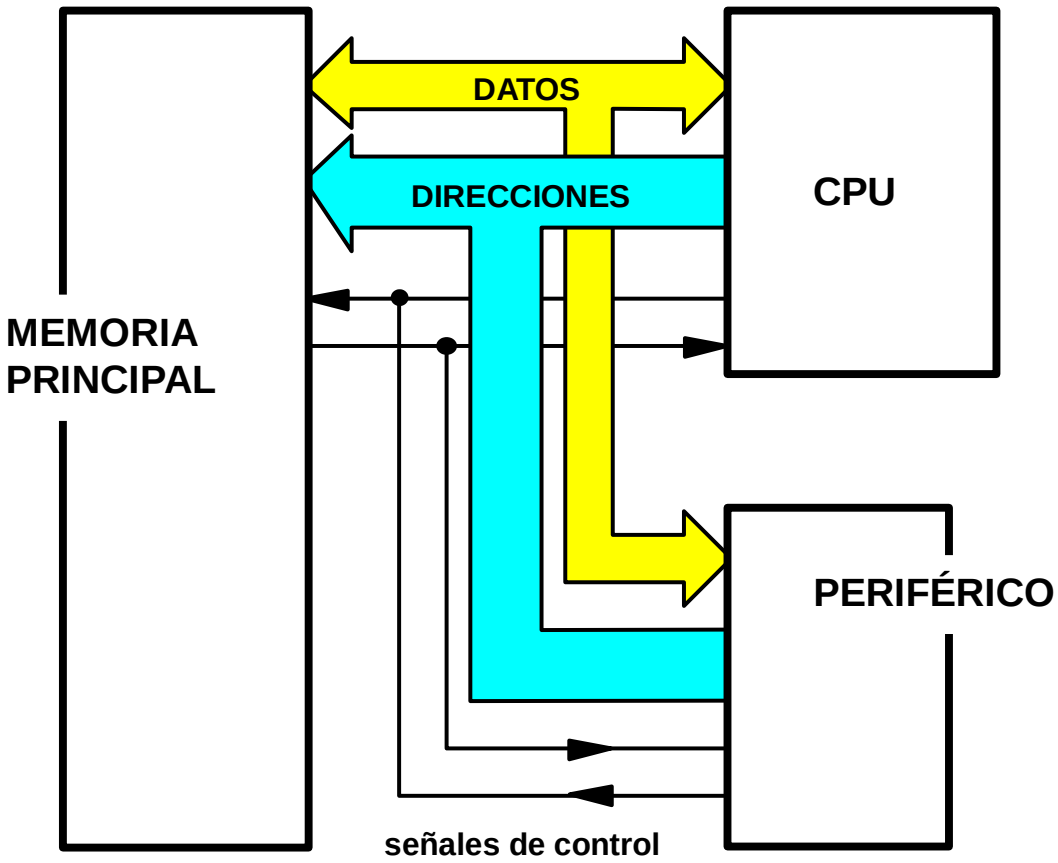
4) **Se programa el controlador de DMA mediante una interrupción (consume tiempo de programación)**
   - ➔ Cantidad de datos a transferir
   - ➔ Posición de memoria
   - ➔ Puerto de E/S
   - ➔ Modo de transferencia

4) **Se realiza la transferencia (tiempo de transferencia)**

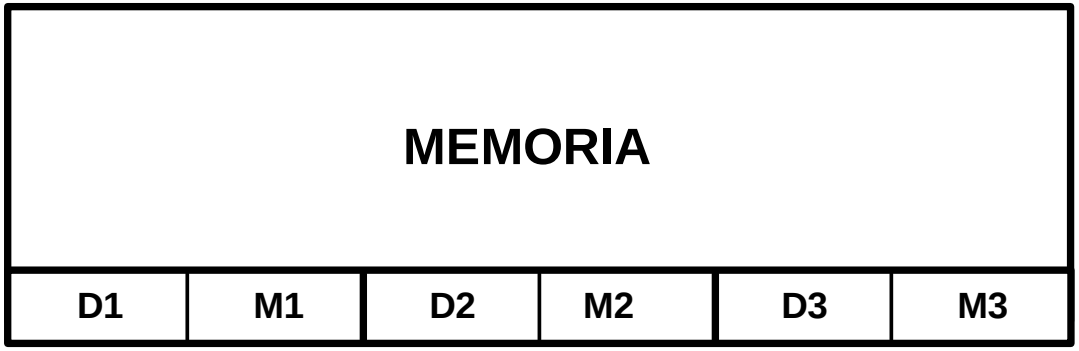
5) **El final de la transferencia se notifica mediante una interrupción (otra fuente de consumo de tiempo)**

Acceso directo a memoria

- **Soporte *hardware***
  - **Existen dos formas básicas de realizar el acceso directo a memoria:**
    - por memoria multipuerta
    - por robo de ciclo

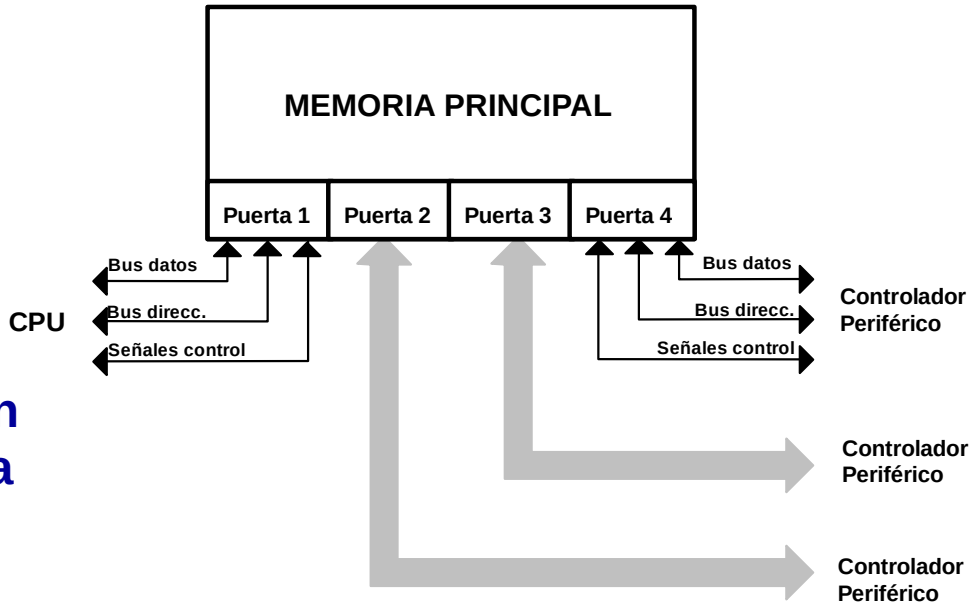


- **Acceso directo a memoria multipuerta (I)**
  - ➔ una memoria multipuerta es aquella que tiene posibilidad de realizar transferencias simultáneas con el exterior por medio de varias puertas
  - ➔ a cada puerta se le asigna un registro de datos y otro de direcciones que indicarán el dato transferido y la dirección de transferencia respectivamente



- **Acceso directo a memoria multipuerta (II)**
  - ➔ **La memoria multipuerta se basa en estos principios básicos:**
    - ➔ Debe estar dividida internamente en bloques, que puedan operar de forma independiente y simultánea
    - ➔ Existirá una colisión o conflicto de acceso cuando se soliciten simultáneamente accesos a un mismo bloque por dos o más puertas. Ha de estar previsto un mecanismo que controle las prioridades de acceso en estas circunstancias

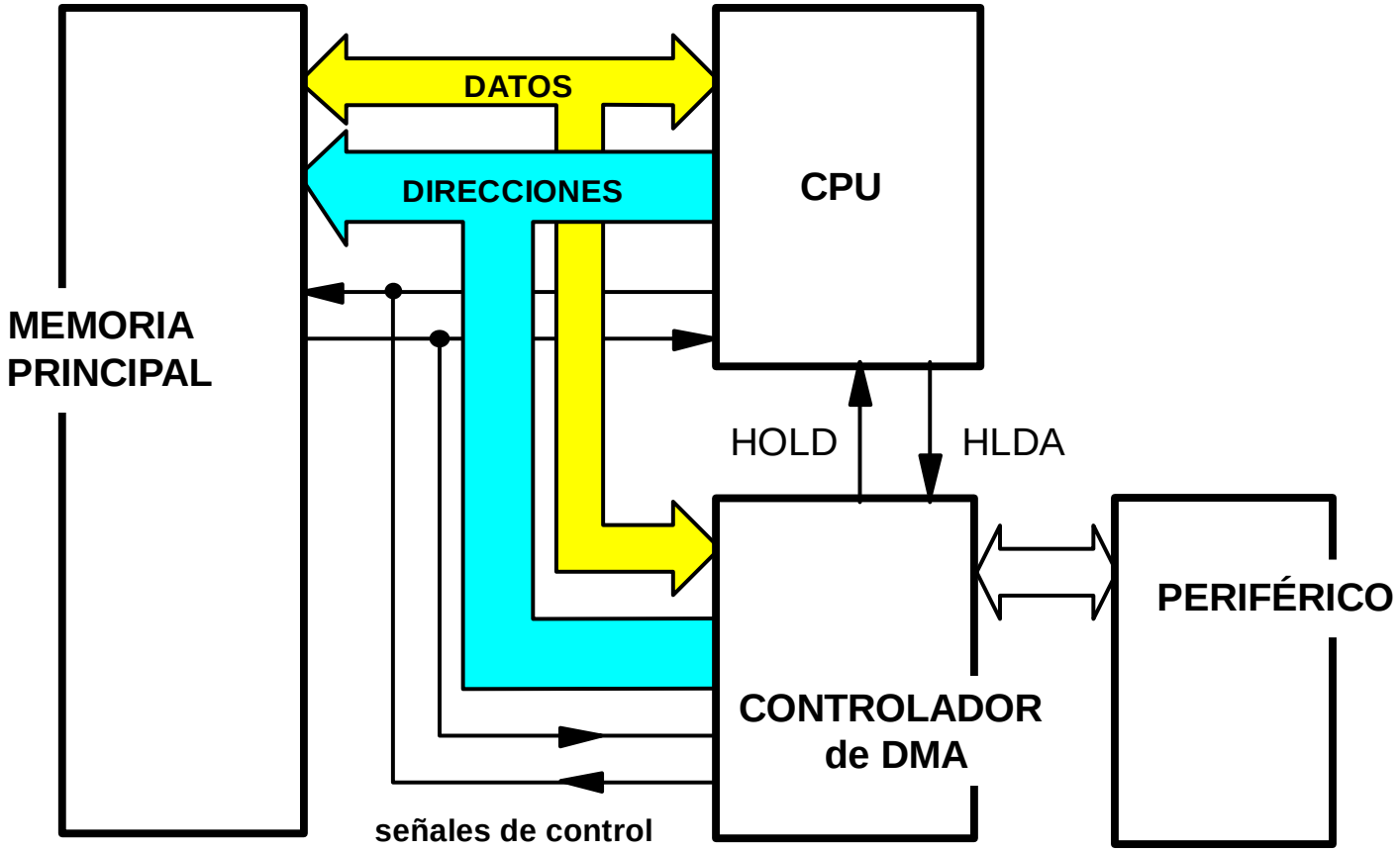
- **Acceso directo a memoria multipuerta (III)**
  - El DMA cuando la memoria es multipuerta permite a los periféricos la manipulación de memoria principal sin intervención de la CPU sin más que conectarse a una de las puertas
  - Ejemplos: memoria de video (VRAM); *mail box* de comunicación entre procesadores



- **Acceso directo a memoria por robo de ciclo (I)**
  - ➔ **La memoria tiene una sola puerta que es compartida por el controlador de DMA y la CPU**
  - ➔ **Siempre que el periférico requiera una transferencia DMA debe hacer una petición de robo de ciclo a la CPU**
  - ➔ **Cuando la CPU hace la concesión, permite que el controlador de DMA gobierne los buses de direcciones y datos y las señales de control de memoria y de E/S**

- **Acceso directo a memoria por robo de ciclo (II)**

Acceso directo a memoria



- **Acceso directo a memoria por robo de ciclo (III)**

  - **Modos:**

    - **DMA por robo de ciclo**
    - **DMA por ráfaga**
    - **DMA transparente**

- DMA por robo de ciclo → se transfiere una palabra de datos y se devuelve el control al procesador que ejecuta una instrucción y pasa el control al DMA; así sucesivamente hasta completar el bloque

  → Se consigue una alta disponibilidad del bus del sistema para la CPU (la interferencia con la CPU es muy baja), aunque la transferencia de los datos será considerablemente lenta

  → Este modo prima la ejecución de la tarea principal sobre la E/S

Acceso directo a memoria

- DMA por ráfaga → consiste en enviar el bloque de datos solicitado mediante una ráfaga, ocupando el bus del sistema hasta finalizar la transmisión completa de la E/S
  - → Se consigue la máxima velocidad de E/S
  - → La CPU no podrá usar el bus durante todo ese tiempo, por lo que permanecería inactiva
  - → Se prima la E/S sobre la ejecución de la tarea principal

- DMA transparente → es como el robo de ciclo salvo que cuando el control de los buses lo tiene el DMA, si el procesador encuentra una instrucción que no requiere el uso de los buses, la ejecuta (digamos "furtivamente") con lo que consigue avanzar en la tarea principal

  - → La velocidad de transferencia es baja

  - → Este modo prima la ejecución de la tarea principal sobre la E/S y la refuerza respecto al robo de ciclo propiamente dicho

Entrada/salida
4. Representación visual

- **La representación visual afecta al sentido de la vista**
  - ➔ **Es el sentido humano más perfecto y evolucionado**
  - ➔ **Se usa tanto en video como impresa**
  - ➔ **Está implícita en los formatos de ficheros gráficos y de video**

- **Existen dos MODOS DE VÍDEO:**

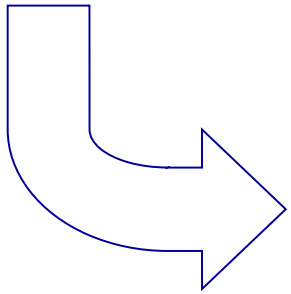
  - **TEXTO → pantalla dividida en celdas formada por intersección de 25 filas y 40 u 80 columnas**
    - En cada celda se presenta un carácter

  - **GRÁFICO → pantalla dividida en puntos o pixels**
    - Cada píxel representa un color

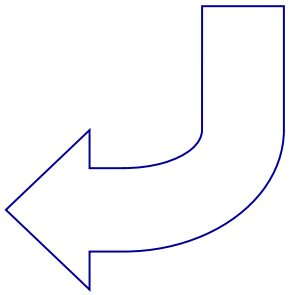
Modo texto

- Cada celda presenta un carácter mediante dos bytes:
  - Byte de carácter; y
  - Byte de atributo:

7	6	5	4	3	2	1	0
parpadeo	color fondo			intensidad	color carácter		



000	Negro
001	Azul
010	Verde
011	Cyan
100	Rojo
101	Magenta
110	Marrón
111	blanco



- **Paleta de color en modo texto**

  - ➔ **8 colores combinación de los 3 primarios**
    - ➔ Cada primario tiene valor 0 ó 1

  - ➔ **8 colores con sobreintensidad que dan lugar a una paleta basada en la anterior pero con tonalidades más claras**
    - ➔ Sólo para el caracter

Value	*Color*
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Light Gray
8	Dark Gray
9	Light Blue
10	Light Green
11	Light Cyan
12	Light Red
13	Light Magenta
14	Yellow
15	White

- La pantalla de video que vemos en cada instante está almacenada en un área de memoria (memoria de video)

  ➔ Por cada celda tenemos 2 bytes (carácter y atributo)

  ➔ La controladora de video lee la memoria de video cada vez que refresca la imagen en el monitor

Modo gráfico

- **Cada pixel representa un color codificado**

  - ➔ **La codificación del color se fundamenta en la descomposición en colores primarios**

    - ➔ La obtención de cualquier color, tanto en video como impresa, se basa en la composición de primarios

    - ➔ La codificación de cada color primario es un problema de representación de conjuntos

- **Un poco de teoría del color**

  - **Existen 2 formas de considerar los colores según la fuente de la luz**

    - Luz reflejada → colores sustractivos

    - Luz emitida → colores aditivos

- **Sustractivo** hace referencia a la percepción de los colores al reflejarse la luz sobre un objeto, lo que produce que éste absorba ciertas longitudes de onda, reflejando otras, cuya mezcla va a dar lugar al color final del objeto
  - ➔ Son sustractivos porque tienden a sustraer el blanco de la luz eliminando ciertas longitudes de onda
  - ➔ Los colores primarios en este caso son el **amarillo**, el **magenta** y el **cyan** (empleados en pintura y artes gráficas)
  - ➔ Si mezclamos los 3 colores sustractivos primarios nos dará un sucio marrón que tiende a negro

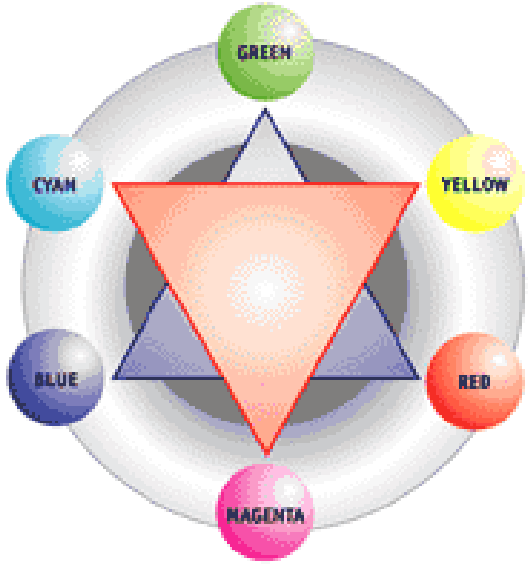
- **Aditivo: los colores se perciben como longitudes de onda que pasan directamente a nuestros ojos, sin ser reflejadas previamente por ningún cuerpo, como es el caso de la luz directamente percibida de una fuente luminosa (bombilla, pantalla)**

  ➔ **Los colores primarios pasan a ser entonces el rojo, el verde y el azul**

  ➔ **La suma de los 3 da el blanco, y la ausencia de los 3 el negro**

- **A partir de la mezcla de colores primarios obtenemos los colores secundarios, y mezclando estos obtenemos los terciarios**

  ➔ **Los colores secundarios por adición se corresponden con los primarios en sustracción, y viceversa**



- **La codificación del color se realiza en base a los aditivos primarios (rojo, verde y azul) y se conoce como RGB (Red, Green, Blue)**

  - ➔ **Cada color es la resultante de combinar una determinada "cantidad" de cada primario RGB**
    - ➔ Esa "cantidad" es la intensidad

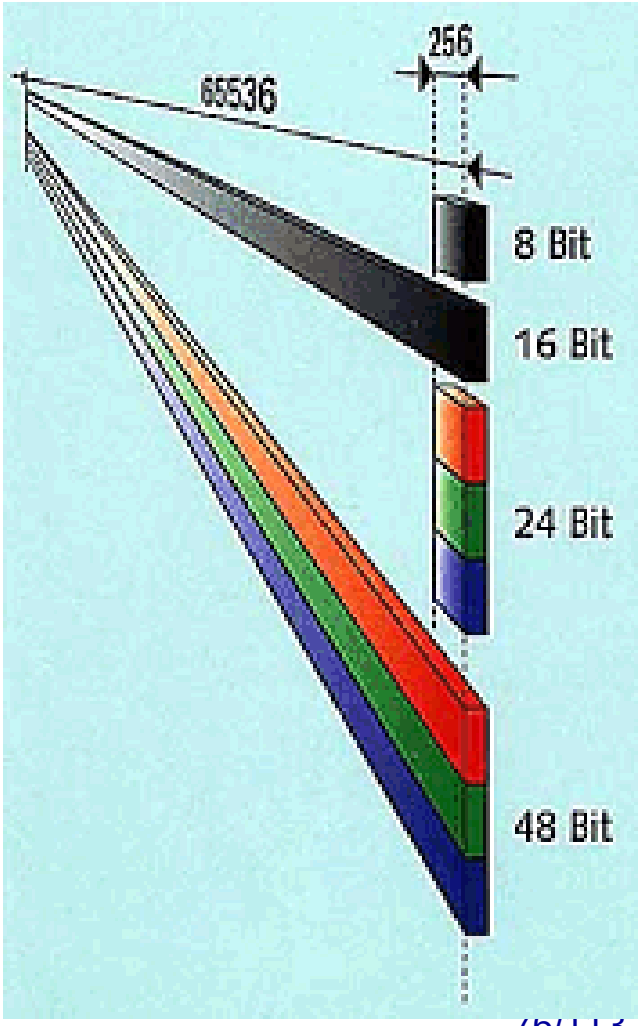
  - ➔ **Está orientada a la presentación en video**
    - ➔ Requiere una conversión para la presentación impresa

- La pregunta ahora es ¿qué variedad de colores deseamos representar?

  ➔ Esto es lo que se conoce como **PROFUNDIDAD DE COLOR**

  ➔ La cantidad de colores vendrá dada por la cantidad de intensidades diferentes que podamos distinguir en cada primario

- **Profundidad de color (I)**
  - ➔**Tamaño de la representación de la intensidad**
    - ➔En grises se construye una escala que va del blanco al negro con “cantidad” igual de RGB para cada tono
    - ➔En color se aplica a cada color primario (RGB)
    - ➔A mayor número de niveles de intensidad mayor variedad de colores



- **Profundidad de color (II)**
  - ➔ **8 bits ➔ 256 colores**
  - ➔ **16 bits ➔ 65.535 colores**
  - ➔ **24 bits ➔ 16.777.216 colores**

  - ➔ **El ojo humano puede distinguir entre 7 y 10 millones de colores diferentes**

  - ➔ **Asignando 1 byte a cada color RGB distinguimos millones de colores y usamos el tamaño normalizado por excelencia (byte)**

    - ➔ A esto se le llama **COLOR VERDADERO**

- **Profundidad de color (III)**

  ➔ **Todos los periféricos de la representación visual trabajan con color verdadero**

  ➔ **Las profundidades de color de 8 y 16 bits se construyen con “paletas de color”**

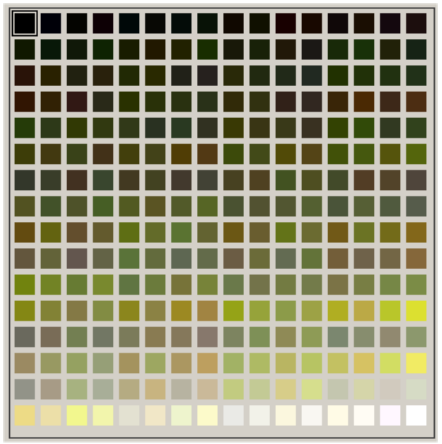
- **Color verdadero y paletas de color**
  - **Color verdadero ➔ a cada color le corresponde un código RGB**
    - ➔Rojo puro RGB(255,0,0)
    - ➔Verde puro RGB(0,255,0)
    - ➔Azul puro RGB(0,0,255)
  - **Paleta de color ➔ cada color es un índice de entrada a una tabla (paleta) que devuelve como salida un código RGB**
    - ➔Es un sistema más lento ya que requiere el acceso a una tabla
    - ➔Ahorra espacio de representación

- **Ejemplo (I):**
  - **La fotografía tiene una codificación en color verdadero de 2^{24} colores (16.777.216 colores)**
  - **Fichero en formato gráfico:**
    - Por cada píxel necesito almacenar 24 bits
    - El tamaño del fichero vendrá dado por el tamaño de la fotografía en pixels multiplicado por 24 bits

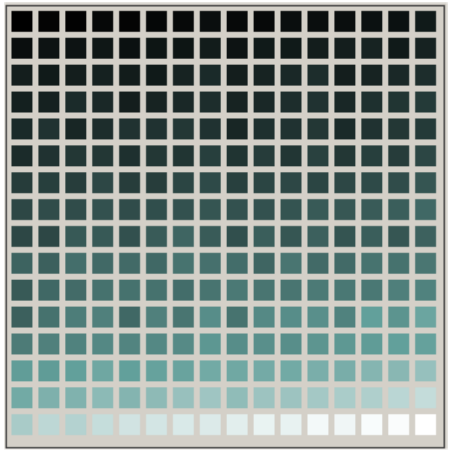
- **Ejemplo (II):**
  - **Los colores diferentes realmente usados son 59.880, es decir, 0,35% del total de los posibles**
  - **Con una 'paleta' de 64K colores necesito 16 bits por píxel más una tabla de 64Kx24 bits**
  - **El método es bueno si se cumple:**

#pixels x 24 > (#pixels x 16) + tabla

#pixels aproximado → 450 x 450



Paleta de 256 colores



Modificación de la paleta de 256 colores

- **Número de pixels o puntos que se pueden distinguir por unidad de superficie**
- **Cuanto más pequeño sea el pixel mayor es la calidad de la imagen**
  - En formatos gráficos, cuanta más información se haya almacenado más resolución tenemos

- **¿Por qué se suele hablar de resolución-profundidad como una pareja?**
  - ➔ **A mayor profundidad de color es necesario un mayor espacio de representación**
  - ➔ **A mayor resolución hemos de representar un mayor número de puntos y, por tanto, necesitamos más espacio de almacenamiento**
  - ➔ **SIN EMBARGO, ¡la memoria es de tamaño fijo para ambas!**
    - ➔ Se ha de repartir entre la profundidad de color y la resolución
    - ➔ Si aumentamos la resolución perdemos posiblemente profundidad de color

- **Dos enfoques**
  - ➔ **Mapas de bits**
    - ➔ Salvamos la información de color de cada pixel
      - Gran volumen de información
      - Resolución fija
  - ➔ **Vectorial**
    - ➔ Describimos una imagen como una serie de líneas o figuras (rellenas o no)
    - ➔ Cada vector es una figura con sus parámetros y su posición dentro de la imagen
      - Ejemplo: Circle(10, x, y, red)
      - Admite cualquier resolución ya que la imagen se redibuja cada vez

- **Ventajas e inconvenientes de cada enfoque**

  - **Mapas de bits**
    - Carga de almacenamiento grande
    - Carga computacional pequeña

  - **Vectorial**
    - Carga computacional grande
    - Carga de almacenamiento pequeña

- **Los ficheros gráficos se suelen comprimir**

  - ➔ ***Run-Lenght*** **(RLE) ➔ un código (de color) repetido consecutivamente se sustituye por un único código más el número de veces que se repite**
  - ➔ **Ejemplo:**

    - abbbbbbbccddddeeddd ➔ a1b7c2d4e23d

  - ➔ Sombras y manchas de color son buenos candidatos
  - ➔ TIFF, PCX, BMP usan este método de compresión

- La memoria principal es volátil…

  ➔ Necesitamos un almacenamiento secundario capaz de salvar datos y programas

- La memoria virtual utiliza un mapa de memoria que excede el tamaño de la memoria principal…

  ➔ El almacenamiento secundario se utiliza como repositorio de páginas de memoria virtual

- **El almacenamiento masivo utiliza las siguientes tecnologías:**
  - **Magnética**
    - Discos duros
    - Discos flexibles
    - Cintas
  - **Óptica**
    - CD
    - DVD
  - **Memorias de estado sólido no volátiles**
    - Flash

- **Grabación magnética**
  - ➔ **Sobre un soporte adecuado, se deposita un material magnético**
  - ➔ **Se disponen un gran número de pequeñas áreas llamadas celdas que pueden magnetizarse independientemente en un sentido u otro**

- **Es un dispositivo de almacenamiento masivo no volátil**
  - ➔ **También: almacenamiento secundario**
  - ➔ **Pertenece al sistema de E/S y no está en comunicación directa con el procesador**
    - ➔ La información que contiene debe ser copiada en memoria principal para que se pueda realizar sobre ella cualquier tipo de tarea computacional
    - ➔ Guarda las imágenes de los programas que ejecuta la máquina

- **Es de acceso aleatorio por bloques**

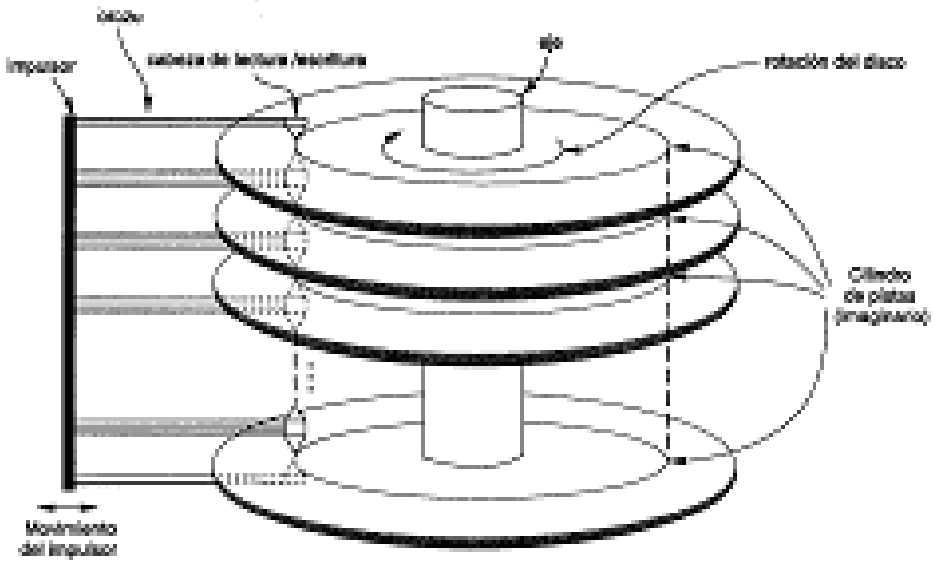
  ➔ **Se accede a cualquier bloque en cualquier orden**

  ➔ **No se accede a bytes o palabras sino a bloques completos (aunque sólo se necesite un dato de tamaño byte o palabra)**

ESTRUCTURA INTERNA

- Platos finos, generalmente de aluminio, recubiertos por un material sensible a alteraciones magnéticas

- Los discos, cuyo número varía según la capacidad de la unidad, se encuentran agrupados por un eje y giran continuamente a gran velocidad (entre 7.200 y 10.000r.p.m.)



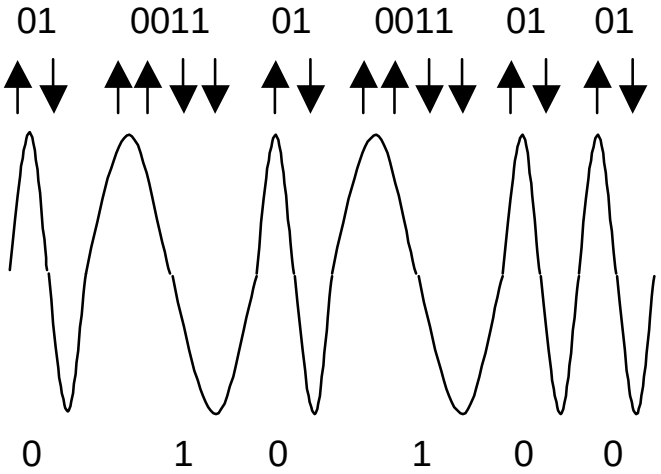
- Cada disco posee dos diminutos cabezales de lectura/escritura, uno en cada cara
  - ➔ Se encuentran flotando sobre la superficie del disco sin llegar a tocarlo, a una distancia de unas 0,1micras
  - ➔ Generan señales eléctricas que alteran los campos magnéticos del disco orientando las partículas y codificando la información en escritura
  - ➔ Durante el proceso de lectura, el campo magnético de las partículas induce una corriente en las cabezas que sirve para decodificar la información

- Las partículas orientadas magnéticamente en un sentido u otro pueden representar dos estados, 0 y 1
- Ahora bien, el sistema de lectura del cabezal se basa en la inducción de una corriente como consecuencia de la variación del campo magnético

$$\vec{I} \propto d\vec{B}$$

- Esto significa que no podemos guardar secuencias de 0 ó 1 ya que no producen variaciones del campo y por tanto no inducen corrientes en el cabezal (se pierde esa información)

- **La solución consiste en salvar códigos con transiciones, por ejemplo, podemos representar el 0 como 01 (transición rápida) y el 1 como 0011 (transición lenta). Así aseguramos que, aún cuando haya que salvar series de bits con el mismo valor, se producen continuas variaciones del campo magnético**



➔ **Esto no es nada más que una modulación en frecuencia**

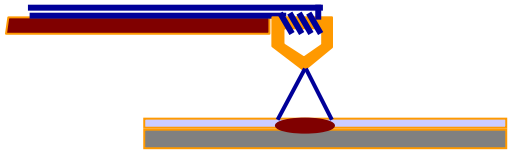
- Se requiere un proceso de modulación (MFM)

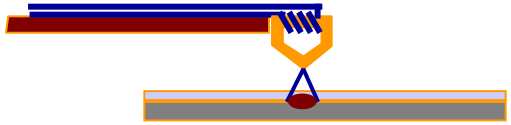
  ➔ Por tanto, es un dispositivo analógico

  ➔ Se puede almacenar más información trabajando con múltiples frecuencias

  ➔ Junto a la modulación se suele incorporar algún algoritmo de compresión (RLL)

- La distancia entre el cabezal y el plato del disco también determinan la densidad de almacenamiento del mismo, ya que cuanto más cerca estén el uno del otro, más pequeño es el punto magnético y más información podrá albergar





- **La información se almacena en la superficie del disco en círculos concéntricos muy estrechos, cada uno de los cuales tiene un diámetro distinto y recibe el nombre de pista**

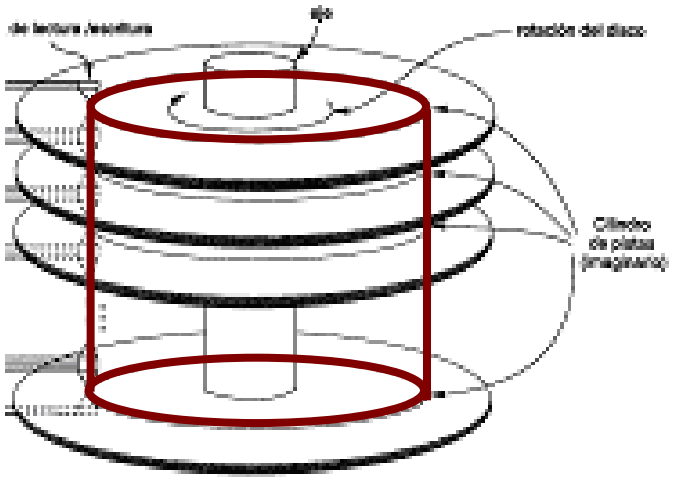
  ➔ **El acceso a una determinada pista se hace situando el cabezal de lectura/escritura en el radio adecuado**

- **Todas las pistas contienen la misma cantidad de datos de manera que hay un mayor empaquetamiento cuanto más cerca del eje nos encontremos.**

  ➔ **Dado que una pista contiene una gran cantidad de información, se suele dividir en *sectores*. Los sectores subtienden un ángulo fijo desde el eje**

- **En el caso de los paquetes de discos, las pistas del mismo diámetro en las diversas superficies constituyen un cilindro, por la forma que tendrían si se les conectara en el espacio**

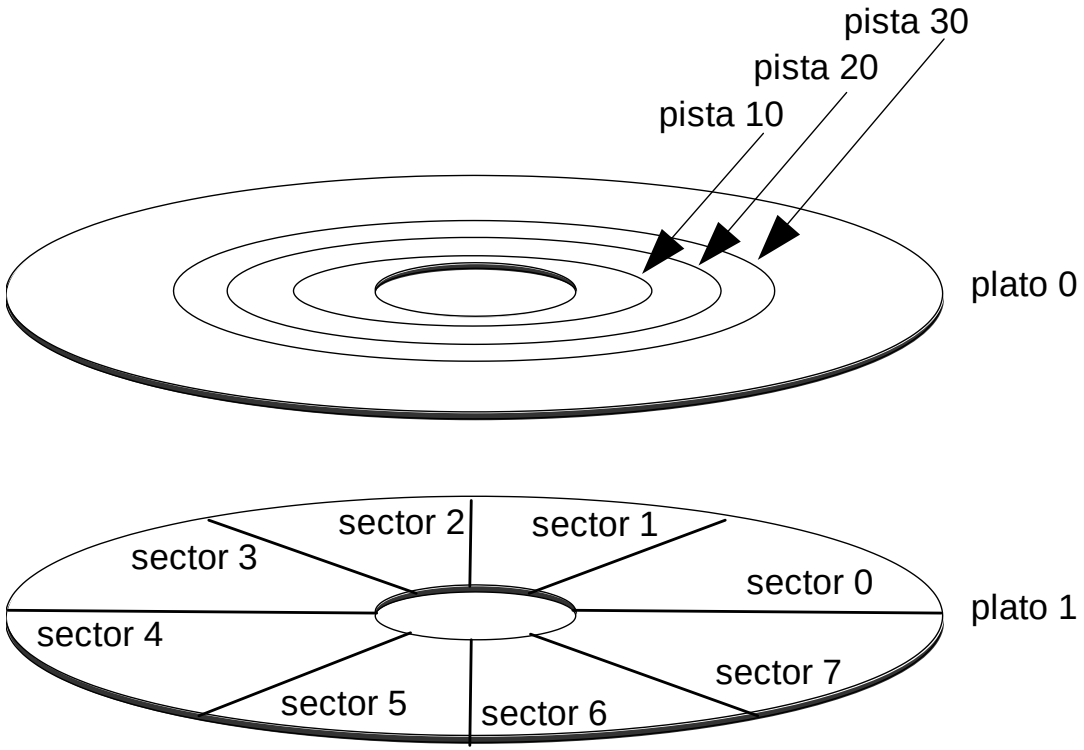
  ➔ **El concepto de cilindro es importante porque los datos almacenados en el mismo cilindro se pueden leer con mucha mayor rapidez que si estuvieran distribuidos entre distintos cilindros**

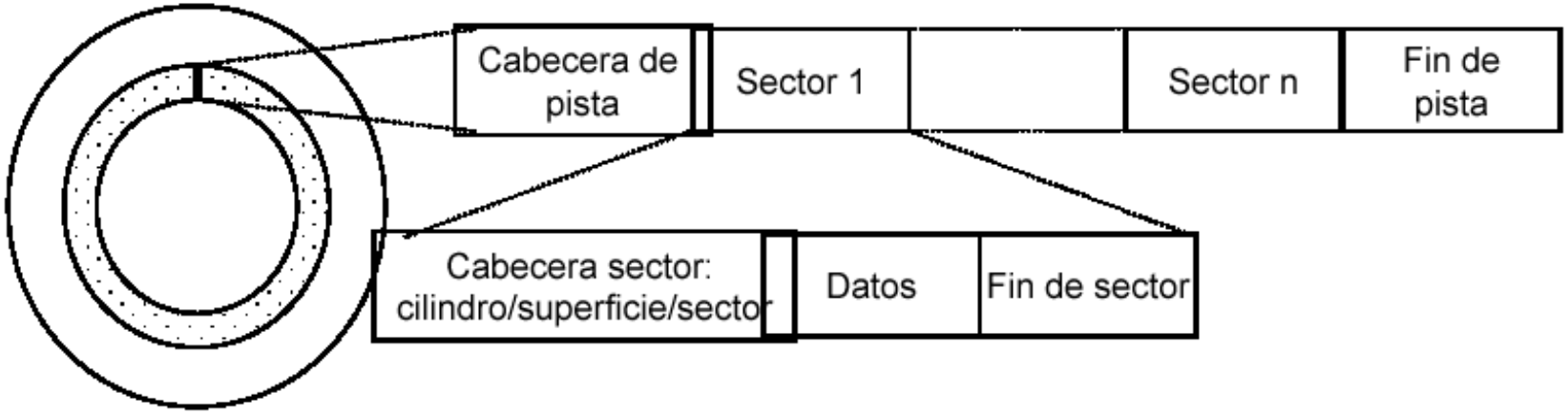


ESTRUCTURA LÓGICA

- Cada plato, cada pista y cada sector llevan un código escrito que los identifica de manera unívoca
  - ➔ Todas las pistas con el mismo código pertenecen al mismo cilindro y se diferencian unas de otras por el número de plato
  - ➔ Los sectores también se numeran y se diferencian unos de otros por el plato y la pista

- Esta información está grabada y se llama formato de bajo nivel
  - ➔ Hoy en día viene grabada de fábrica pero en otros tiempos era necesario hacerla como una tarea previa en el proceso de instalación de este tipo de dispositivos





- **Cada conjunto de datos lleva un identificador único formado por los tres códigos (plato, pista, sector)**
  - ➔ **Ahora bien, no todos los fabricantes los organizan de la misma forma. En realidad este identificador puede generarse así:**

    `número de plato, número de pista, número de sector`

  - ➔ **o así:**

    `número de sector, número de pista, número de plato`

  - ➔ **o cualquier otra combinación. En todos los casos el código es único pero el valor del mismo es distinto**

- **Los sectores suelen agruparse en *clusters* o unidades de asignación (depende del S.O.)**

  - **Un *cluster* es la mínima unidad de lectura o escritura del disco**

    - El número de sectores que forman un *cluster* es variable; lo determina el S.O. en función de la capacidad total del HD
    - Los ficheros se reparten en la cantidad de *clusters* que sea necesaria
    - Siempre queda un *cluster* final no totalmente ocupado
    - Si los *clusters* son grandes el tamaño total depreciado a lo largo de todo el disco duro puede ser considerable

- **El formato del sistema de ficheros depende del Sistema Operativo**

  - ➔FAT12
  - ➔FAT16
  - ➔FAT32
  - ➔NTFS
  - ➔ext2
  - ➔ext3
  - ➔…

- **Proceso de lectura de un fichero**
  - Buscar fichero en la tabla
  - Leer los *clusters* donde se ubica
  - Acceder a los *clusters* indicados

- **Proceso de escritura de un fichero**
  - Asignar entrada en la tabla
  - Buscar *clusters* libres
  - Escribir los *clusters* libres y anotarlos en la tabla

TIEMPO DE ACCESO

- El parámetro más usado para medir la velocidad de un disco duro es el tiempo de acceso, que en realidad es la suma de varios factores:

  ➔ el tiempo medio de búsqueda;

  ➔ la latencia o retardo rotacional; y

  ➔ el tiempo de transferencia

- Ubicar mecánicamente la cabeza de lectura/escritura sobre la pista correcta ➔ tiempo de búsqueda
- Después, hay otro lapso de espera mientras el principio del sector deseado gira hasta colocarse bajo la cabeza de lectura/escritura ➔ retardo rotacional o latencia
- Por último, se requiere un lapso adicional para transferir los datos ➔ tiempo de transferencia de bloque

- El tiempo de búsqueda y el retardo rotacional suelen ser mucho mayores que el tiempo de transferencia de bloque

- **Para hacer más eficiente la transferencia de múltiples sectores, se acostumbra transferir varios sectores consecutivos de la misma pista o cilindro**
  - Esto elimina el tiempo de búsqueda y el retardo rotacional para todos los sectores, con excepción del primero, lo que da pie a un ahorro sustancial de tiempo cuando se transfieren muchos sectores contiguos
  - Por lo regular, el fabricante del disco indica una velocidad de transferencia masiva para calcular el tiempo que requiere la transferencia de sectores consecutivos

- El tiempo requerido para localizar y transferir un sector es del orden de milisegundos (entre 15 y 60 ms)
- En el caso de sectores contiguos, el primer bloque tarda entre 15 y 60 ms, pero la transferencia de los siguientes podría tardar sólo 1 ó 2 ms por bloque
- Un tiempo de transferencia de milisegundos se considera bastante alto en comparación con el tiempo de acceso a la memoria principal (entre 10 y 60 ns)
- Por tanto, la localización de datos en el disco es un *cuello de botella* importante (especialmente en aplicaciones que lo requieren por esencia como las bases de datos)