GESTIÓN DE MEMORIA

Universidad San Pablo-CEU
Escuela Politécnica Superior
Rodrigo García Carmona

OBJETIVOS

- Conocer las características de la **memoria principal** y la forma en la que los procesos se ubican en ella.
- Entender el propósito de un **gestor de memoria** y conocer las **funciones** que realiza.
- Conocer los diferentes **modelos de gestión de memoria**:
  - Particionamiento fijo.
  - Particionamiento dinámico.
  - Paginación.
  - Segmentación.
- Entender en qué consiste la **memoria virtual** y las ventajas que trae consigo.
- Comprender el **principio de localidad**.
- Analizar la memoria virtual basada en paginación.
- Ser consciente de las **limitaciones** y los problemas a resolver en un esquema de **memoria virtual con paginación**:
  - Velocidad de acceso.
  - Tamaño de la tabla de páginas.
  - *Trashing*.
  - Compartición de páginas.

CONTENIDO

- Memoria y procesos
- Gestores de memoria
- Particionamiento fijo
- Particionamiento dinámico
- Paginación
- Segmentación
- Paginación y segmentación
- Memoria virtual
- Paginación en memoria virtual
- Problemas de la memoria virtual
  - Velocidad
  - Espacio
  - *Trashing*
  - Compartición

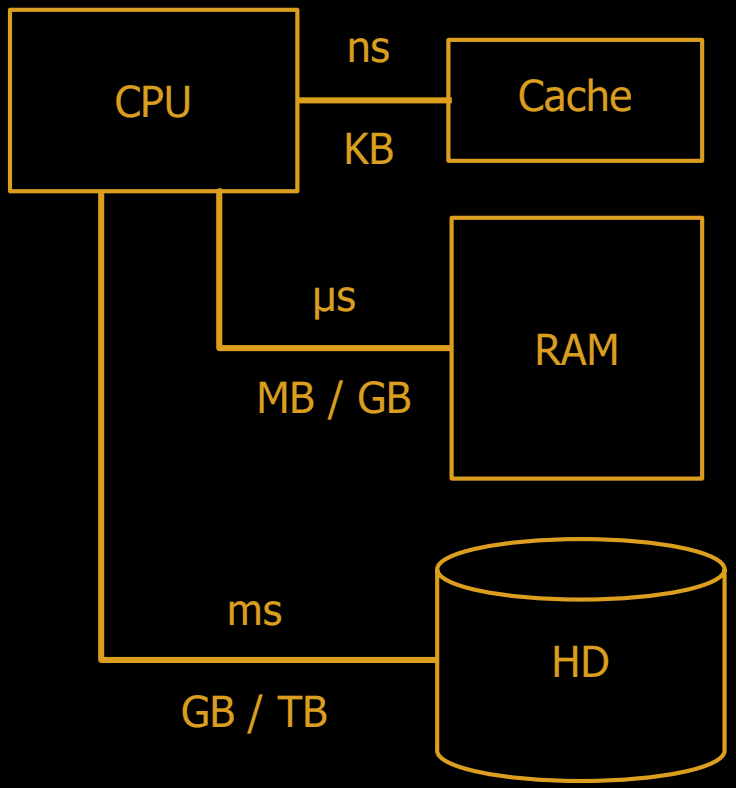
Bibliografía

- W. Stallings:
  Sistemas Operativos.
  - Capítulo 7 y 8.
- A.S. Tanenbaum:
  Modern Operating Systems.
  - Capítulo 3.

MEMORIA Y PROCESOS

JERARQUÍA DE MEMORIA

- En un ordenador la memoria está organizada en una jerarquía de varios niveles.
- Compromiso:
  - Capacidad…
  - …o velocidad.
- Volatilidad.
- **Memoria primaria:** RAM
  - Gestor de memoria.
- **Memoria secundaria:** HD

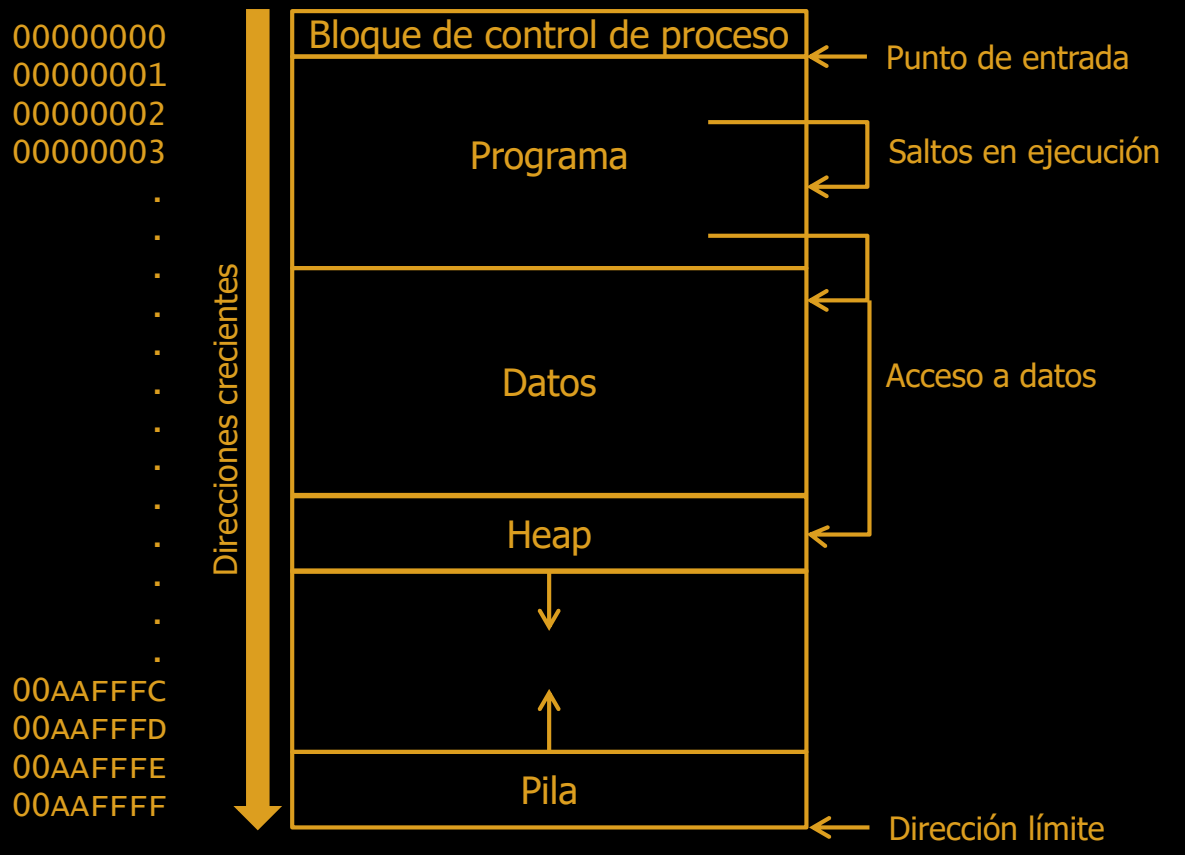


ESTRUCTURA FÍSICA DE LA MEMORIA

- Memoria principal.
- **Palabras:**
  - Mínima unidad de asignación
  - Tamaño fijo por hardware.
    - n = 8-64 bits.
- **Direcciones:**
  - Apunta a una palabra.
  - Tamaño máximo fijo por hardware.
    - m = 8-64 bits.
- Capacidad memoria: 2^{n+m}

ESTRUCTURA EN MEMORIA DE UN PROCESO

- PCB:
  - Información del sistema operativo.
- Programa.
- Datos.
- Heap:
  - Datos creados dinámicamente.
- Pila.



GESTORES DE MEMORIA

GESTOR DE MEMORIA

- Componente del sistema operativo.
- Encargado de controlar todos los aspectos relacionados con la memoria principal.
  - Si se usa memoria virtual también utiliza la memoria secundaria.
  - Abstrae a los procesos de la estructura real de la memoria.
- Colabora con el hardware:
  - Registros especiales de la CPU.
  - MMU: Unidad de gestión de memoria.
  - Es necesario el soporte hardware para asegurar un rendimiento adecuado.

RESPONSABILIDADES

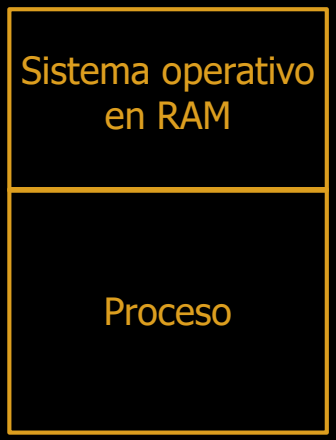
- Capacidades necesarias de un sistema de gestión de memoria para funcionar en un entorno con multiprogramación:
  - **Ubicación:** Situar un proceso en la memoria cuando se va a ejecutar y retirarlo cuando se suspende.
  - **Protección:** Evitar que un proceso pueda leer o modificar posiciones de memoria para las que no tiene acceso.
  - **Compartición:** Ofrecer la posibilidad de que varios procesos compartan las mismas posiciones de memoria.
  - **Particionamiento:** Dividir la memoria en bloques lógicos para su administración y mantenimiento. Permitir que un proceso se almacene en memoria correctamente independientemente de su tamaño.

SIN GESTIÓN DE MEMORIA

- Cada proceso ocupa una posición de memoria fija.
  - Controlada por el programador…
  - …o fijada en tiempo de compilación/enlace.
- **No se utiliza ningún tipo de abstracción.**
- **No hay traducción de direcciones.**
- Los procesos pueden:
  - Acceder a posiciones de memoria de otros procesos.
  - Acceder a posiciones de memoria del SO.
- Los procesos no pueden:
  - Retirarse de memoria.
  - Reubicarse en otra posición.
- No suelen soportarse múltiples procesos.

ORGANIZACIÓN DE LA MEMORIA SIN UN GESTOR

- El sistema operativo puede:
  - Estar en memoria junto al proceso(s).
  - Residir en la ROM.
  - Estar en memoria y acceder a drivers almacenados en la ROM.



CAPACIDADES DE UN SISTEMA SIN GESTIÓN DE MEMORIA

- ¿Proporciona un sistema sin gestión de memoria las capacidades necesarias para un sistema multiprogramado?
- **Ubicación:** Los procesos ocupan una posición de memoria fija e inmutable, decidida por el programador.
  - *No proporcionada.*
- **Protección:** Los procesos pueden acceder sin ningún tipo de protección a las posiciones de memoria de otros procesos y/o el sistema operativo.
  - *No proporcionada.*
- **Compartición:** La única compartición disponible es la que aporta el propio programador.
  - *No proporcionada.*
- **Particionamiento:** La memoria no está particionada.
  - *No proporcionada.*

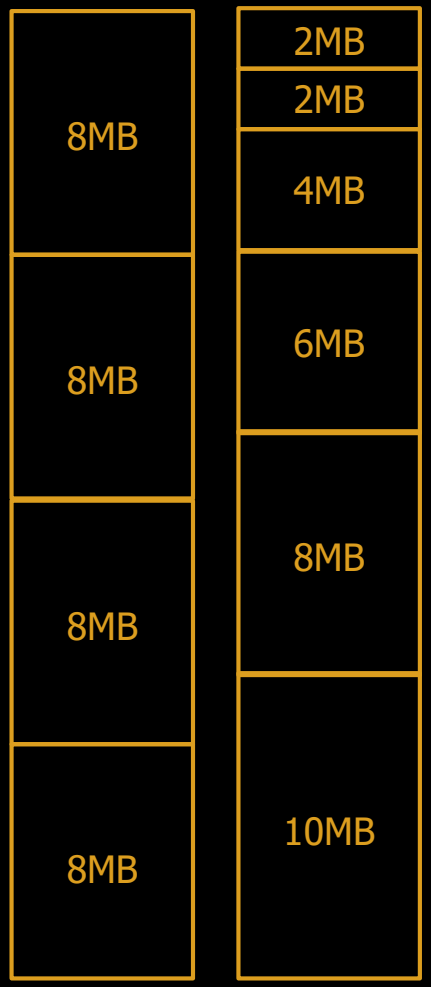
TIPOS DE GESTORES DE MEMORIA

- Resulta evidente que para un entorno con multiprogramación es necesaria la presencia de un gestor de memoria.
- **Modelos de gestión de memoria:**
  - Particionamiento fijo.
  - Particionamiento dinámico.
  - Paginación.
  - Segmentación.
  - Paginación y segmentación

PARTICIONAMIENTO FIJO

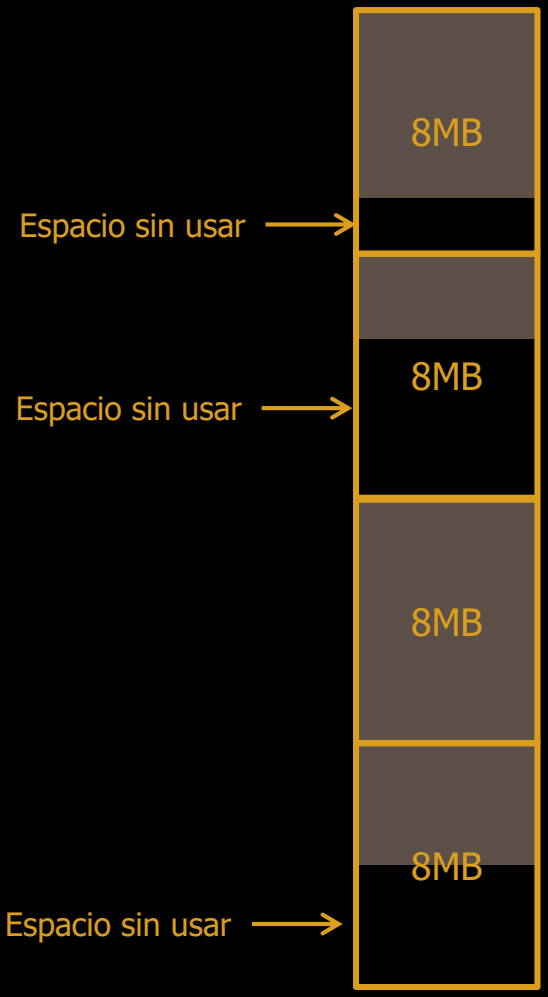
PARTICIONAMIENTO FIJO

- La memoria se divide en bloques:
  - Bloques de tamaño fijo.
  - Un proceso en cada bloque.
  - Creados de antemano.
- Posibles distribuciones:
  - Bloques de igual tamaño.
  - Bloques de múltiples tamaños.
- **Limitaciones:**
  - El tamaño de los bloques limita el de los procesos.
  - Fragmentación interna.



FRAGMENTACIÓN INTERNA

- **Fragmentación interna:** Espacio en el interior de un bloque que no puede ser utilizado.
- Se produce al cargar un proceso de un tamaño inferior al bloque asignado.
- Dicho espacio no puede ser asignado a otro proceso.
- Se produce en la mayoría de los bloques.
- **Un proporción de memoria muy alta se desperdicia.**



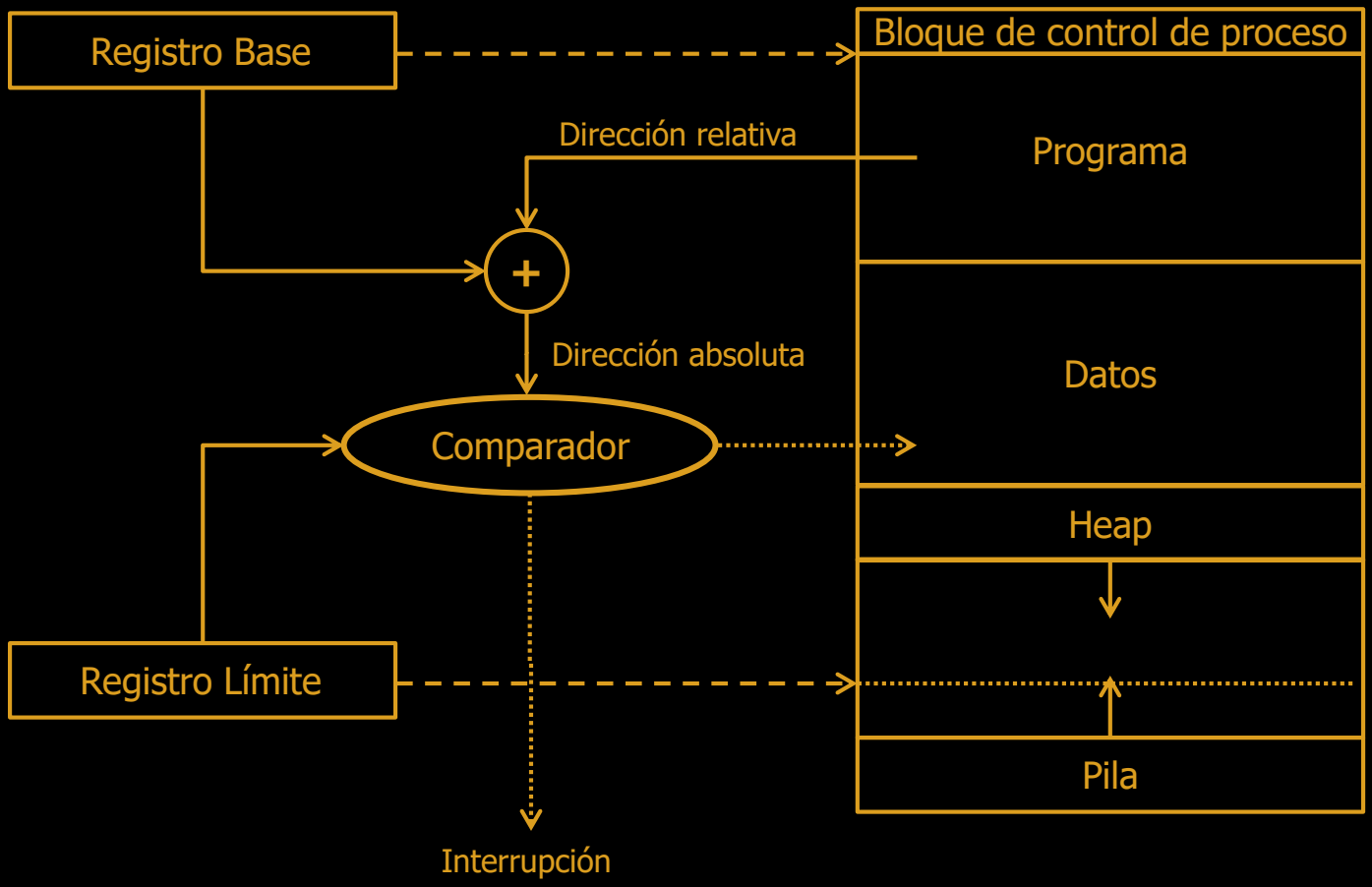
REUBICACIÓN

- En el particionamiento fijo los procesos:
  - Pueden estar ubicados en cualquiera de los bloques del tamaño adecuado.
  - Pueden ser suspendidos y retirados de memoria.
  - Pueden ser activados y colocados en memoria.
- Dónde resida un proceso es **transparente para él**.
- **Swap:** Acto de…
  1. Retirar un proceso suspendido de memoria.
  2. Colocar un proceso reactivado en memoria.
- Se puede trabajar con procesos que, **en conjunto**, sean **mayores que el tamaño real de la memoria**.
- Colas de reubicación:
  - Bloques de igual tamaño: 1 cola.
  - Bloques de múltiples tamaños: 1 cola por tamaño de bloque.

REGISTROS BASE Y LÍMITE

- **Registro base:**
  - Dirección de memoria real.
  - Punto inicial del proceso.
  - Permite la **traducción de direcciones**.
  - Dirección real = dirección relativa + registro base.
- **Registro límite:**
  - Dirección de memoria real.
  - Punto final del proceso.
  - Permite implementar la **protección**.
  - Dirección real $\leq$ registro límite.

FUNCIONAMIENTO DE LOS REGISTROS BASE Y LÍMITE



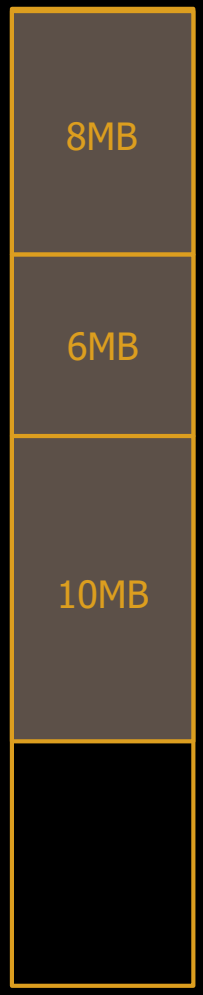
CAPACIDADES DE UN GESTOR DE PARTICIONAMIENTO FIJO

- **Ubicación:** Los procesos pueden ocupar varias posiciones de memoria, y ser desplazados de una a otra.
  - ✓ *Proporcionada.*
- **Protección:** Los procesos no pueden acceder a posiciones de memorias de otros procesos o del sistema operativo.
  - ✓ *Proporcionada.*
- **Compartición:** No se ofrecen mecanismos de compartición.
  - ➢ *No proporcionada.*
- **Particionamiento:** La memoria está particionada en bloques de tamaño fijo. Cada proceso ocupa uno de estos bloques, desperdiciándose espacio por causa de la fragmentación interna. **El tamaño máximo de proceso está limitado por el tamaño de bloque.**
  - ❖ *No proporcionada por completo.*

PARTICIONAMIENTO DINÁMICO

PARTICIONAMIENTO DINÁMICO

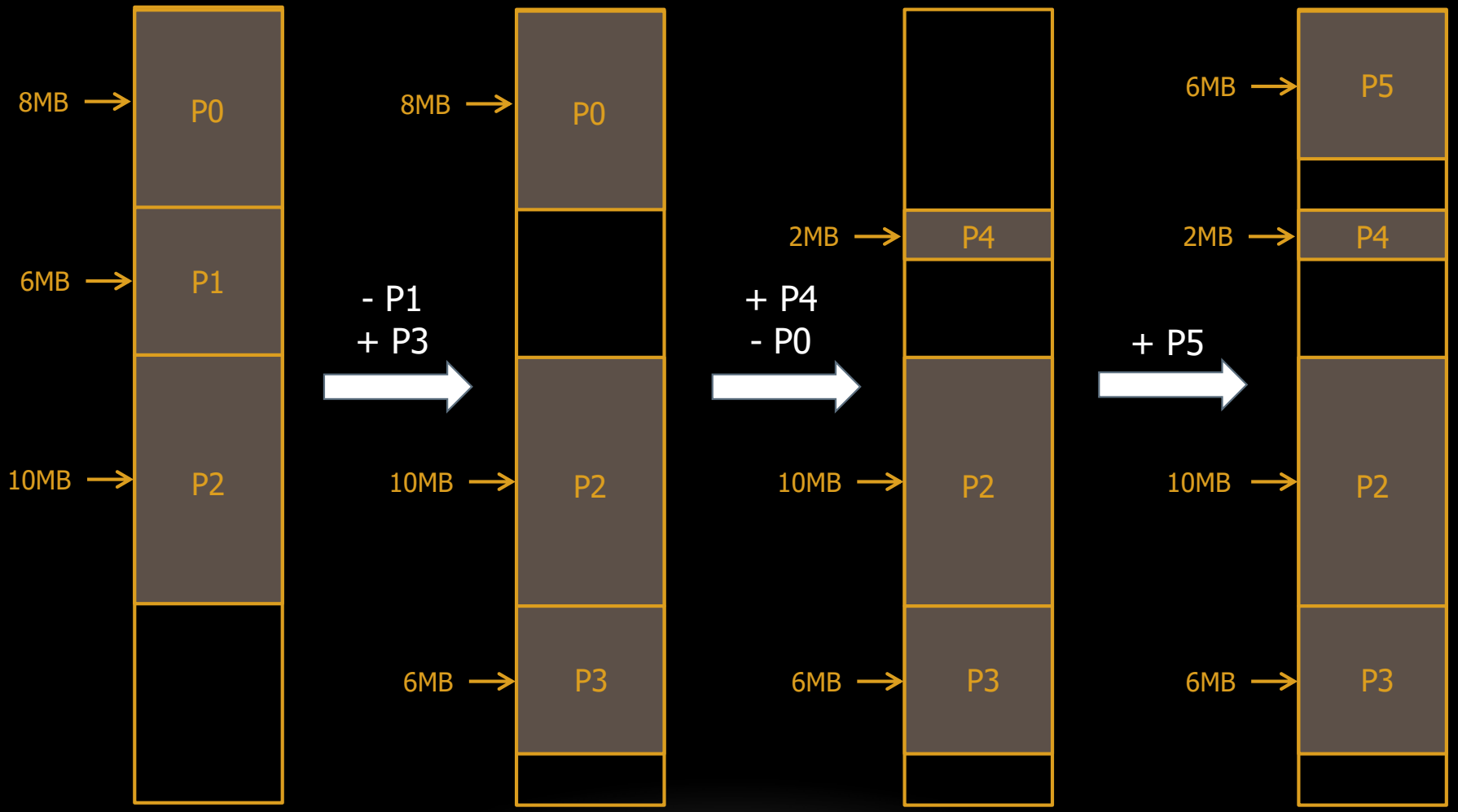
- La memoria se divide en particiones:
  - Un proceso por cada partición.
  - La partición es del tamaño exacto del proceso.
  - Creados cada vez que se carga un proceso.
- Ya no hay que preocuparse de…
  - Fragmentación interna.
  - Procesos de tamaño superior al bloque.
- **Limitaciones:**
  - El tamaño de la memoria limita el de los procesos.
  - Fragmentación externa.



FRAGMENTACIÓN EXTERNA

- **Fragmentación externa:** Espacio en la memoria que no puede ser utilizado.
- Se produce al cargar y retirar procesos de memoria durante un tiempo prolongado.
- Quedan pequeños “huecos” en la memoria demasiado pequeños para almacenar ningún proceso.
- **Un proporción de memoria muy alta se desperdicia.**
  - …pero esta vez hay solución.

FRAGMENTACIÓN EXTERNA



SOLUCIONES A LA FRAGMENTACIÓN EXTERNA

- **Compactación:**
  - Proceso que se ejecuta periódicamente.
  - Desplaza los procesos en memoria.
  - Exige capacidad de reubicación.
  - Alto consumo de recursos.
- **Algoritmos de ubicación:**
  - *Best-fit* (mejor): Espacio de tamaño más parecido.
  - *Worst-fit* (peor): Espacio de tamaño menos parecido.
  - *First-fit* (primero): Espacio con dirección de memoria inferior.
  - *Next-fit* (siguiente): Espacio con dirección de memoria siguiente a la última colocación.
  - Un buen uso minimiza la necesidad de compactación.

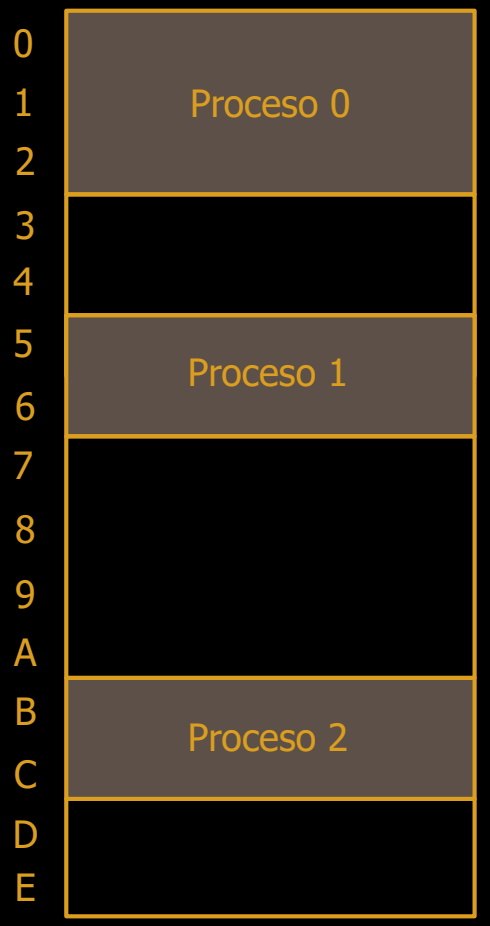
GESTIÓN DE LA MEMORIA LIBRE CON PARTICIONAMIENTO DINÁMICO

- Para la gestión de la memoria libre en un esquema de particionamiento dinámico hay dos alternativas:
- **Bitmaps (*mapas de bits*):**
  - Array de 0s y 1s.
  - Cada entrada indica si una posición de memoria está libre (0) u ocupada (1).
  - Puede modificarse para que cada entrada se refiera a un grupo de posiciones.
- **Listas enlazadas:**
  - Lista de elementos enlazados con el anterior y el siguiente.
  - Cada elemento indica:
    - Si se trata de un proceso (P) o un agujero (H).
    - Dirección de memoria en la que comienza.
    - Cuánto se extiende (un número).

BITMAPS Y LISTAS ENLAZADAS

- ¿Cómo sería el bitmap del particionamiento de la derecha?

- ¿Y la lista enlazada?



MODELO BUDDY

- Híbrido entre particionamiento fijo y dinámico:
  - Bloques de memoria como en particionamiento fijo…
  - …pero de tamaño variable como en el dinámico.
- Bloques de tamaño 2^k:
  - $L \leq K \leq U$
  - 2^L: Bloque de tamaño mínimo asignado.
  - 2^U: Tamaño total de la memoria.
- **Asignación de un proceso:**
  1. Se busca un bloque tal que $2^{k-1} <$ tamaño $\leq 2^k$
  2. Si no existe se divide el bloque más pequeño en una "pareja" de dos bloques de la mitad de tamaño y se vuelve al paso 1.
- **Reunión de bloques:**
  - Si dos bloques de una "pareja" no tienen un proceso asignado se reúnen en un bloque del doble de tamaño.

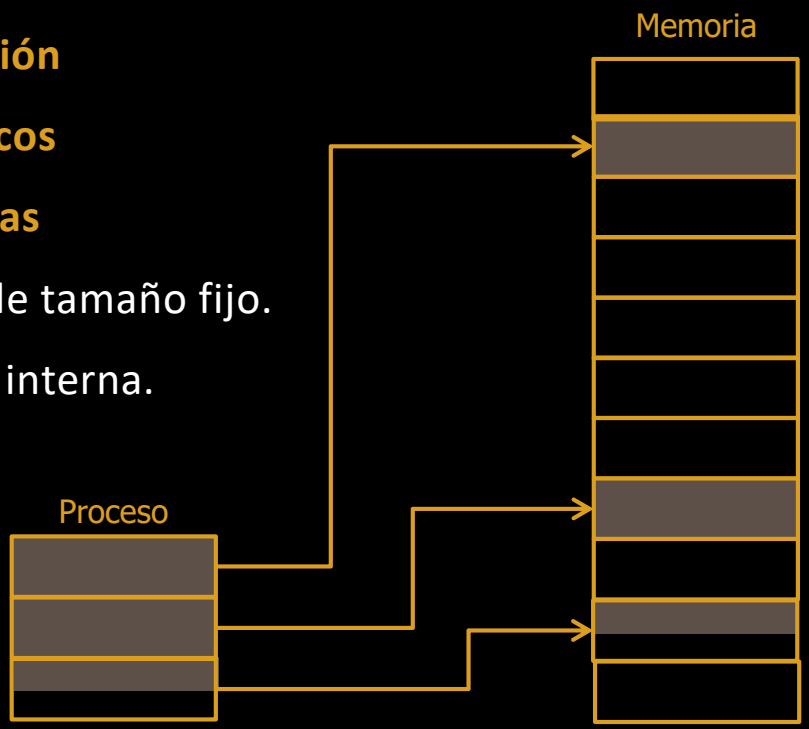
CAPACIDADES DE UN GESTOR DE PARTICIONAMIENTO DINÁMICO

- **Ubicación:** Los procesos pueden ocupar varias posiciones de memoria, y ser desplazados de una a otra.
  - ✓ *Proporcionada.*
- **Protección:** Los procesos no pueden acceder a posiciones de memorias de otros procesos o del sistema operativo.
  - ✓ *Proporcionada.*
- **Compartición:** No se ofrecen mecanismos de compartición.
  - ➢ *No proporcionada.*
- **Particionamiento:** La memoria está particionada en bloques del tamaño del proceso que lo ocupa. Se desperdicia espacio por causa de la fragmentación externa. **El tamaño máximo de proceso está limitado por el tamaño de bloque.**
  - ❖ *No proporcionada por completo.*

PAGINACIÓN

PAGINACIÓN

- Tanto el particionamiento fijo como el dinámico tienen desventajas.
- ¿Y si dividimos un proceso en partes más pequeñas?
  - Partes de tamaño fijo: **Paginación**
- Memoria dividida en bloques: **Marcos**
- Proceso dividido en bloques: **Páginas**
- Marcos y páginas son pequeños y de tamaño fijo.
- Se reduce mucho la fragmentación interna.
  - Prácticamente despreciable.
- Necesita soporte hardware.



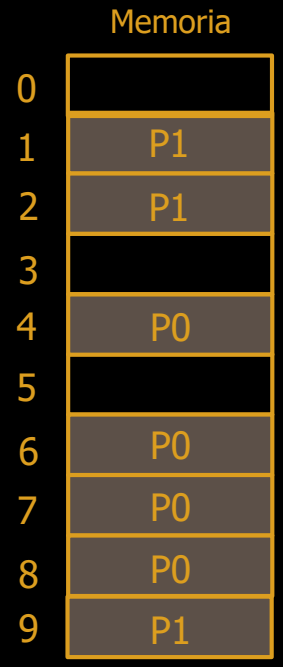
ESTRUCTURAS DE DATOS

- Para hacer posible la paginación se usan dos tipos de estructuras de datos:
- **Tabla de páginas:**
  - Contiene la dirección del marco en el que está almacenada cada página.
  - Una tabla por proceso.
  - Tantas entradas como páginas tiene el proceso.
  - Permite traducir entre direcciones lógicas y físicas.
- **Lista de marcos libres:**
  - Contiene los marcos que no han sido asignados.
  - Una tabla para todo el sistema.
  - Tantas entradas como marcos libres.

Tabla de Páginas del proceso 0

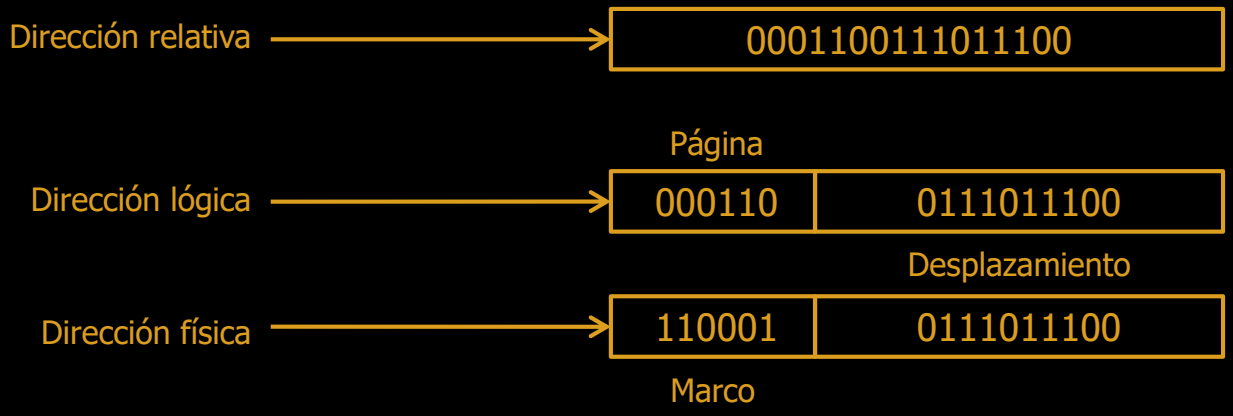
Tabla de Páginas del proceso 1

Lista de marcos libres

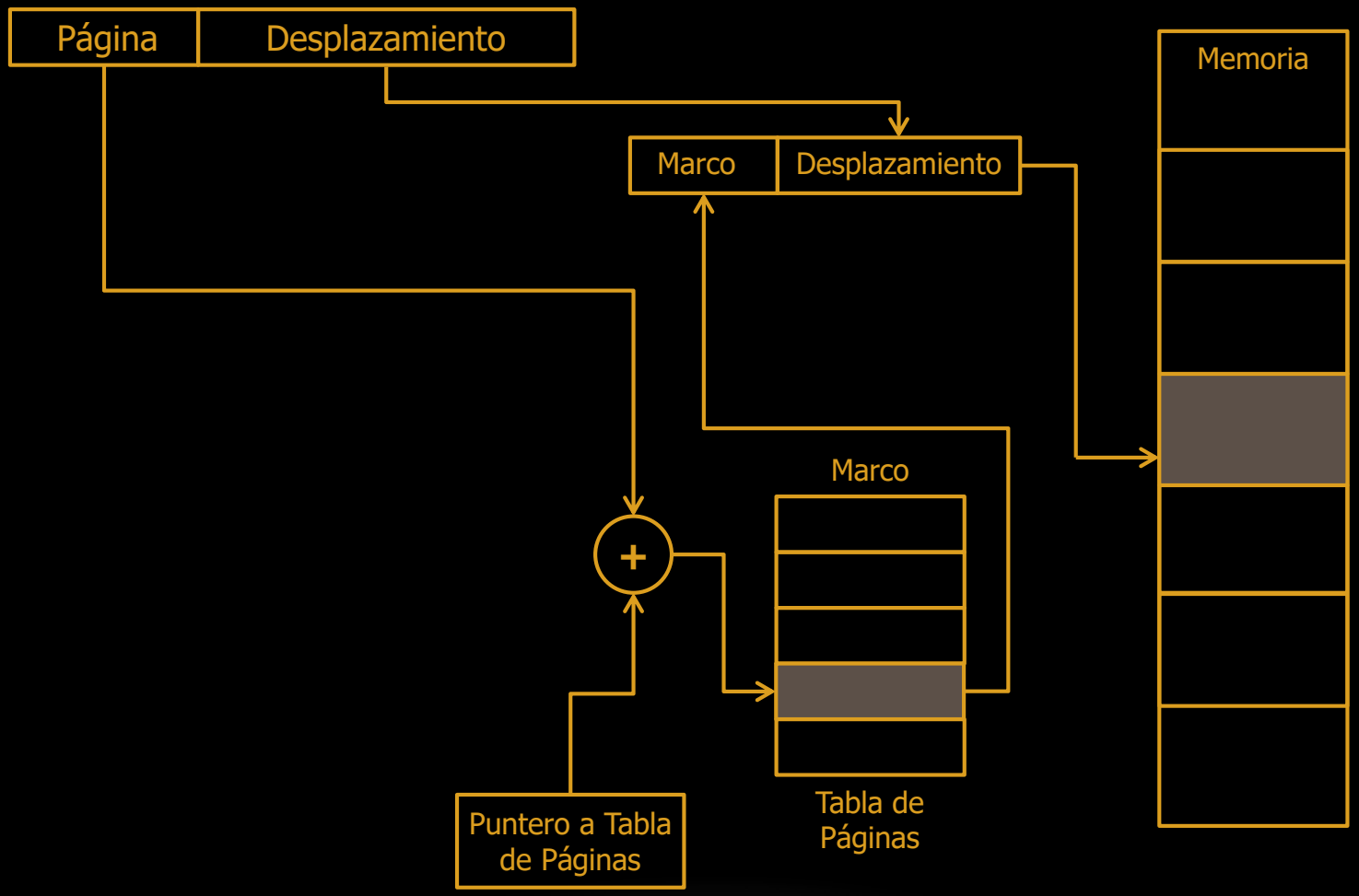


DIRECCIÓN LÓGICA

- **Dirección lógica:** Compuesta de dos partes:
  - Número de página.
  - Desplazamiento.
- Si el tamaño de marco/página es 2^n…
  - Dirección lógica == Dirección relativa
  - ¡Es transparente para el programador!



TRADUCCIÓN DE UNA DIRECCIÓN USANDO PAGINACIÓN



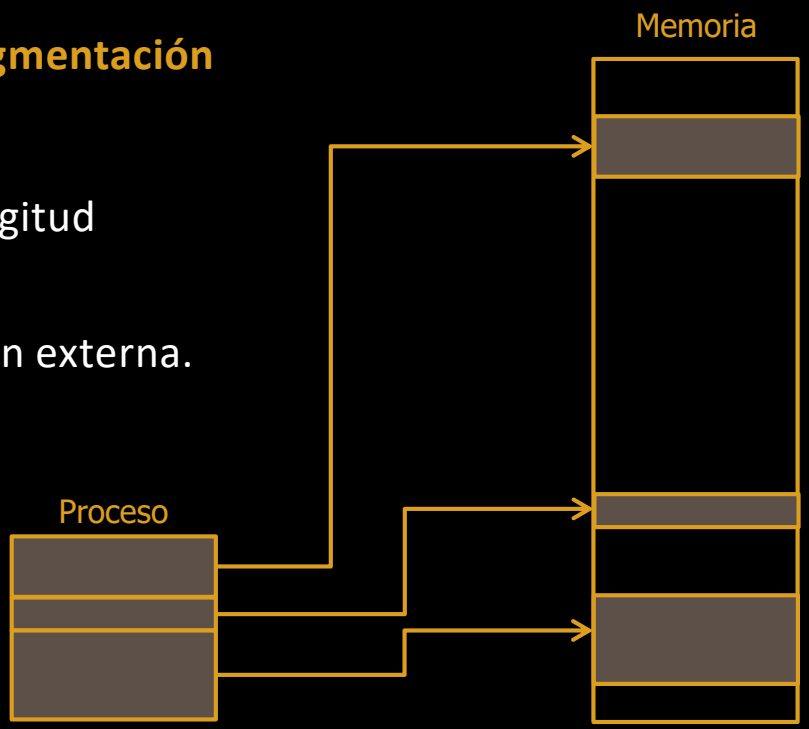
CAPACIDADES DE UN GESTOR DE MEMORIA CON PAGINACIÓN

- **Ubicación:** Los procesos pueden ocupar varias posiciones de memoria, y ser desplazados de una a otra.
  - ✓ *Proporcionada.*
- **Protección:** Los procesos no pueden acceder a posiciones de memorias de otros procesos o del sistema operativo.
  - ✓ *Proporcionada.*
- **Compartición:** No se ofrecen mecanismos de compartición.
  - ➢ *No proporcionada.*
- **Particionamiento:** La memoria está particionada en bloques de tamaño fijo llamados marcos. Los procesos en bloques de igual tamaño llamados páginas. No se desperdicia prácticamente espacio. **El tamaño máximo de proceso no está limitado por el tamaño de bloque.**
  - ✓ *Proporcionada.*

SEGMENTACIÓN

SEGMENTACIÓN

- Tanto el particionamiento fijo como el dinámico tienen desventajas.
- ¿Y si dividimos un proceso en partes más pequeñas?
  - Partes de tamaño variable: **Segmentación**
- Memoria **no** dividida en bloques.
- Proceso dividido en bloques de longitud variable: **Segmentos**
- Se reduce en parte la fragmentación externa.
  - Sigue siendo necesario recurrir a la compactación.
- Necesita soporte hardware.

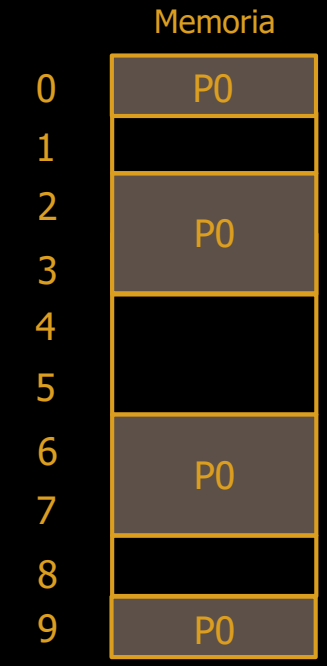


ESTRUCTURAS DE DATOS

- Para hacer posible la segmentación se usan dos estructuras de datos:
- **Tabla de segmentos:**
  - Contiene la longitud de cada segmento.
  - Contiene la dirección base a partir de la cual se encuentra cada segmento en la memoria física.
  - Una tabla por proceso.
  - Tantas entradas como segmentos tiene el proceso.
  - Permite traducir entre direcciones lógicas y físicas y comprobar que la dirección se encuentra dentro del rango válido.
- **Lista de bloques libres:**
  - Contiene los bloques de memoria que no han sido asignados.
  - Una tabla para todo el sistema.
  - Tantas entradas como “huecos” libres.

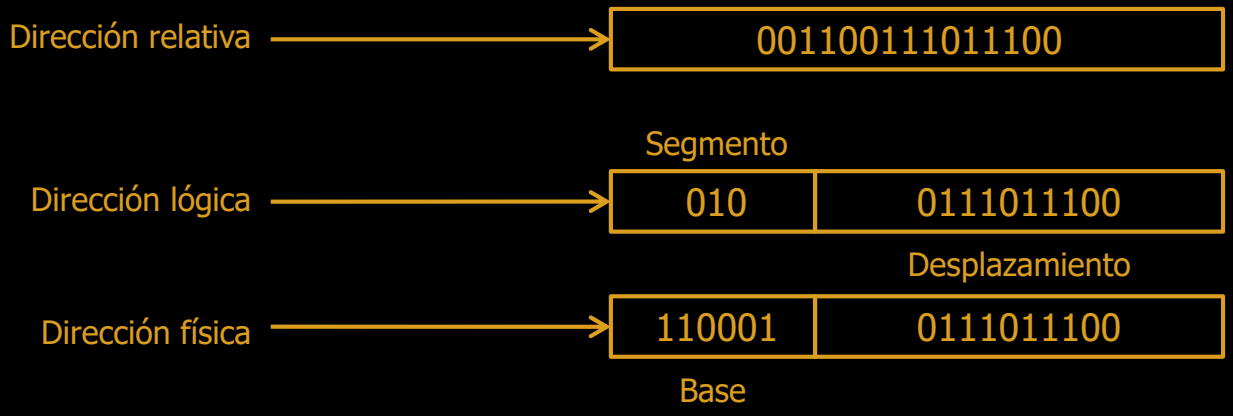
Tabla de Segmentos del proceso 0

Lista de bloques libres

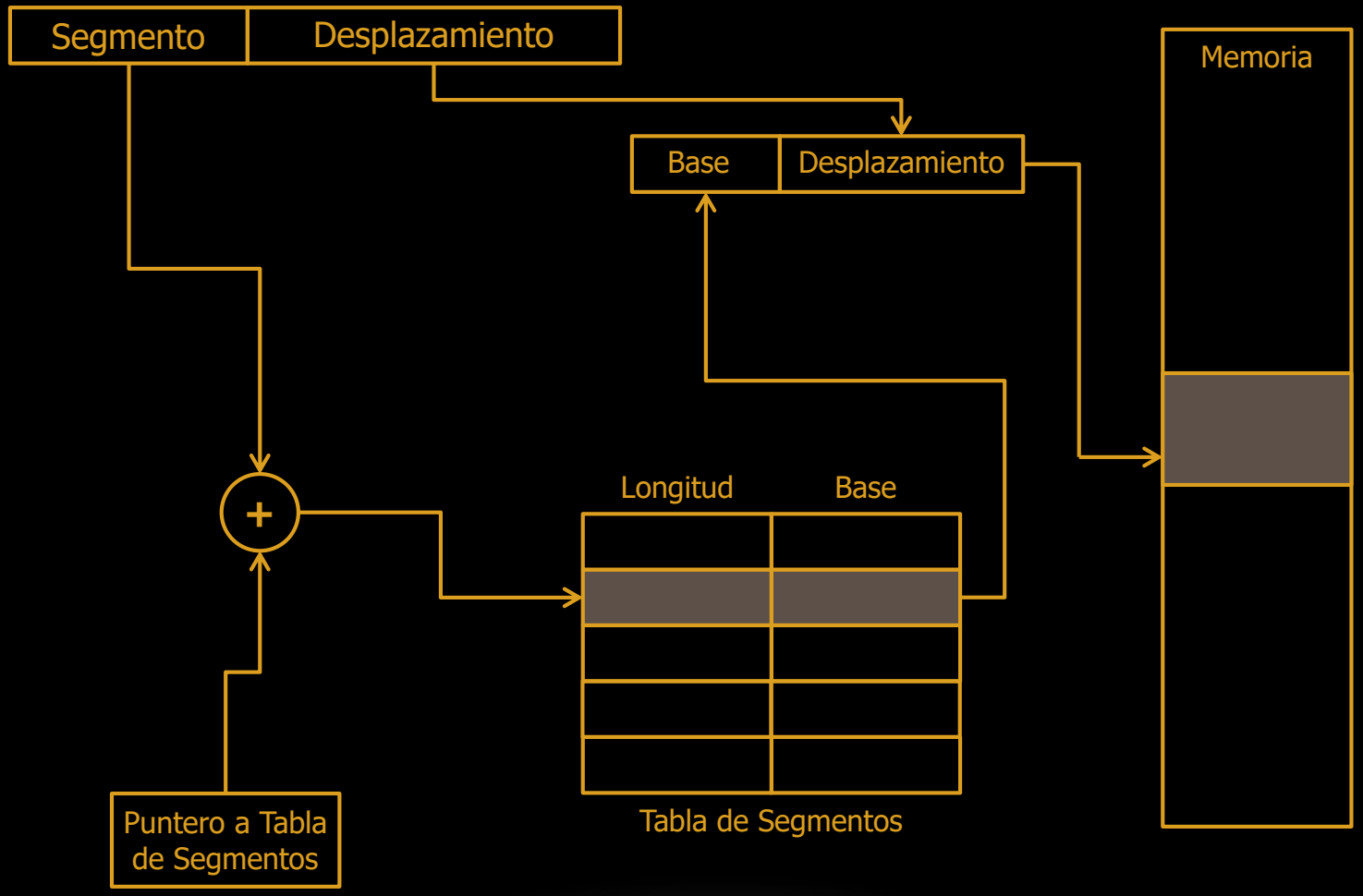


DIRECCIÓN LÓGICA

- **Dirección lógica:** Compuesta de dos partes:
  - Número de segmento.
  - Desplazamiento.
- Dirección lógica != Dirección relativa:
  - ¡No es transparente para el programador!
  - El programador puede controlar la configuración de compartición de cada segmento y dividir el programa en segmentos según su función.



TRADUCCIÓN DE UNA DIRECCIÓN USANDO SEGMENTACIÓN



CAPACIDADES DE UN GESTOR DE MEMORIA CON SEGMENTACIÓN

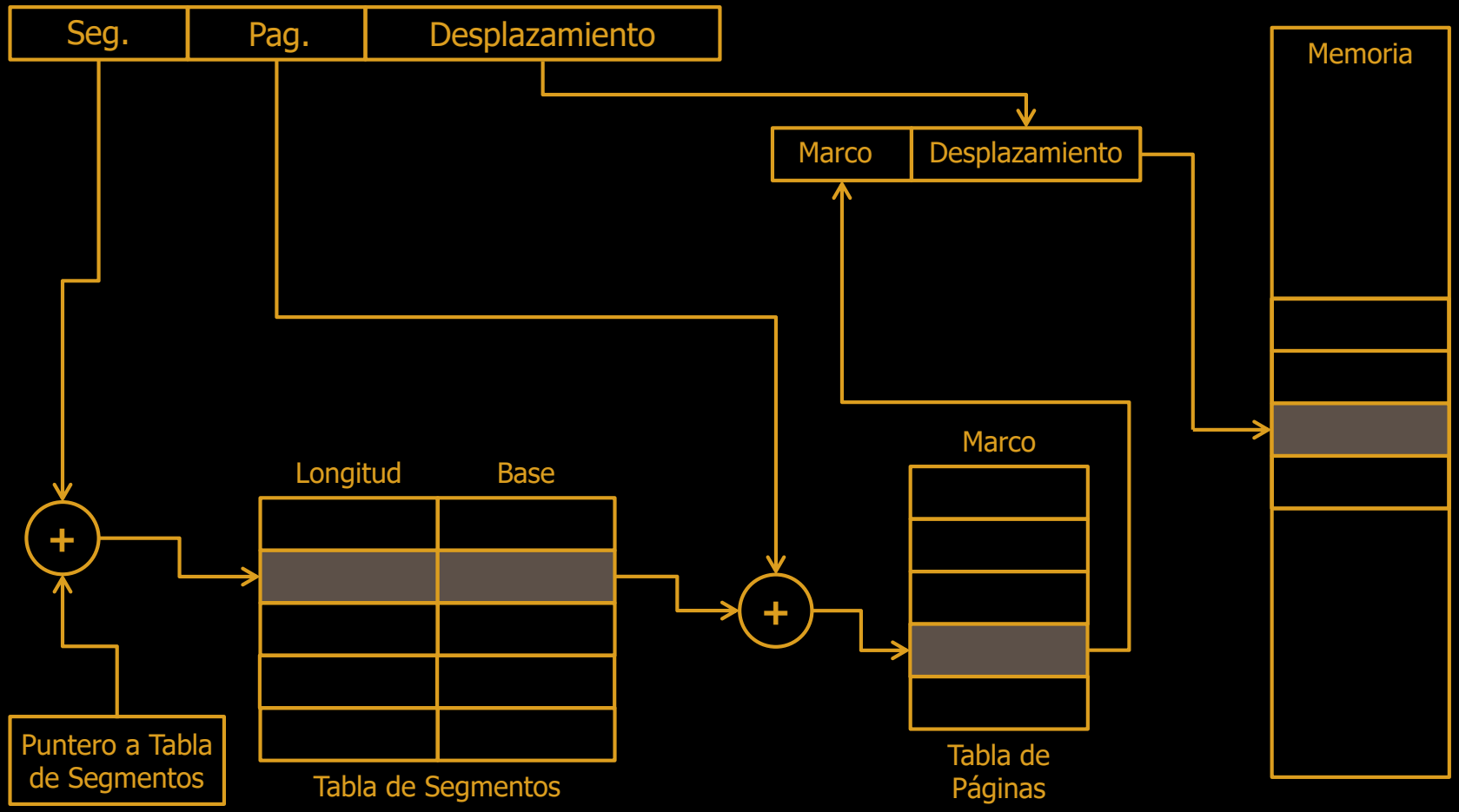
- **Ubicación:** Los procesos pueden ocupar varias posiciones de memoria, y ser desplazados de una a otra.
  - ✓ *Proporcionada.*
- **Protección:** Los procesos no pueden acceder a posiciones de memorias de otros procesos o del sistema operativo.
  - ✓ *Proporcionada.*
- **Compartición:** Se pueden compartir segmentos entre procesos.
  - ✓ *Proporcionada.*
- **Particionamiento:** Los procesos están particionados en pequeños bloques de llamados segmentos. Se produce fragmentación externa, pero en menor medida que con el particionamiento dinámico. **El tamaño máximo de proceso no está limitado por el tamaño de bloque.**
  - ✓ *Proporcionada.*

PAGINACIÓN Y SEGMENTACIÓN

PAGINACIÓN Y SEGMENTACIÓN

- Los esquemas de segmentación y paginación pueden combinarse.
- **Paginación segmentada:**
  - Cada página se divide en segmentos.
  - En la práctica no se usa.
- **Segmentación paginada:**
  - Cada segmento se divide en páginas.
  - Dirección compuesta de:
    - Número de segmento.
    - Número de página dentro del segmento.
    - Desplazamiento dentro de la página.

SEGMENTACIÓN PAGINADA



VENTAJAS DE LA SEGMENTACIÓN PAGINADA

- **De la paginación:**
  - Sin fragmentación externa.
  - Soporte a funciones hardware.
- **De la segmentación:**
  - Soporte a estructuras de datos cambiantes.
  - Soporte a protección.
  - Soporte a compartición.
- Desde el punto de vista del programador es indistinguible de la segmentación.

CAPACIDADES DE UN GESTOR DE MEMORIA CON SEGMENTACIÓN PAGINADA

- **Ubicación:** Los procesos pueden ocupar varias posiciones de memoria, y ser desplazados de una a otra.
  - ✓ *Proporcionada.*
- **Protección:** Los procesos no pueden acceder a posiciones de memorias de otros procesos o del sistema operativo.
  - ✓ *Proporcionada.*
- **Compartición:** Se pueden compartir segmentos entre procesos.
  - ✓ *Proporcionada.*
- **Particionamiento:** No se produce fragmentación externa ni interna. **El tamaño máximo de proceso no está limitado por el tamaño de bloque.**
  - ✓ *Proporcionada.*

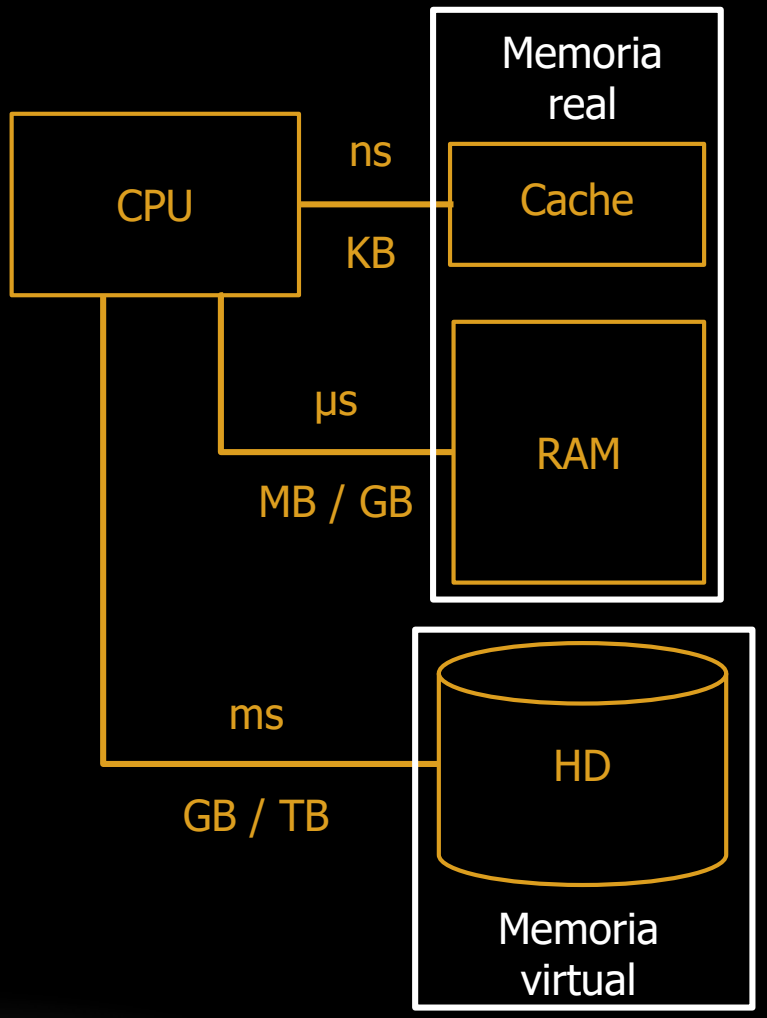
MEMORIA VIRTUAL

MEMORIA VIRTUAL

- Tanto en la paginación como en la segmentación…
  - Las referencias a memoria son direcciones lógicas.
  - Los procesos están divididos en porciones.
- Estas dos características hacen posible que…
  - Sólo una parte de cada proceso esté presente en memoria: **conjunto residente**
- Y esto tiene dos implicaciones:
  1. Los procesos pueden ser mayores que la memoria principal.
  2. Más procesos pueden encontrarse a la vez en la memoria principal.
- Un esquema como éste es el de un **gestor de memoria con memoria virtual**.

MEMORIA REAL Y VIRTUAL

- En este esquema la memoria se divide en dos bloques: real y virtual
- **Memoria real (principal):**
  - Cache y RAM.
  - Dónde residen las porciones que están en uso.
  - Alta velocidad, baja capacidad.
- **Memoria virtual (secundaria):**
  - Disco duro.
  - Dónde residen las porciones que no están en uso.
  - Dónde residen los procesos suspendidos.
  - Alta capacidad, baja velocidad.



PRINCIPIO DE LOCALIDAD

- Lo que hace posible este esquema es el **principio de localidad**:
  - Las referencias dentro de un programa, tanto a memoria como a datos, tienden a agruparse.
- Por tanto, la mayoría de los accesos a direcciones de un proceso serán a la propia página o a las más próximas.
  - No es necesario tener en memoria física mas que estas direcciones próximas.
- Principio del 90/10: Durante el 90% del tiempo sólo se accederá al 10% del espacio de direcciones de un proceso.
- Se discute si sigue siendo así en los lenguajes orientados a objetos:
  - Pero los grandes tamaños de memoria actuales hacen que este principio se mantenga.

PAGINACIÓN EN MEMORIA VIRTUAL

MEMORIA VIRTUAL CON PAGINACIÓN

- La **paginación** es la forma más común de implementar la memoria virtual.
- El gestor de memoria se encarga de mover páginas entre la memoria real y virtual.
- Es necesario ampliar la tabla de páginas:
  - **Bit de presencia:** Present/Valid
    Indica que el contenido de la tabla de páginas es válido.
  - **Bit de modificación:** Modified
    Indica si la página ha sido modificada

	Valid	Modified	Frame
0	1	0	00FFAB
1	1	1	46FF00
2	0	1	0600AF
3	0	0	000000

Tabla de Páginas del proceso 0

PROBLEMAS DE LA MEMORIA VIRTUAL CON PAGINACIÓN

- La memoria virtual con paginación presenta una serie de problemas:
  - Relacionados con el uso de paginación:
    - **Velocidad:** acceso a la dirección de memoria real.
    - **Espacio:** tamaño de la tabla de páginas.
    - **Compartición:** no soportada por la paginación.
  - Relacionados con el uso de memoria virtual.
    - Una de las tareas más importantes del gestor de memoria será decidir **cuándo y cómo mover páginas** entre la memoria real y la virtual.
- La forma en la que se ataquen estos problemas marcará la diferencia entre una gestión adecuada o deficiente:
  - Mecanismos para atacar cada uno de los problemas.
  - Decisiones de diseño.
  - “Trucos” para aprovechar mejor la memoria.

PROBLEMAS: VELOCIDAD

EL PROBLEMA DE LA VELOCIDAD

- En la paginación, para obtener una dirección física es necesario:
  1. Consultar el puntero a tabla de páginas.
  2. Consultar la tabla de páginas. ⟵ **Acceso a memoria**
  3. Concatenar el marco y el desplazamiento.
- Por cada acceso a memoria, es necesario otro: **La velocidad se divide por 2.**
- Posibles soluciones a este problema:
  - Implementar la tabla de páginas en hardware:
    - Mediante el uso de registros.
    - × **Inviable:** La tabla de páginas es ahora demasiado grande.
      32 bits y 4KBs por marco: 1 millón de páginas.
      64 bits y 16KBs por marco: 1.000 billones de páginas.
  - ✓ **Utilizar TLB:** Traslation Lookaside Buffer.

TLB: TRANSLATION LOOKASIDE BUFFER

- El equivalente a la memoria cache para la tabla de páginas.
  - Pequeño tamaño y alta velocidad.
  - Almacena las páginas más recientemente usadas.
  - Una para todos los procesos.
- Añade dos elementos a la tabla:
  - **Process:** Identificador del proceso.
  - **Page:** Identificador de la página dentro del proceso.

	Valid	Modified	Process	Page	Frame
0	1	0	0001	A001	00FFAB
1	1	1	00A4	BFFA	46FF00
2	1	1	490B	0006	0600AF
3	1	0	9085	0AB5	000000

CARACTERÍSTICAS DE LA TLB

- Suele implementarse usando memoria asociativa:
  - Se consultan varias entradas a la vez.
- Tiene un tamaño pequeño: 32 o 64 entradas.
- Comportamiento:
  - Cuando una página no se encuentra en la TLB se produce un **fallo de página**:
    - Se busca la página en la memoria física.
    - Se reemplaza una de las páginas de la TLB por la nueva.
    - Si ha sido modificada se escribe la vieja en memoria.
  - Dos posibles fallos de página:
    - **Soft miss** (fallo blando): La página está en memoria principal.
      Tiempo de reemplazo: nanosegundos.
    - **Hard miss** (fallo duro): La página está en memoria secundaria
      Tiempo de reemplazo: milisegundos.

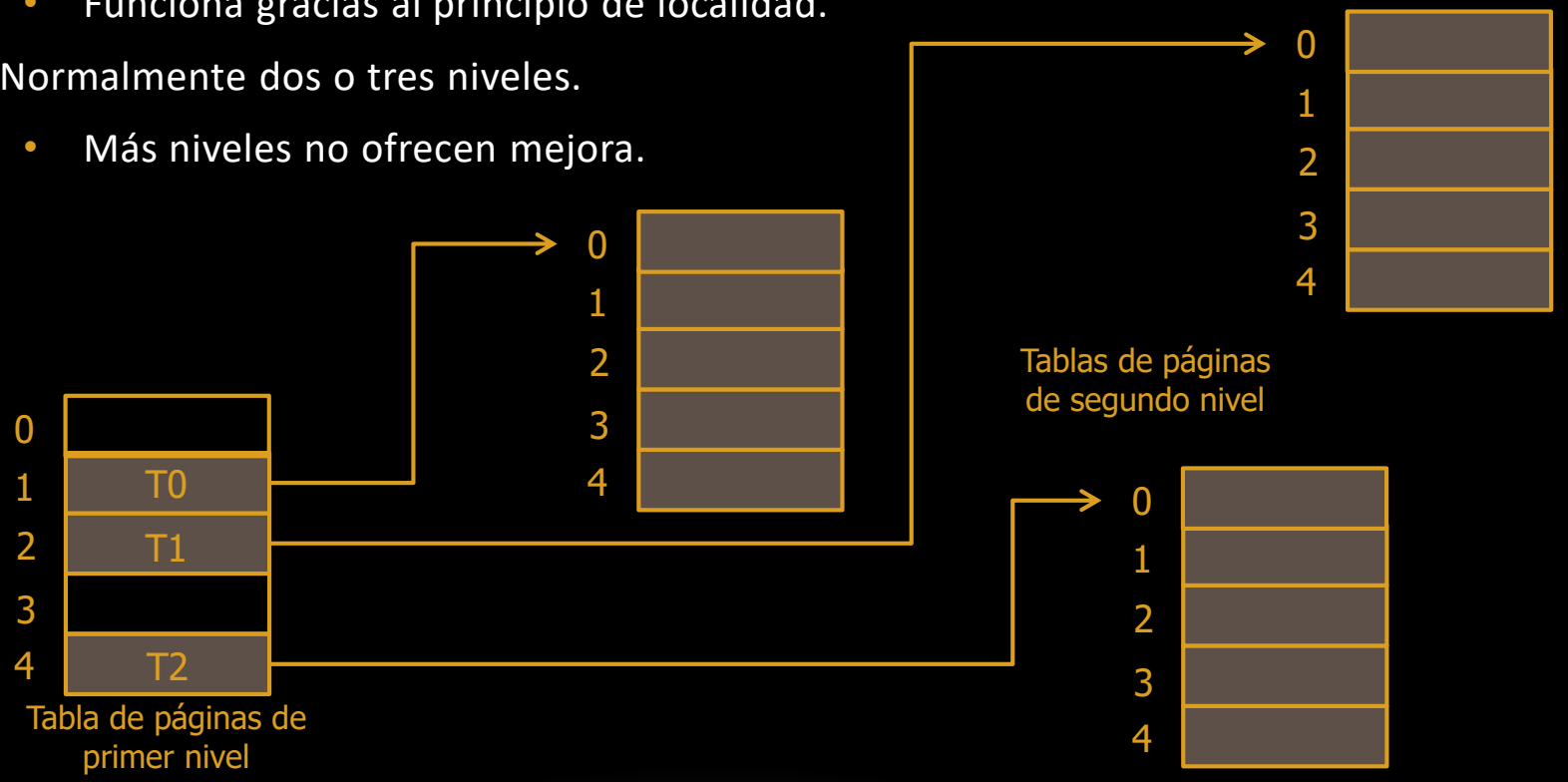
PROBLEMAS: ESPACIO

EL PROBLEMA DEL ESPACIO

- En la paginación, es necesario contar con una tabla de páginas para cada proceso.
- Usando memoria virtual el número de procesos presentes en memoria al mismo tiempo puede ser muy grande.
- Además, cada proceso puede tener un espacio de direcciones muy grande.
- Las tablas de páginas pueden alcanzar **tamaños enormes**:
  - Con 32 bits: Megabytes.
  - Con 64 bits: ¡Petabytes!
- Es inviable mantener tablas de páginas tan grandes en memoria.
- Posibles soluciones:
  - Almacenar la tabla de páginas en la memoria secundaria:
    - × **Inviable:** El tiempo de acceso empeoraría en órdenes de magnitud.
  - ✓ **Utilizar tablas de página multinivel.**
  - ✓ **Utilizar tablas de página invertidas.**

TABLAS DE PÁGINA MULTINIVEL

- Tablas de páginas que contienen referencias a otras tablas de páginas.
- Sólo se mantienen en memoria las tablas de páginas que se están usando.
  - Funciona gracias al principio de localidad.
- Normalmente dos o tres niveles.
  - Más niveles no ofrecen mejora.



TRADUCCIÓN DE DIRECCIONES EN TABLAS DE PÁGINAS MULTINIVEL

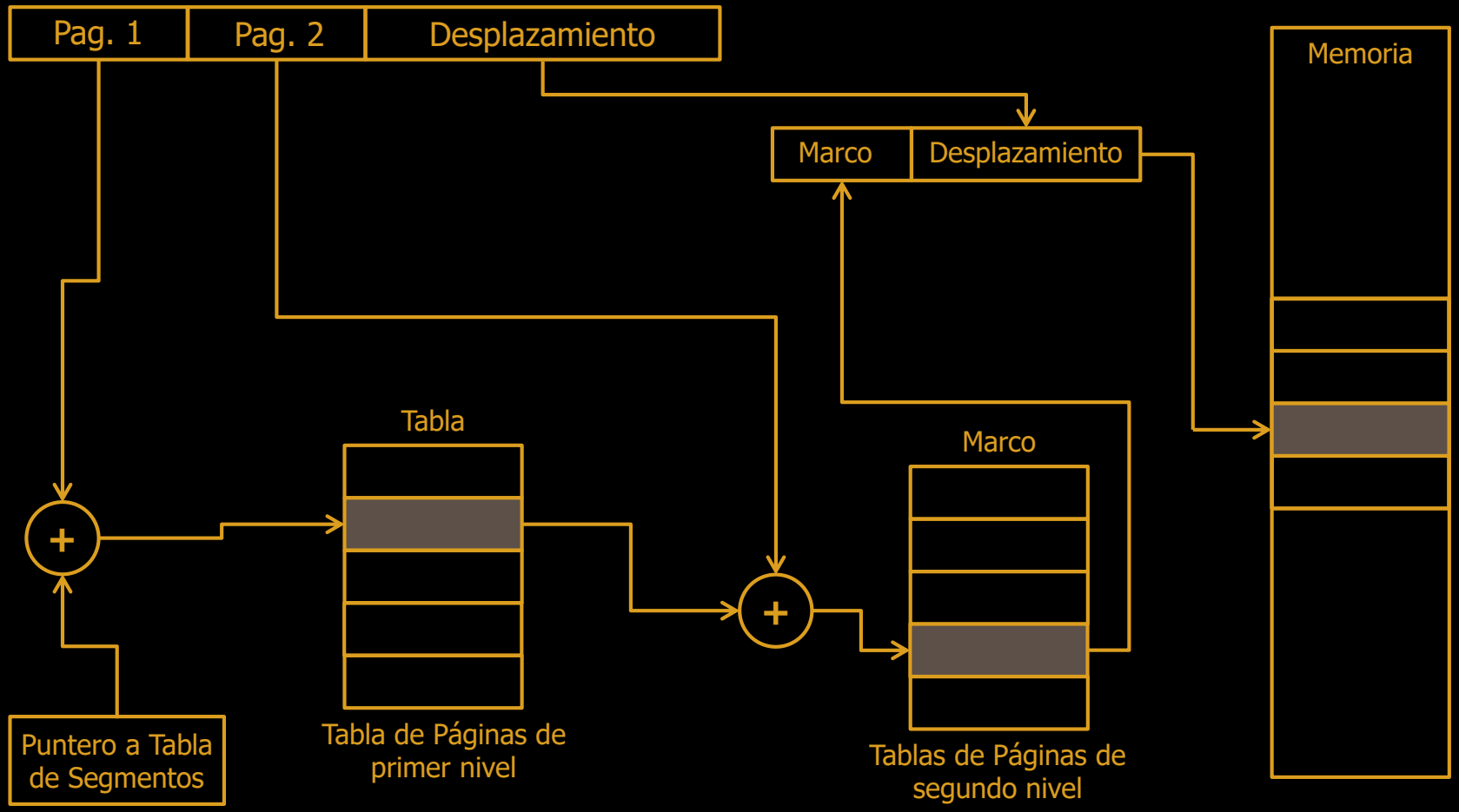


TABLA DE PÁGINAS INVERTIDA

- Ni siquiera con la paginación multinivel basta para espacios de direcciones de 64 bits. Es necesario tomar medidas adicionales.
- **Tabla de páginas invertida:**
  - Una para todos los procesos.
  - Una entrada para cada marco.
- Tamaño fijo, el de la memoria física.
  - 1GB con marcos de 4KB equivale a sólo 2^{18} entradas (antes 2^{52}).

Frame	Valid	Modified	Process	Page
0	1	0	0001	A001
1	1	1	00A4	BFFA
2	1	1	490B	0006
3	1	0	9085	0AB5

PROBLEMAS DE LA TABLA DE PÁGINAS INVERTIDA

- Se ahorra muchísimo espacio, pero tiene inconvenientes:
  - Para cada acceso a memoria es necesario buscar en ¡toda la tabla!
  - Coste en velocidad altísimo.
  - No es factible implementar una tabla invertida usando memoria asociativa.
- **Solución:** Usar un hash para identificar la posición en la tabla.
  - Función matemática. Se genera una para cada página en la tabla.
  - Hash de tamaño M, no mucho mayor que el tamaño de la tabla.
  - Si coinciden dos hashes se establece un puntero a una nueva dirección.

Frame	Hash	Valid	Modified	Process	Page	Pointer
0	32	1	0	0001	A001	34
1	33	1	1	00A4	BFFA	-
2	34	1	1	490B	0006	-
3	67	1	0	9085	0AB5	-

PROBLEMAS: *TRASHING*

TRASHING

- Si se producen continuamente fallos de página el sistema invierte más tiempo en cargar y retirar páginas de la memoria física que en ejecutar código.
- A esta situación se la denomina ***trashing*** (traducido como trasiego).
  - El rendimiento cae abruptamente.
- Puede producirse por:
  - Una memoria física pequeña.
  - La ejecución de demasiados procesos.
  - **Una mala construcción del sistema:**
    - Algoritmo de reemplazo inadecuado.
    - Malas decisiones de diseño.

ALGORITMOS DE REEMPLAZO

- Selección de la página a reemplazar cuando se carga una nueva en la memoria física.
- Su objetivo es seleccionar la página que vaya a tardar más en ser accedida de nuevo.
- Para evaluar su calidad se comparan con el algoritmo "**reemplazo óptimo**":
  - Se obtiene de la ejecución previa de un proceso.
  - Indica el menor número de reemplazos posible.
  - No es un algoritmo real, simplemente se usa como referencia.
- Para implementar muchos algoritmos de reemplazo se añade un bit extra a la tabla de páginas.
  - **Bit de referencia:** Indica si la página ha sido referenciada recientemente.

	Valid	Referenced	Modified	Frame
0	1	1	0	00FFAB
1	1	0	1	46FF00

Tabla de Páginas del proceso 0

ALGORITMOS BÁSICOS

- **Least Recently Used (LRU):** Usado menos recientemente.
  - Se reemplaza la página que fue usada por última vez hace más tiempo.
  - Ofrece un buen resultado.
  - Muy costosa de implementar si no la soporta el hardware:
    - Instante de tiempo de último uso de cada página.
    - Lista encadenada.
- **First-In First-Out (FIFO):** Primero en entrar primero en salir.
  - Se reemplaza la página que hace más tiempo que fue cargada.
  - Muy fácil de implementar: Buffer circular de tamaño fijo.
  - Se basa en una premisa errónea:
    - Que la página más antigua es la menos accedida.

ALGORITMOS AVANZADOS

- **Not Recently Used (NRU):** No usado recientemente.
  - Aproximación a LRU.
  - Se elimina una página al azar en el siguiente orden:
    - Una página con bits de referencia y modificación a 0.
    - Una página con bit de referencia a 0 y bit de modificación a 1.
    - Una página con bit de referencia a 1 y bit de modificación a 0.
    - Una página con bits de referencia y modificación a 1.
  - El bit de referencia se pone a 0 periódicamente.
- **Second Chance:** Segunda oportunidad.
  - Versión modificada del FIFO.
  - Antes de eliminar una página se comprueba el bit de referencia:
    - Si es 0: Se elimina la página.
    - Si es 1: Se pone el bit a 0 y se coloca la página al final de la cola.

ALGORITMO DEL RELOJ

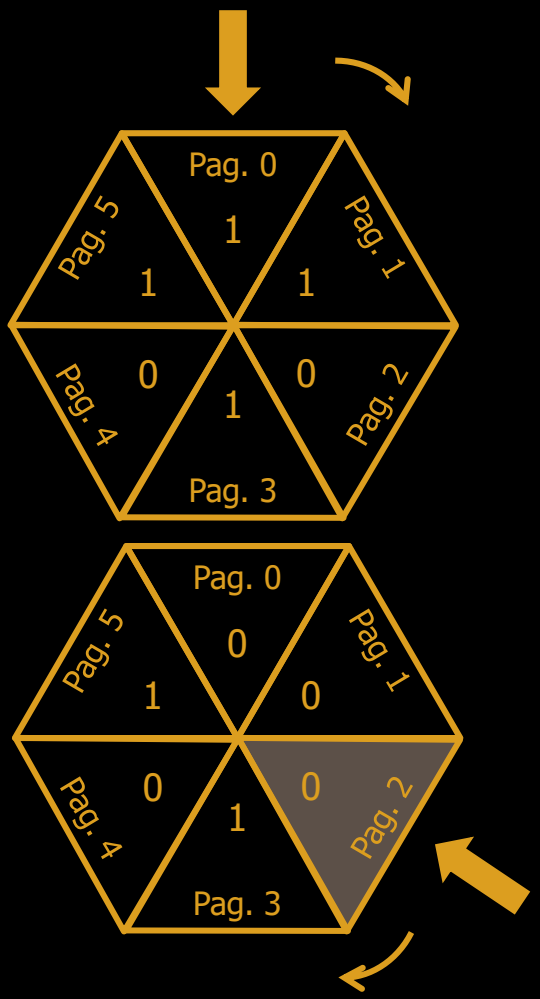
- Versión avanzada del FIFO.
- Buffer circular.
- Cuando se produce un reemplazo se recorre el buffer circular a partir de la posición del último reemplazo:

  1.A Si el bit de referencia de la entrada está a 1, se pone a 0.

  1.B Si el bit de referencia de la entrada está a 0, se reemplaza.

  2 Se avanza a la siguiente entrada.

- Se puede mejorar cambiando el número de bits de referencia.



OTRAS DECISIONES DE DISEÑO (I)

- Además del algoritmo de reemplazo, pueden tomarse otras decisiones de diseño importantes:
- **Tamaño del conjunto residente:** Número de marcos asignados a un proceso.
  - **Fijo:** Pueden quedarse marcos sin usar.
  - **Variable:** Más compleja.
- **Ámbito de reemplazo:**
  - **Local:** Sólo se reemplaza una página del mismo proceso que la página cargada. Utilización subóptima de la memoria.
  - **Global:** Se reemplazan páginas de cualquier proceso. Más compleja. Incompatible con un tamaño de conjunto residente fijo.
- Estas dos decisiones pueden complementarse con un **buffer de páginas**, dónde se almacenan las páginas más recientemente reemplazadas.

OTRAS DECISIONES DE DISEÑO (II)

- **Política de recuperación:** Cuándo se traen a memoria las páginas.
  - **Bajo demanda:** Cuando son referenciadas.
    Los procesos tardan mucho en arrancar ya que al principio se producirán muchos fallos de página.
  - **Adelantada:** Cuando se cargan páginas próximas.
    Es producen cargas ¡y expulsiones! innecesarias.
- **Política de limpieza:** Cuándo se escriben en memoria secundaria las páginas modificadas.
  - **Bajo demanda:** Cuando son reemplazadas.
    Siempre sucede justo antes de la lectura de la página a cargar, el peor momento.
  - **Adelantada:** Cuando hay ciclos libres.
    Se llevan a cabo escrituras innecesarias.

OTRAS DECISIONES DE DISEÑO (III)

- **Tamaño de página:**
  - **Grande:** Fragmentación interna.
  - **Pequeña:** Tabla de páginas grande. Uso ineficiente de dispositivos de acceso como discos duros.
- **Control de carga:** Cantidad de procesos en ejecución:
  - **Pocos:** Se desperdicia tiempo de procesador.
  - **Muchos:** Se producen muchos fallos de página.

PROBLEMAS: COMPARTICIÓN

COMPARTICIÓN

- En principio la paginación no soporta la compartición de páginas.
- Pero es una característica deseable para ahorrar espacio en memoria.
- Varios procesos podrían contener referencias a las mismas páginas.
- **Modificaciones que hacen posible la compartición:**
  - Dividir los espacios de direcciones de un proceso en:
    - Espacio de instrucciones.
    - Espacio de datos.
  - Modificar la tabla de páginas para dar permisos sobre determinadas páginas:
    - Lectura (R), escritura (W) y ejecución (X).

	Valid	Referenced	Modified	Permissions	Frame
0	1	1	0	R_X	00FFAB
1	1	0	1	RW_	46FF00

PROBLEMAS DE LA COMPARTICIÓN

- No permite distinguir entre grupos de “usuarios”:
  - Sólo entre los niveles de acceso que proporcione el hardware.
  - Sería necesaria una implementación más complicada.
- La política de reemplazo se complica:
  - Varias tablas de páginas pueden hacer referencia al mismo marco.
- La política de limpieza se complica:
  - Es necesario controlar quién escribe y cuándo.
  - A veces puede ser necesario duplicar espacios de datos.
- **Alternativa:** Compartir procesos enteros. Su espacio completo de direcciones.
  - **Shared libraries:** Bibliotecas compartidas.