Sistemas Operativos I

Memoria Principal

Objetivos

- Proveer una descripción de la distintas formas de organizar el hardware de memoria
- Discutir las distintas técnicas de manejo de memoria, incluidos paginación y segmentación

Conceptos

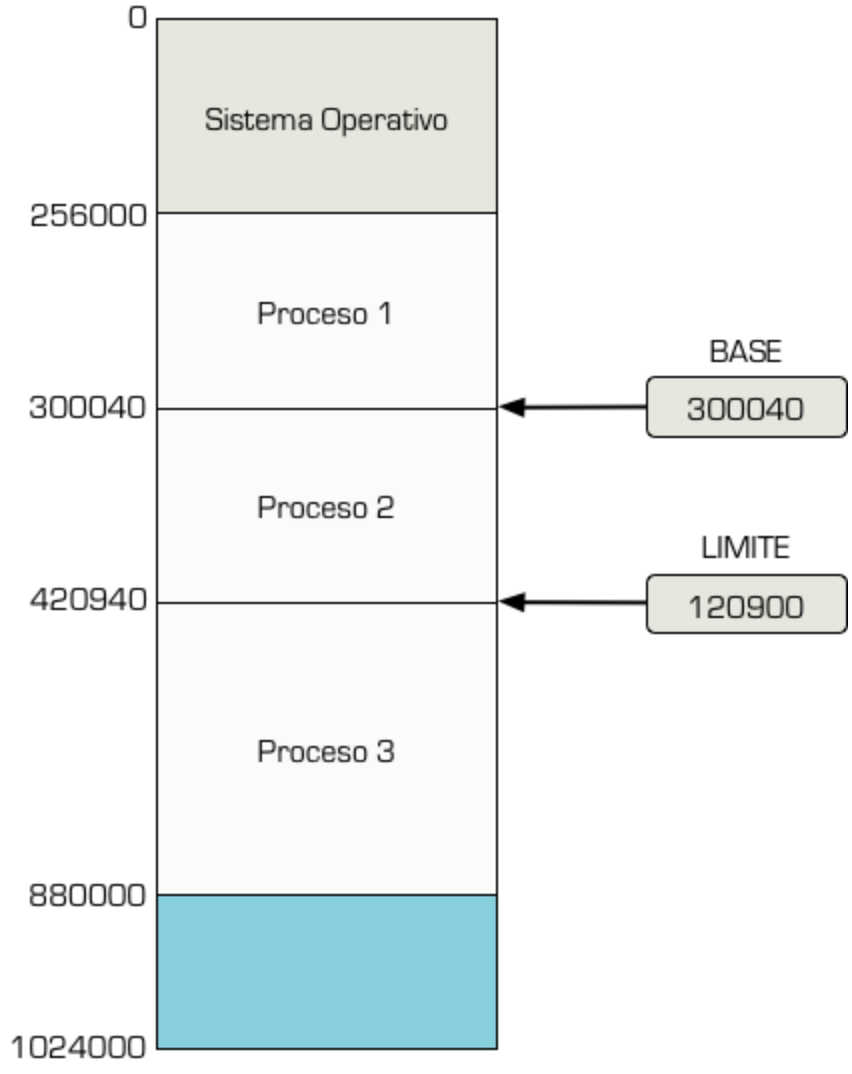
- La memoria principal y registros son el único almacenamiento que la CPU puede acceder directamente
- Los programas deben estar en memoria para ser ejecutados
- Se necesita protección de memoria para la correcta operación
- Velocidad de acceso:
    - Los registros se acceden en 1 ciclo de CPU
    - El acceso a memoria principal puede tomar varios ciclos
    - El cache se encuentra entre la memoria principal y los registros de CPU

Vinculación de Instrucciones y Datos

- La vinculación (binding) de instrucciones y datos a direcciones de memoria puede suceder en 3 etapas diferentes:
    - **Tiempo de compilación**: si la posición de memoria se conoce a priori, se puede generar código absoluto; se debe recompilar si la dirección de inicio cambia
    - **Tiempo de carga**: debe generar código reubicable si la posición de memoria no se conoce en tiempo de compilación
    - **Tiempo de ejecución**: la vinculación es postergada hasta el tiempo de ejecución si el proceso puede moverse durante su ejecución. Necesita soporte de hardware (p. ej., registros base y límite)

Registros Base y Límite

- Un par de registros base y límite definen el espacio lógico de direcciones



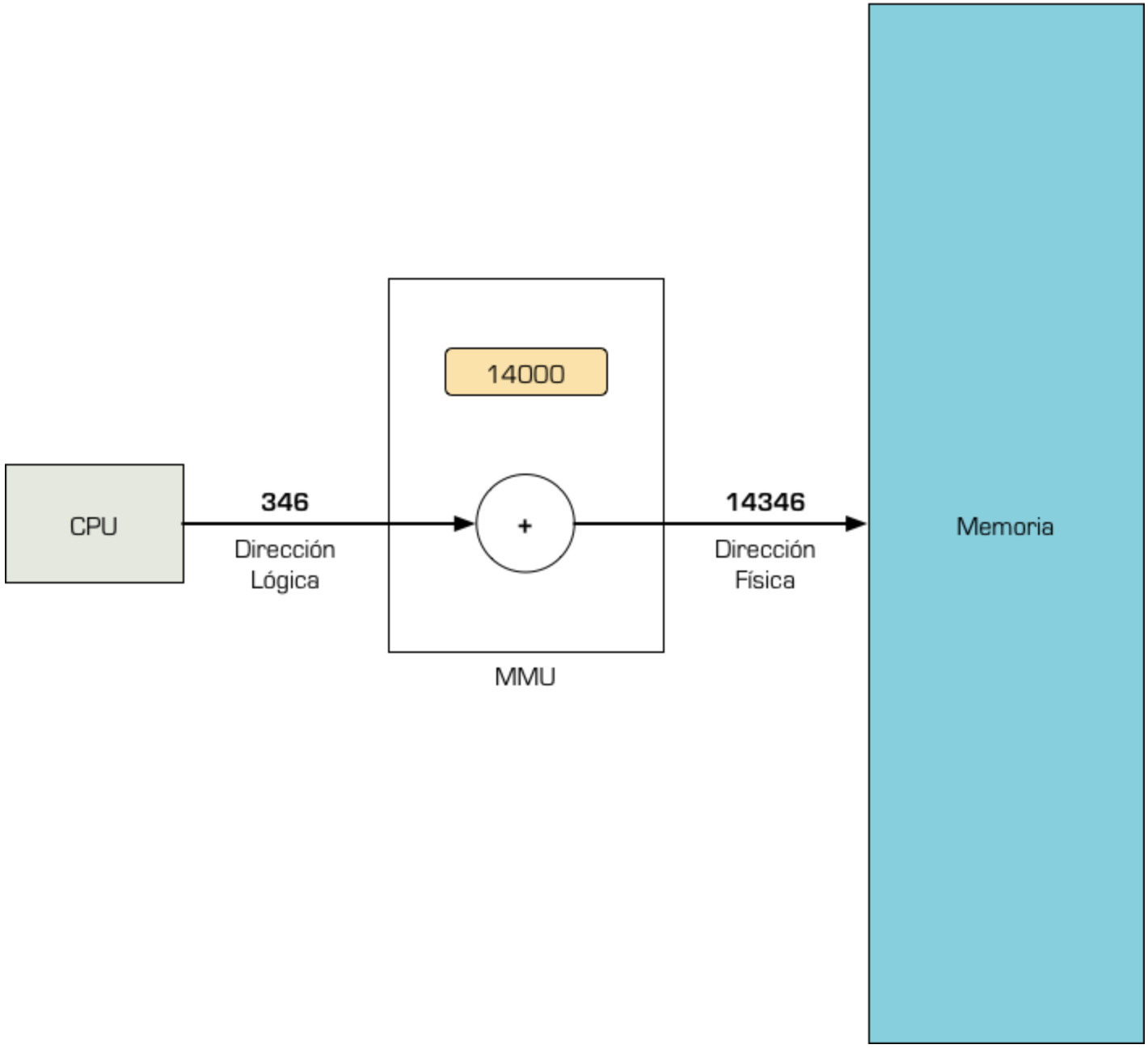
Espacio de Direcciones Lógico vs. Físico

- El concepto de espacio de direcciones lógico que está separado de un espacio físico es central en la administración de memoria
  - **Dirección lógica** – generada por la CPU; también conocida como dirección virtual
  - **Dirección Física** – dirección que ve la unidad de memoria
- Las direcciones lógicas y físicas son las mismas en la vinculación de tiempo de compilación y carga; y difieren en la vinculación de tiempo de ejecución

Unidad de Administración de Memoria (MMU)

- Dispositivo de hardware que mapea la memoria virtual o lógica con la memoria física
- En un esquema con MMU, el valor del registro de re-ubicación es sumado a cada dirección generada por un proceso del usuario cuando se envía a memoria
- Los programas del usuario trabajan con direcciones lógicas; nunca ven las direcciones físicas reales

Re-ubicación Dinámica



Carga Dinámica

- La rutina no es cargada hasta que no se invoca
- Mejor aprovechamiento del espacio de memoria; una rutina que no se utiliza nunca es cargada
- Útil cuando grandes partes de código son necesarias para manejar casos que ocurren infrecuentemente
- No se necesita soporte especial de SO; se puede implementar a través del diseño del programa

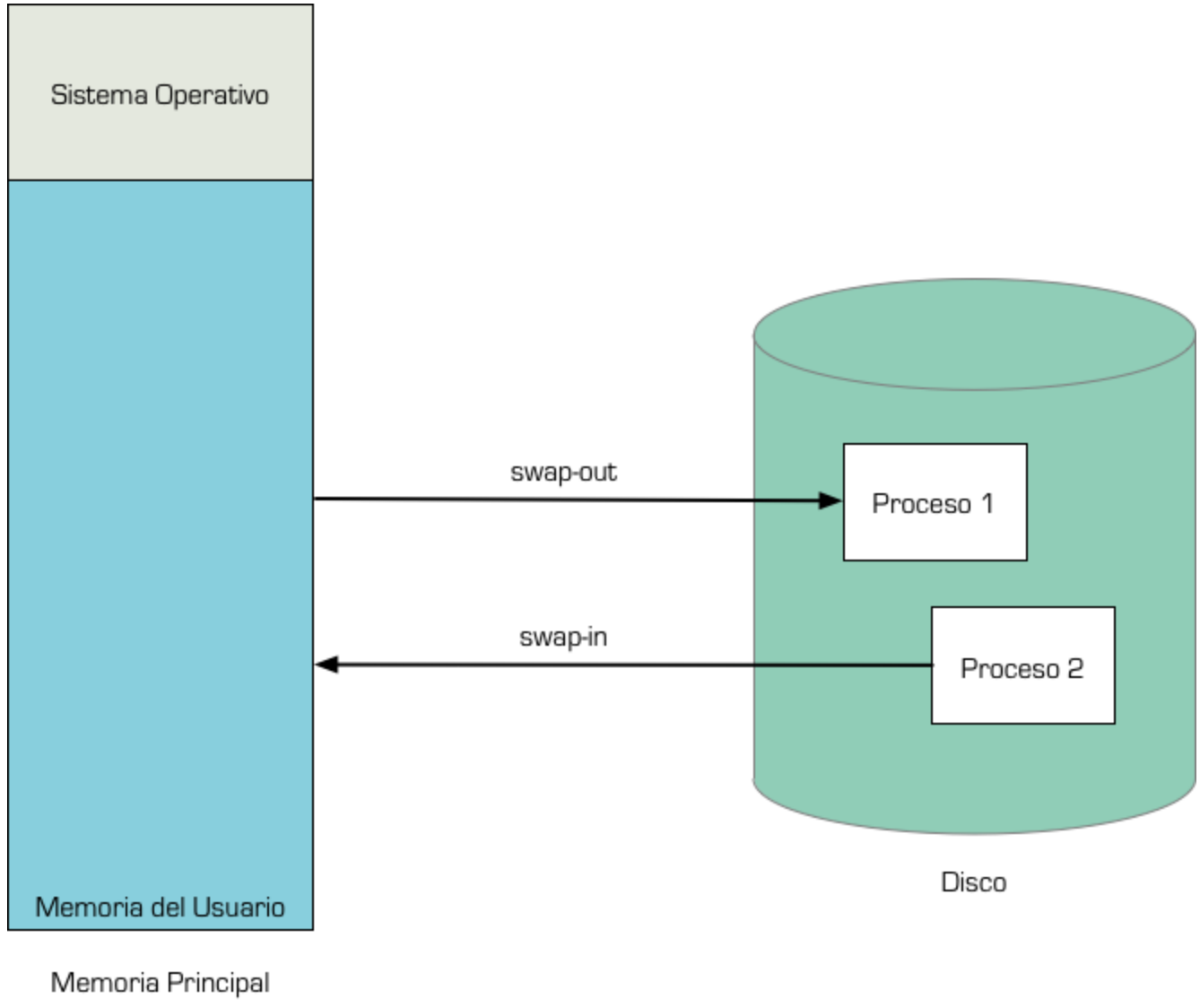
Enlace Dinámico

- El enlace se pospone hasta el tiempo de ejecución
- Una pequeña pieza de código, **stub**, se usa para encontrar la rutina de librería apropiada
- El **stub** se reemplaza a si mismo con la dirección de la rutina, y ejecuta la rutina
- Los sistemas operativos necesitan verificar si la rutina está en la memoria del proceso
- El enlace dinámico es particularmente útil para librerías

Swapping

- Un proceso puede ser sacado temporalmente de memoria a almacenamiento secundario, y traído nuevamente a memoria para continuar su ejecución
- Almacenamiento secundario – discos rápidos y suficientemente grandes para acomodar copias de la imagen de memoria; debe proveer acceso directo
- Roll out, roll in – variante de swapping usado para planificadores por prioridad; los procesos de menor prioridad son desalojados de forma que los de mayor prioridad puedan ejecutarse
- La mayor parte del swap es tiempo de transferencia; proporcional al tamaño de memoria

Diagrama de Swapping

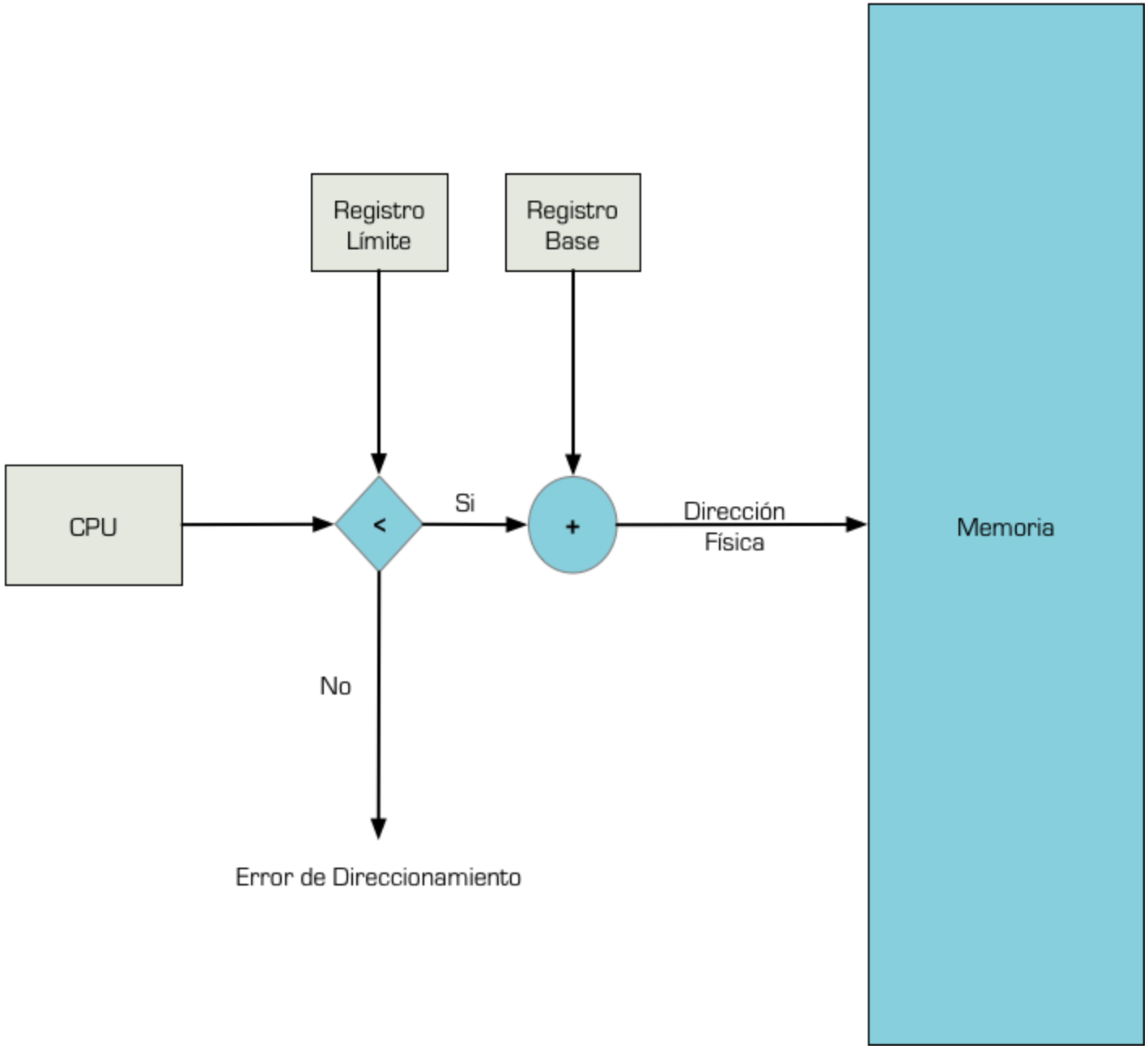


Asignación Continua

Asignación Continua

- La memoria se divide generalmente en 2 particiones:
  - SO residente, generalmente en la parte baja de memoria junto al vector de interrupciones
  - Procesos del usuario
- Se utilizan registros de re-ubicación para proteger los procesos entre si, y al sistema operativo
  - **Registro base** – contiene el valor de la menor dirección física
  - **Registro límite** – contiene el rango de direcciones lógicas; las direcciones lógicas deben ser menor que este valor
  - La MMU mapea direcciones lógicas en forma dinámica

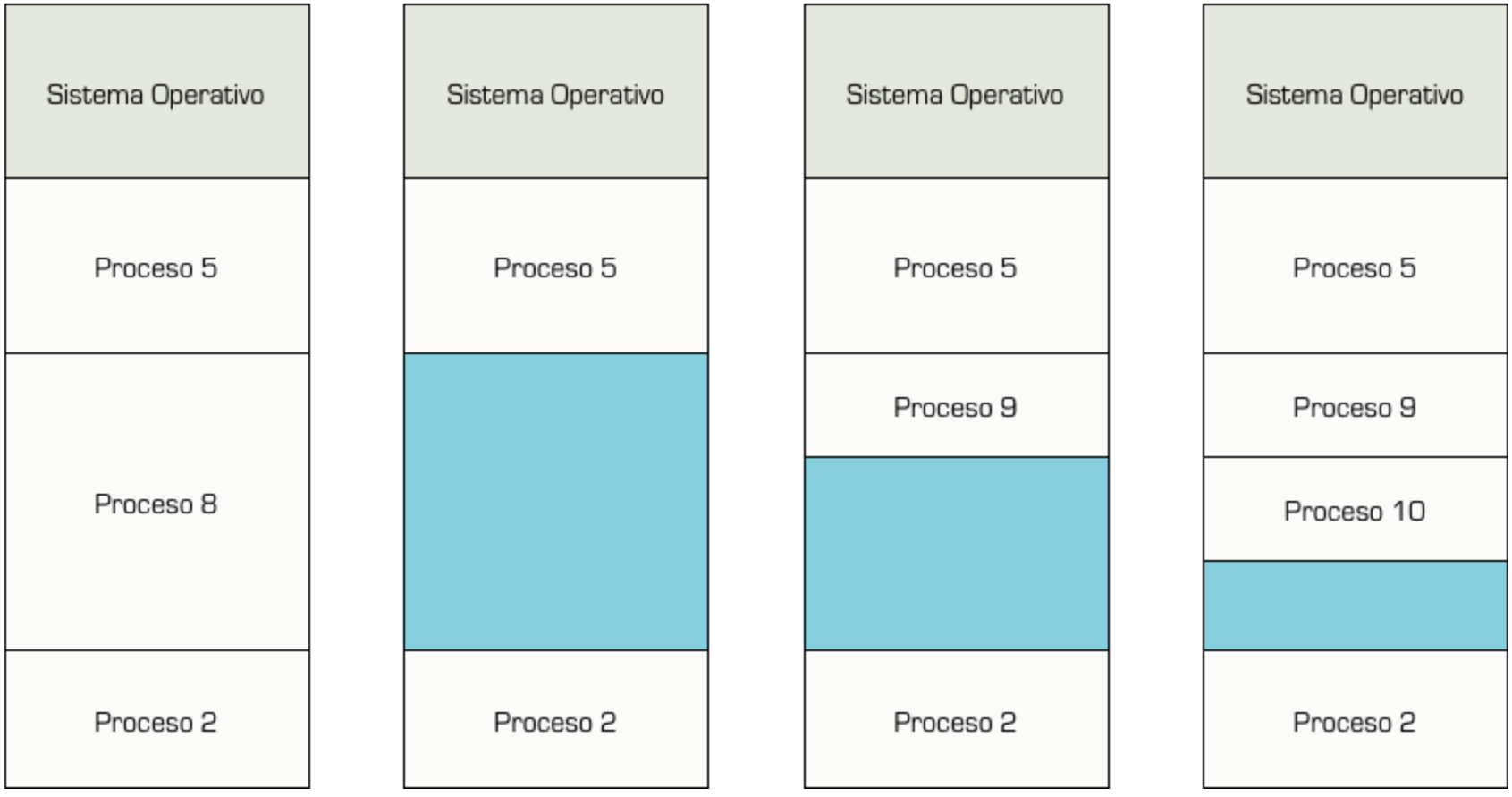
Soporte de Hardware



Asignación Continua

- Asignación con múltiples particiones
  - **Hueco** – bloque de memoria disponible; de varios tamaños desparramados por memoria
  - Cuando un proceso llega, se le asigna un hueco lo suficientemente grande
  - El SO mantiene información de: **a)** particiones asignadas **b)** particiones libres (huecos)

Múltiples Particiones



Problema de Asignación Dinámica

> Cómo satisfacer un pedido de tamaño n con una lista de huecos

- **First-fit**: asigna el primer hueco libre que es suficientemente grande
- **Best-fit**: asigna el menor hueco que es suficientemente grande para contener al proceso; debe buscar en toda la lista, salvo que este ordenada
    - Produce el desperdicio más chico por hueco
- **Worst-fit**: asigna el bloque más grande; debe recorrer la lista completa
    - Deja el desperdicio más grande

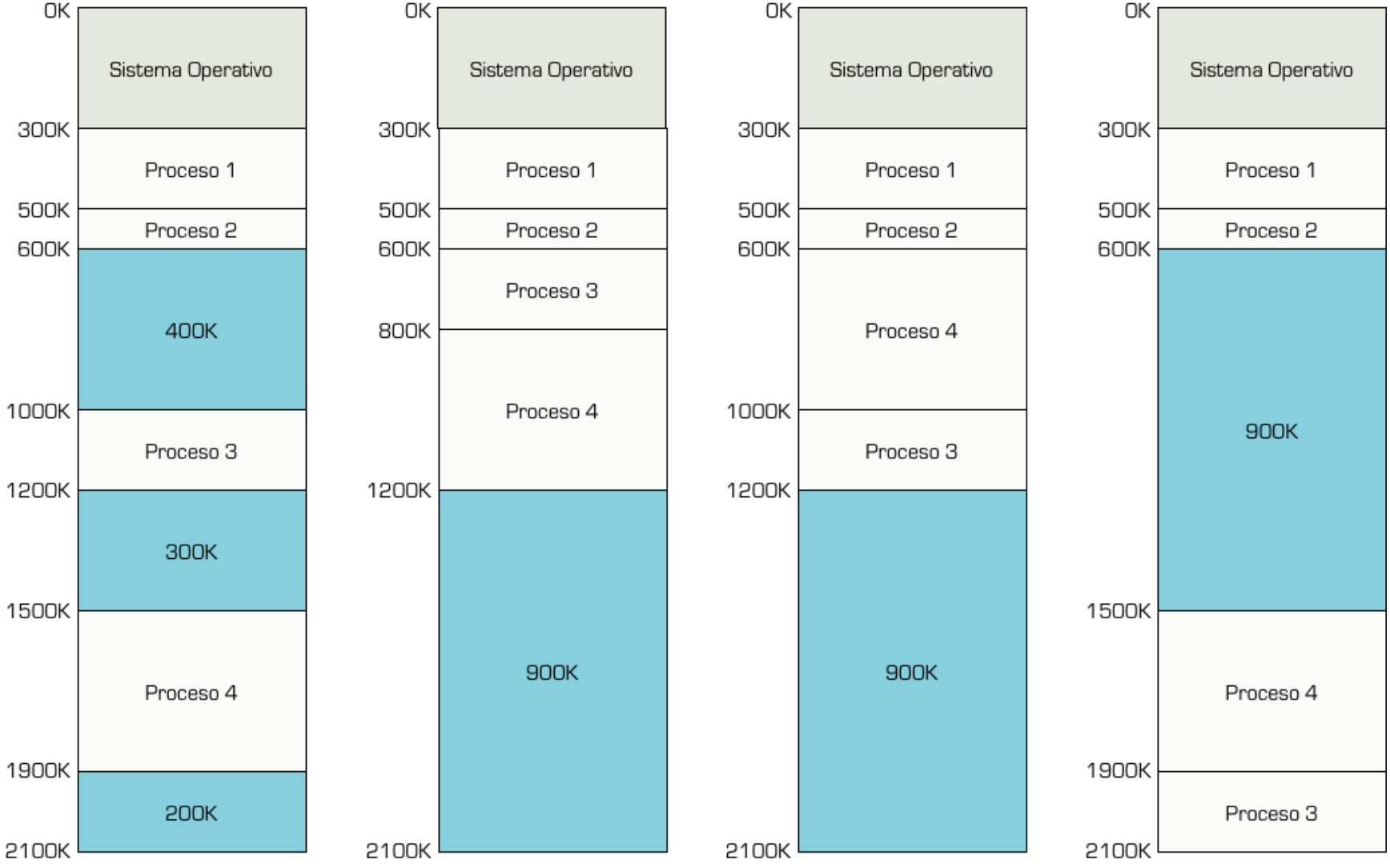
Fragmentación

- **Fragmentación Externa** – existe suficiente para satisfacer un pedido, pero no en forma continua
- **Fragmentación Interna** – la memoria asignada puede ser algo mayor que la requerida; esta diferencia es interna a una partición pero no es usada

Compactación

- Reduce la fragmentación externa por medio de la compactación
- Mueve el contenido de memoria para ubicar toda la memoria libre junta
- La compactación es posible sólo si tenemos re-ubicación dinámica, y el binding es hecho en tiempo de ejecución
- Problemas de I/O
    - Anclar el proceso en memoria si está haciendo E/S
    - Hacer E/S sólo a buffers del SO

Ejemplo: Compactación



Paginación

Paginación

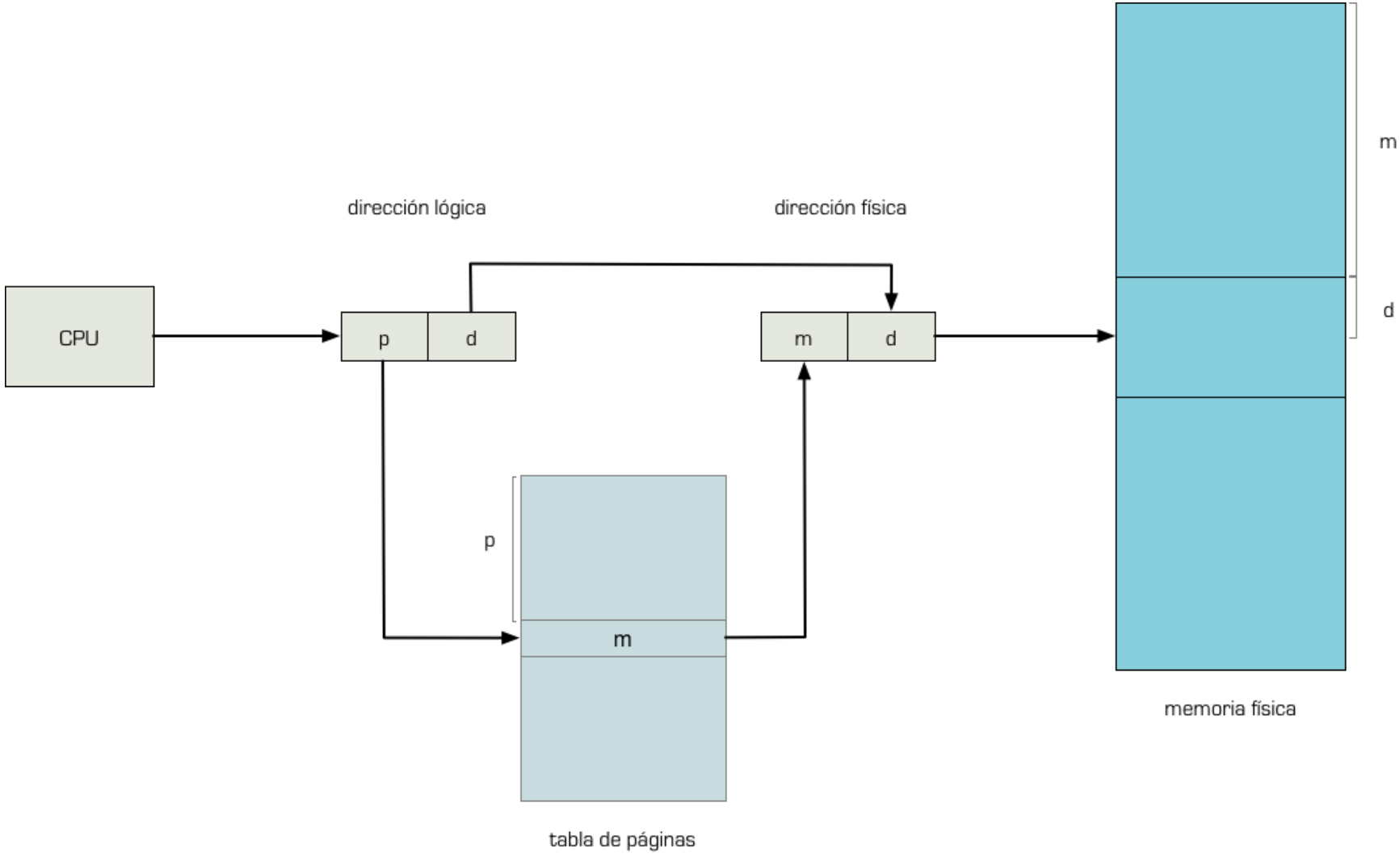
- El espacio físico de direcciones puede no ser continuo; se le asigna memoria a un proceso siempre que haya disponible
- Divide la memoria física en bloques de tamaño fijo llamados marcos (frames) (el tamaño es una potencia de 2, entre 512 bytes y 8192 bytes)
- Divide la memoria lógica en bloques de igual tamaño llamados páginas
- Mantiene registro de todos los marcos vacíos
- Para ejecutar un programa de n páginas, necesita encontrar n marcos libres y cargar el programa
- Inicializar una tabla de páginas que permita traducir una dirección lógica a una física
- Fragmentación interna

Esquema de Traducción

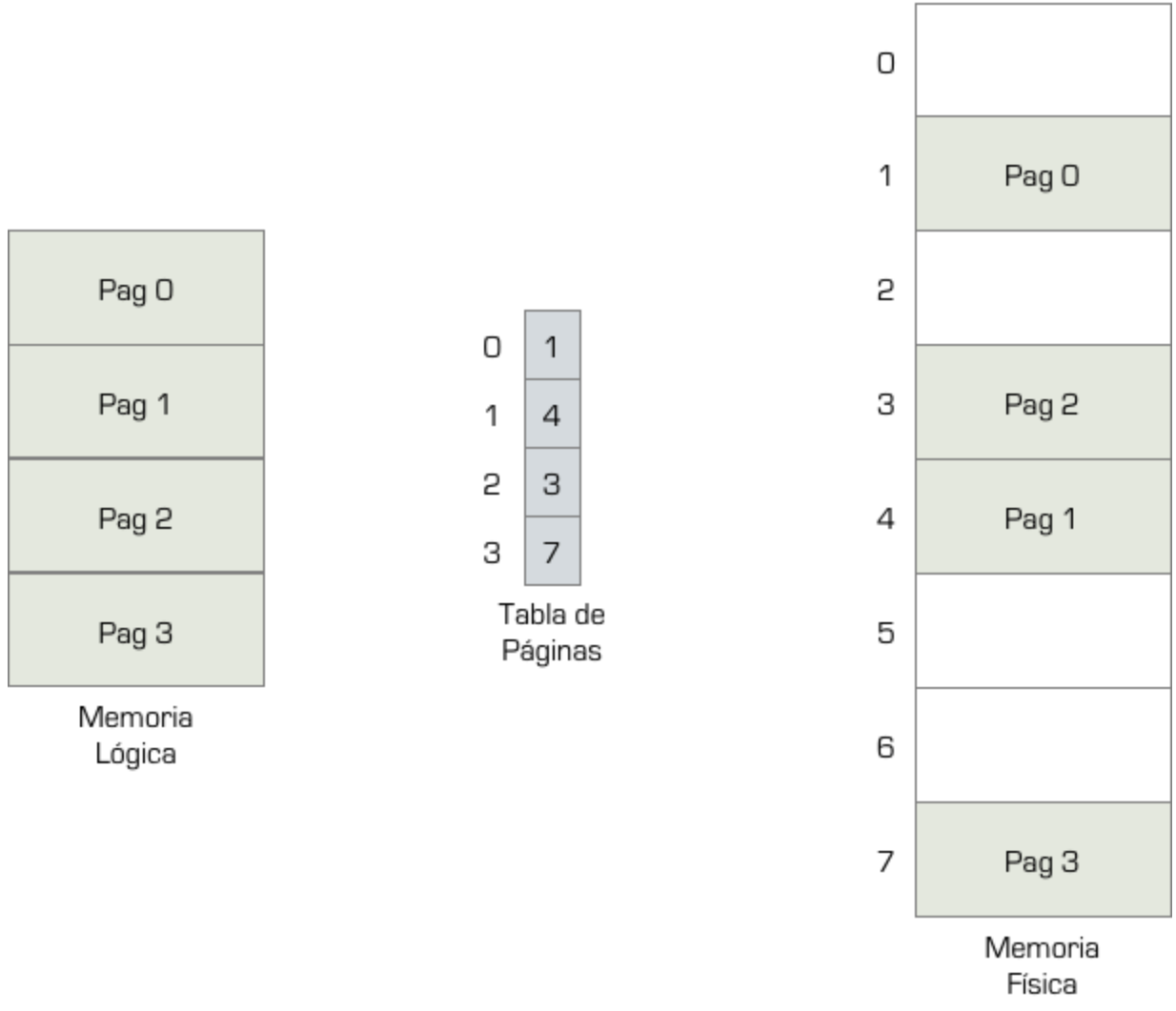
- Las direcciones generadas por la CPU son divididas en:
- **Número de Página (p)** – usado para indexar dentro de la tabla de página que contiene la dirección base de cada marco
- **Desplazamiento de Página (d)** – combinado con la dirección base para definir la dirección física de memoria que se envía a la unidad de memoria

- Para un espacio de direcciones lógico $$2^m$$ y un tamaño de página de $$2^n$$

Hardware de Paginación



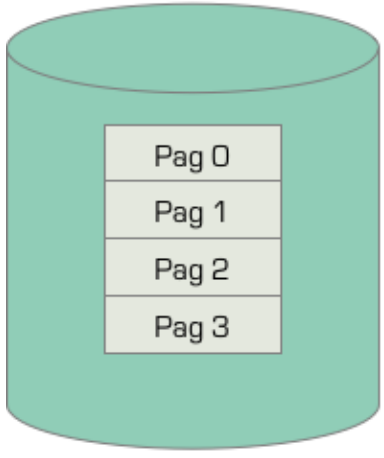
Memoria Lógica y Física

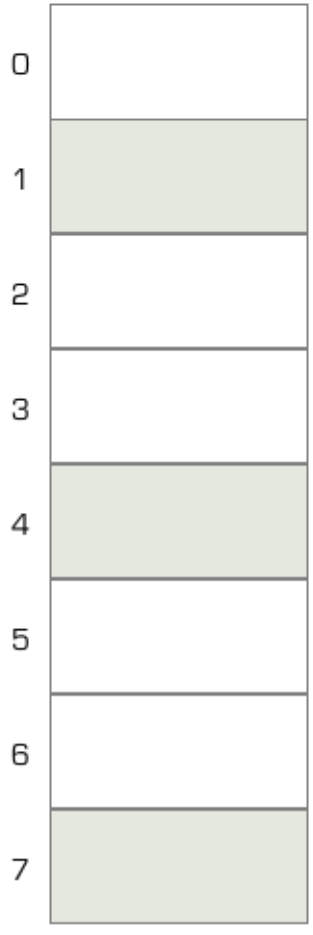


Marcos Libres

Marcos Libres

6
2
5
3
0





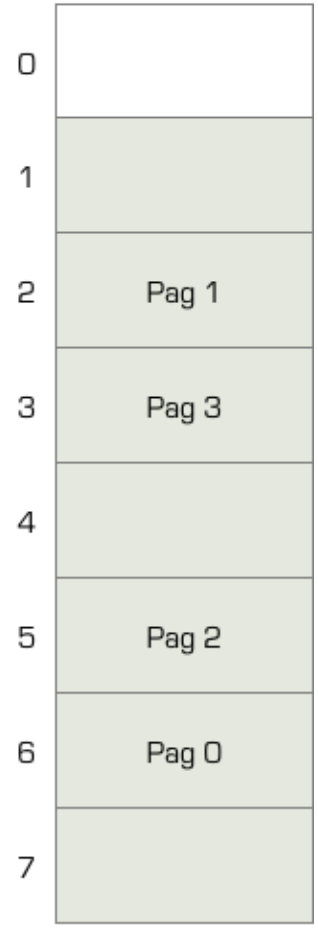
Memoria Física

Marcos Libres

0

0	6
1	2
2	5
3	3

Tabla de Páginas



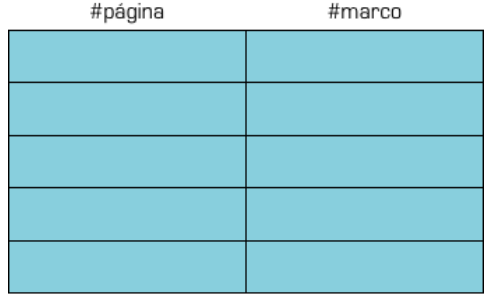
Memoria Física

Implementación

- La tabla de página se mantienen en memoria principal
- Page-table base register (PTBR) apunta a la tabla
- Page-table length register (PRLR) indica el tamaño de la tabla
- En este esquema cada instrucción o dato requiere 2 accesos a memoria. Uno a la tabla de páginas y otro al dato o instrucción

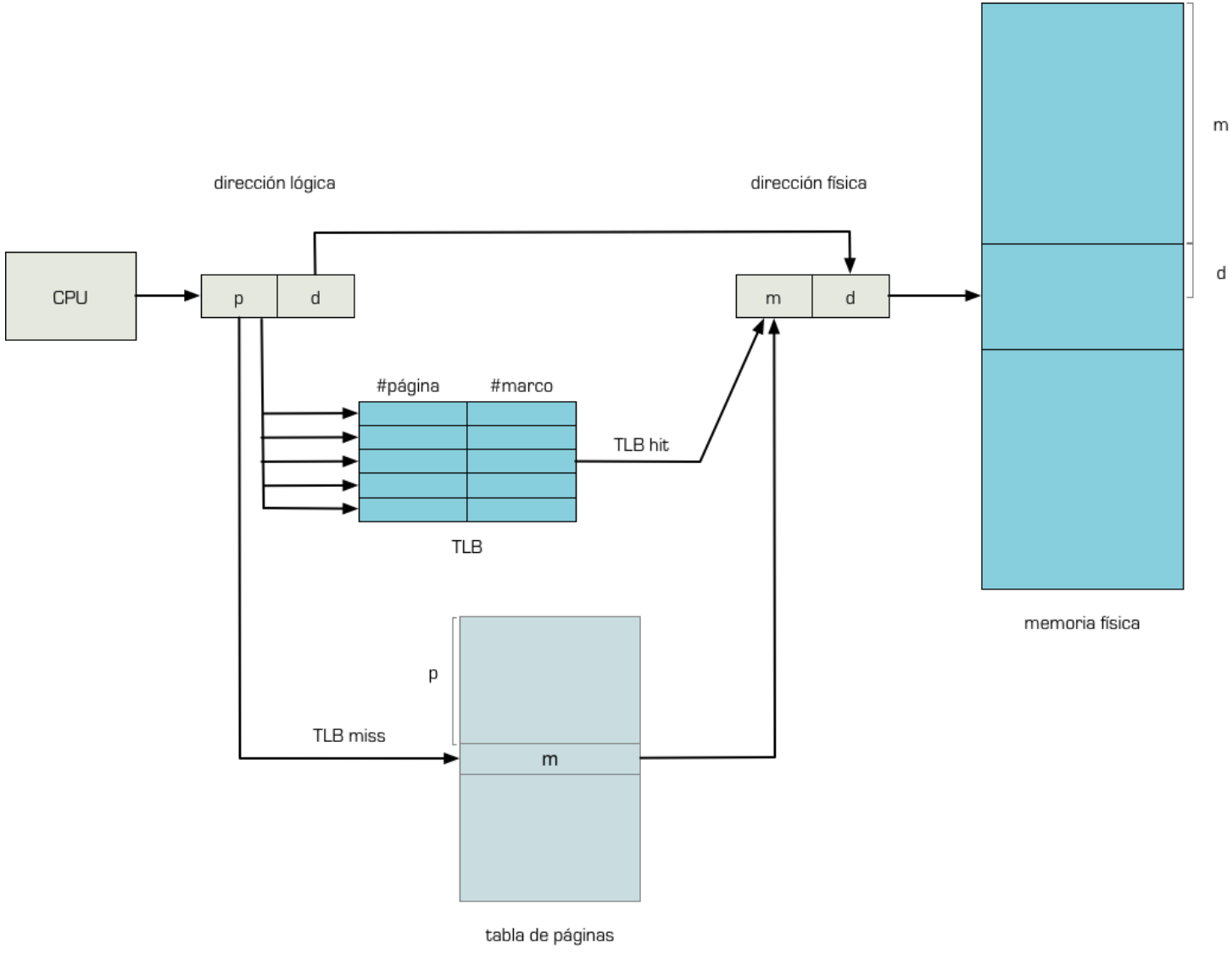
Memoria Asociativa (TLB)

- **Memoria asociativa** – búsqueda
- Traducción (p, d)



- Si p está en memoria asociativa, obtiene el # de marco
- Si no, obtiene el # de marco de la tabla en memoria
- Entre 64 y 2048 entradas

Hardware de Paginación con TLB



^ Invalidación de TLB (PTBR)

Tiempo de Acceso Efectivo

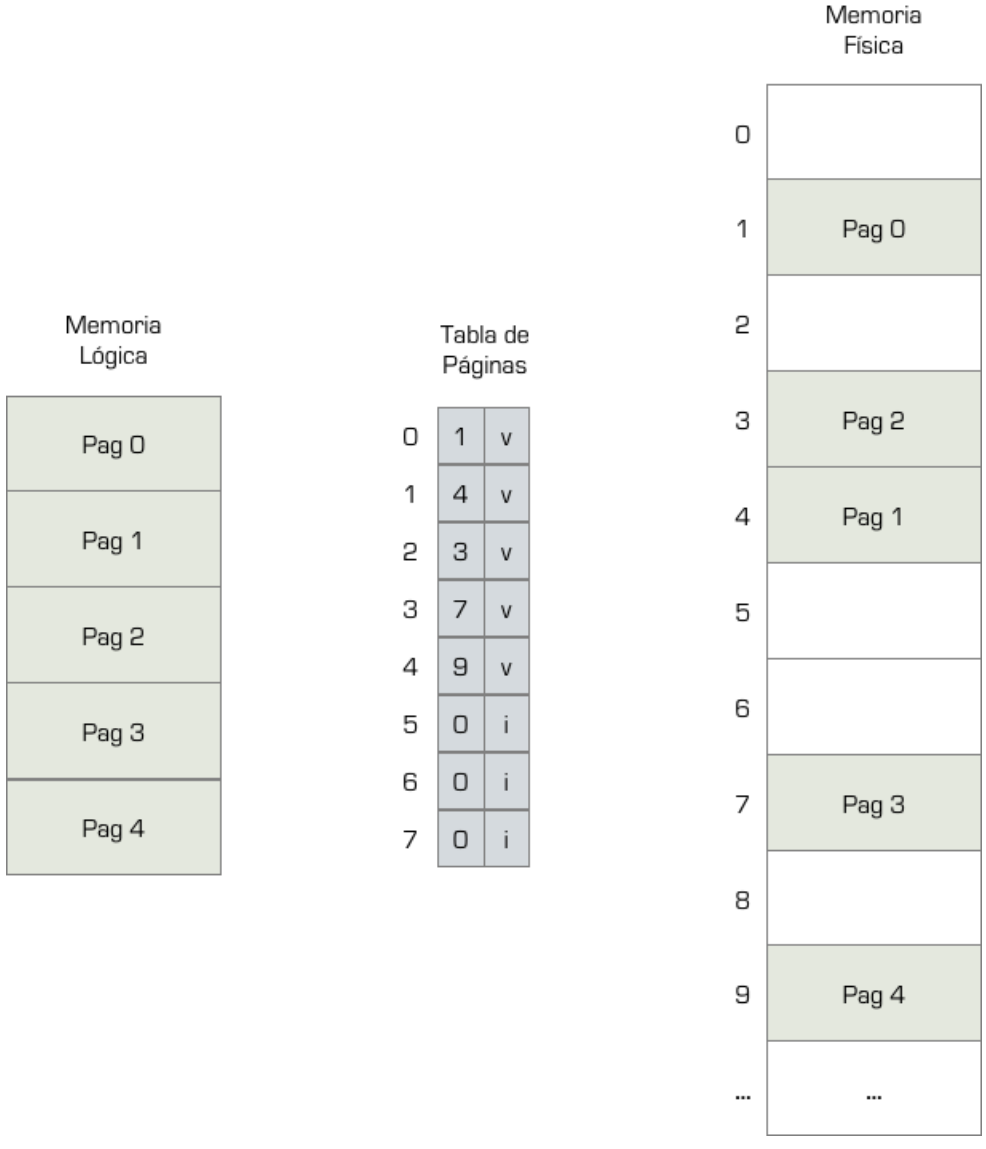
- **Tasa de aciertos** – porcentaje de las veces que la página es encontrada en la memoria asociativa
- **Búsqueda asociativa** = ε unidades de tiempo
- Asumiendo que el acceso a memoria es 1 µseg
- Tasa de aciertos = α
- **Tiempo de Acceso Efectivo** (TAE): TAE = (1 + ε) α + (2 + ε)(1 – α) = 2 + ε – α

> *si α = 85% y ε=0.1 EAT=1.25, α = 98% EAT=1.12*

Protección de Memoria

- La protección de memoria se implementa asociando bits de protección a cada marco
- Un bit válido-invalido se asocia a cada entrada en la tabla de páginas:
    - **válido** indica que la página asociada está en el espacio lógico de direcciones del proceso, y es una página válida
    - **inválido** indica que la página no está en el espacio lógico de direcciones del proceso

Bit Válido/Inválido



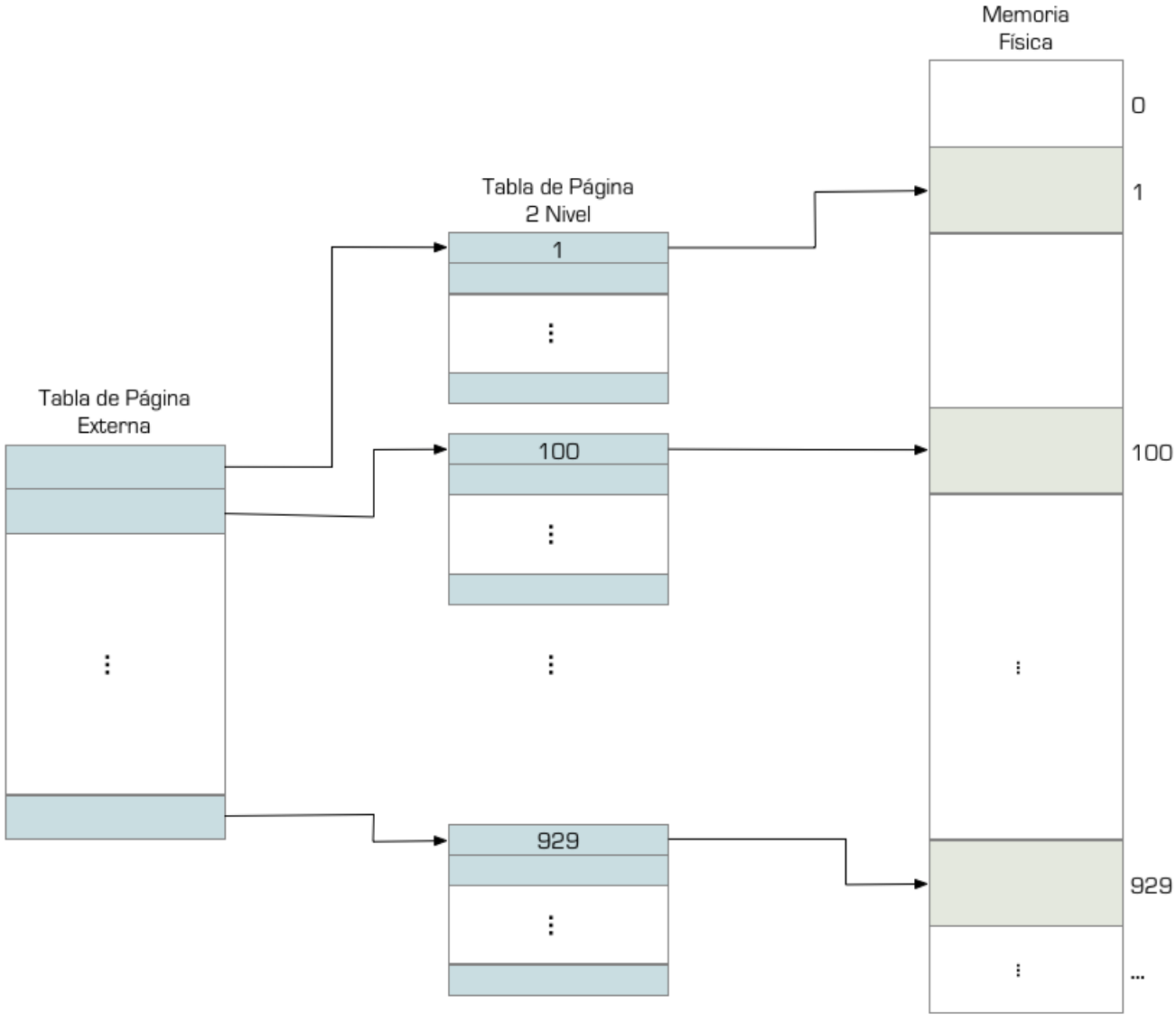
Estructura de la Tabla de Páginas

- Paginación Jerárquica
- Tablas de Páginas Hasheadas
- Tabla de Páginas Invertidas

Tabla de Páginas Jerárquica

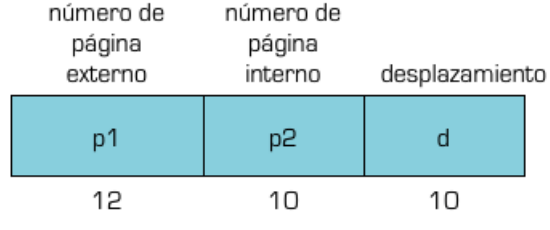
- Parte el espacio lógico de direcciones en múltiples tablas de página
- Una técnica simple es usa una tabla de 2 niveles

Esquema de 2-Niveles



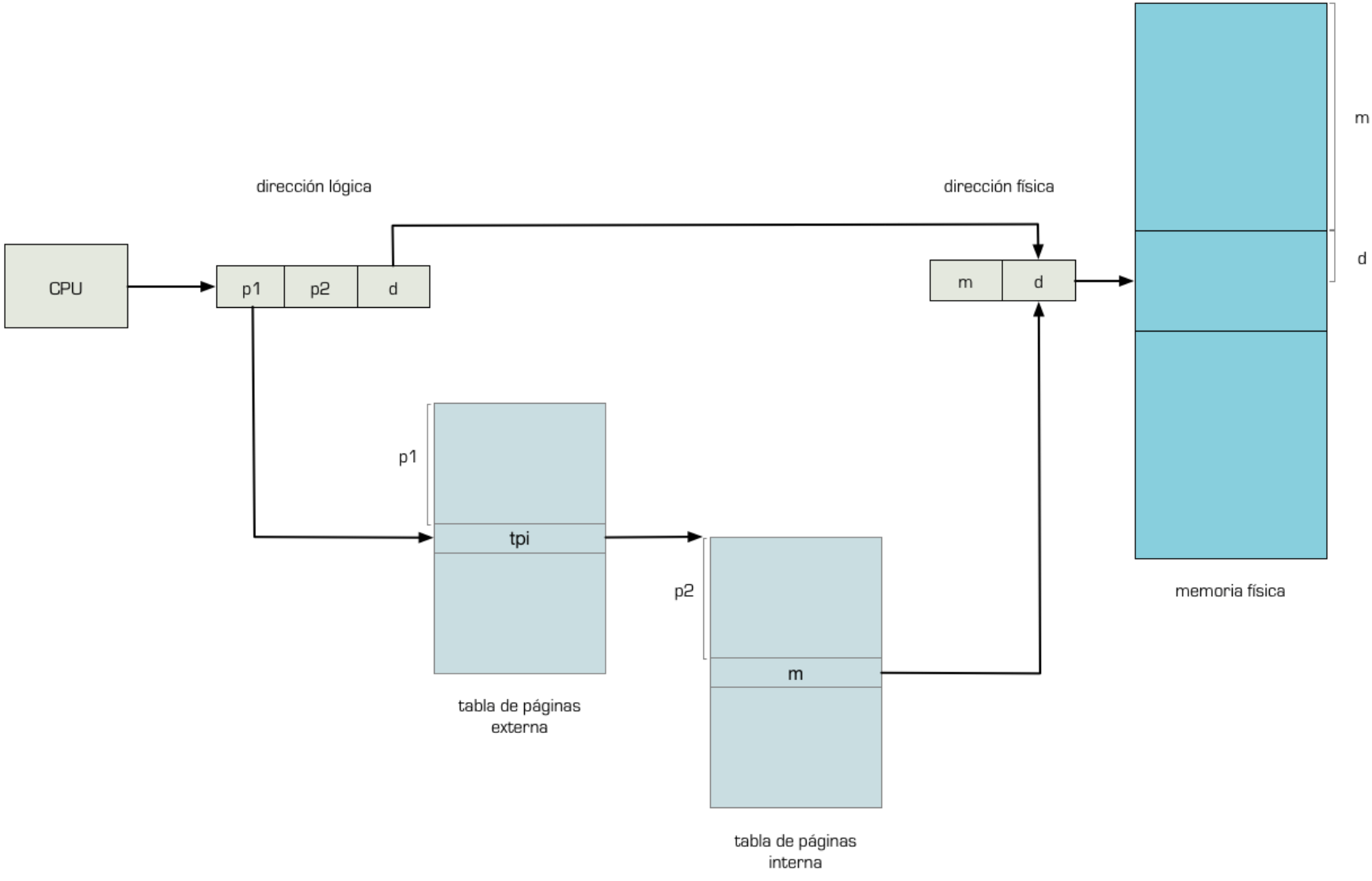
Ejemplo de 2-Niveles

- Una dirección lógica (en una máquina de 32-bits con páginas de 1K) es dividida en:
  - Un número de página que de 22 bits
  - Un desplazamiento de página de 10 bits
- Dado que la tabla de página es paginada, el número de página se divide a su vez en:
  - Un número de página de 12-bits
  - Un desplazamiento de página de 10-bits
- Entonces, una dirección lógica queda:

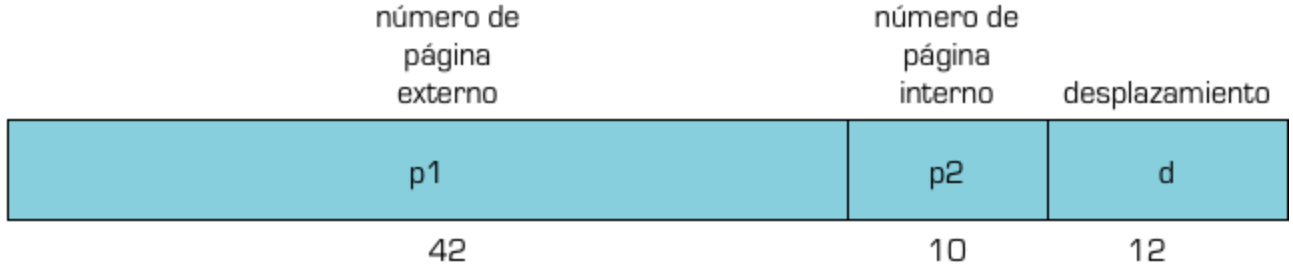


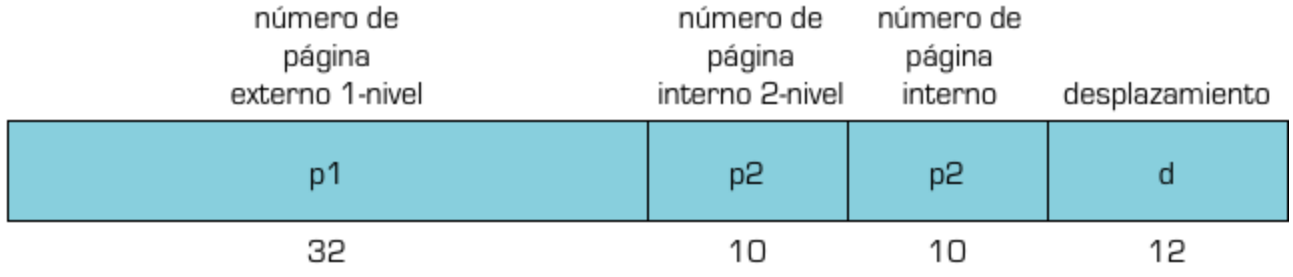
  - donde **p1** es el índice en la tabla de páginas interna, y **p2** es el desplazameinto en la tabla de página externa

Esquema de Traducción 2-Niveles



Esquema de 3-Niveles

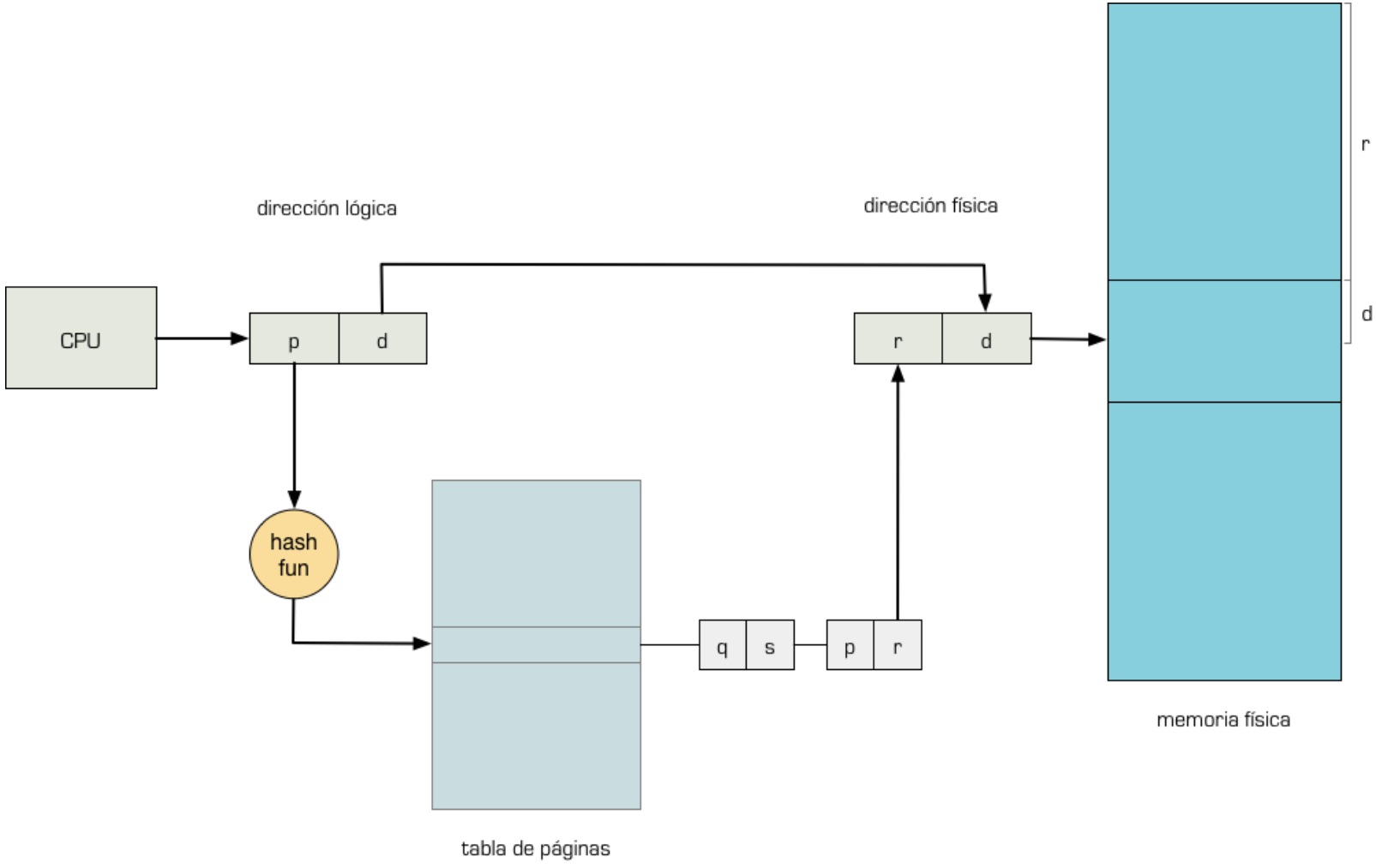




Esquema Hasheado

- Común en espacios de direcciones > 32 bits
- El número de página lógica es hasheado en la tabla de paginas
    - La tabla contiene un lista de elementos hasheados a la misma dirección
- Los números de página lógica es comparado con la lista de páginas
    - Si se encuentra, se extrae el marco físico correspondiente

Esquema de Traducción

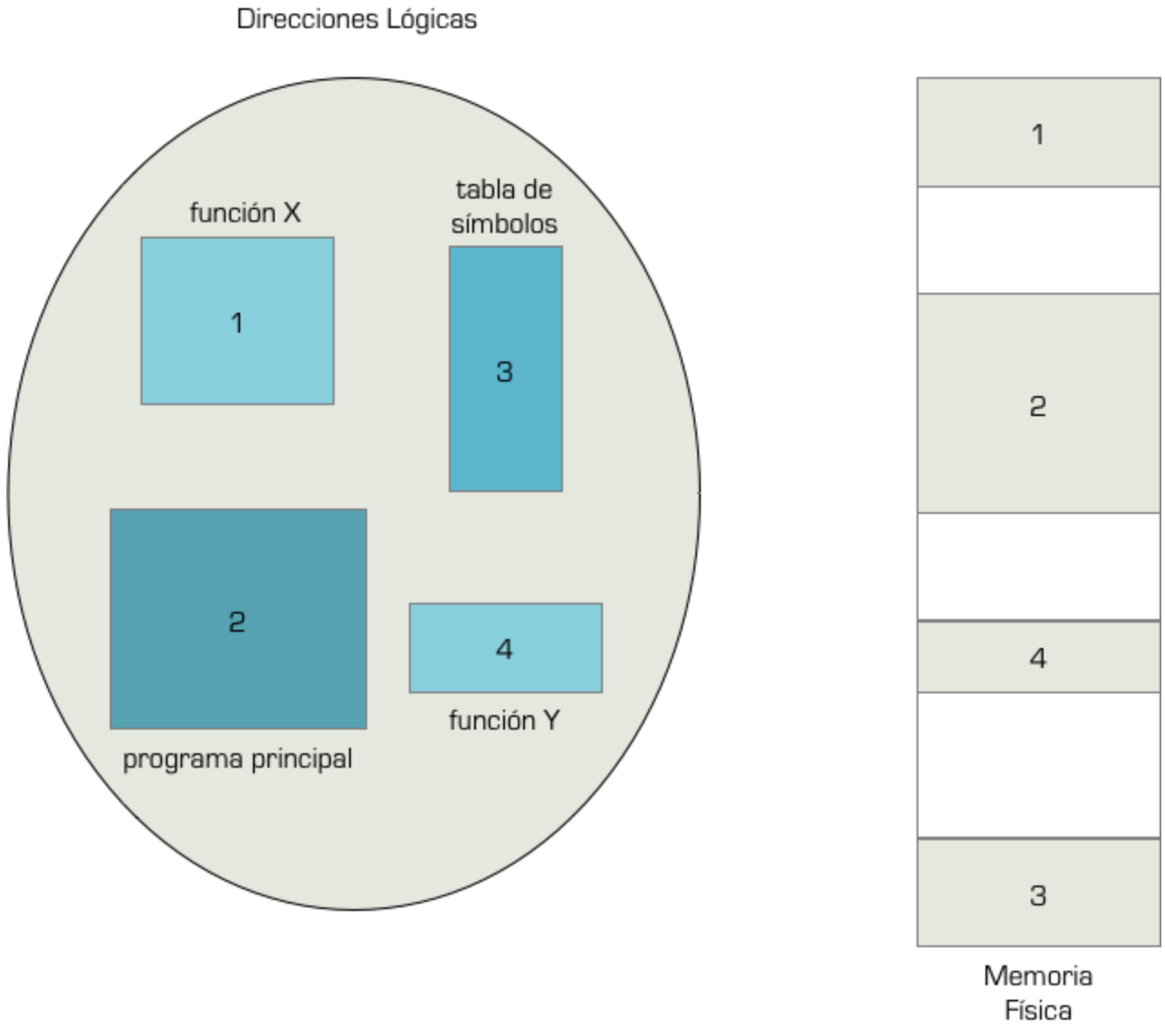


Segmentación

Segmentación

- Esquema de manejo de memoria que soporta la vista del usuario de la memoria
- Un programa es un conjunto de segmentos
- Un segmento es una unidad lógica:
    - programa principal: procedimiento, función, variables locales, variables locales, stack, tabla de símbolos...

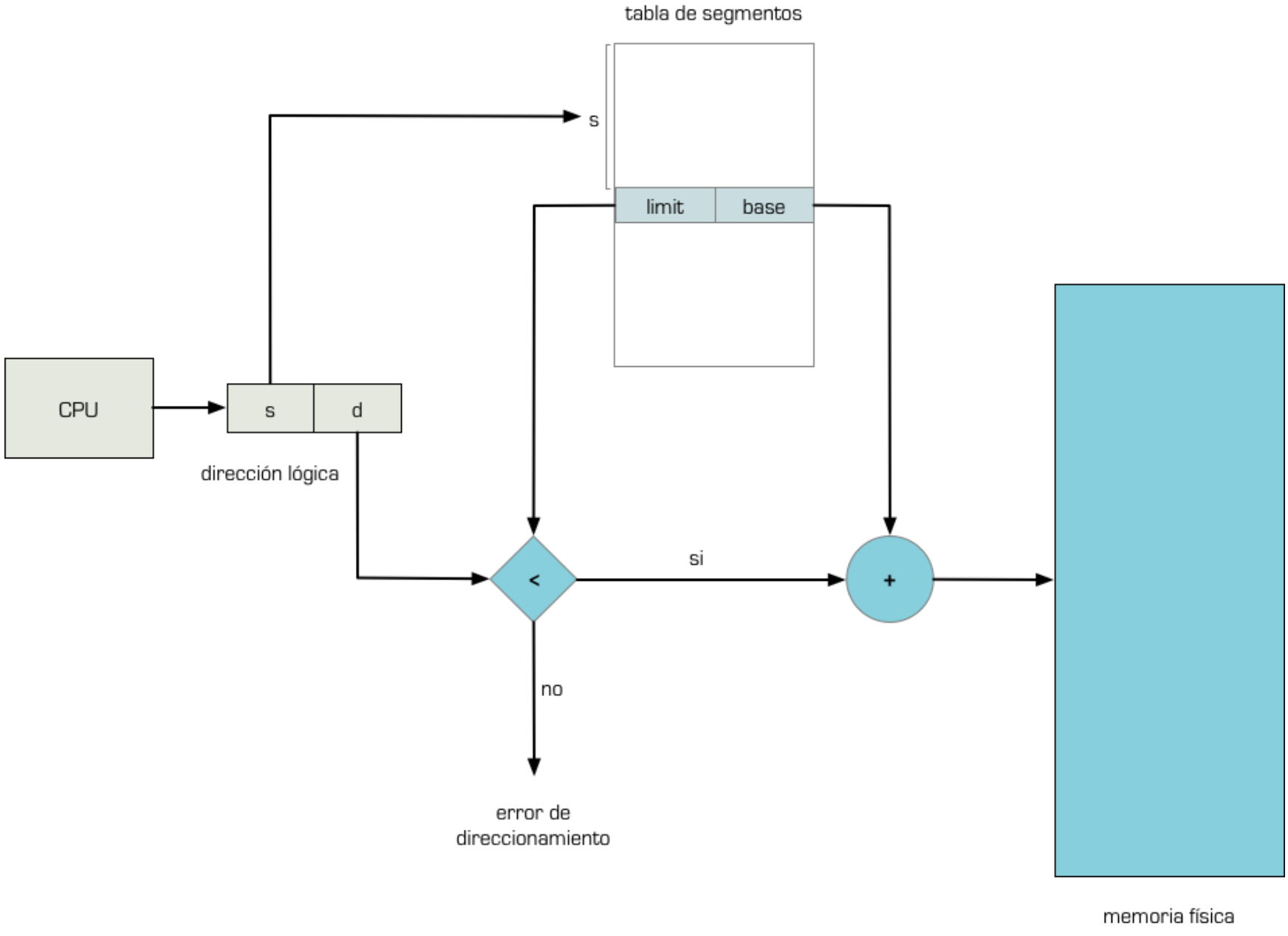
Visión Lógica



Arquitectura de Segmentación

- La dirección lógica ahora consiste de una tupla: `<#segmento, desplazamiento>``
- **Tabla de segmentos** – mapea direcciones físicas bi-dimensioaneles; cada entrada tiene:
    - **base** – contienen dirección física de comienzo del segmento en memoria
    - **límite** – especifica la longitud del segmento
- Registro base de tabla de segmento (STBR) apunta a la ubicación de la tabla de segmentos en memoria
- Registro de tamaño de tabla(STLR) indica el número de segmentos usados por un programa;
    - el #segmento s es válido si s < STLR

Hardware de segmentación



Otras Consideraciones

- Protección
  - Con cada entrada en la tabla de segmentos se asocia:
    bit de validez = 0 ⇒ segmento ilegal
- Privilegios de read/write/execute
- Dado que los segmentos varían en tamaño, la asignación de memoria es un problema de asignación dinámica