# GESTIÓN DE MEMORIA

UNIDAD 8

# GESTIÓN DE MEMORIA.

- Requerimientos de la administración de memoria.
  - Reubicación
  - Protección
  - Compartición
- Organización Física y Organización Lógica.
- Administración de la memoria sin intercambio
  - Monoprogramación sin intercambio.
  - Multiprogramación y uso de memoria.
  - Multiprogramación con particiones fijas
- Intercambio
  - Multiprogramación con particiones variables.
  - Administración de la memoria mediante: Mapa de bits. Listas ligadas. Sistema de Compañeros (Buddy System)
  - Asignación del hueco de intercambio
  - Análisis de sistemas con intercambio: Regla del cincuenta por ciento. Regla de la memoria no utilizada.
- Paginación Pura
- Segmentación Pura
- Segmentación Paginada

# Gestión de la Memoria

- La administración de memoria es una tarea realizada por el sistema operativo que consiste en ***gestionar la jerarquía de memoria, en cargar y descargar procesos en memoria principal para que sean ejecutados***. Para ello el sistema operativo gestiona lo que se conoce como ***MMU*** o ***Unidad de Administración de Memoria***, el cual es un dispositivo hardware que transforma las direcciones lógicas en físicas.

- Su trabajo es seguir la pista de que ***partes de la memoria están en uso y cuales no lo están***, con el fin de poder asignar memoria a los procesos cuando la necesiten, y recuperar esa memoria cuando dejen de necesitarla, así como ***gestionar el intercambio entre memoria principal y el disco*** cuando la memoria principal resulte demasiado pequeña para contener a todos los procesos

# Objetivos de la Gestión de Memoria

- Ofrecer a cada proceso un espacio lógico propio
- Proporcionar protección entre los procesos
- Permitir que los procesos compartan memoria
- Maximizar el rendimiento del sistema

# REQUERIMIENTOS DE LA ADMINISTRACIÓN DE MEMORIA

Reubicación, Protección, Compartición, Organización Física y Lógica

# Requisitos de la Gestión de Memoria

**Reubicación**

- En un sistema multiprogramado la memoria se encuentra compartida por varios procesos, por lo tanto, los procesos deben ser cargados y descargados de memoria.

**Protección**

- En un sistema con multiprogramación es necesario proteger al sistema operativo y a los otros procesos de posibles accesos que se puedan realizar a sus espacios de direcciones.

**Compartición**

- En ciertas situaciones, bajo la supervisión y control del sistema operativo, puede ser provechoso que los procesos puedan compartir memoria.

# Requisitos de la Gestión de Memoria

**Organización Lógica**

- Tanto la memoria principal como la secundaria presentan una organización física similar, como un ***espacio de direcciones lineal y unidimensional***. Debe existir una cierta correspondencia entre el sistema operativo y el hardware al tratar los datos y los programas de los usuarios de acuerdo a la estructura lógica que ellos presenten.

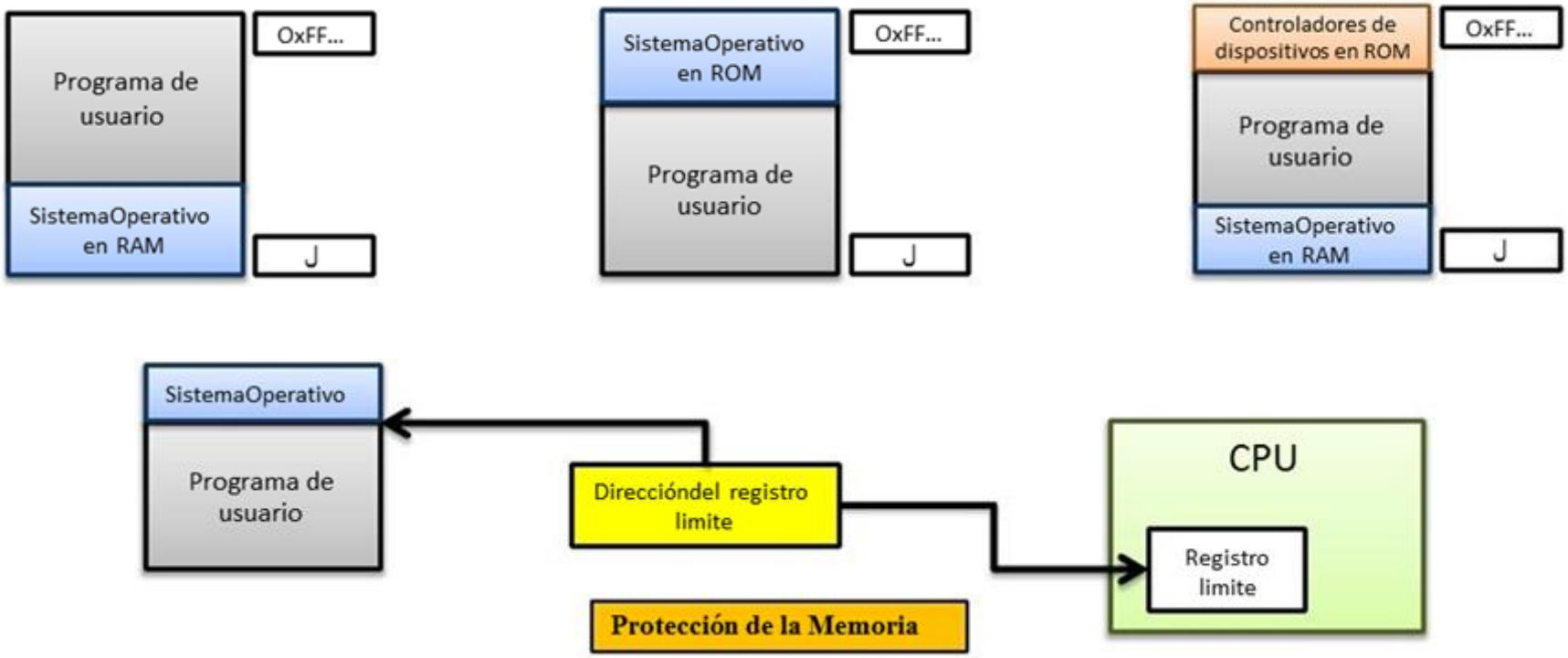
**Organización Física**

- Debe ser parte de la administración de memoria, la organización del flujo de información entre la memoria principal y la memoria secundaria.

# ADMINISTRACIÓN DE LA MEMORIA SIN INTERCAMBIO

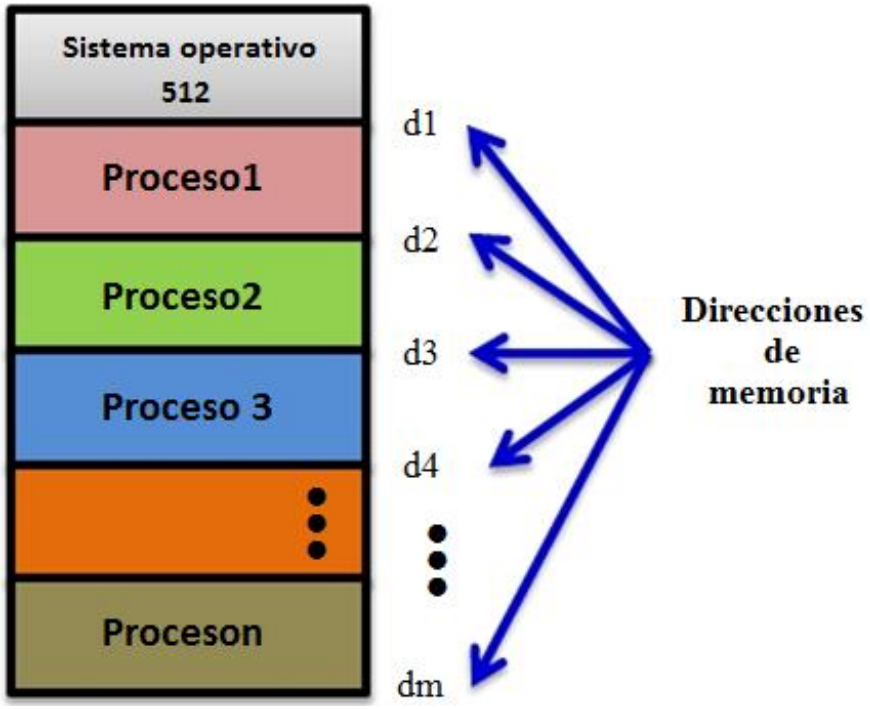
# Monoprogramación sin Intercambio

- Existen tres formas de organización de la memoria RAM, con un sistema operativo y un proceso de usuario.
- Todo el programa se carga en memoria para su ejecución

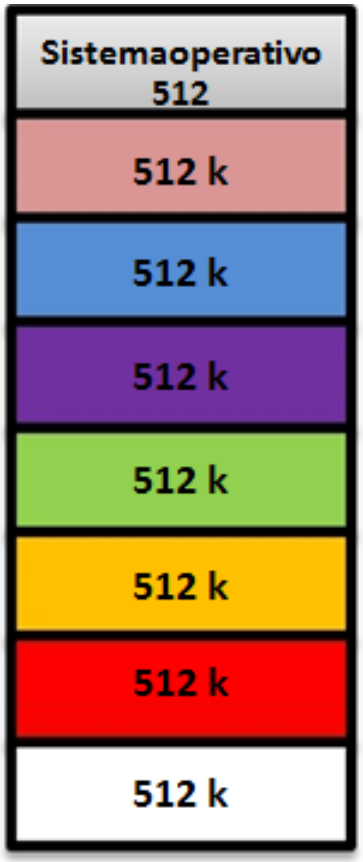


# Multiprogramación

- **Problemas**:
  - Qué proceso ocupa cuál espacio.
  - Reubicación
  - Protección de las zonas de memoria ocupadas por los procesos



# Multiprogramación: Particiones Estáticas




**Organización de la Memoria**

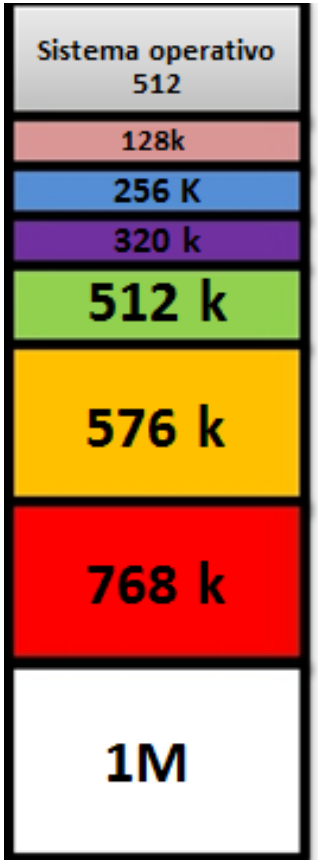
La memoria se divide en regiones con límites fijos, con dos alternativas:

1. Particiones igual tamaño o
2. Particiones de diferentes tamaños.

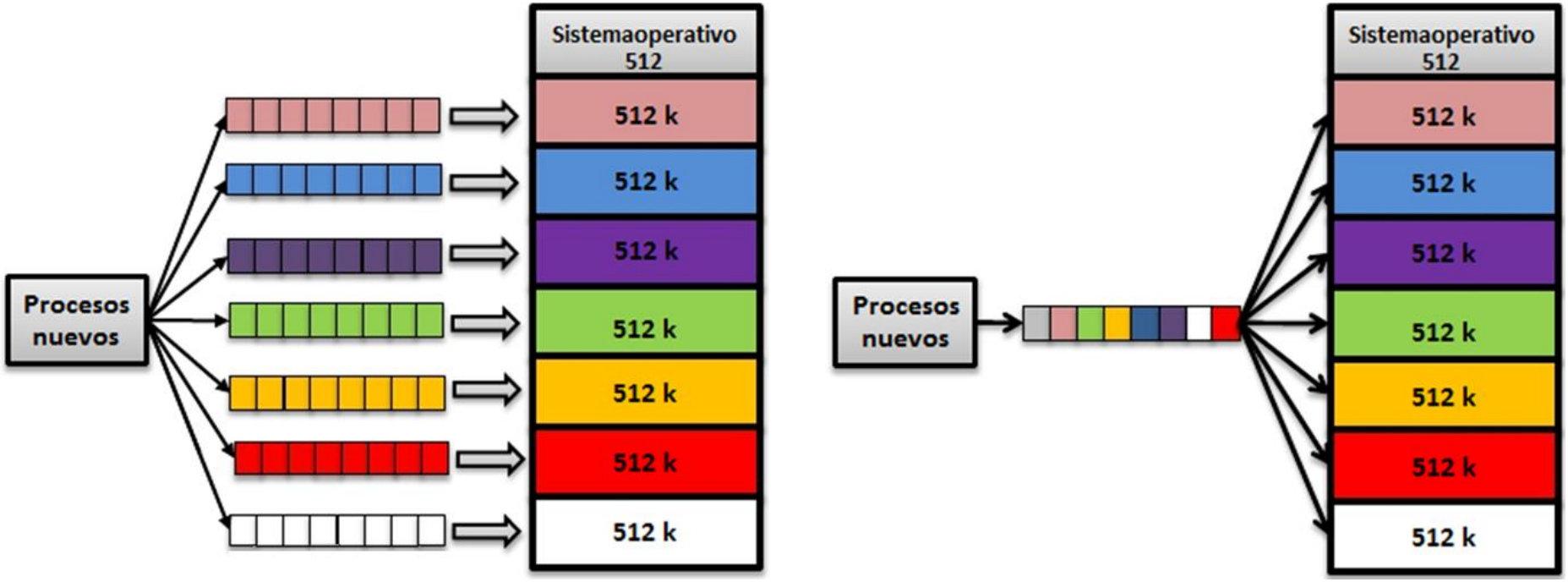
**Asignación de Procesos**

Se pueden tener dos esquemas de asignación de memoria en particiones estáticas utilizando colas de procesos:

1. Colas de planificación para cada partición
2. Una cola de planificación para la memoria (mayor eficiencia)



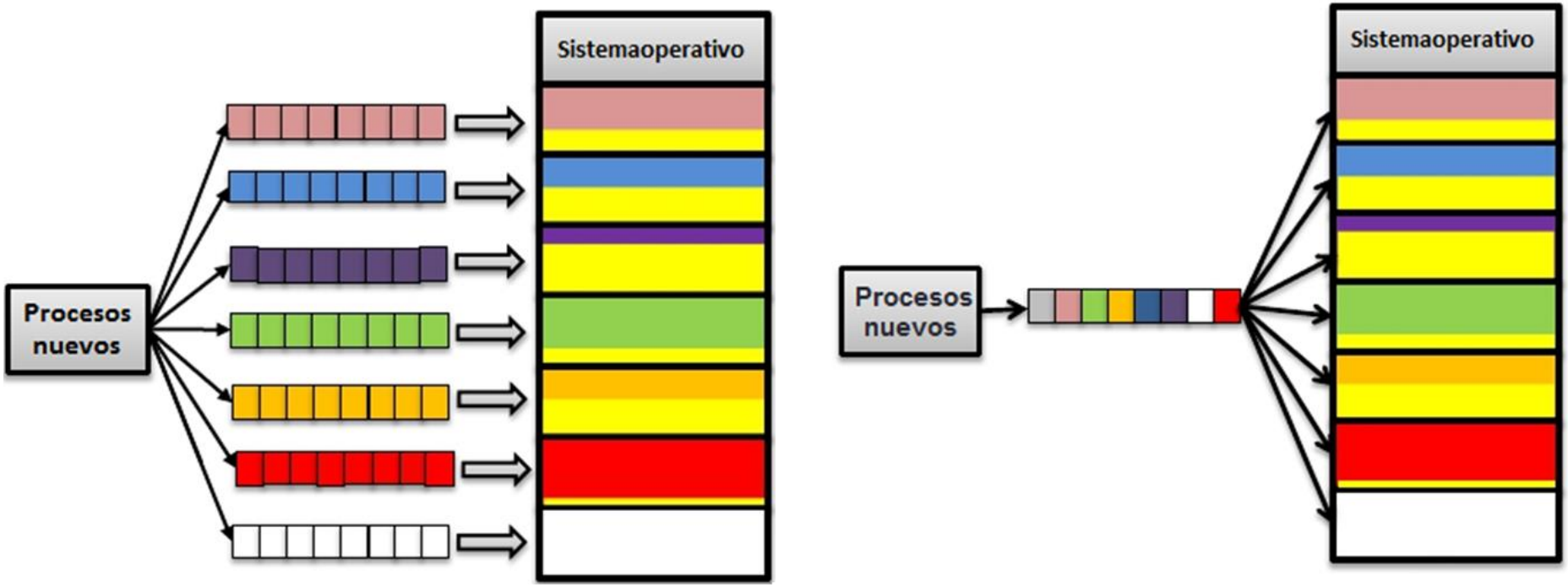
# Multiprogramación: Particiones Estáticas de igual tamaño




Los trabajos se traducían mediante compiladores y ensambladores absolutos para ejecutarse en una partición específica.

***(sin reubicación)***

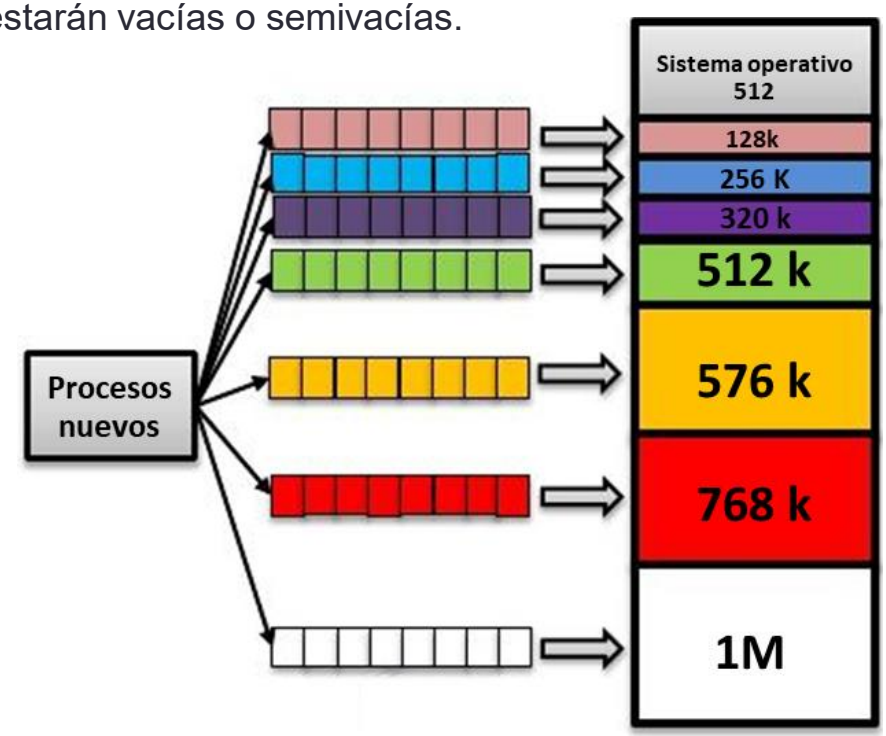
# Multiprogramación: Particiones Estáticas de igual tamaño

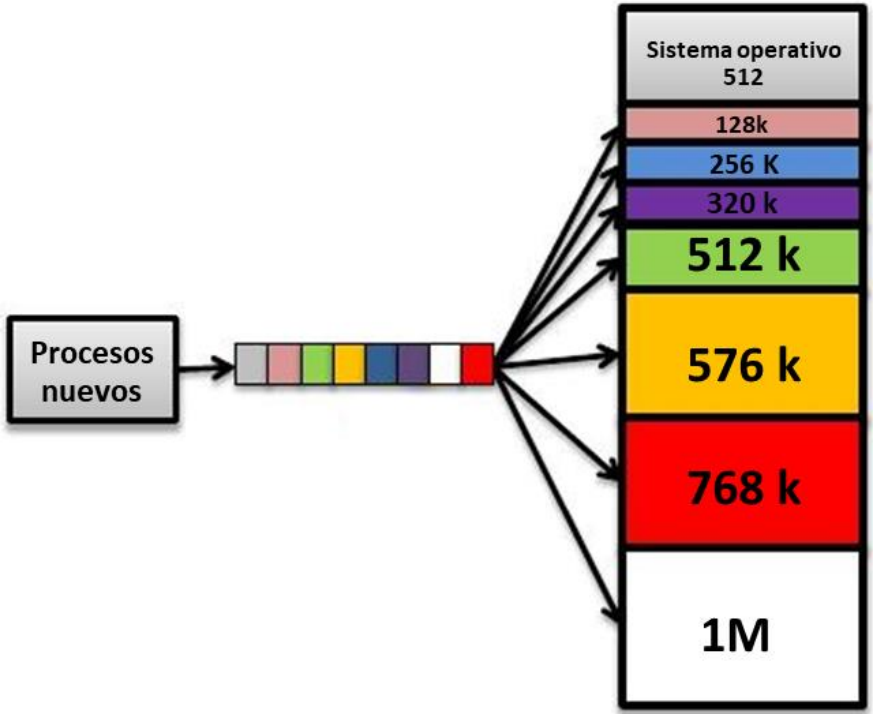



Cuando un proceso se introduce en una partición, lo más probable es que su tamaño no sea el mismo, es decir, sea algo menor) que el de la partición. Esto origina un problema de desperdicio de memoria conocido como ***fragmentación interna.***

# Multiprogramación: Particiones Estáticas de diferentes tamaños

Algunas colas estarán saturadas por procesos que tienen un tamaño similar al de la partición, mientras que otras estarán vacías o semivacías.




Se minimiza la fragmentación interna porque cada proceso es asignado a la partición más pequeña. Para ello se supone que se conoce la cantidad de memoria que ocupa un proceso, lo cual NO siempre es cierto

# Multiprogramación: Particiones Fijas

¿Cómo se lleva el control de la memoria que se encuentra disponible y aquella que está ocupada?

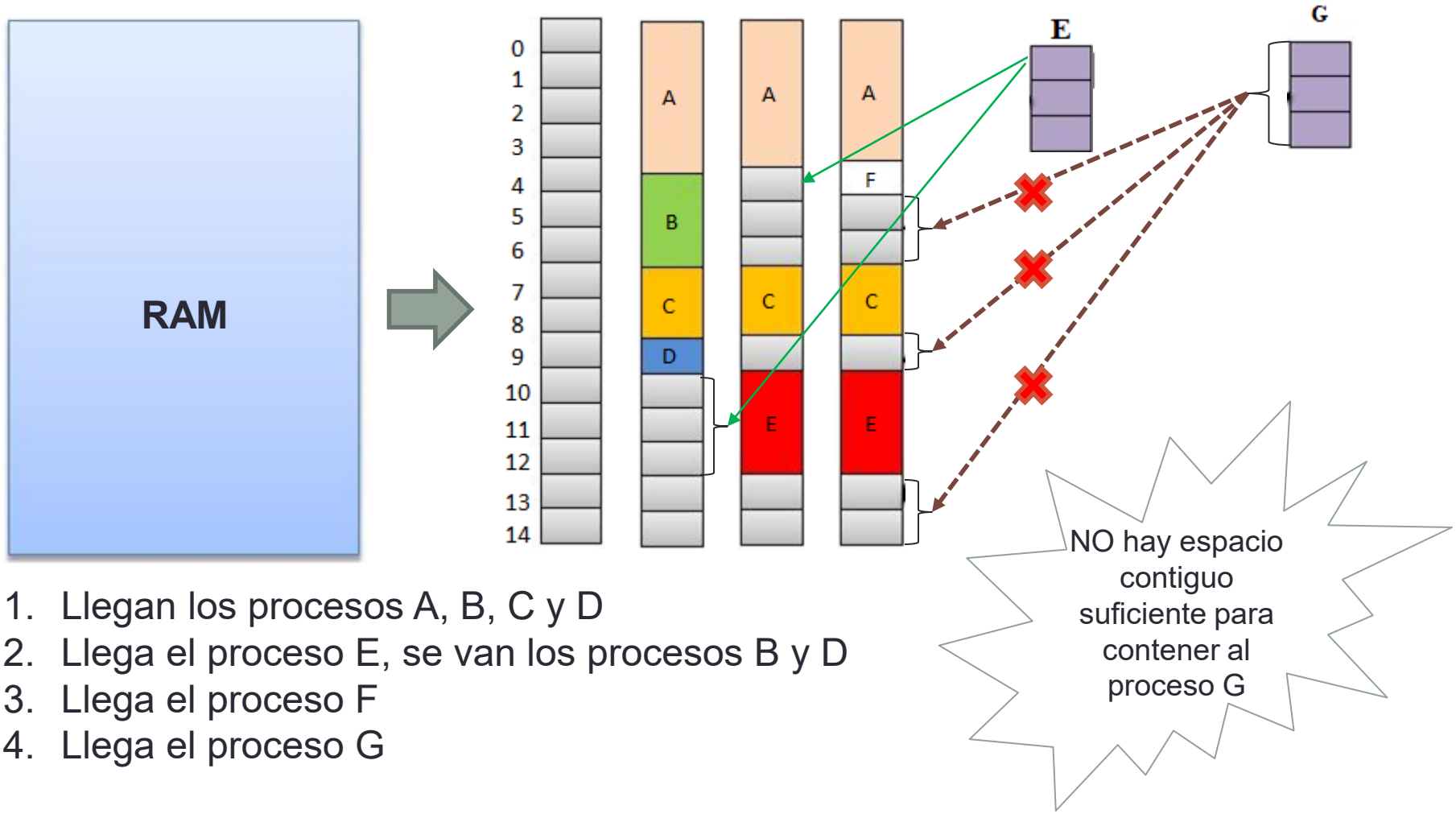
Es suficiente con guardar en una tabla la información sobre la partición que está libre o que está ocupada y qué proceso se encuentra en ella, así como las direcciones de comienzo y fin de la partición.

| No. de partición | Libre/ ocupada | Id del Proceso | Inicio de la partición | Fin de la partición |
|---|---|---|---|---|
| | | | | |
| | | | | |

*Tabla de administración de la memoria*

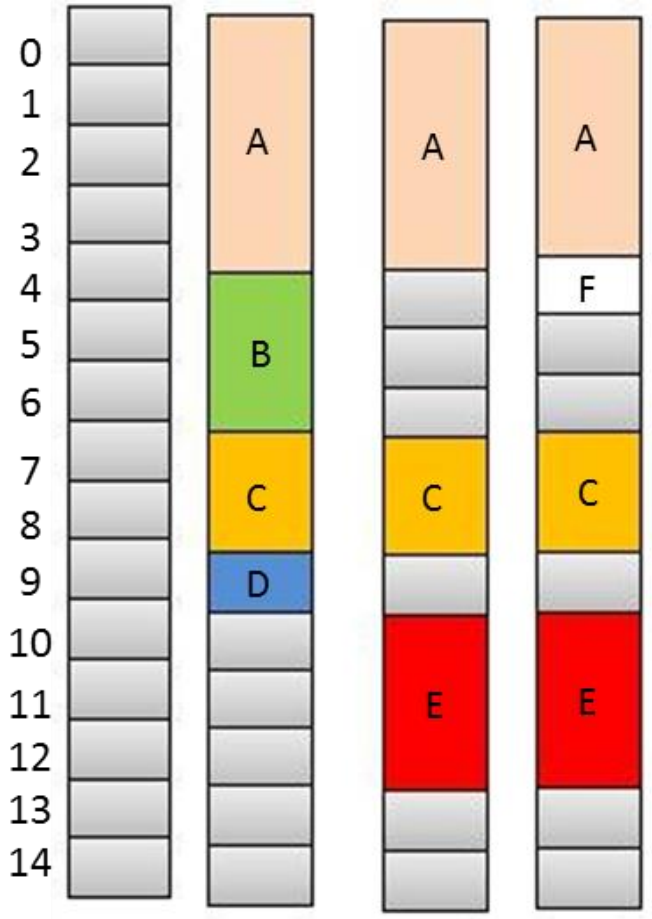
# INTERCAMBIO

# Multiprogramación: Particiones Variables



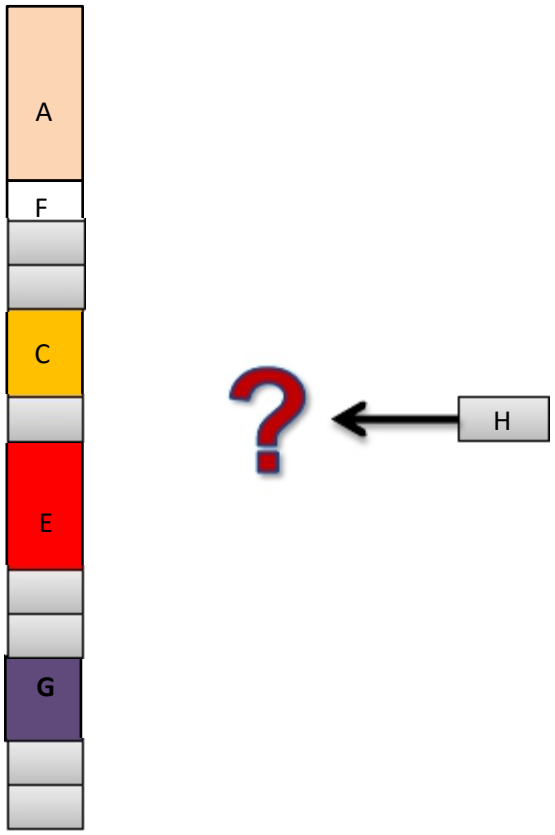

1. Llegan los procesos A, B, C y D
2. Llega el proceso E, se van los procesos B y D
3. Llega el proceso F
4. Llega el proceso G

# Multiprogramación: Particiones Variables




- **Fragmentación externa:** huecos de memoria que fueron asignados a procesos que ya NO están en el sistema.
- **Compactación:** recuperación de los huecos de memoria sin utilizar

# Multiprogramación: Particiones Variables.




**Estrategias de Colocación**

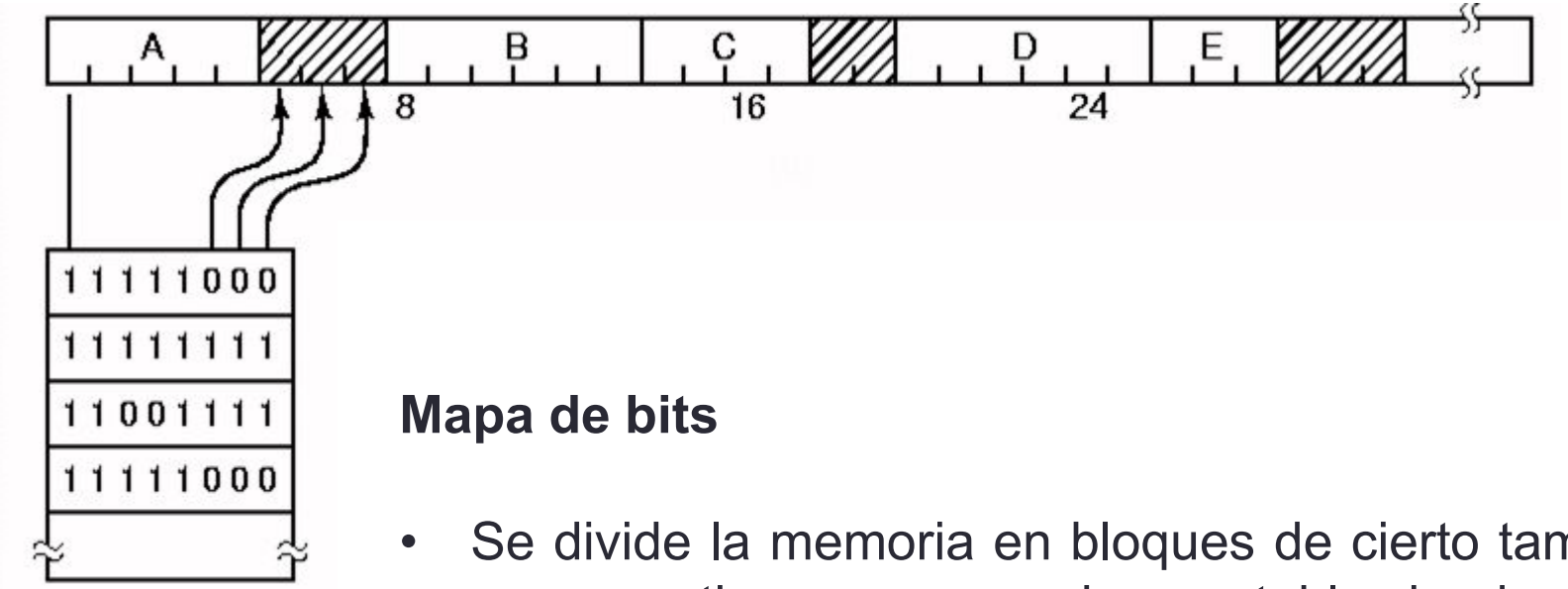
- **Primer ajuste:** Primera partición de memoria que se encuentre libre.
- **Mejor ajuste:** Busca y asigna la partición de memoria que mejor se ajuste al tamaño del proceso tratando de dejar libre la menor cantidad de memoria
- **Peor ajuste:** Se busca y asigna la partición de mayor tamaño para dejar libre la mayor cantidad de memoria para que pueda colocarse otro proceso en ella.
- **Siguiente ajuste:** La primera partición libre, desde la última asignación realizada

# Multiprogramación: Particiones Variables

¿Cómo se lleva el control de la memoria que se encuentra disponible y aquella que está ocupada?

- Mapa de bits.
- Listas enlazadas.
- Sistema de los asociados (Buddy System)

# Multiprogramación: Particiones Variables




**Mapa de bits**

- Se divide la memoria en bloques de cierto tamaño y se mantiene en memoria una tabla donde cada bit se utiliza para identificar un bloque ocupado (1) o libre(0).

- Si las particiones son muy pequeñas se tendrá un mapa de bits muy grande y viceversa

# Multiprogramación: Particiones Variables

**Listasenlazadas**

La lista está compuestapor los siguientes campos:

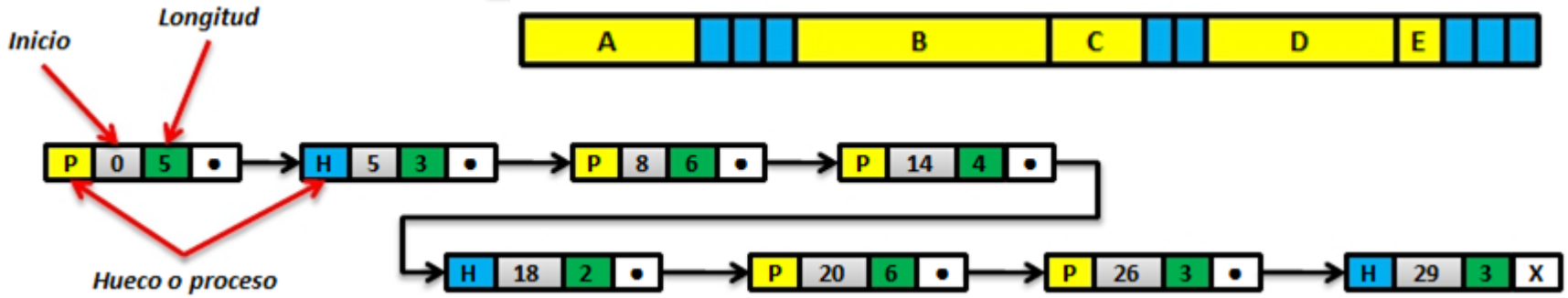
- Bit que representa un proceso o una partición disponible.
- Numero de bloque de inicio.
- Numero de bloques utilizados.
- Puntero al siguiente nodo en la lista.

**Ventajas:**

La actualización de la lista enlazada es directa.

**Desventaja:**

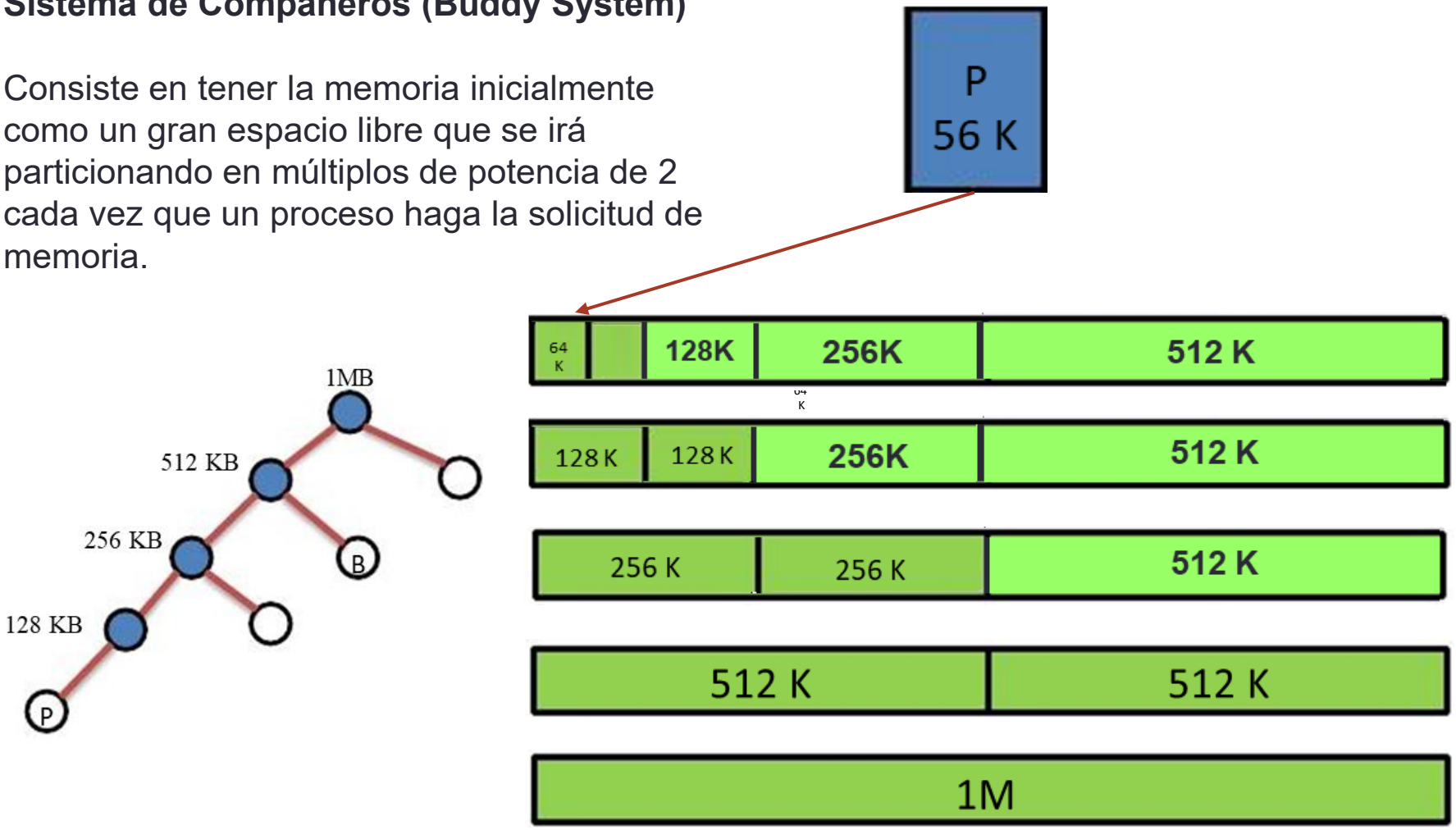
La búsqueda de un hueco que pueda satisfacer la solicitud de memoria para un proceso es lenta. La solución puede ser la creación de dos listas enlazadas, una para los procesos y otra para los espacios libres. Esto provoca que se dé mantenimiento a dos listas cuando un proceso termina y cuando se otorga memoria a otro proceso.



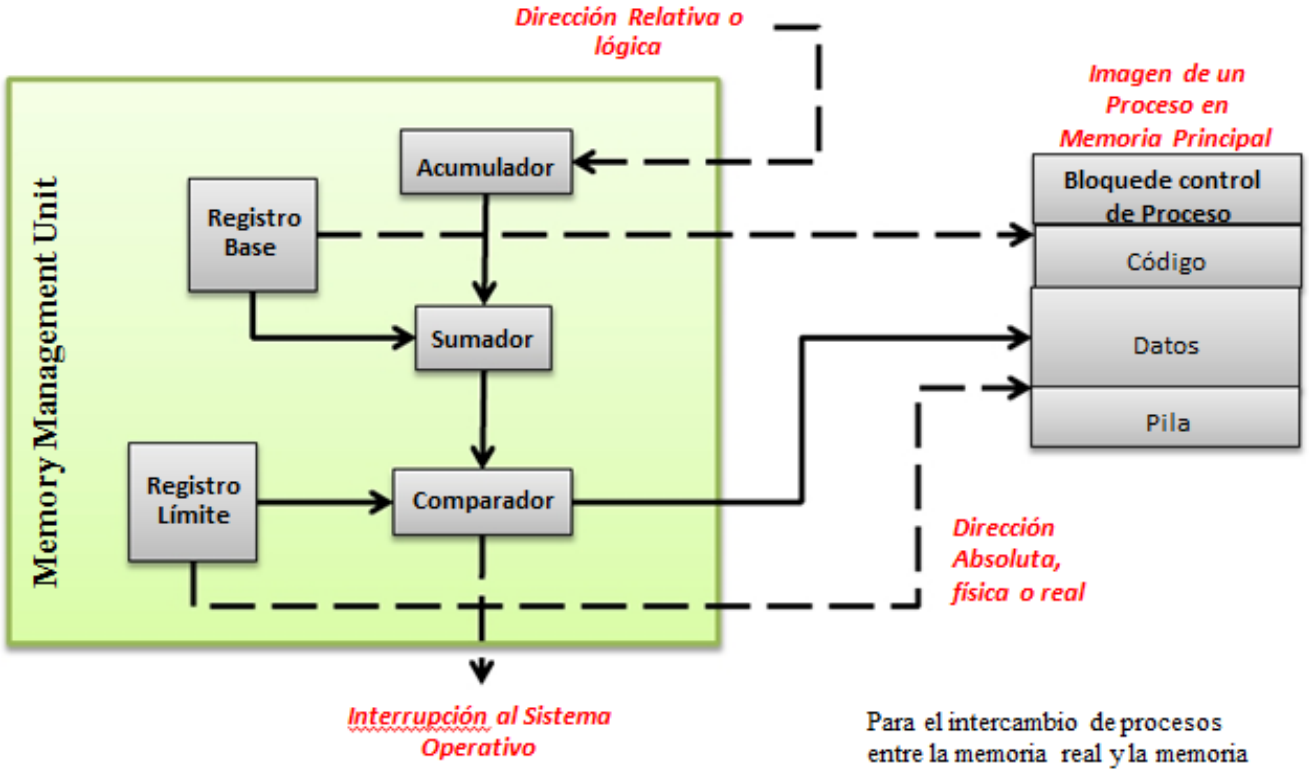
# Multiprogramación: Particiones Variables

**Sistema de Compañeros (Buddy System)**

Consiste en tener la memoria inicialmente como un gran espacio libre que se irá particionando en múltiplos de potencia de 2 cada vez que un proceso haga la solicitud de memoria.



# Soporte del Hardware para la reubicación




La Unidad Manejadora de Memoria (MMU) realiza la traducción de las direcciones relativas en direcciones físicas en la memoria principal.

Para el intercambio de procesos entre la memoria real y la memoria virtual se requiere un mecanismo que reubique los procesos sin perder las referencias que se hacen dentro de él. Los programas que emplean direcciones relativas se cargan mediante cargadores dinámicos durante la ejecución.

# Análisis de Sistemas con Intercambio

- **Regla del 50%:**
  - Al utilizar el mecanismo de **Primer Ajuste** o el de **Mejor Ajuste** puede afectar al grado de fragmentación, porque la estrategia de **Primer Ajuste** es mejor para algunos sistemas, mientras que para otros resulta más adecuada la de **Mejor Ajuste**. Otro factor diferenciador es el extremo de un bloque libre que se asigne ¿cuál es el fragmento que se deja como agujero restante, el situado en la **parte superior** o el situado en la **parte inferior**? Independientemente de qué algoritmo se utilice, la **fragmentación externa** termina siendo un problema.
  - El ***análisis estadístico*** de la estrategia de **Primer Ajuste** revela que si tenemos ***N*** bloques asignados, se perderán otros ***0,5N*** bloques por la fragmentación. Es decir un tercio de la memoria puede no ser utilizable.

# Regla del 50%. Ejemplo

**[ PROCESO 1 ] -> [ HUECO 1 ] -> [ PROCESO 2 ] -> [ PROCESO 3 ] -> [ HUECO 2 ] -> [ PROCESO 4 ]**

- Bloques asignados (N): 4 (Procesos 1, 2, 3 y 4).
- Bloques libres (Huecos): 2 (Hueco 1 y Hueco 2). Esto es el 0.5N.

- Si sumamos el total de bloques independientes en la memoria tenemos:

**Total de bloques** = Procesos + Huecos = N + 0.5N = 1.5N

- Qué fracción representan los huecos del total?, haces la división:

$$\frac{\text{Huecos}}{\text{Total de bloques}} = \frac{0.5N}{1.5N} = \frac{0.5}{1.5} = \frac{1}{3}$$

# INTERCAMBIO

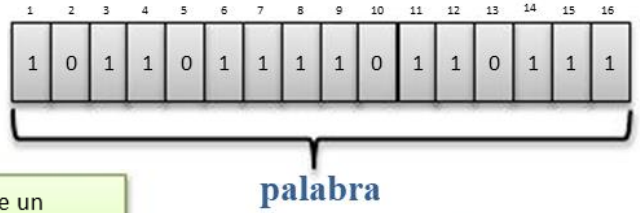
Continuación…

# Paginación Simple

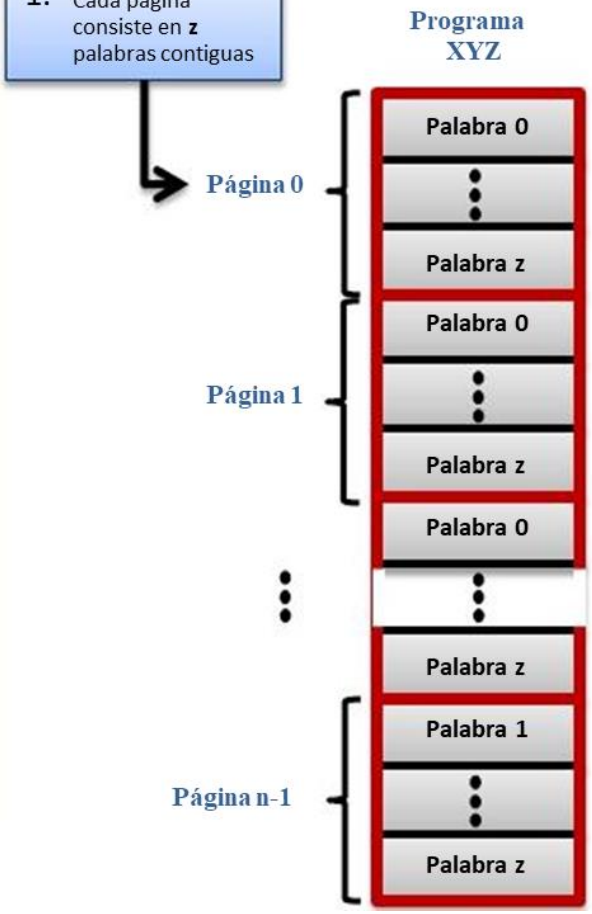
- Permite que la memoria de un proceso no sea contigua
- Hay una distinción entre direcciones lógicas y físicas
- La memoria física la dividimos en bloques de tamaño fijo: marcos
- La memoria lógica:
  - La dividimos  en bloques llamados: páginas
  - De igual tamaño que el marco
- Las páginas de un proceso se cargan en los marcos de la memoria principal que estén disponibles
  - Tenemos trozos del proceso allí donde la memoria está disponible

# Paginación Simple

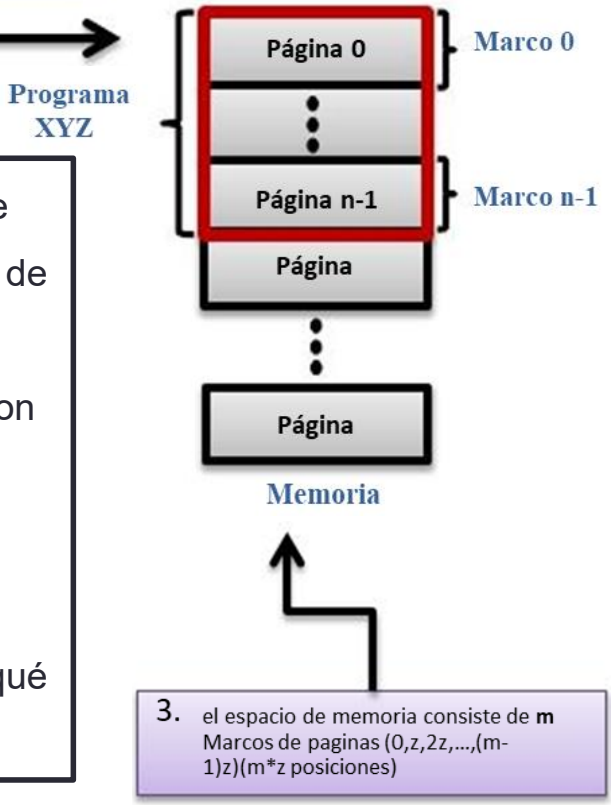
Las computadoras modernas normalmente tienen un tamaño de palabra de 16, 32 ó 64 bits.




1. Cada página consiste en **z** palabras contiguas




2. Un espacio de direcciones **N** de un programa consiste de **n** páginas (0,1,2,3...n-1)(n*z direcciones virtuales)

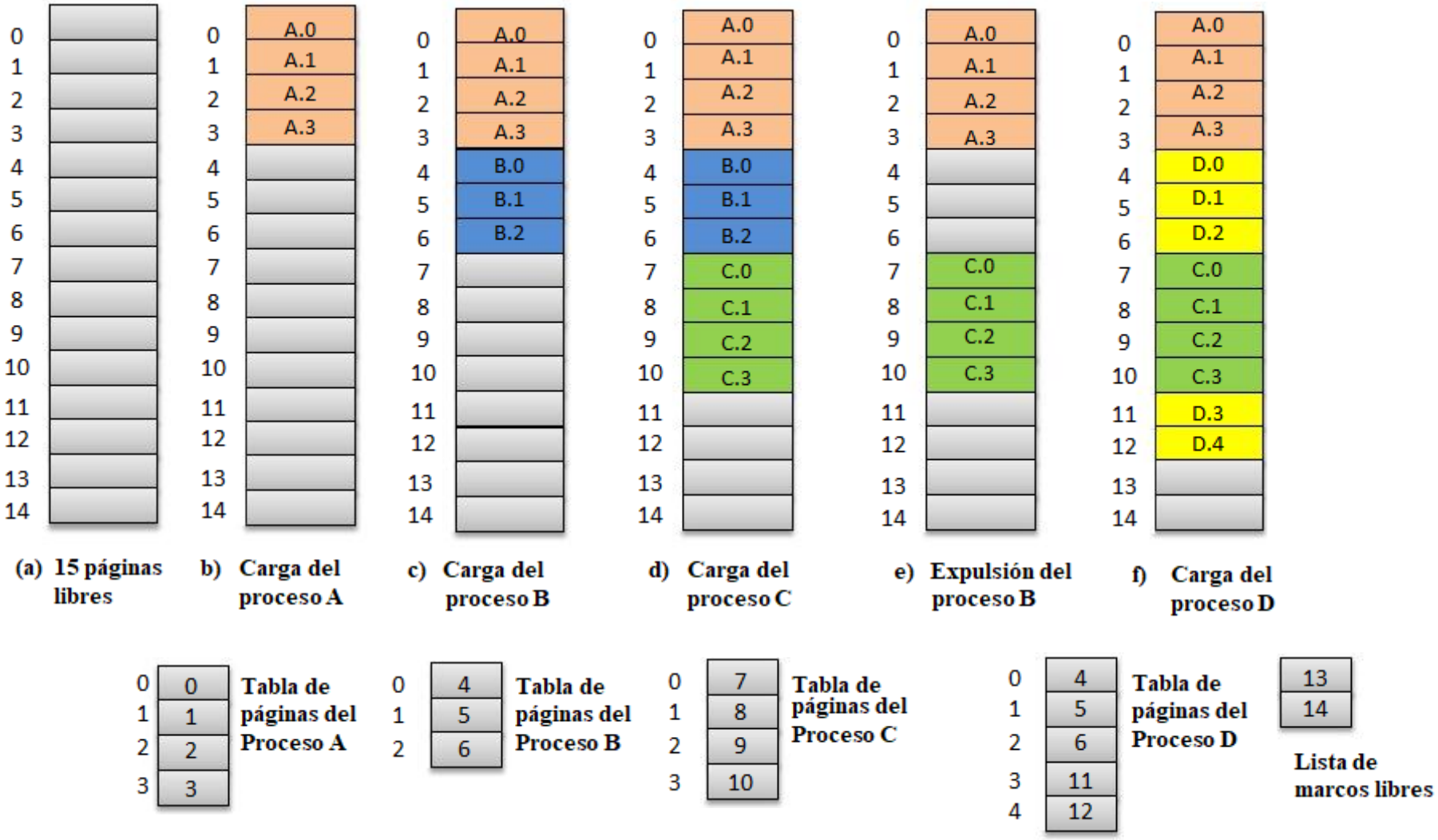



La técnica de paginación simple es similar a la partición estática de igual tamaño.

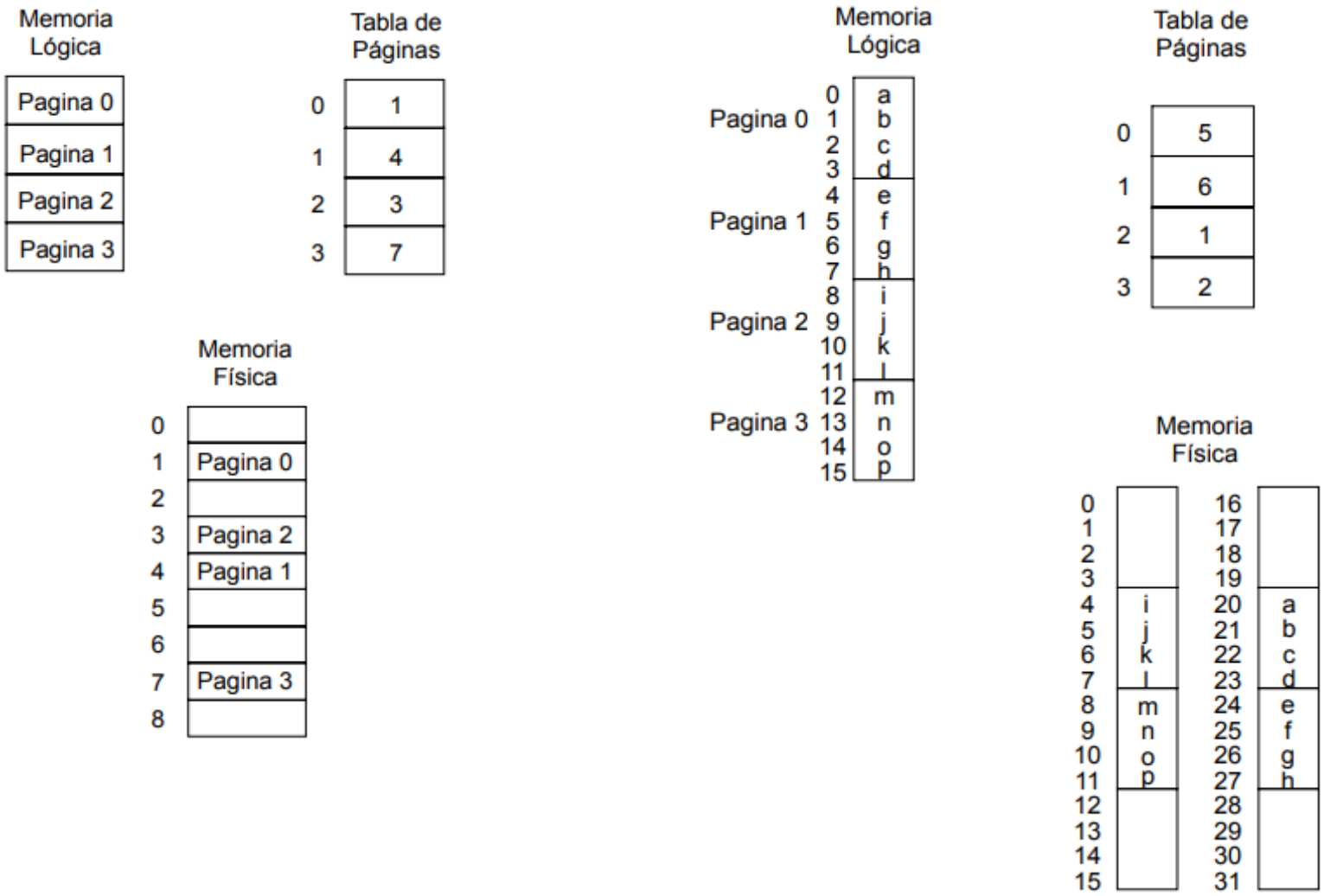
Las diferencias están en que, con Paginación, las particiones son más pequeñas, un programa puede ocupar más de una partición y estas no tienen por qué estar contiguas

3. el espacio de memoria consiste de **m** Marcos de paginas (0,z,2z,...,(m-1)z)(m*z posiciones)

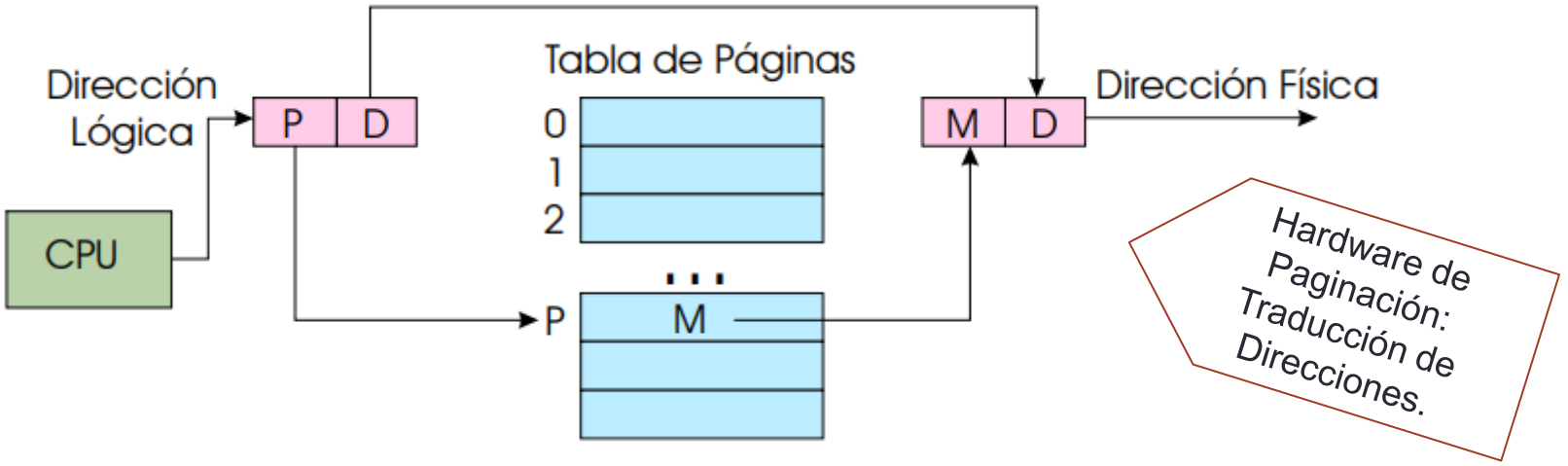
# Paginación Simple



# Paginación Simple. Ejemplo



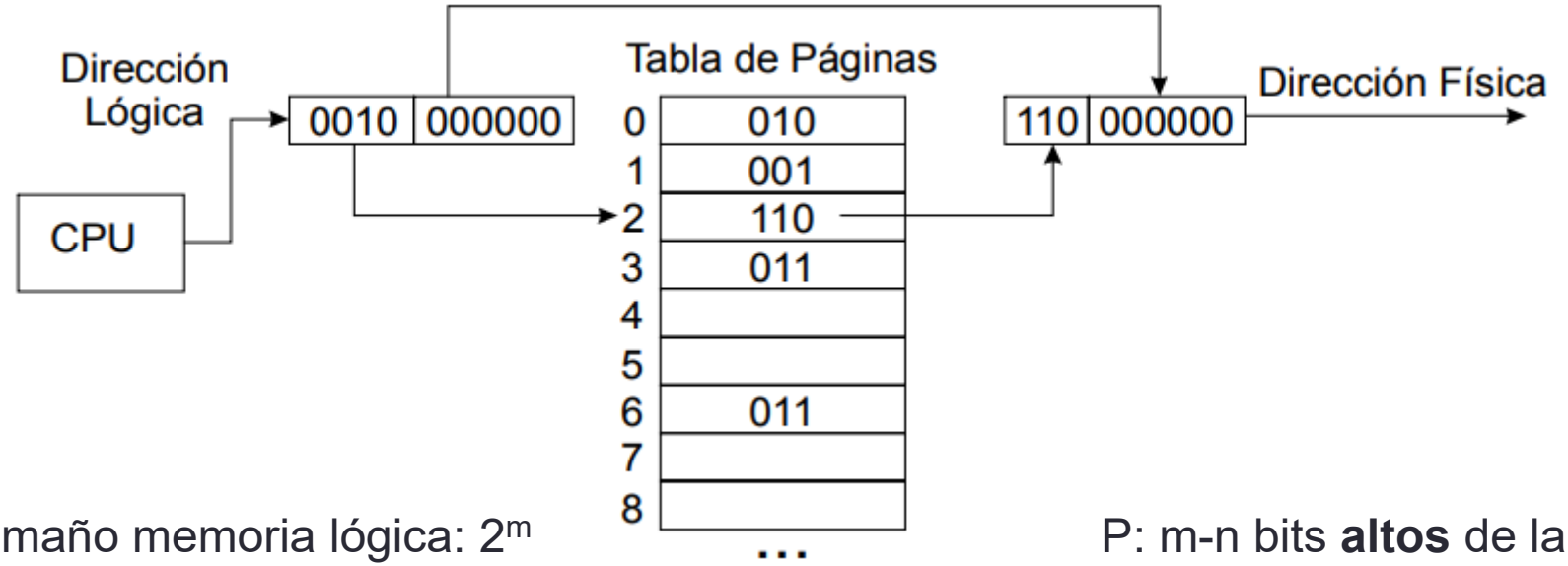
# Paginación Simple




- La **dirección lógica** generada consta de dos partes
  - Número de Página (P)
  - Desplazamiento dentro de la página (D)
- La **tabla de páginas**: (contiene la dirección base en memoria física)
  - Permite establecer una correspondencia entre el número de página y un número de marco de memoria física.
- La **dirección física** es el número de marco y el desplazamiento.

# Paginación Simple

- Tamaño de páginas y marcos definidos por Hardware
- Normalmente se escoge un tamaño de página potencia de 2 (la traducción de DL a DF es más sencilla)

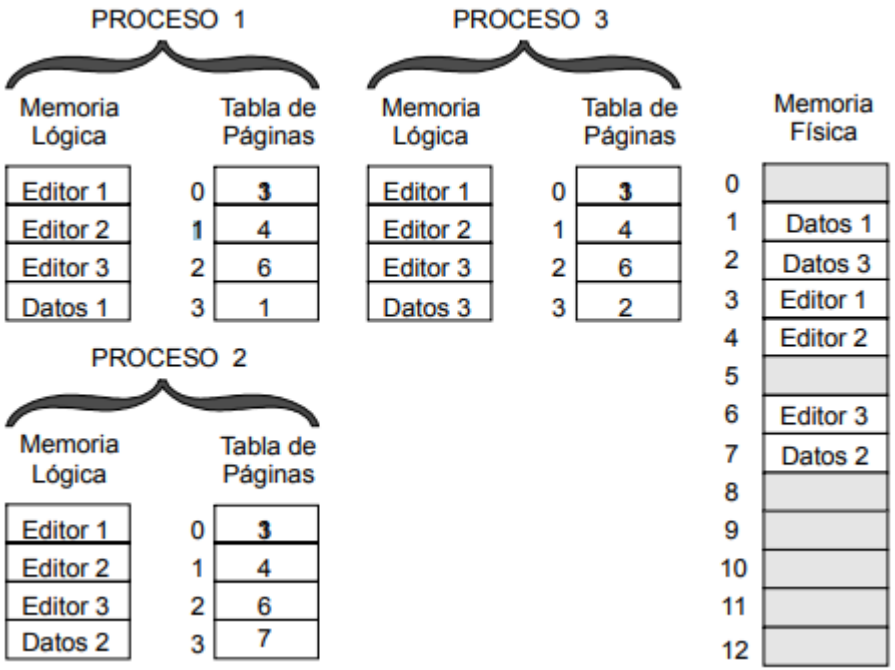



Tamaño memoria lógica: $2^m$
Tamaño página: $2^n$ (bytes o palabras)
P: índice en Tabla de Páginas
D: desplazamiento

P: m-n bits **altos** de la DL
D: n bits **bajos** de la DL

# Paginación Simple. Ventaja

- **Páginas Compartidas:**
- La paginación permite compartir código común entre varios procesos:
  - Solo si el código es reentrante (no se modifica durante la ejecución)
  - El área de datos de los procesos sería diferente
  - Ejemplo: varios procesos ejecutan el mismo editor de texto



# Paginación Simple. Protección

- En la tabla de páginas pueden encontrarse unos **bits de protección** asociados a cada marco.
  - Indican si la página es de sólo lectura o lectura/escritura
- Cuando se consulta el número de marco, se consultan además los **bits de protección**
- Se debe controlar que el número de página no supere el total de páginas usadas por el proceso (sería una dirección incorrecta)

# Paginación Simple. Ventajas

- La paginación no es visible al usuario del SO
- Se elimina la fragmentación externa
- La fragmentación interna solo se produce en la última página.
- Es fácil permitir que los procesos compartan memoria.
- Se pueden proteger las páginas (bits de protección)
- Si las páginas son pequeñas:
  - Se reduce la fragmentación interna
  - Se aumenta el tamaño de la tabla de páginas.

# Segmentación Simple

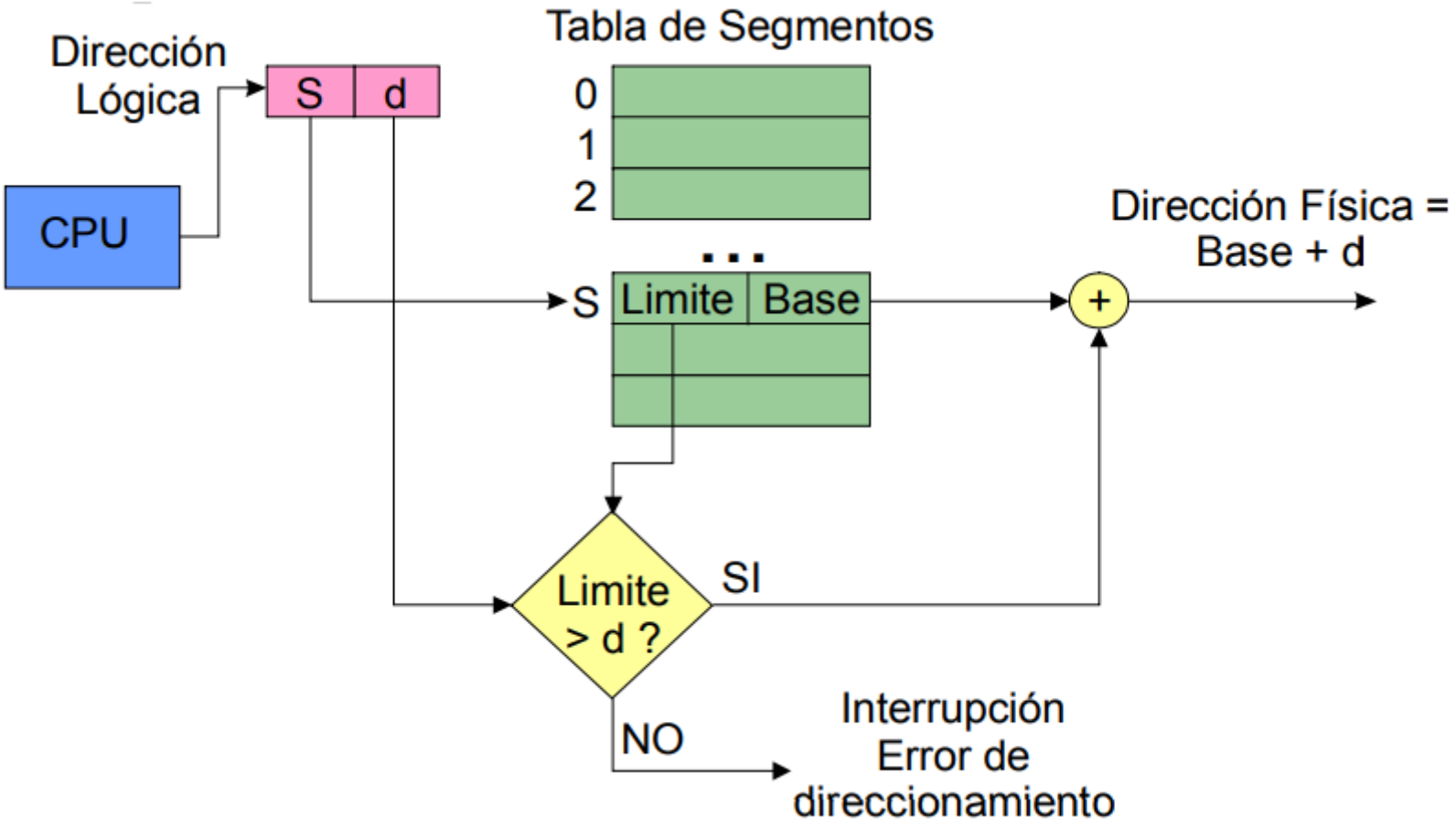
- El espacio de direcciones lógicas se compone de un conjunto de segmentos: Cada uno tiene el nombre y una longitud.
- Para el usuario las direcciones especifican el nombre del segmento y el desplazamiento dentro de él.
- El nombre del segmento se numera

  **<numero de segmento, desplazamiento>**
- El procesador Intel 8086 usa segmentación, los programas se separan en:
  - Segmento de Código
  - Segmento de Datos
  - Segmento de Pila
- Hay una división lógica del proceso en diferentes segmentos

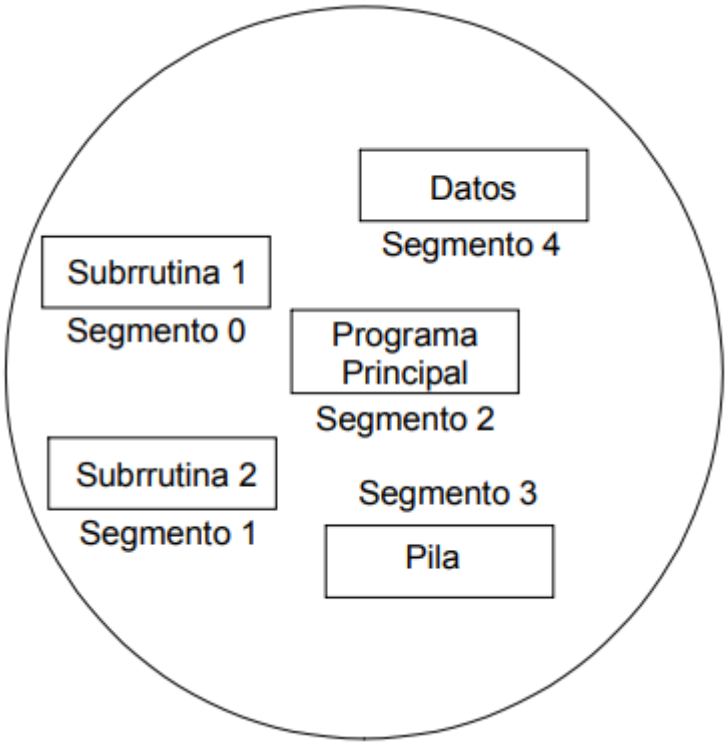
# Segmentación Simple. Hardware

- Tabla de Segmentos:
  - Establece la correspondencia entre direcciones físicas y lógicas
  - Se busca en la tabla de acuerdo con el número de segmento.
  - Cada entrada posee 2 registros:
    - Base (DF inicial del segmento en memoria)
    - Límite de Segmento (Longitud del segmento)
  - Se compara límite del segmento con desplazamiento
  - Si desplazamiento es válido, se suma a la dirección el registro base

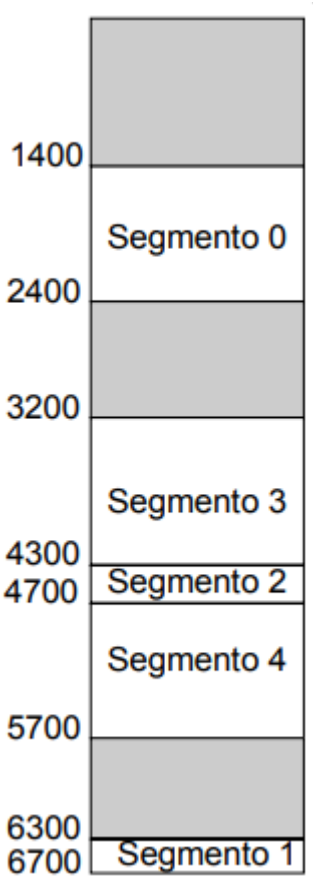
# Segmentación Simple. Hardware



# Segmentación Simple. Ejemplo




| | Limite | Base |
|---|---|---|
| 0 | 1000 | 1400 |
| 1 | 400 | 6300 |
| 2 | 400 | 4300 |
| 3 | 1100 | 3200 |
| 4 | 1000 | 4700 |



# Segmentación Simple

- Implementación Hardware de la Tabla de Segmento
  - Se puede ubicar en registros rápidos o memoria (como paginación)
  - Si está en memoria:
    - Un registro base STBR (segment table base register) indica inicio de la tabla de segmentos en memoria
    - Un registro límite indica longitud de la tabla de segmentos
  - Protección
    - Bits de protección: Segmento de solo lectura o lectura/escritura.
    - Se consultan antes de acceder al segmento
  - Compartición de código
    - Puede realizarse a nivel de segmento (código o datos)
    - Cada proceso tendrá una tabla de segmento
    - Compartir un segmento significa que una entrada de la tabla de segmentos coincide en varios procesos (igual posición física)

# Segmentación Simple. Compartición

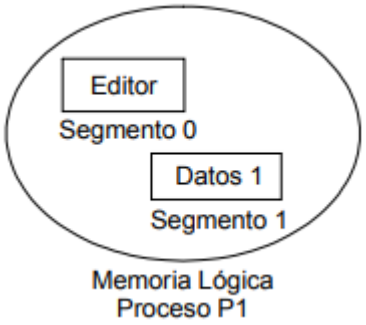



Tabla de Segmentos
Proceso P1

| | Limite | Base |
|---|---|---|
| 0 | 25286 | 43062 |
| 1 | 4425 | 68348 |

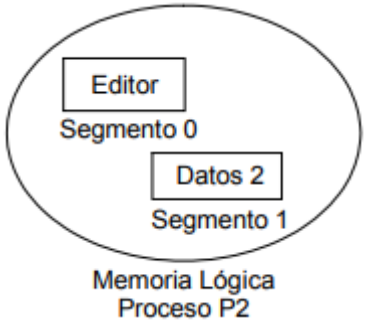
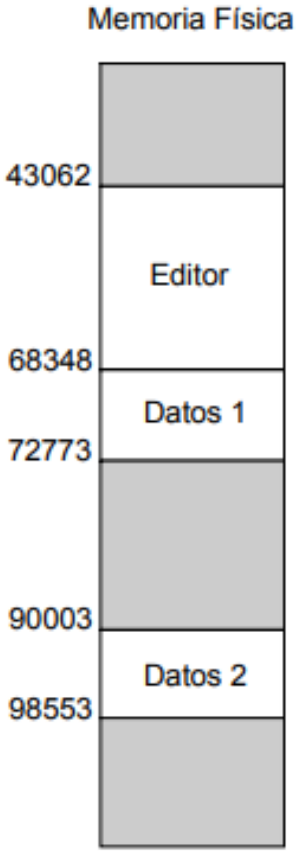



Tabla de Segmentos
Proceso P2

| | Limite | Base |
|---|---|---|
| 0 | 25286 | 43062 |
| 1 | 8550 | 90003 |



# Segmentación Simple. Fragmentación

- Los segmentos son de tamaño variable:
  - Puede haber fragmentación externa
  - Bloques de memoria demasiado pequeños para contener un segmento.
- Solución: Se puede compactar la memoria (segmentación usa reubicación dinámica)
- Problema de fragmentación:
  - Cada proceso un segmento: particiones variables
  - Cada palabra (byte) un segmento:
    - No habría fragmentación externa
    - Se necesita de una tabla de segmentos del tamaño de la memoria

# Segmentación Simple. Ventajas e Inconvenientes

- Facilita la compartición: se comparten unidades lógicas o segmentos (datos, texto).
- Se pueden proteger los segmentos (bits de protección)
- Facilita la ampliación de las estructuras de datos (solo se ampliaría el segmento correspondiente)
- Visión de proceso tal y como lo ve el usuario
- Fragmentación externa: Se pueden aplicar las mismas soluciones vistas en Particiones Variables.

# Segmentación Paginada Simple

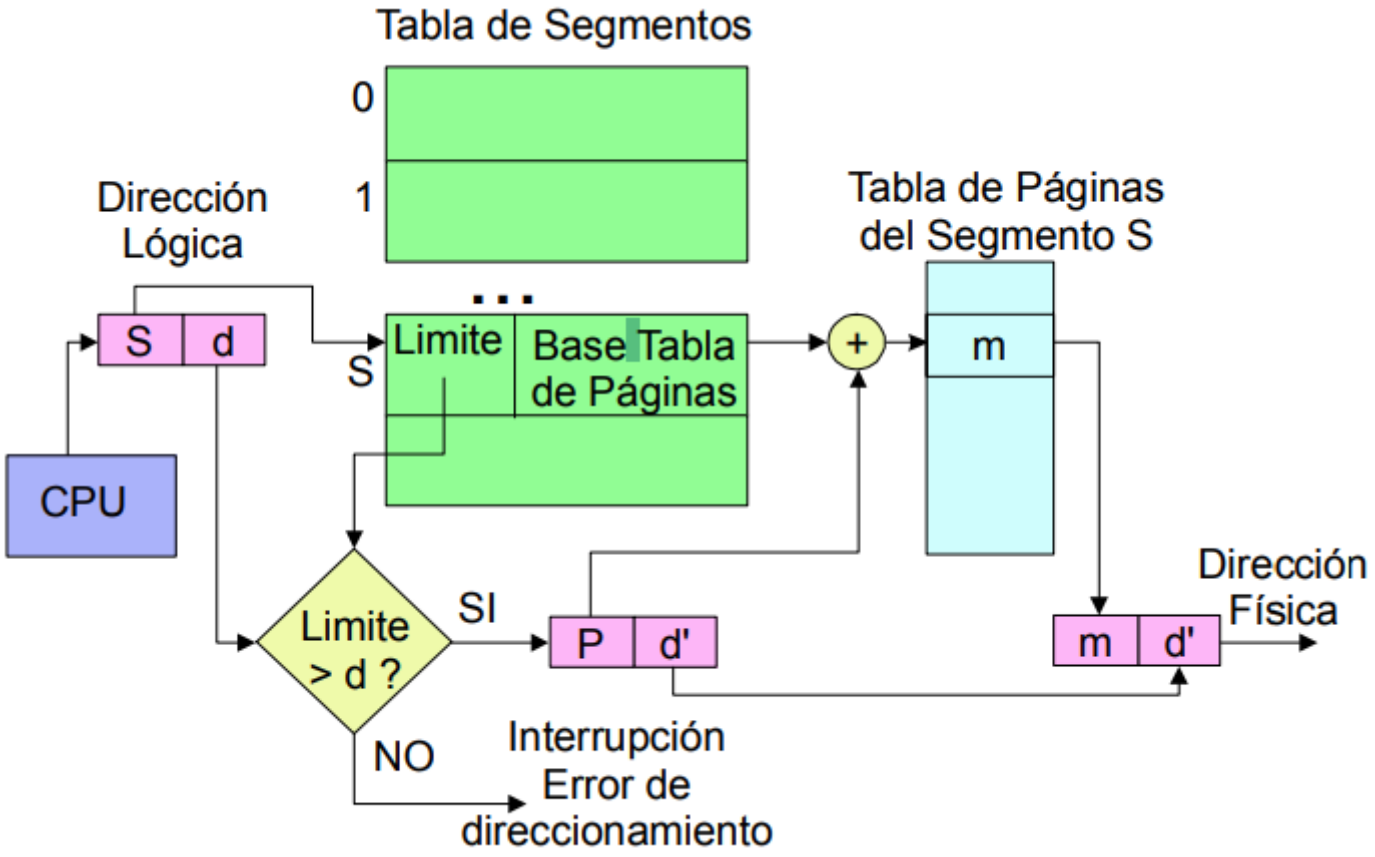
- La Memoria lógica está dividida en bloques llamados segmentos que contienen las regiones de un proceso
- Dirección lógica:

  <NroDeSegmento,Desplazamiento> = <S,d>
- Los segmentos están divididos en páginas de igual tamaño que los marcos
- Las páginas de un proceso se cargan en marcos de la memoria principal
- Cada segmento tiene asociada una tabla de páginas
- Se usa un registro límite y base de la tabla de páginas para cada segmento

# Segmentación Paginada. Esquema de Traducción de direcciones

- Dirección lógica: <S,d>
  - S: entrada a la Tabla de segmento, la cual contiene
    - el límite del segmento
    - La dirección base de una tabla de páginas
      - Habrá una tabla de pagina por cada segmento
  - d: incluye
    - P: número de pagina
    - d': desplazamiento dentro de la página

# Segmentación Paginada. Esquema de Traducción de direcciones



# Resumen de la Unidad 8

- **Requerimientos:** Reubicación, Protección, Compartición, Organización Física y Lógica.
- **Sin intercambio:** Monoprogramación → Multiprogramación con particiones estáticas (fijas).
- **Con intercambio:** Particiones variables, fragmentación externa, estrategias de colocación.
- **Control de memoria:** Mapa de bits, Listas enlazadas, Sistema de Compañeros (Buddy System).
- **Paginación Simple:** Elimina fragmentación externa. Memoria no contigua. Tabla de páginas.
- **Segmentación Simple:** División lógica del proceso. Tabla de segmentos (Base + Límite).
- **Segmentación Paginada:** Combina ambos esquemas. Elimina fragmentación externa de la segmentación.
- **Concepto clave:** Cada técnica resuelve limitaciones de la anterior, a costa de mayor complejidad en hardware y administración.