# INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 3: MEMORÍA

***Memoría***:

- La organización y la administración de la "memoria principal" resultan ser uno de los factores MÁS importantes en el diseño de un sistema operativo.
- Todo programa y datos de estos deben estar en el almacenamiento principal (RAM) para:
    - Poder ejecutar dichos programas.
    - Referenciarlos de forma directa.
- El kernel se encarga de administrar los espacios de direcciones de la memoria RAM.
    - Administración eficiente → implica mejor funcionamiento.

**Tarea del sistema operativo:**

- Debe llevar el registro de las partes de memoria que están en uso y aquellas las cuales no están en uso.
- Debe asignar espacio en memoria principal (RAM) a los procesos cuando estos necesitan el espacio.
- Debe liberar espacio de memoria asignada a procesos que han terminado.
- Se espera que el sistema operativo haga un uso eficiente de la memoría con la finalidad de alojar la mayor cantidad de procesos posible.
- Adicionalmente:
    - El SO debe lograr que el programador se **abstraiga** de la alocación de los programas.
    - Debe brindar **seguridad** entre procesos para que no haya accesos indebidos (un proceso violando el espacio privado de otro).
    - Brindar la posibilidad de **acceso compartido** a determinadas partes de la memoria (librerías, código en común, etc).
    - Garantizar la **performance**.

---

---

***Administración de memoría***

- Hay una división lógica de la memoría física para alojar múltiples procesos:
    - Se garantiza protección.
    - Depende del mecanismo de administración provisto por el Hardware.
    - ¿Cómo va a parar lo lógico a lo físico? → depende del hardware y de un conjunto de estructuras utilizadas por el mismo.

- Asignación eficiente:
    - Contener el mayor número de procesos para garantizar el mayor uso de la CPU por los mismos.
    - Asignar de manera eficiente la RAM.

**Requisitos para la administración de la RAM:**

➔ Reubicación
- El programador no debe ocuparse de saber donde será colocado el programa en la RAM. Es decir, el programa es indiferente a la ubicación física.
- Mientras un proceso es ejecutado, puede ser sacado y traído a la memoria (swap), y posiblemente ser colocado en diferentes direcciones → el proceso debe ser independiente de la ubicación.
- Las referencias a la memoría se deben "traducir" según la ubicación actual del proceso.

➔ Protección
- Los procesos NO deben referenciar o acceder a direcciones de memoria de otros procesos. (SALVO si tienen permiso para ello).
- El chequeo se debe realizar durante la ejecución.
    - No es posible anticipar todas las referencias a memoria que un proceso puede realizar (es dinámico), por eso se realiza en run-time.

➔ Compartición
- Consiste en permitir que varios procesos accedan a la misma porción de memoría.
    - Ej: rutinas comunes, librerías, espacios explícitamente compartidos para los procesos, etc.
- Permite un mejor aprovechamiento de la memoria RAM, evitando copias repetidas de instrucciones que son innecesarias.

---

---

**Abstracción – espacio de direcciones**

- El espacio de direcciones es el rango de direcciones a memoria posibles que un proceso puede utilizar para direccionar sus instrucciones y datos.
- El rango de direcciones lógicas esta definida por la arquitectura.
    - Procesador de 32 bits: 0.. $2^{32}$ -1
    - Procesador de 64 bits: 0.. $2^{64}$ -1
- El direccionamiento es independiente de la ubicación "real" del proceso en la memoria RAM, **la dirección 100 en el rango puede no ser igual a la dirección 100 en la física**.

# INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 3: MEMORÍA

## Direcciones

- Lógicas (la que usan los procesos)
  - Referencia a una localidad de memoria independiente de la asignación actual de los datos en la memoria.
  - Representa una dirección en el "Espacio de direcciones del proceso"
- Físicas
  - Referencia a una localidad en la memoria física (RAM).
  - Dirección absoluta.
- **Cuando uso direcciones lógicas, necesito algún tipo de conversión a direcciones físicas para hallar el dato.**

## Conversión de direcciones

➔ Cada dirección lógica que se genera se debe convertir a una dirección física.

➔ Una forma simple de hacer esto es utilizando registros auxiliares.
  - Registro base:
    - Dirección de comienzo del espacio de direcciones del proceso en la RAM.
  - Registro límite:
    - Dirección final del proceso o tamaño de su espacio de direcciones.
  - A partir de los valores de esos dos registros, podemos tomar la dirección lógica como un desplazamiento desde la base, y controlar que dicha suma (registro base + dirección lógica=dirección física) no exceda el registro límite. Si la dirección supera el límite → excepción.

➔ Los valores de ambos registros se fijan cuando el espacio de direcciones del proceso se carga en la memoria.

---

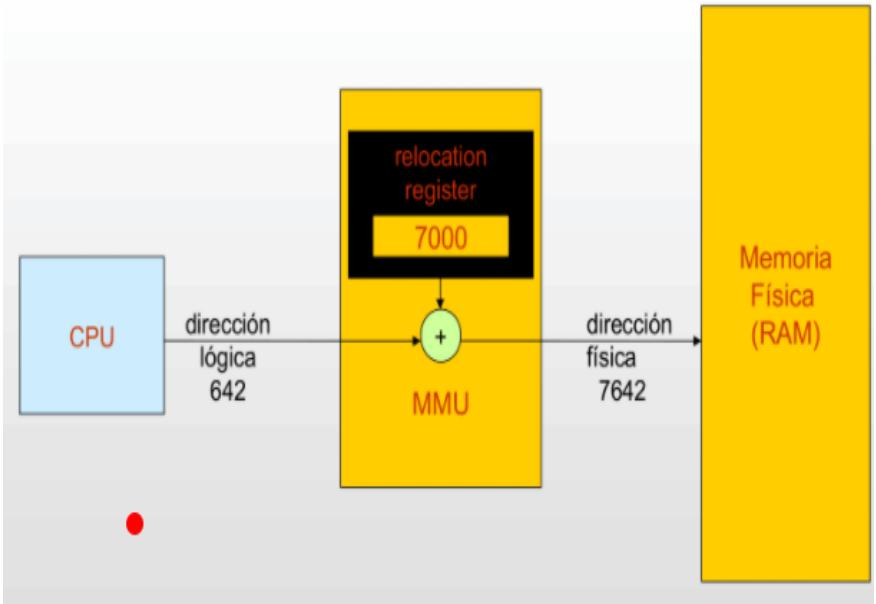
---

## Direcciones lógicas vs Físicas

- Si la CPU trabaja con direcciones lógicas, para acceder a la memoria principal, se deben transformar en direcciones físicas.
  - Resolución de direcciones (address-binding): transformar la dirección lógica en la dirección física que corresponde.
- Resolución en momento de compilación o en tiempo de carga del proceso (no usado en la actualidad):

- Direcciones lógicas y físicas son idénticas.
- Para reubicar un proceso es necesario recompilarlo o recargarlo.

- Resolución en tiempo de ejecución
  - Direcciones lógicas y físicas son diferentes.
  - Las lógicas son llamadas direcciones virtuales.
  - La reubicación se puede realizar fácilmente.
  - La resolución debe ser rápida en velocidad, a la par del CPU.
  - El mapeo entre “virtuales” y “físicas” es realizado por hardware.
    - Memory Management Unit (MMU).
      - Es un dispositivo de hardware que mapea direcciones virtuales a físicas (convierte.)
        - Es parte del procesador. (Hardware)
        - Re-programar el MMU es una operación privilegiada.
          - SOLO SE REALIZA EN MODO KERNEL.
      - El valor en el “registro de realocación” es sumado a cada dirección generada por el proceso de usuario al momento de acceder a la memoria.
        - Los procesos nunca usan direcciones físicas.



# INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 3: MEMORÍA

***Mecanismos de asignación de memoría:***

- ➔ Particiones fijas: el primer esquema que se implementó.
  - La memoría se divide en particiones o regiones de tamaño fijo (todos del mismo tamaño o no).
  - Cada partición aloja un proceso.
  - Cada proceso se coloca de acuerdo a algún criterio (first fit, best fit, worst fit, next fit) en alguna partición (son algoritmos*).*
  - Al ser fijo podría ocurrir fragmentación interna (espacio desperdiciado).
- ➔ Particiones dinámicas: es la evolución del esquema anterior.
  - Las particiones varían en tamaño y en número.
  - Cada una aloja un proceso.
  - Cada partición es generada de forma dinámica con el tamaño justo que necesita el proceso.
  - Generación fragmentación externa.
  - Se recomienda peor ajuste, ya que la que me va a quedar libre es probable que sea la mas grande.
- ➔ Fragmentación (problemas de las particiones)
  - Producida cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua.
  - Fragmentación interna:
    - Producido en particiones fijas.
    - Porción de partición que queda sin utilizar.
  - Fragmentación externa:
    - Producido en particiones dinámicas.
    - Son huecos que van quedando en la memoria a medida que los procesos terminan.
    - Estos huecos si bien son memoria libre, al no estar contiguos entre sí un proceso no puede ser alocado.
    - El problema se soluciona compactando → pero es costosa.

***Problemas del esquema registro base y límite***

- El esquema de registro base + registro límite presenta problemas:
  - Necesidad de almacenar el espacio de direcciones de forma **continua** en memoria física.
  - Los primeros SO definían particiones fijas de memoria hasta evolucionar en particiones dinámicas.

- Fragmentación.
- Mantener partes del proceso que sean innecesarias.
- Los esquemas de particiones fijas y dinámicas NO son usados hoy en día.

- Solución:
  - Paginación
  - Segmentación

**Paginación**

➔ La memoria física es dividida lógicamente en pequeños trozos de igual tamaño (marcos).
➔ La memoria lógica (de los procesos) es dividida en trozos de igual tamaño que los marcos (esto genera paginas).
➔ El SO debe mantener una tabla de paginas por cada proceso, donde cada entrada de dicha tabla contiene el marco en la que se coloca dicha página. Se guarda la base del marco que se cargo en la PCB. Hay puntero a la tabla.
➔ La dirección lógica es interpretada como:
  - Un numero de pagina y un desplazamiento dentro de la misma.


➔
➔ Como se puede observar, se rompe la continuidad.
➔ Puede causar fragmentación interna (menos rompe bolas que la externa).
➔ No es muy importante esa fragmentación interna.

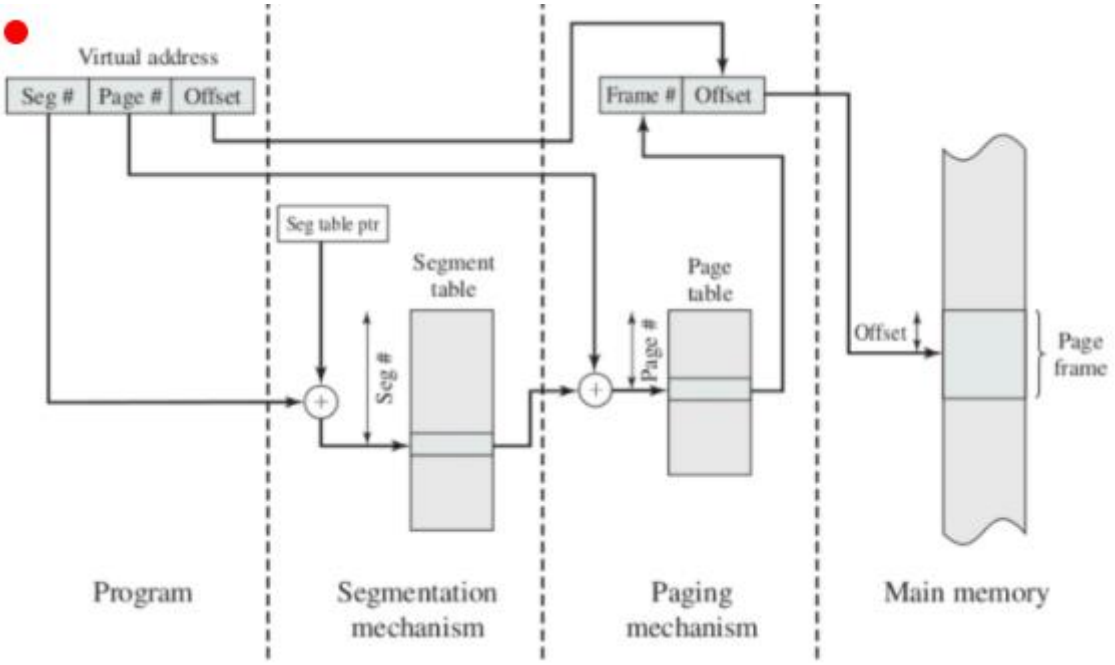
**Segmentación**

- ➔ Esquema que se asemeja a la visión del usuario. El programa es dividido en partes/secciones, donde en cada sección se guardan datos similares.
- ➔ Un programa es una colección de segmentos. Un segmento es una unidad lógica como:
    - Programa Ppal, procedures, funciones, variables locales y globales, stack, etc.
    - Cada segmento tiene un registro base y un registro límite.
    - Por lo que se genera una tabla de segmentos con esos dos valores para cada segmento por proceso.
- ➔ Puede causar fragmentación EXTERNA.
- ➔ Todos los segmentos de un programa pueden NO tener el mismo tamaño (códigos, datos, rutinas). La base y límite del segmento son dinámicos.
- ➔ Las direcciones lógicas consisten en 2 partes:
    - Selector de segmento.
    - Desplazamiento dentro del segmento (sobre registro base y límite).
- ➔ La segmentación posee ventaja sobre la paginación respecto de: la protección de espacios de memoria y la compartición de bloques de memoría.
- ➔ Cuando uno compila → el compilador deja huellas del segmento.

---

**Segmentación paginada (mix de las dos anteriores):**

- ➢ Paginación
    - Transparente al programador.
    - Elimina fragmentación externa.
- ➢ Segmentación
    - Visible al programador.
    - Facilita modularidad, estructuras de datos grandes y da mejor soporte a la compartición y protección.
- ➢ Segmentación paginada: cada segmento es dividido en páginas de tamaño fijo.
    - Cada segmento tiene una tabla de páginas que le corresponde.
    - Toma las ventajas de ambas: compartición, protección(de parte de la segmentación), evitar fragmentaciones (de parte de la paginación).
    - Se sigue guardando en paginas
    - La unidad de trabajo para subir o bajar de la RAM es la pagina


Virtual address
Seg #
Page #
Offset
Frame #
Offset
Seg table ptr
Segment table
Page table
Seg #
Page #
Offset
Page frame
Program
Segmentation mechanism
Paging mechanism
Main memory