
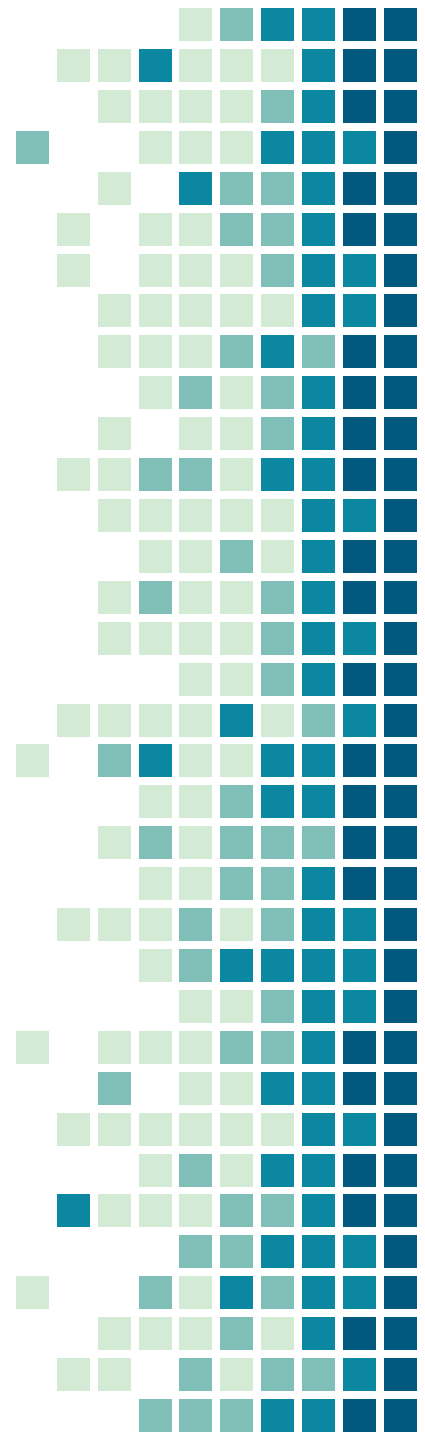


Memoria virtual



Agenda

- **Multiprogramación:**
 - Memoria virtual
 - Particiones estáticas
 - Particiones dinámicas
 - Paginación
 - **MMU**
 - **TLB**
- 



Multiprogramación

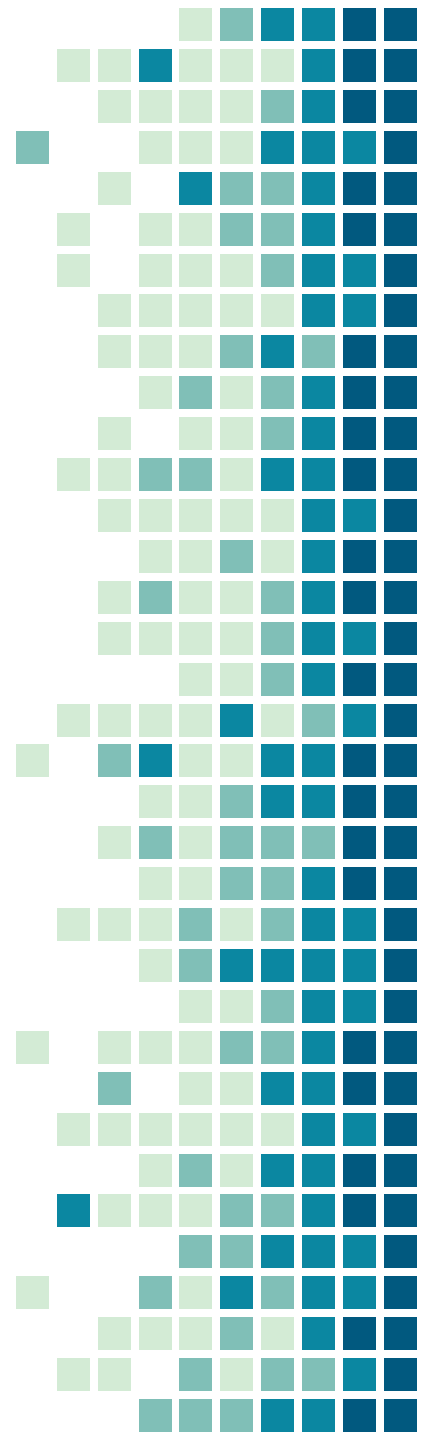
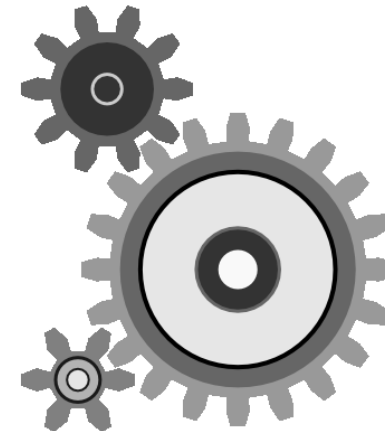


— Multiprogramación —

Paradigma en el que un **mismo computador** se comparte por **múltiples programas**.

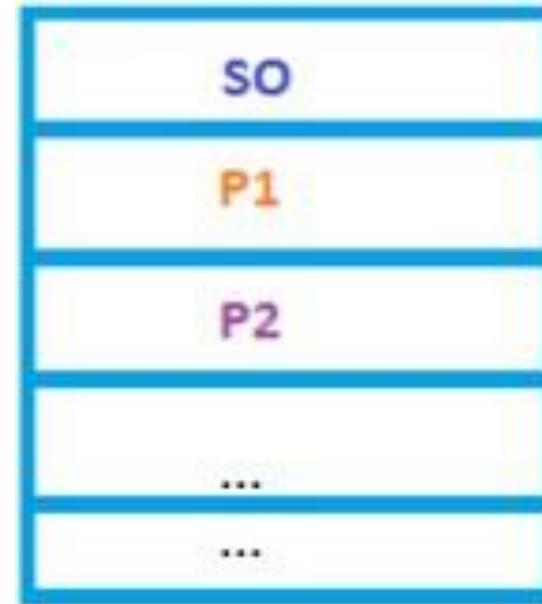
- Cada programa o proceso comparte los elementos de computación así como de almacenamiento (memoria).

Deben existir mecanismos de **protección** para los elementos compartidos.



Protección

La condición fundamental de la multiprogramación es que **todos** los procesos se deben ejecutar de **la misma manera** que si se estuvieran ejecutando independientemente en el computador

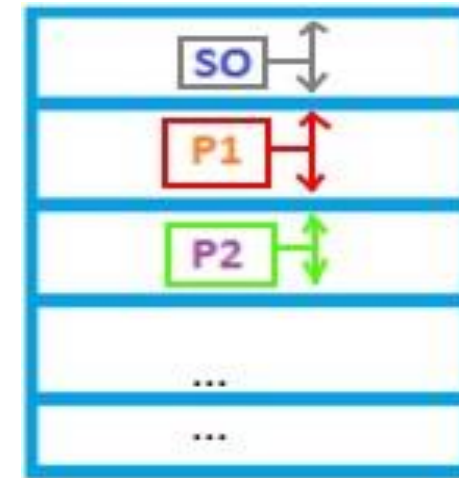


➤ Esto se logra a través de la **protección**: *brinda garantía de esa condición.*

Memoria virtual

Forma de brindar protección de memoria.

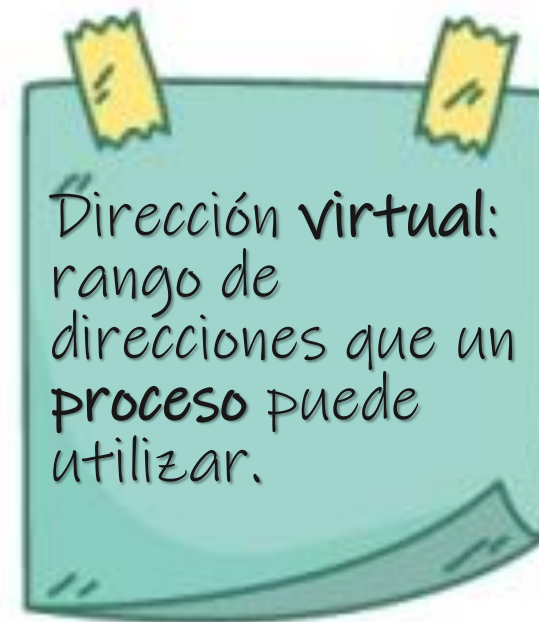
Idea principal: Dividir la memoria principal en **bloques** o particiones en donde se ejecutarán diferentes procesos, donde los procesos **no pueden acceder** a direcciones fuera del rango asignado.



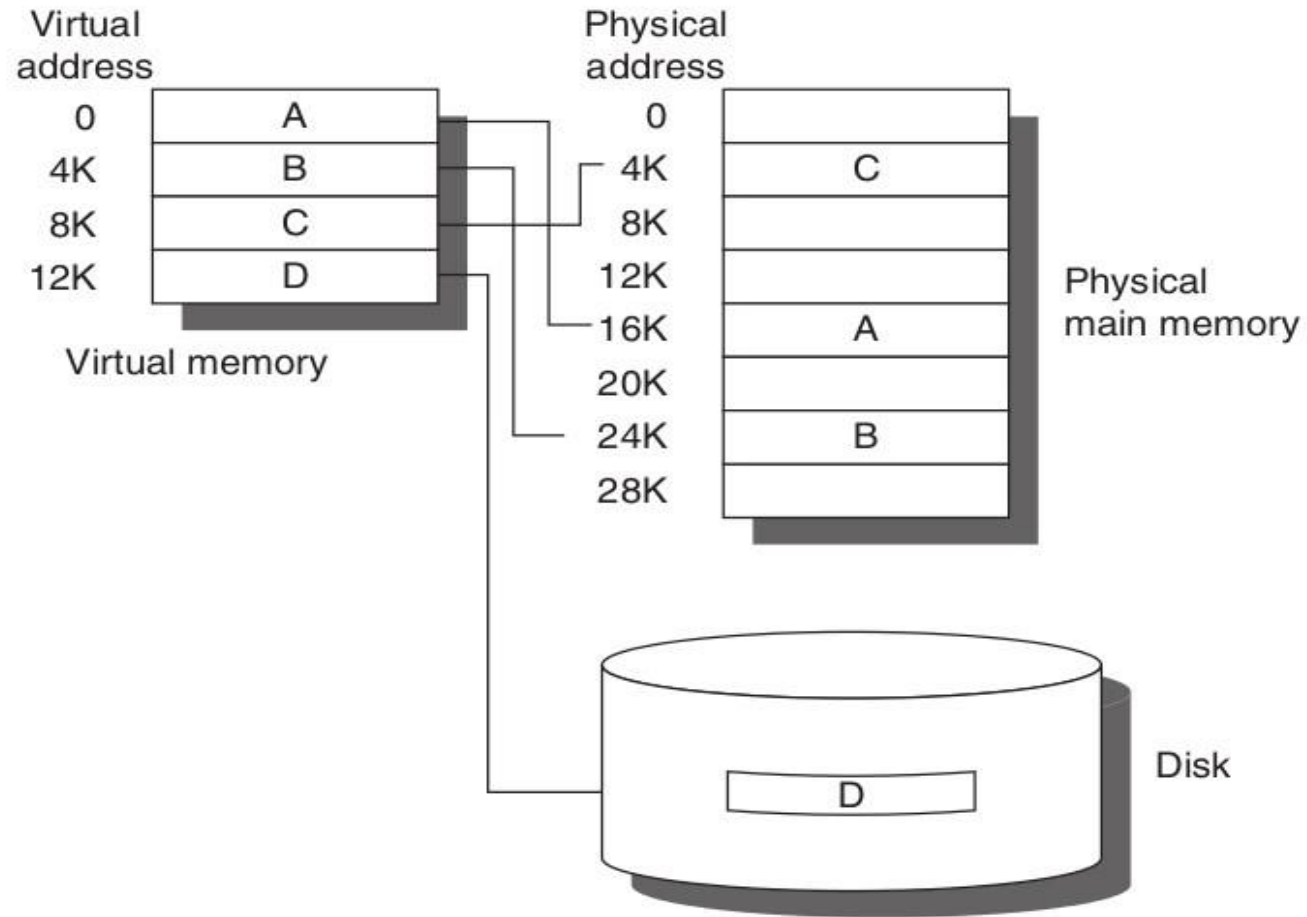
Dirección física vs Dirección virtual



VS



Memoria virtual



Direcciones virtuales: características

- ☞ Permiten que cada programa pueda ejecutarse en computadores con **diferente** tamaño de memoria física. **Independencia.**
- ☞ Direcciones virtuales de diferentes procesos corresponden a diferentes direcciones físicas. **Protección.**

Tipos de particiones

Tres tipos de principales de particiones o formas de brindar protección de memoria a través de memoria virtual:



Particiones estáticas

Particiones dinámicas
(segmentación)

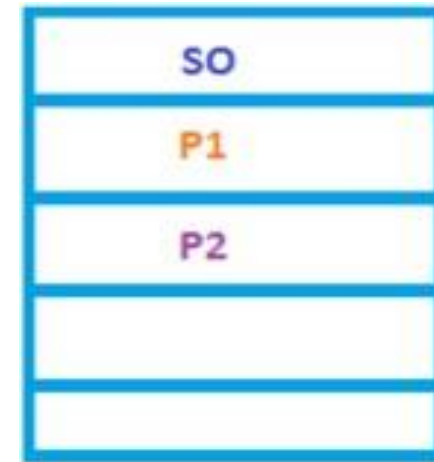
Páginas (paginación)

Particiones estáticas



Primer modelo de distribución de memoria en S.O

En las particiones de memoria estática, la memoria se asignaba **estáticamente** a los procesos según las particiones totales y las particiones libres.



Ventajas

Desventajas

Las particiones
estáticas son simples
y rápidas

Se reduce la
posibilidad de
error

Desperdicio de
memoria

Estático: las particiones
cuentan con un tamaño
definido y de igual
manera los procesos

Una partición por
proceso, si el proceso no
alcanza en la partición, no
se podrá ejecutar nunca

¿Solución?



Particiones dinámicas (Segmentación)



Las particiones de memoria para los procesos no se encuentran predefinidas, sino que se le **asigna memoria** a los procesos según la necesidad, de acuerdo con un algoritmo específico:

Best fit

First fit

Worst fit.



Ventajas

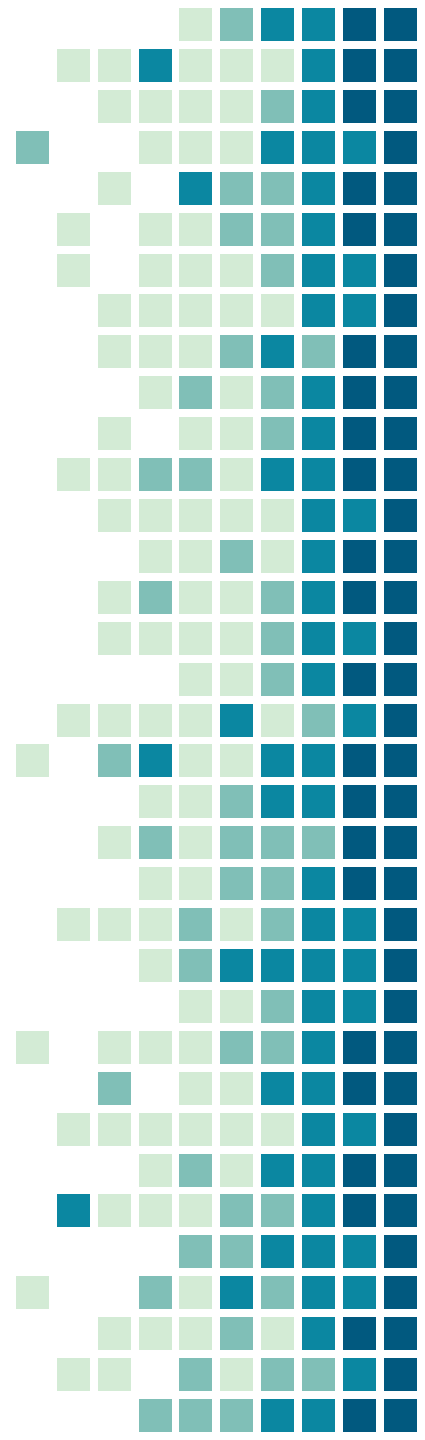
Se **aprovecha**
mejor el espacio

**Favorece
multiprogramación:**
Mayor cantidad de
procesos
ejecutándose
simultáneamente

Desventajas

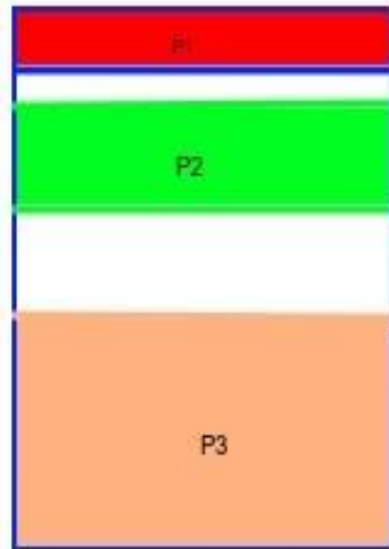
**Espacio continuo para
cada proceso:** procesos
pueden no alcanzar los
espacios libres, por lo
que deben esperar a
que la memoria se libere

**Fragmentación
de memoria**



Fragmentación

Situación no deseada en que la suma de espacios libres en memoria alcanzaría para ejecutar un proceso, pero este **no** puede ser ejecutado ya que **ningún espacio** independiente es **suficiente** para el proceso.



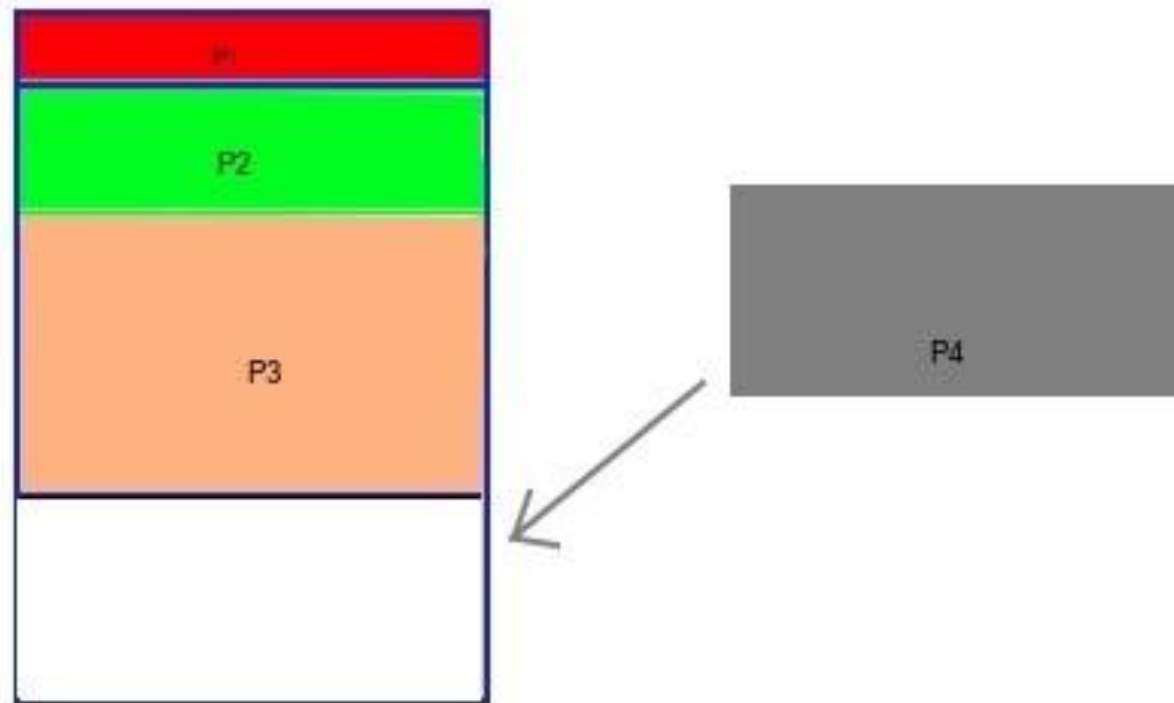
Solución: Compactación



Reacomodo dinámico de la memoria, en general es costoso en términos de tiempo ya que genera **overhead**.

- ➡ **Overhead:** El overhead es un mal necesario, representa un recurso (tiempo para este caso) requerido para administrar el sistema sin ser directamente productivo para el proceso / aplicación.

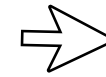
Compactación de memoria



Paginación



La **paginación** rompe ese requisito:



- Desde un **punto de vista lógico** se divide el programa en porciones de memoria de tamaño fijo llamado páginas (pages).
- Con este método, un programa **no tiene que** estar contiguo, se divide en páginas y cada página puede estar **físicamente separada en memoria.**

Problema
existente con
esquemas
anteriores: Cada
proceso requiere
estar
**físicamente
unido** en
memoria
(contiguos)

Paginación

	P3(2)		P1(1)
	P2(1)		P2(2)
P3(1)	P3(3)		P1(2)

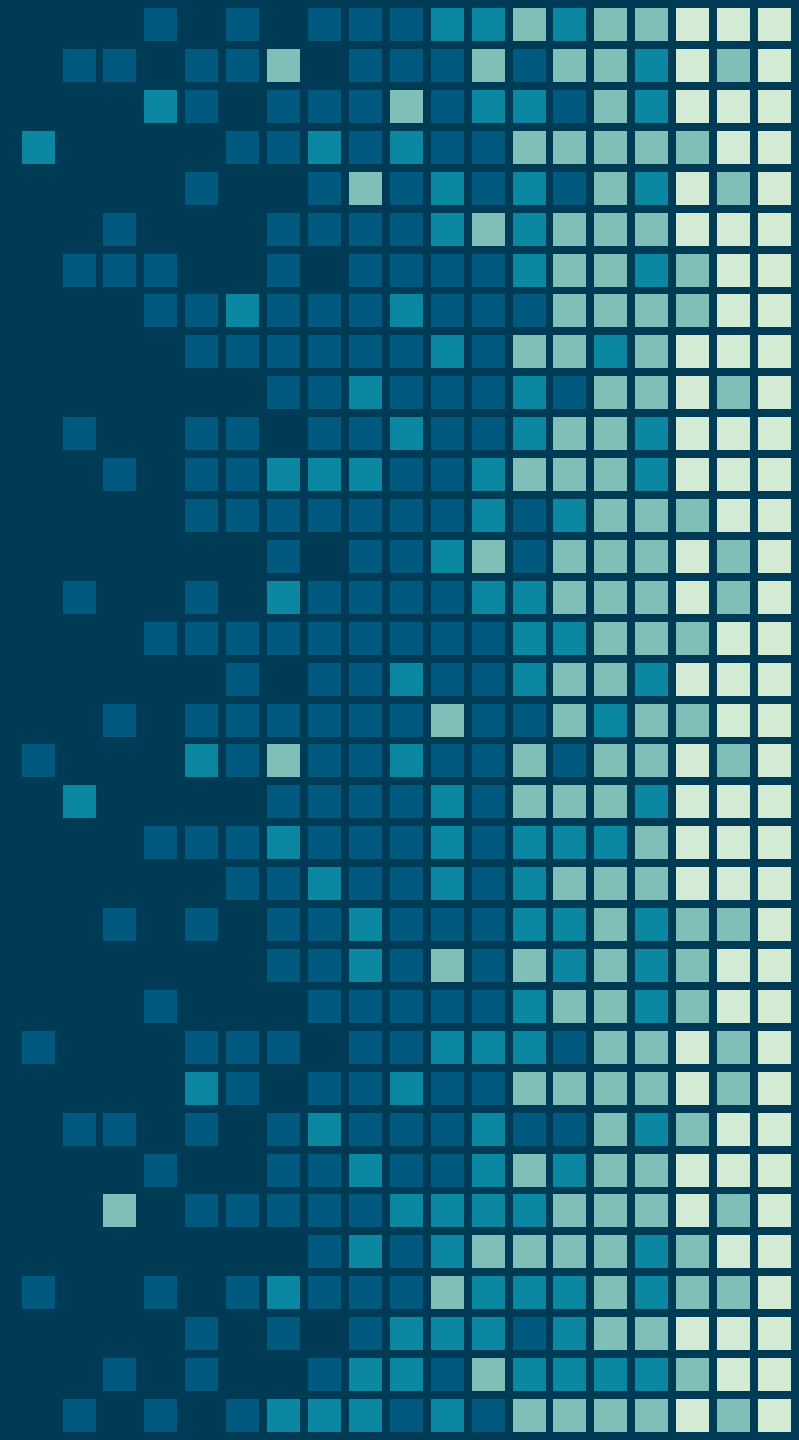


Se elimina la fragmentación: no es necesario realizar compactación. Disminuye el Overhead.



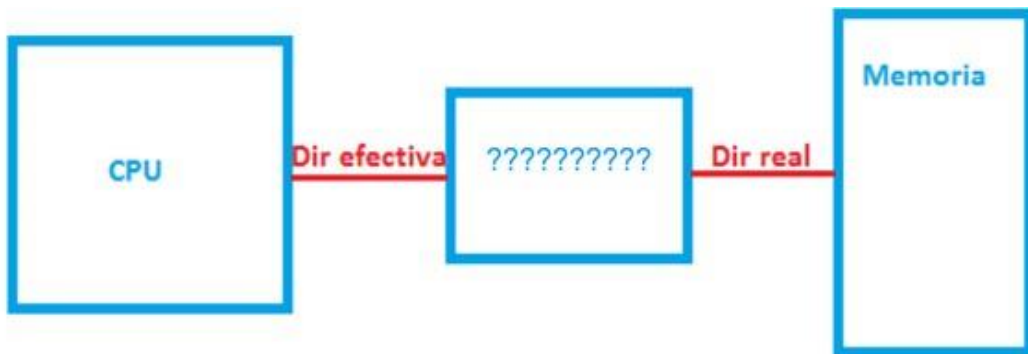
Fragmentación interna: Al final de cada página se desperdicia memoria.

MMU



Conversión entre direcciones

La paginación divide la memoria en secciones (páginas) **lógicas o virtuales**. Debe existir un mecanismo que se encargue de **convertir** las direcciones virtuales (dadas por procesador) para cada proceso en **direcciones reales (físicas)**.



Unidad de manejo de memoria

La MMU se encarga de:

- **Traducción de memoria:**
Convierte direcciones virtuales en direcciones físicas

- **Protección:** Realiza chequeos de privilegios y protección de lectura/escritura de memoria para el procesador en un proceso específico

- **Control de caché:**
Diferentes regiones de memoria pueden requerir diferentes tratos en caché.

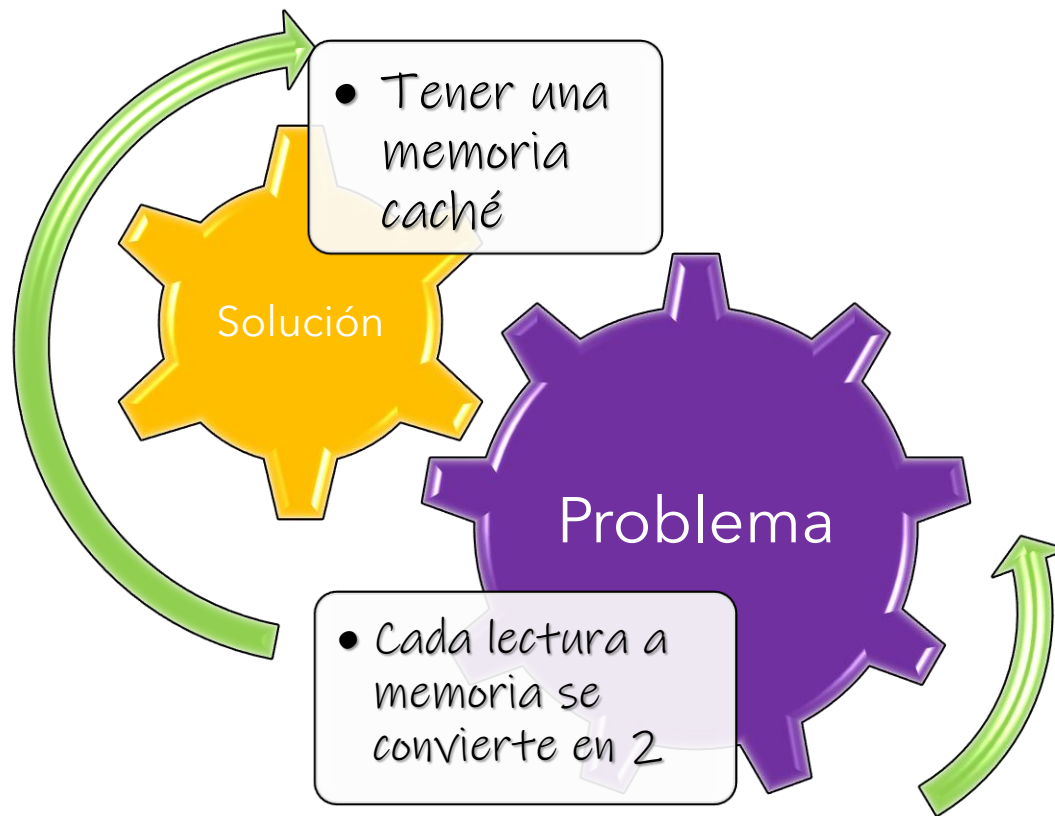
Unidad de Manejo de Memoria

La MMU recibe el número de página y offset (dir efectiva) y se encarga de generar (por medio de indexar una tabla de página) la nueva dirección real.

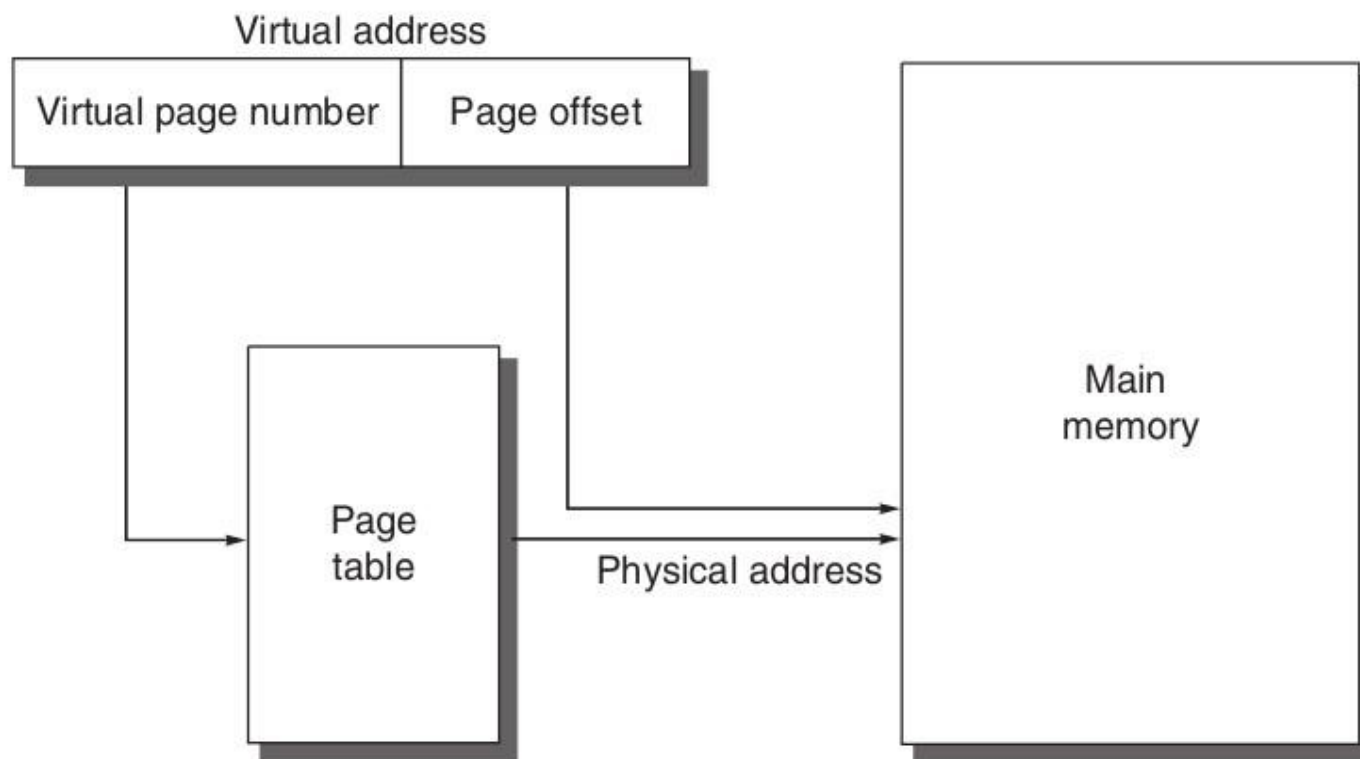
Tabla de Página: contiene la información para traducir de número de página virtual a real.

- Es cargada por la MMU para realizar dicha conversión de manera **transparente** para las aplicaciones.
- La tabla de página se encuentra en **la memoria principal**

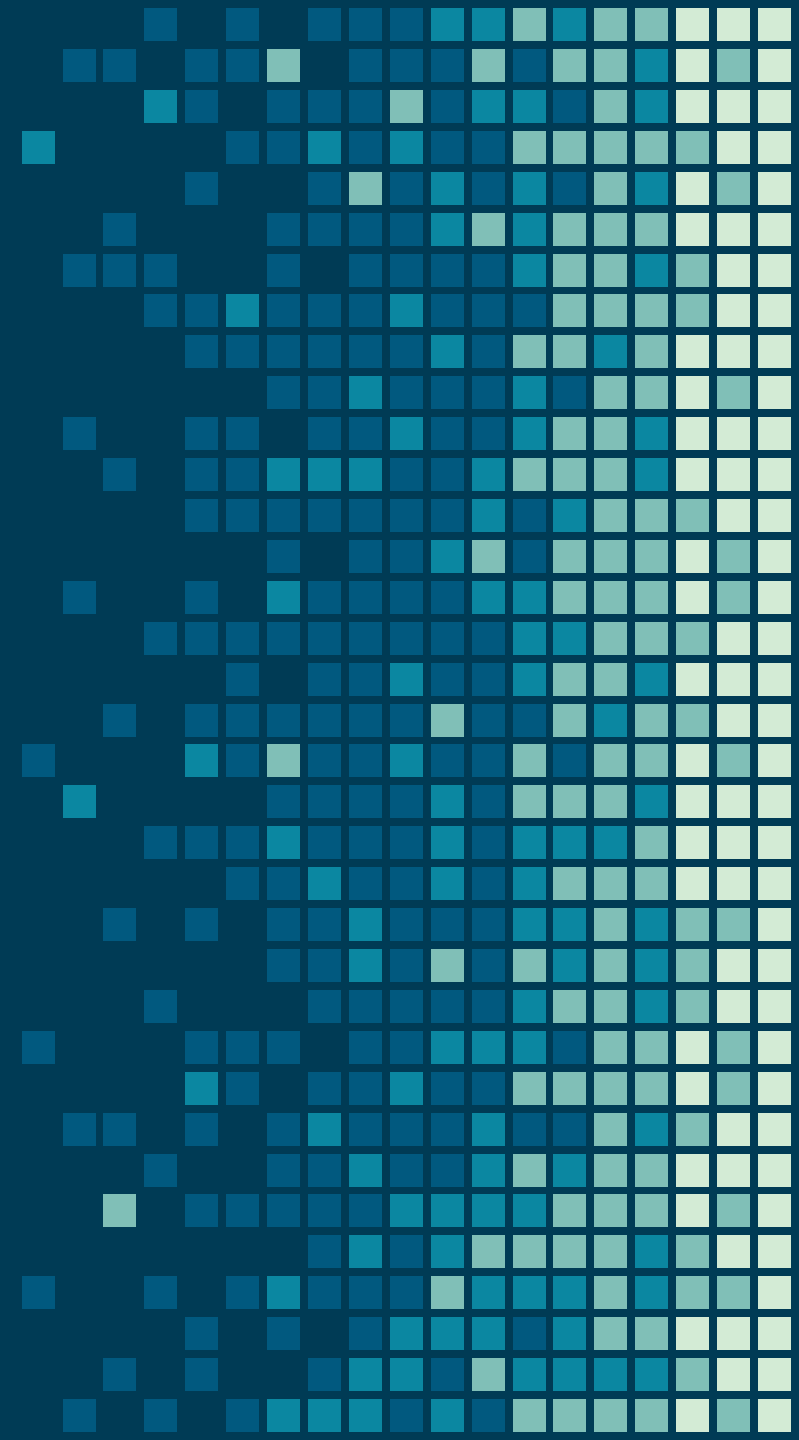
Desafíos



Traducción virtual/real




TLB (Translation Look-aside Buffer)




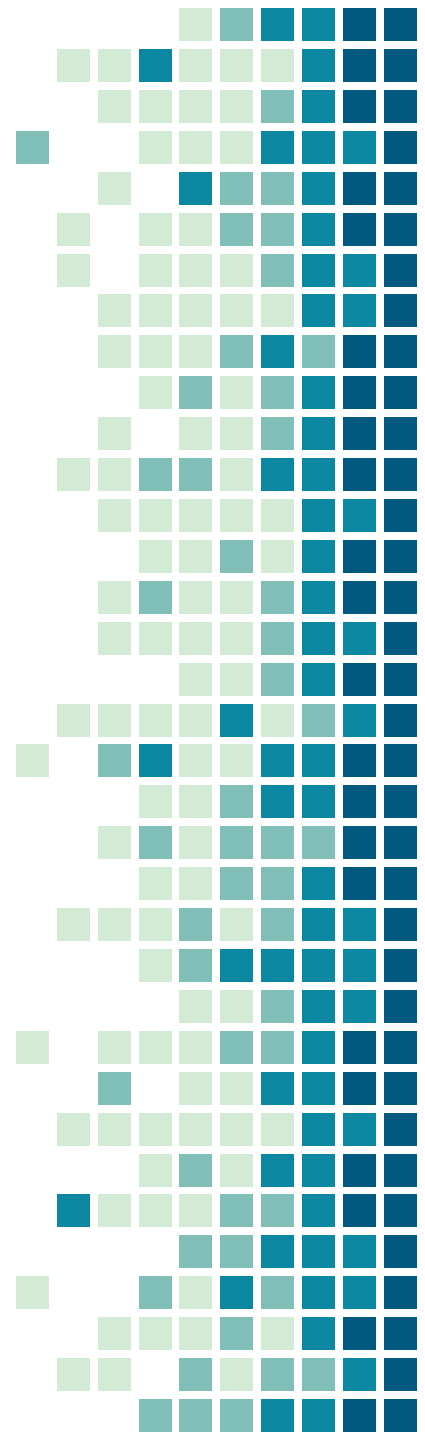
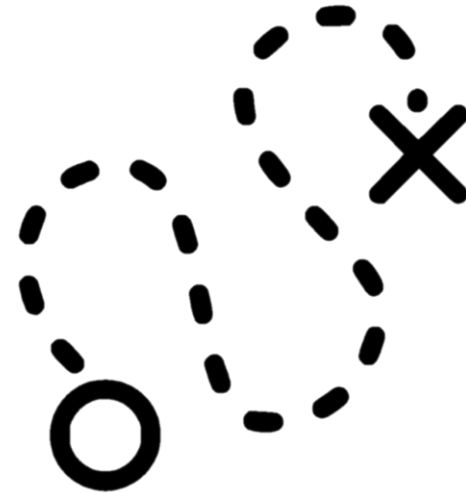


La tabla de página para cada proceso se mapea por segmentos a una **memoria caché** dentro de la **MMU**.

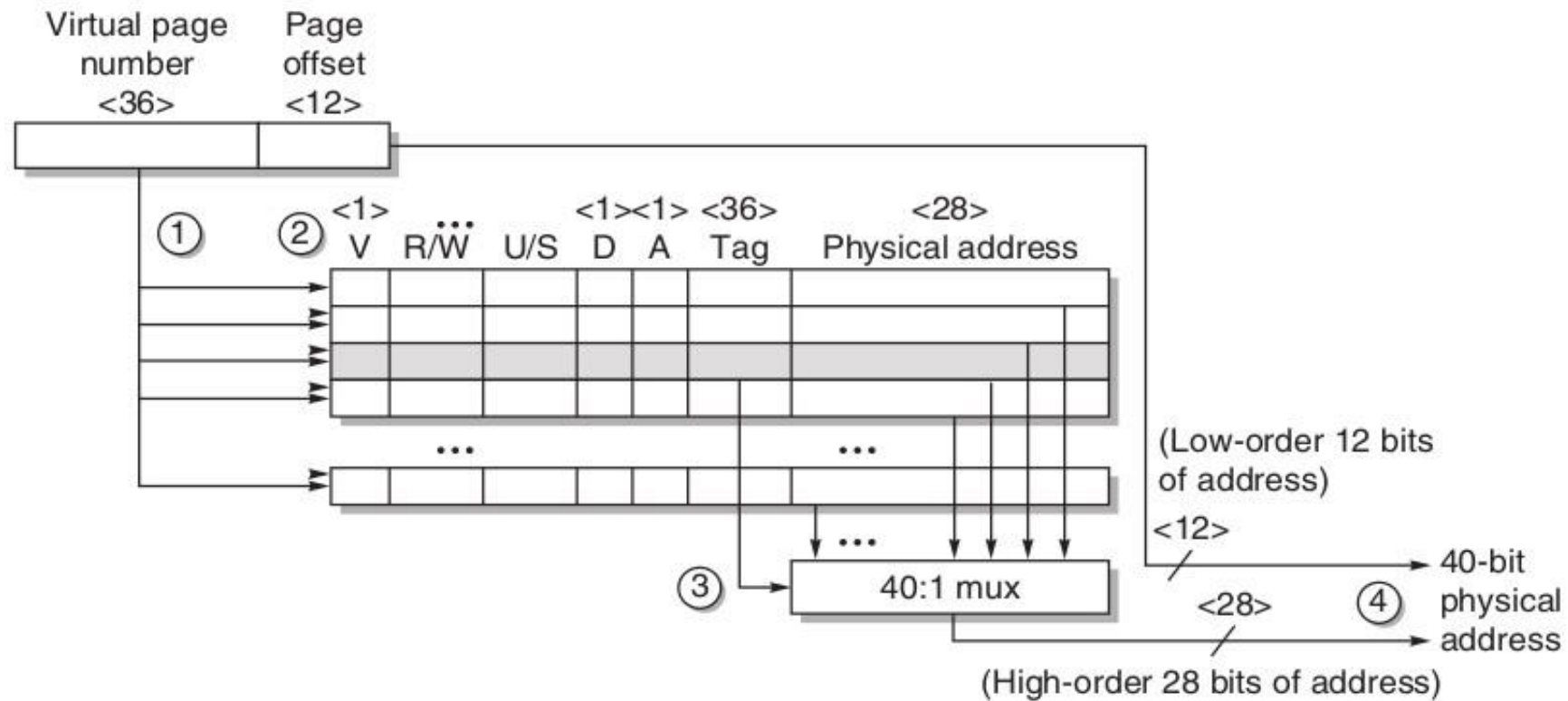
Cada vez que el CPU genere una **nueva dirección de memoria** (para leer o escribir) en la entrada de memoria:

 Primero se busca en la TLB, si está (hit) se realiza la conversión inmediatamente.

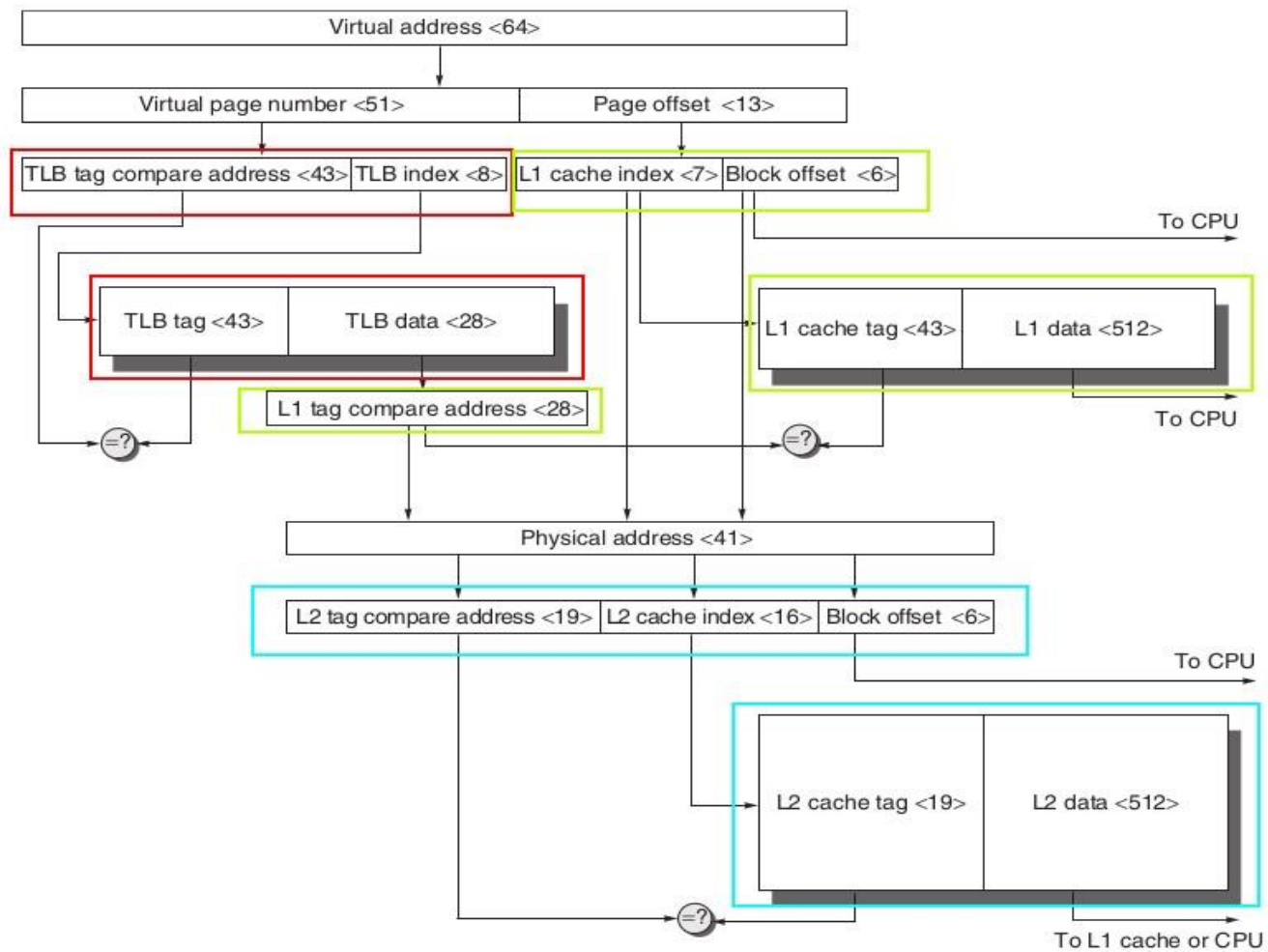
 Si no está, se debe traer el segmento de la tabla de página de memoria principal a la caché de la MMU (miss) y luego realizar la conversión.



Organización. AMD Opteron

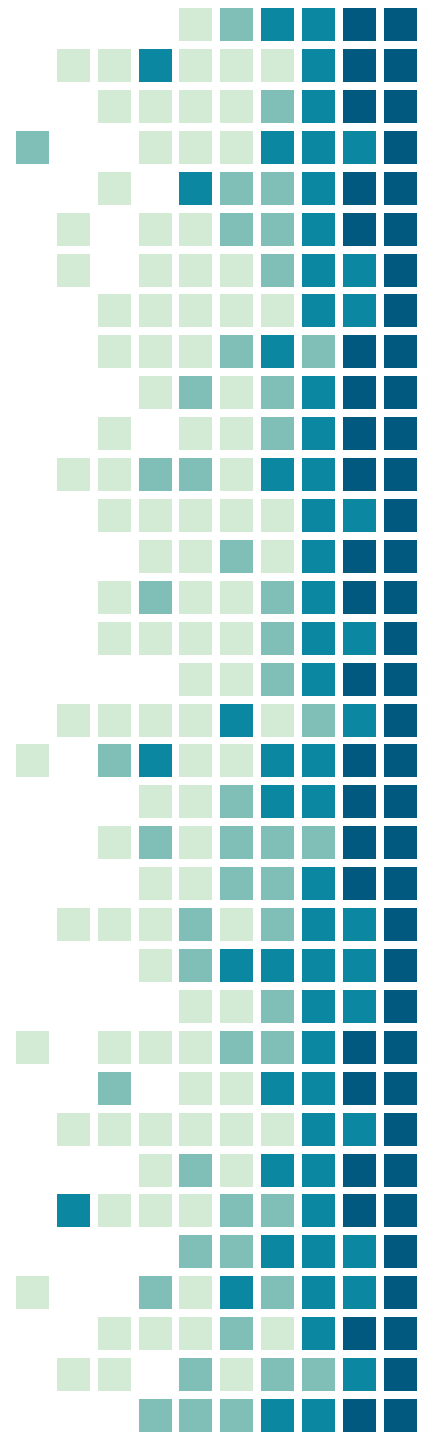


TLB y caché



Referencias

-► J Hennesy and David Patterson (2012)
Computer Architecture: A Quantitative Approach. 5th Edition. Elsevier – Morgan Kaufmann.
-► William Stallings (2010)
Computer organization and architecture: designing for performance. Pearson Education India
-► Peter Barry and Patrick Crowley (2012)
Modern Embedded Computing: Designing Connected, Pervasive, Media-Rich Systems.



¿Preguntas?

Realizado por: Jason Leitón Jiménez.

Tecnológico de Costa Rica

Ingeniería en Computadores

2024

