# SISTEMAS OPERATIVOS

3004610 - 1

**German Sánchez Torres, I.S., M.Sc., Ph.D.**

Profesor, Facultad de Ingeniería - Programa de Sistemas

Universidad del Magdalena, Santa Marta.

Phone: +57 (5) 4214079 Ext 1138 - 301-683 6593

Edificio Docente, Cub 3D401.

Email: sanchez.gt@gmail.com –gsanchez@unimagdalena.edu.co


# Jerarquía de Memoria

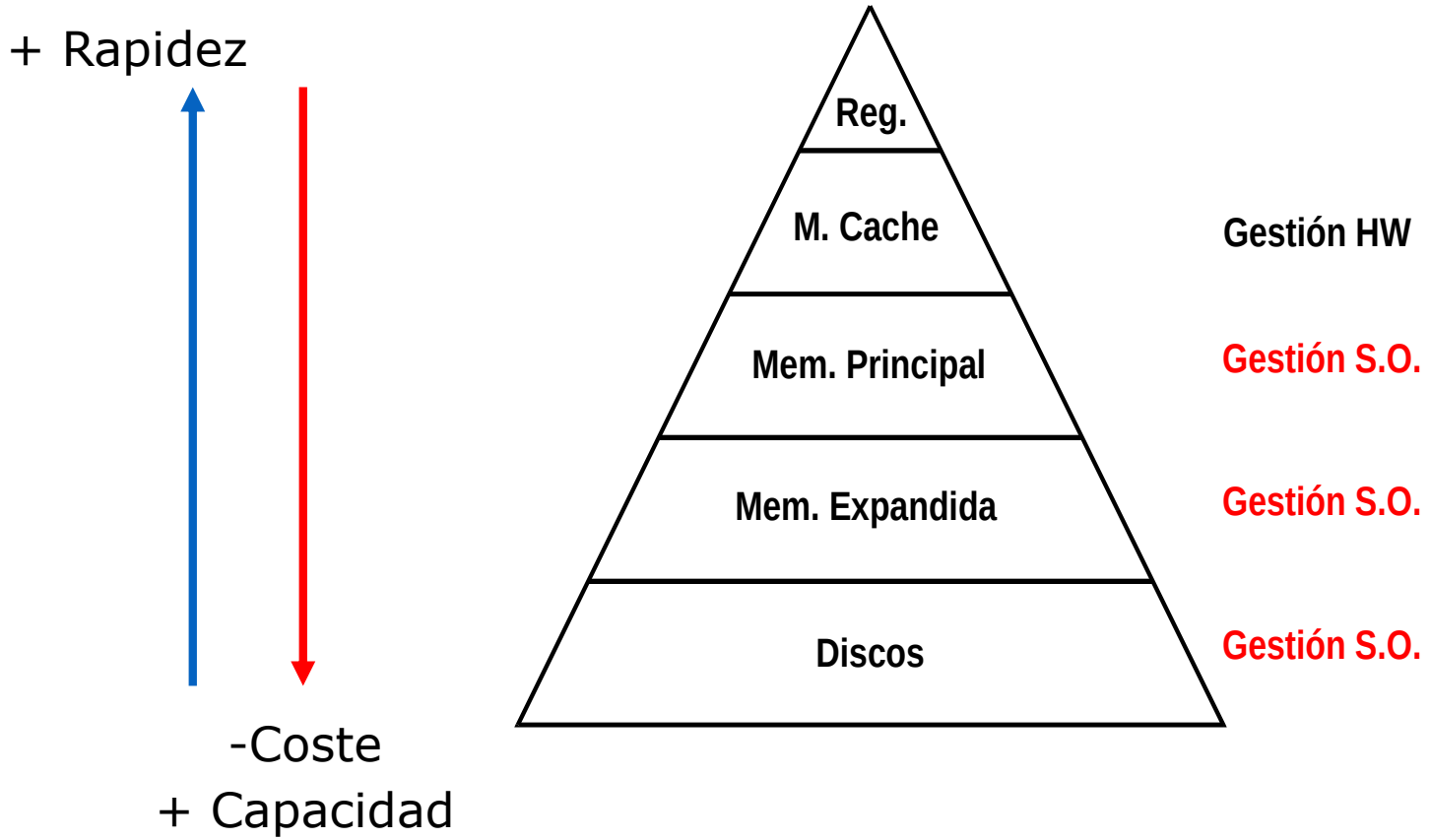
# Introducción

Objetivo: conseguir tiempos de acceso aceptables a costos razonables

- Elementos de información replicados en varios niveles de la jerarquía. Cuando los datos van a ser usados se promueven a niveles más rápidos (hacia arriba). Cuando se modifica o crea información en niveles rápidos y quiere hacerse permanente, se promueve a niveles inferiores

- Problema de coherencia: necesidad de actualizar las diferentes copias de los mismos datos, de forma que se asegure que siempre que se use se tomarán los valores correctos

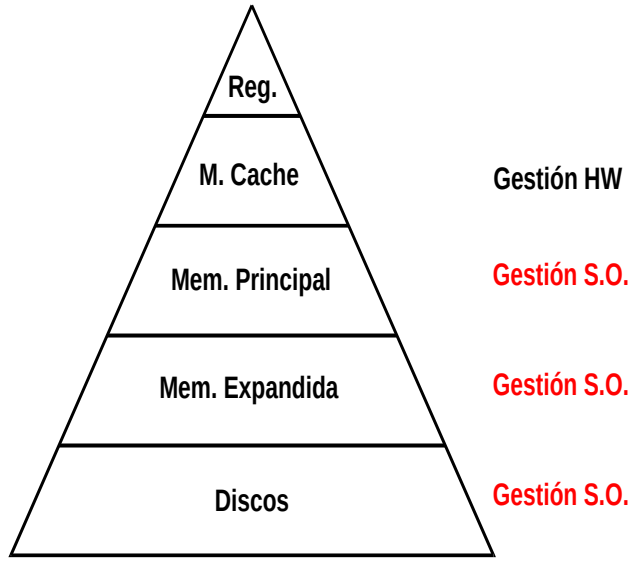
- Traducción de direcciones: las copias de la misma información, en niveles distintos, tendrán direcciones diferentes. Necesidad de traducir


+ Rapidez
-Coste
+ Capacidad
Reg.
M. Cache
Mem. Principal
Mem. Expandida
Discos
Gestión HW
Gestión S.O.
Gestión S.O.
Gestión S.O.

# Migración

Objetivo: que la información esté en el nivel adecuado cuando se necesite

- Migración de la información realizada de forma:
  - **Automática**: se realiza de forma transparente al programa. Se produce en las memorias caché y en la memoria virtual
  - **Por demanda explícita**: el programa solicita explícitamente el movimiento de la información.

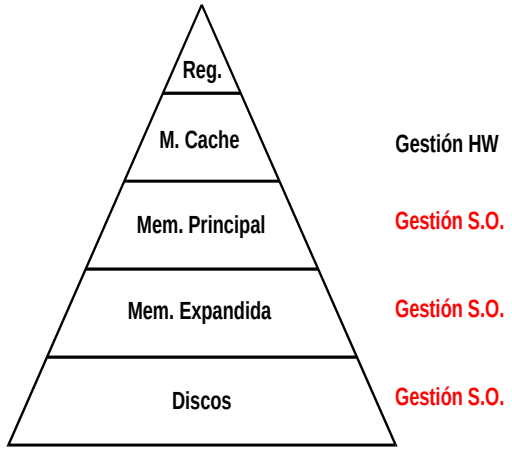


El mecanismo de migración se basa en los siguientes aspectos:

•**tamaño de bloques transferidos**: cantidad de información movida entre niveles. Disminuye a medida que nos acercamos a la cúspide

•**política de extracción**: qué información sube al nivel k desde el nivel k+1 y cuándo. Solución más común: por demanda. Se promueva aquella información solicitada por el programa y cuando se referencia. Se aprovecha la proximidad espacial, por lo que no sólo se promueve el dato solicitado, sino también las direcciones cercanas (bloque, línea, página, ....)

•**política de reemplazo**: el nivel k tiene menor tamaño que el nivel k+1, por lo que estará lleno cuando se quiera llevar información a él, y habrá que extraer datos ya presentes



# Direccionamiento

El programa en ejecución genera la dirección X relativa al dato A. Esta dirección X se refiere al nivel k+1. Sin embargo, hay una copia del dato en el nivel K, por lo que se deseará acceder a la copia en este nivel. Este dato tiene dirección Y en el nivel K.


Se trata de un problema no trivial, ya que puede haber notables diferencias de tamaño entre niveles:

Cuantos bit contendrá una dirección en cada nivel?

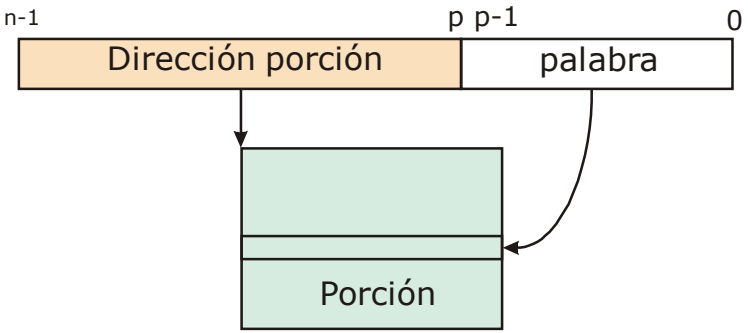
k+1 (2GB, n=?)

k (8M, m=?)

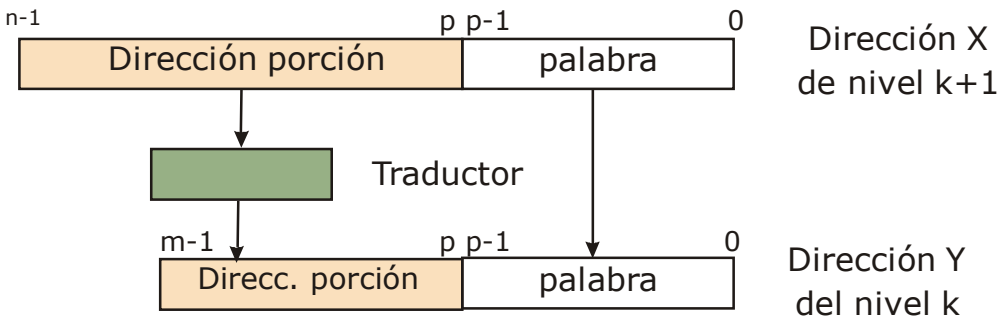
$$2^n = x \qquad n = \log_2 x \qquad \log_b x = \frac{\log_k x}{\log_k b}$$

Para simplificar la traducción y aprovechar la proximidad espacial, se dividen los mapas de direcciones en porciones de tamaño fijo ($2^p$). Estos bloques constituyen la unidad mínima de información transferida entre niveles. Al tener la unidad de información este tamaño, la dirección puede descomponerse en dos partes:

m-p bits: identifican el bloque
p bits: identifican el byte dentro del bloque




**División de la dirección**




**Traducción de la dirección**



Tasa de aciertos, tiempo medio efectivo

# Proximidad

**Proximidad referencial**: un programa en ejecución sólo usa en cada momento una pequeña parte de toda la información que maneja. Las referencias usadas muestran además ciertas relaciones.

Se denomina traza a la lista ordenada, en el tiempo, de las direcciones de memoria direccionadas. La traza R está compuesta tanto por direcciones de instrucciones como por instrucciones de datos

$R_e = r_e(1), r_e(2), r_e(3), ....., r_e(j)$

$r_e(i)$: i-ésima dirección ejecución del programa e

Sean u y v dos direcciones: $d(u,v)=|u-v|$, por lo que

$$d(r_e(j),r_e(k))=|r_e(j)-r_e(k)|$$

- ***Proximidad espacial***: dadas dos referencia $r_e(j)$ y $r_e(i)$ próximas en el tiempo (i-j es pequeño), existe alta probabilidad de que su distancia $d(r_e(i),r_e(j))$ sea muy pequeña

    - ***Proximidad secuencial***: ya que muchos trozos de programas y estructuras de datos se recorren de forma secuencial, existe alta probabilidad de que la referencia siguiente a $r_e(j)$ coincida con la siguiente dirección de memoria

- ***Proximidad temporal***: los programas suelen referenciar direcciones empleadas en un pasado próximo. Es decir, existe alta probabilidad de que la próxima referencia a $r_e(j+1)$ esté entre las n referencias previas:

$$r_e(j-n+1),r_e(j-n+2),...,r_e(j-1),r_e(j)$$