# Guía compacta de Memoria Virtual

## 1. ¿Qué es la memoria virtual?

Abstracción que permite a cada proceso “creer” que dispone de un espacio de direcciones continuo y amplio. Proporciona aislamiento entre procesos y hace posible utilizar más espacio lógico que RAM física, almacenando páginas poco usadas en disco.

## 2. Paginación: páginas ↔ frames

|  |  |  |
| --- | --- | --- |
| Concepto | En memoria virtual | En memoria física |
| Bloque | Página | Frame |

El tamaño de página y de frame es idéntico, por lo que el desplazamiento (offset) es el mismo en ambas direcciones.

## 3. Formato de las direcciones

Para una página de 2^p bytes dentro de un espacio de 2^m bytes:

Dirección virtual (m bits) = [Número de página (m-p bits)] | [Offset (p bits)]

Dirección física (n bits) = [Número de frame (n-p bits)] | [Offset (p bits)]

## 4. Traducción paso a paso

1. TLB: si hay hit, obtenemos el número de frame de inmediato.  
2. Si hay miss en TLB, se consulta la tabla de páginas.  
3. Si la entrada es válida, actualiza el TLB; si no, ocurre un page fault.  
4. Con el frame + offset se genera la dirección física y se accede a caché/RAM.

## 5. Page fault

El sistema operativo copia la página desde disco a un frame, actualiza la tabla de páginas y el TLB, y reintenta la instrucción.

## 6. Tiempo de acceso efectivo (EAT)

EAT = T\_TLB + (1 - hit\_TLB) \* T\_PT + tasa\_fallos \* T\_PF

## 7. Interacción con la caché

Después de traducir la dirección física, se busca en la caché (hit → pocos ns, miss → acceso a RAM y carga del bloque).

## 8. Políticas de miss de escritura

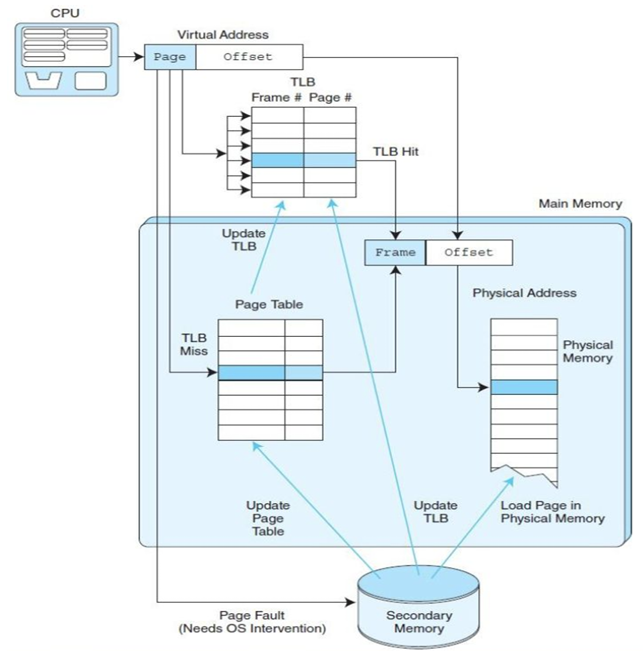
Write-allocate: trae el bloque al miss; Write-around: escribe directo en RAM.

## 9. Políticas de escritura

Write-through: actualiza caché y RAM al mismo tiempo.  
Write-back: solo actualiza la caché y escribe a RAM al reemplazar el bloque (dirty bit).

## 10. Flujo completo (resumen visual)

El diagrama siguiente muestra el camino de la dirección virtual a la física con TLB, tabla de páginas, caché y manejo de page fault:



**Lectura paso a paso del diagrama**

*(numeramos las flechas azules para que sigas el recorrido lógico)*

| **#** | **Qué componente interviene** | **Qué ocurre exactamente** |
| --- | --- | --- |
| **①** | **CPU** → emite una **dirección virtual**.  Está dividida en: • **Page** (n.º de página virtual). • **Offset** (desplazamiento dentro de la página). | Ningún dato real se toca aún; solo se forma la dirección. |
| **②** | **TLB** (Translation Look-aside Buffer) | Es una mini-caché de traducciones página ↔ frame.  Busca el n.º de página en sus entradas. • **TLB hit** → ya conoce el **frame** físico → salta al paso ④.  • **TLB miss** → pasa al paso ③. |
| **③** | **Tabla de páginas (Page Table)** en la RAM | Se indexa con el n.º de página que vino de la CPU.  Hay dos casos:  **a)** **Entrada válida** (la página ya está en RAM) → devuelve el n.º de frame, **actualiza el TLB** (flecha “Update TLB”) y salta al paso ④.  **b)** **Entrada inválida** → ocurre un **page fault** → salta al paso ⑤. |
| **④** | **Formación de la dirección física** | Offset (no cambia nunca) se concatena con el frame físico obtenido.  Esa dirección física se usa para acceder a **caché L1/L2** y, si es necesario, a la **RAM** (bloque resaltado en azul). |
| **⑤** | **Page fault** → Interviene el Sistema Operativo | 1. El SO detiene la CPU.  2. Localiza la página en **disco/SSD** (círculo “Secondary Memory”).  3. Elige un frame libre o expulsa uno (si expulsa un frame “dirty”, lo escribe a disco).  4. **Carga** la página desde disco a ese frame (flecha “Load Page in Physical Memory”). |
| **⑥** | **Actualizaciones** | Después de cargarla:  • Marca la entrada correspondiente como **válida** en la tabla de páginas (flecha “Update Page Table”).  • Inserta la nueva traducción en el **TLB** (flecha “Update TLB”).  • Retoma la instrucción fallida → ahora volverá al paso ④ y seguirá normalmente. |