UAI Universidad Abierta Interamericana

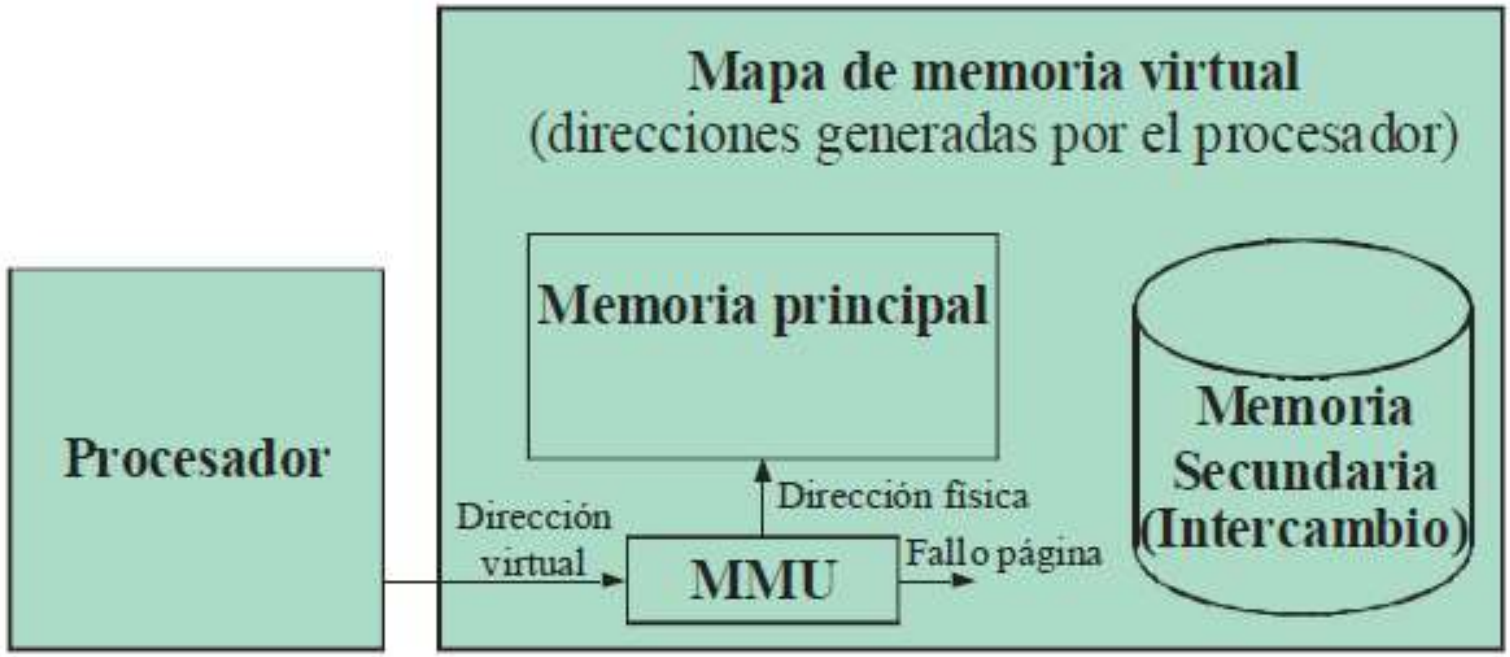
ISTEMAS OPERATIVOS

Mg. Leandro Ezequiel Mascarello

<leandro.mascarello@uai.edu.ar>

UAIOnline ultra

- Gestión automática de la parte de la Jerarquía de Memoria formada por los niveles de memoria principal y de disco.



Gestión de Memoria Virtual

- **Jerarquía de memoria**: Niveles de almacenamiento
- Procesos exhiben **proximidad de referencias**
- **Memoria virtual**
  - Transferencias M. principal y M. secundaria (*swap*)
- Basada en **paginación**:
  - Página no residente se marca ausente
  - Se guarda dir. de bloque de *swap* que la contiene
- **Disco ⟶ M. principal**: por demanda:
  - Acceso a pág. no residente: **Fallo de página**
  - S.O. lee página de disco
- **M. principal ⟶ Disco**: por expulsión::
  - No hay espacio en M. principal para traer página
  - Se expulsa (reemplaza) una página residente
  - S.O. escribe página expulsada a disco

- Beneficios:
  - Protección y reubicación de procesos
  - Aumento grado de multiprogramación
    - ¡Cuidado con "hiperpaginación" [*trashing*]!
  - Ejecución de programas que no caben en M. ppal.
- El proceso sólo ve **direcciones virtuales**.
- El mapa virtual de un proceso está soportado en memoria secundaria (*swap*).
- No es necesario que todo el proceso esté cargado en memoria =>
  - Más procesos en MP, uso más eficiente del procesador.
  - Un proceso puede ser mayor que la MP.
  - Arranque más rápido.
- **Conjunto residente**: parte del proceso cargado en MP.

Paginación

- **Hardware** [¡imprescindible!] más sofisticado
- Unidad de asignación: **página: 2^p**
- Mapa de memoria dividido en páginas
- **Memoria física** dividida en **marcos de página**
- **Tabla de páginas** (T.P.) por proceso:
  - Relaciona cada página con el marco que la contiene
- **MMU** usa T.P. para traducir direcciones lógicas
- S.O. mantiene T.P. de procesos y notifica a MMU cuál debe usar

Paginación

- Entrada de la T.P.
  - Protección (RWX)
  - "Bit de residencia": Página presente (**P**) o ausente **(A**)
- Fragmentación interna
  - En cada zona de memoria, puede desperdiciarse parte de último marco asignado a proceso
- ¿Cumple requisitos?
  - Espacios independientes: Mediante T.P.
  - Protección: Mediante T.P.
  - Compartir memoria: Páginas asociadas a mismo marco
  - Soporte de regiones: se usa información de la T.P.
    - Tipo de acceso no permitido: Protección
    - Accesos a huecos: Página presente
    - No se reserva espacio para huecos
  - Maximizar rendimiento
    - Buen aprovechamiento
    - Permite memoria virtual

Ejecución de Programas

- Creación de **regiones** iniciales desde ejecutable
- No se asigna espacio en M. principal
- No se carga nada en M. principal
  - Se traerá bajo demanda
- Se rellena cada entrada de T.P.:
  - Protección: Depende de tipo de región
  - Ausente
  - Dir. en Disco donde está almacenada
- Último valor depende del tipo de **gestión de *swap***

Gestión del Espacio de SWAP

- Asignación de espacio de *swap* cuando se crea región
- Dos alternativas: con y sin preasignación
- Creación de región **con preasignación**
  - Se asigna espacio de *swap*
  - Se copia en él contenido inicial desde disco
  - Páginas se traen por demanda desde *swap*
  - En expulsión ya tiene espacio reservado
- Creación de región **sin preasignación**
  - No se asigna espacio de *swap*
  - Páginas se traen por demanda desde disco
  - En expulsión se reserva espacio de *swap* (si pág. "sucia")
- *Sin preasignación se usa más actualmente*
- Región compartida con soporte no usa *swap*

Con Preasignación de SWAP

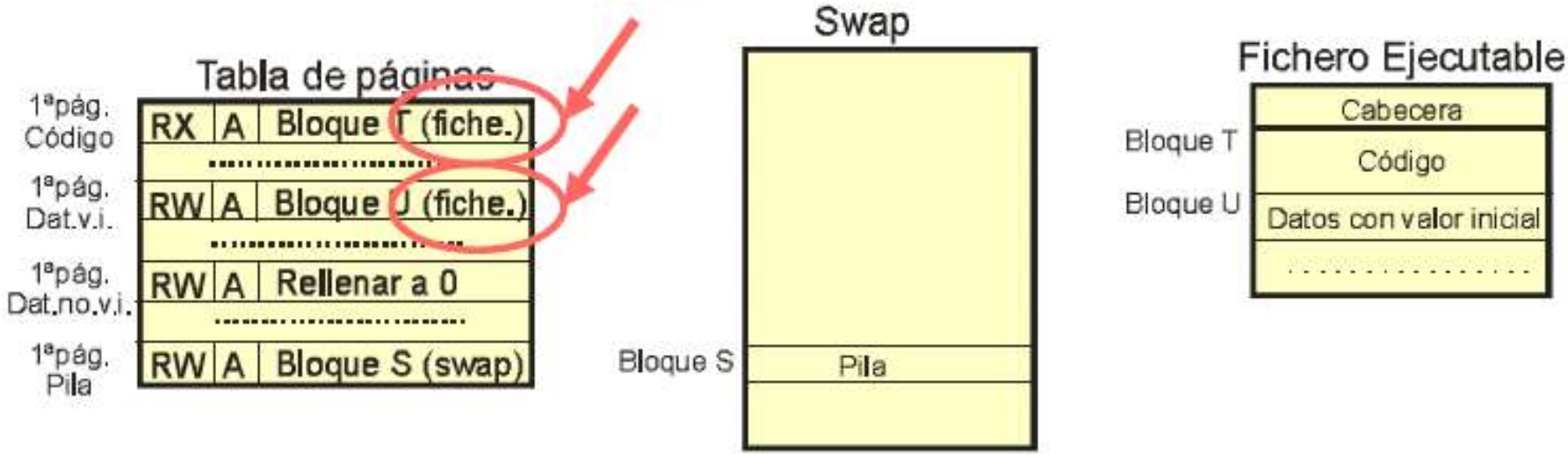
- S.O. reserva espacio en *swap* para regiones
- Copia del soporte al swap:
  - Código: del ejecutable
  - Datos "inicializados": del ejecutable
  - Pila: argumentos del programa
  - Datos no "inicializados": rellenar a 0
- Entradas T.P referencian a bloques del *swap*

Sin Preasignación de Código

- Región de código: compartida con soporte
- No es necesario usar *swap*
  - Se usa directamente ejecutable

Sin Preasignación de SWAP

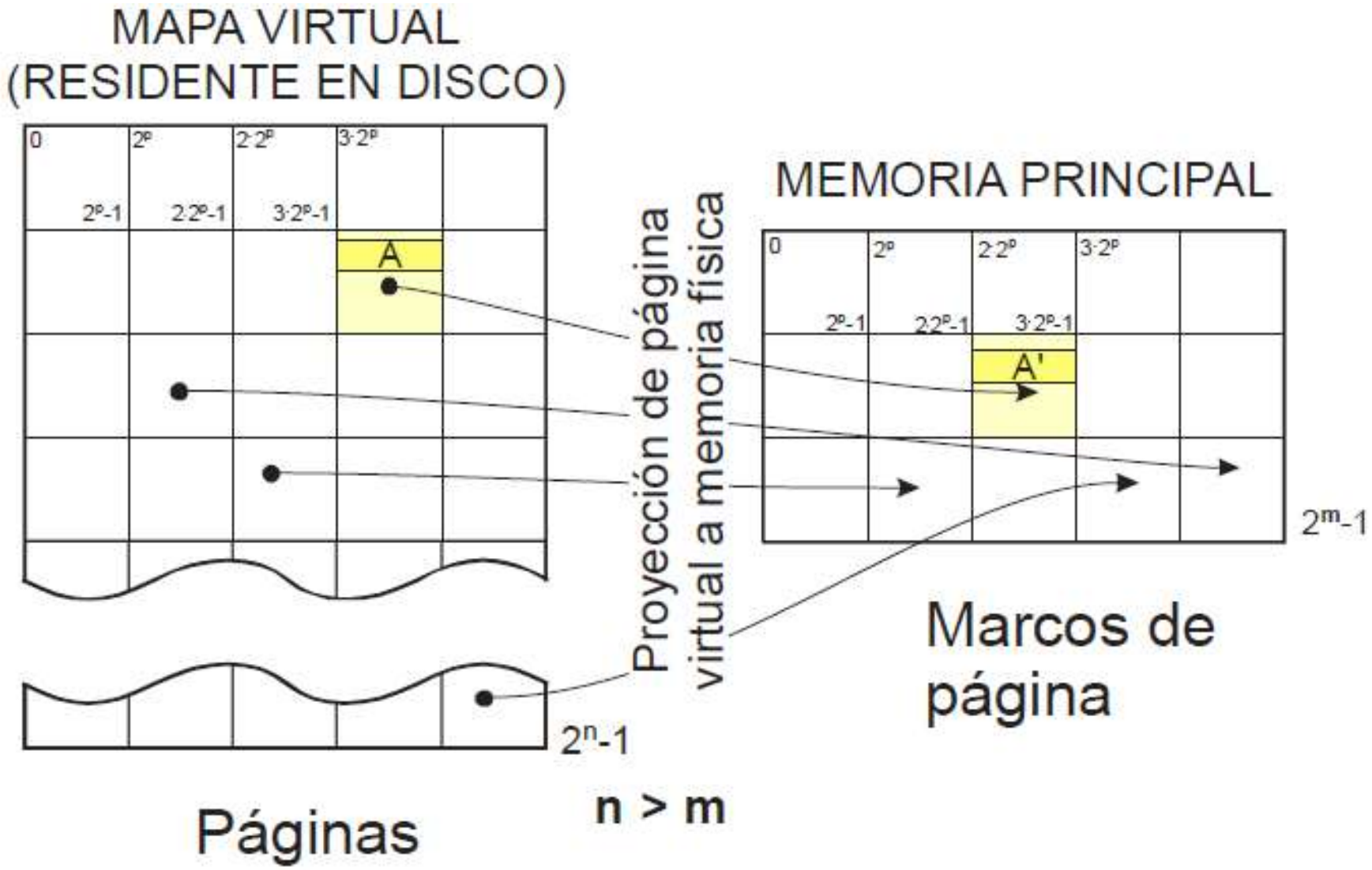
- Entradas T.P. referencian a bloques del soporte (si hay):
  - Código: bloques del ejecutable
  - Datos “inicializados”: bloques del ejecutable
  - Datos no “inicializados”: valor que indica rellenar a 0
  - Pila: bloque del *swap* con argumentos del programa
- Al expulsar por primera vez una página modificada se reserva espacio de *swap*
  - Excepto si compartida y con soporte -> se actualiza soporte



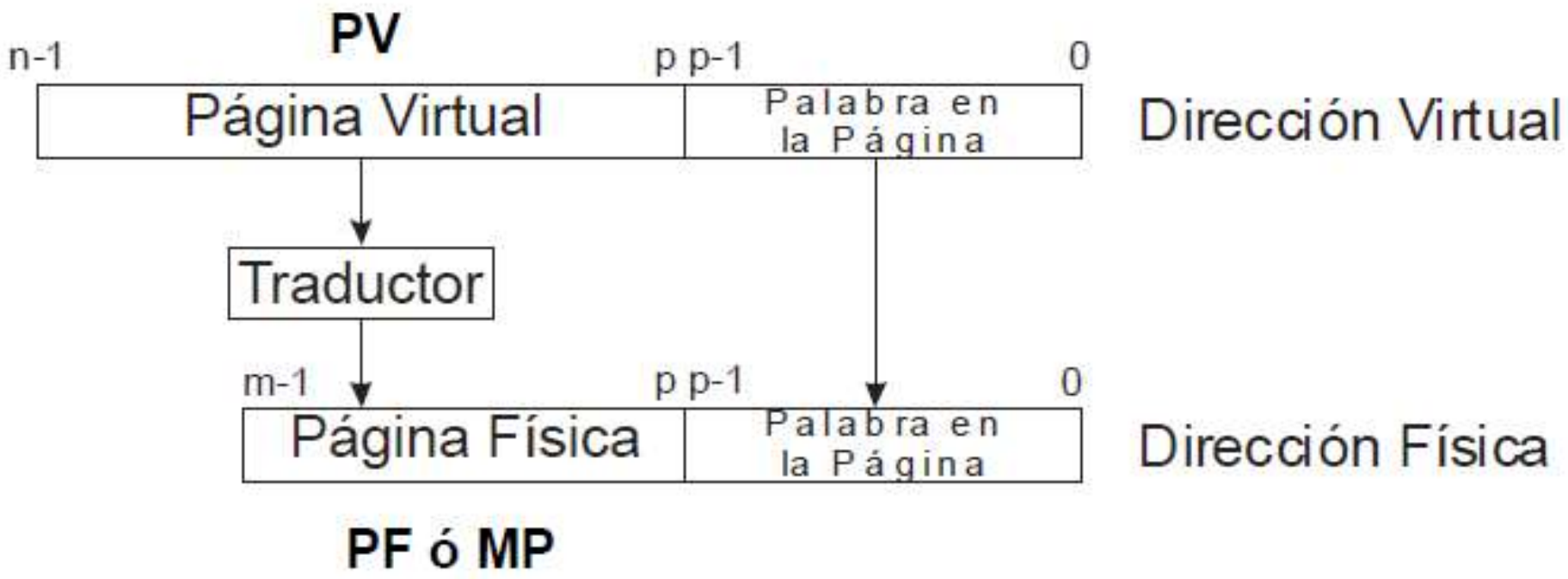
Requisitos de la Memoria Virtual

- Soporte en almacenamiento secundario.
- Espacio virtual mayor que la memoria física => fallos en los accesos a memoria.
- Cuando se produce un fallo de acceso => **tra**p al SO.
- Hay que continuar o reiniciar la instrucción en la que se produjo el fallo de acceso.
  - MOV (R1), (R2) => tres *posibles* fallos de página.
- Esquemas HW de memoria virtual:
  - Paginación.
  - Segmentación.
  - Segmentación paginada.

Paginación


MAPA VIRTUAL
(RESIDENTE EN DISCO)
0
2ᵖ
2·2ᵖ
3·2ᵖ
2ᵖ-1
2·2ᵖ-1
3·2ᵖ-1
A
2ⁿ-1
Páginas
Proyección de página
virtual a memoria física
MEMORIA PRINCIPAL
0
2ᵖ
2·2ᵖ
3·2ᵖ
2ᵖ-1
2·2ᵖ-1
3·2ᵖ-1
A'
2ᵐ-1
Marcos de
página
n > m

Paginación: Mecanismo de Traducción

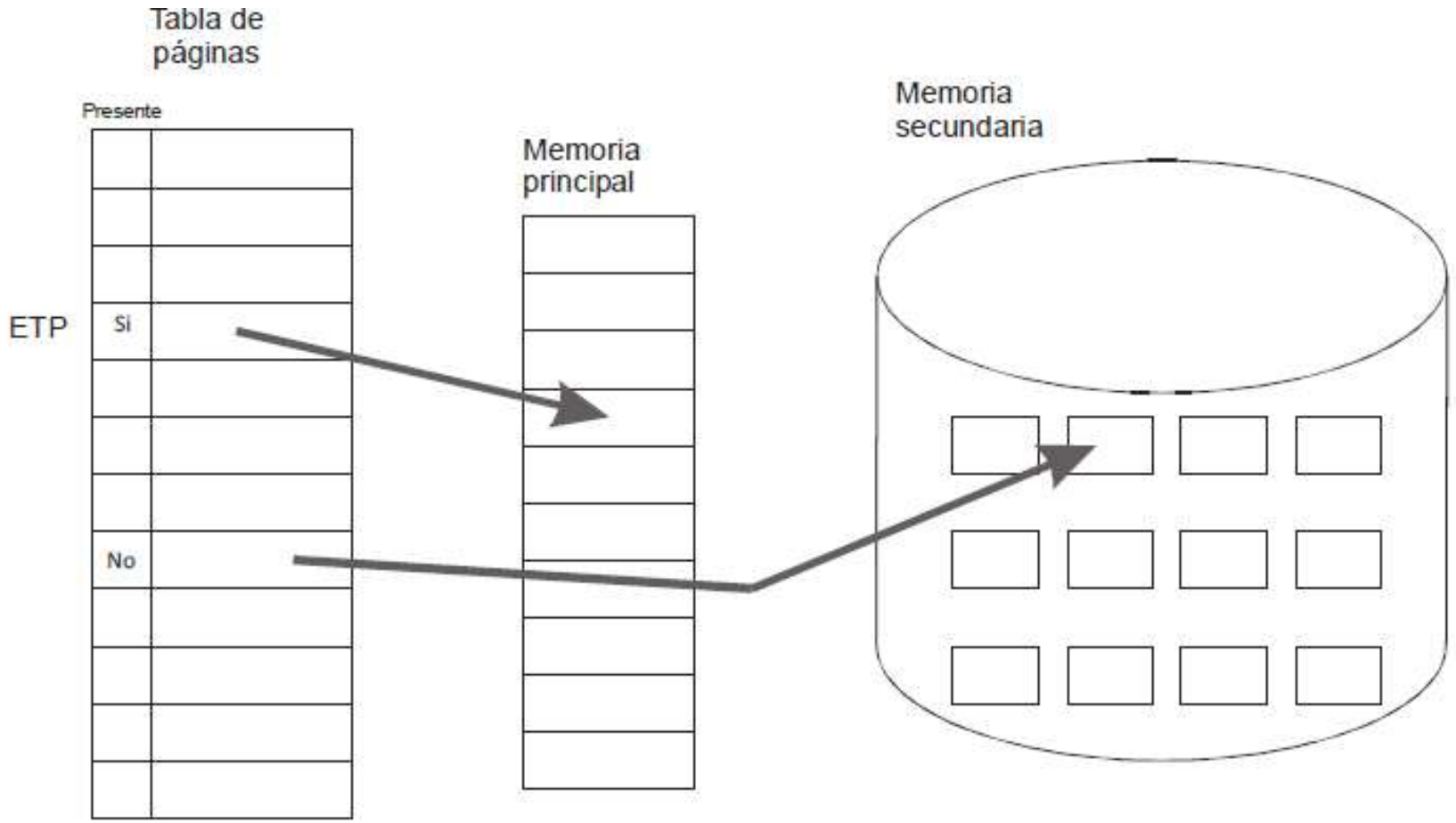


- Necesario una Tabla de Páginas, TP.

Paginación: Tabla de Paginas

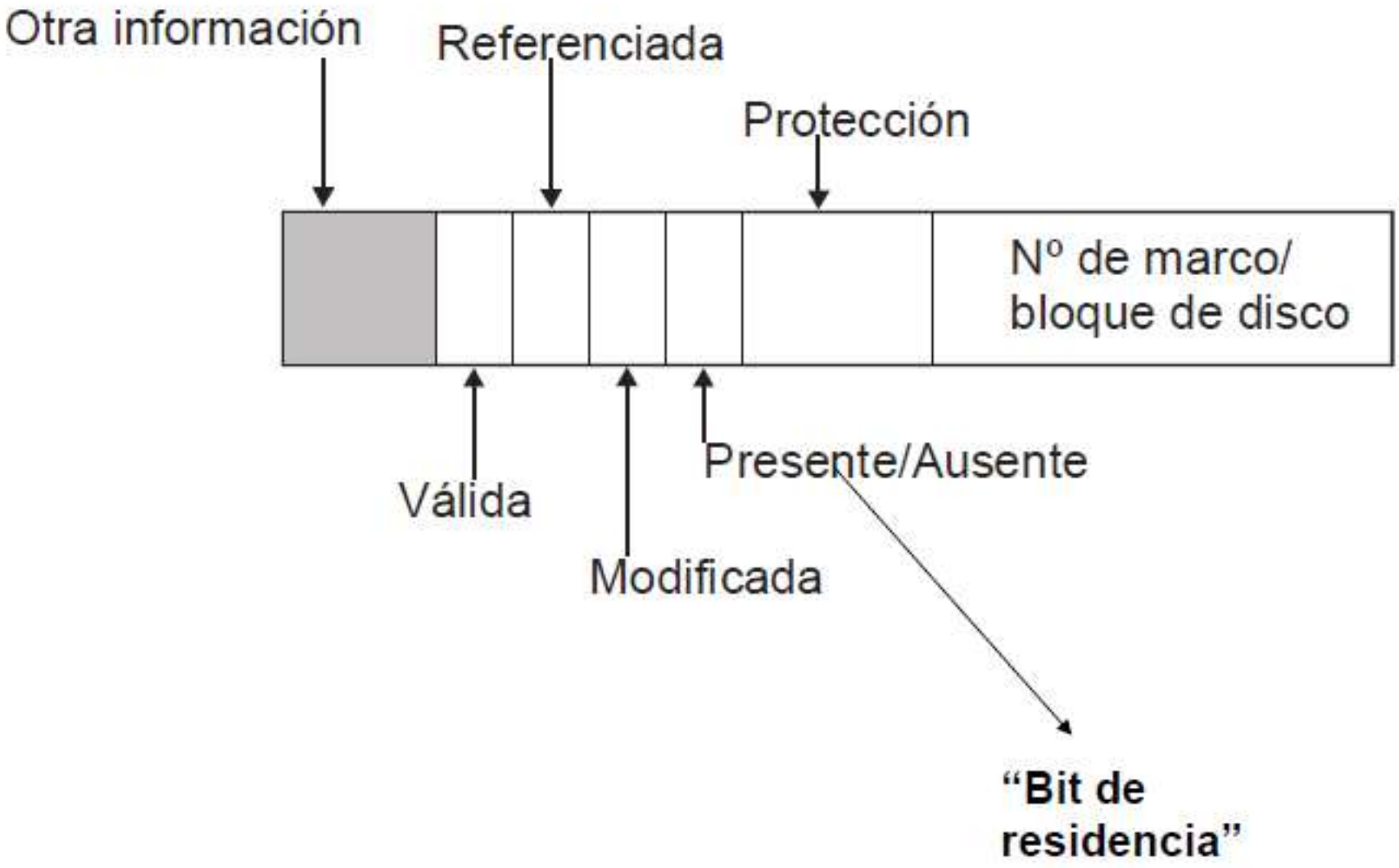
- La información de traducción se organiza en *tablas de páginas*.
- La tabla de páginas está formada por **entradas de tablas de página** (ETP).
- Cada entrada permite traducir una página virtual.
- Permite saber si una página:
  - Está en Memoria => marco de página (MP).
    - Está en almacenamiento secundario => página en disco.

Paginación: Tabla de Paginas



- Si la página no se encuentra en MP => **fallo de página** (*trap* al SO).

Información en una ETP



Información en una ETP

- Otra información:
  - *Copy-on-write*
  - No paginada (fija en memoria física).
  - Cache desactivada.
  - Rellenar a ceros.
  - Rellenar de fichero.

Paginación por Demanda

- Procesos no se carga inicialmente: ninguna página en MP.
- SO: crea previamente la TP: todas ETP a no residentes
- A medida que se producen fallos de página se van subiendo.
- Ventajas:
  - Sólo se cargan en MP las páginas que se referencian.
  - Acelera la carga de un proceso (no se cargan páginas que no se referencian).
  - Mejor uso de la MP.
- Esquema implementado en los SO actuales.
  - UNIX.
  - LINUX.
  - Windows NT y siguientes.

Tratamiento de los fallos de página

- Hw → *trap* "fallo de página".
- UC: Salva estado del proceso actual y bifurcar a la Rut. de Gestión Fallo de Página.
- SO: qué página hace falta => el Hw lo pasa como parámetro.
- SO: el acceso está permitido (consulta TP)?. Si no está permitido, se "mata" al proceso o se envía una señal.
- SO: busca MP libre. Si no hay, se ejecuta el **algoritmo de reemplazo**.
- SO: Si MP "víctima" está "**sucio**" → escribir en disco: activar otro proceso.
- SO:Cuando el MP queda libre, se bloquea en M. Pral, se manda leer: se activa otro proceso.
- SO: Cuando **interrupción de E/S** de la lectura, se actualiza la ETP correspondiente: MP se desbloquea.
- SO: El proceso pasa a **ejecutable.**
- Se **continúa** o se **reinicia** la instrucción que produjo el fallo.

Ejecución de un programa en LINUX

- Ejemplo de código.

```
#include <math.h>
#include <stdio.h>

main()
{
   double x = 30;
   double res;
   void *p;

   p= sbrk(40000);

   res = sin(x);
   printf("res = %f \n", res);
}
```

- Bibliotecas:
  - `libc.so , libm.so`

Strace

- Mandato **strace**
  - `open("/lib/libm.so", O_RDONLY)`
  - `mmap()` => se proyecta el código.
  - `open("/lib/libc.so", O_RDONLY)`
  - `mmap()` => se proyecta el código.
  - Comienza a ejecutar el proceso
- Fichero `/proc/PID/maps`

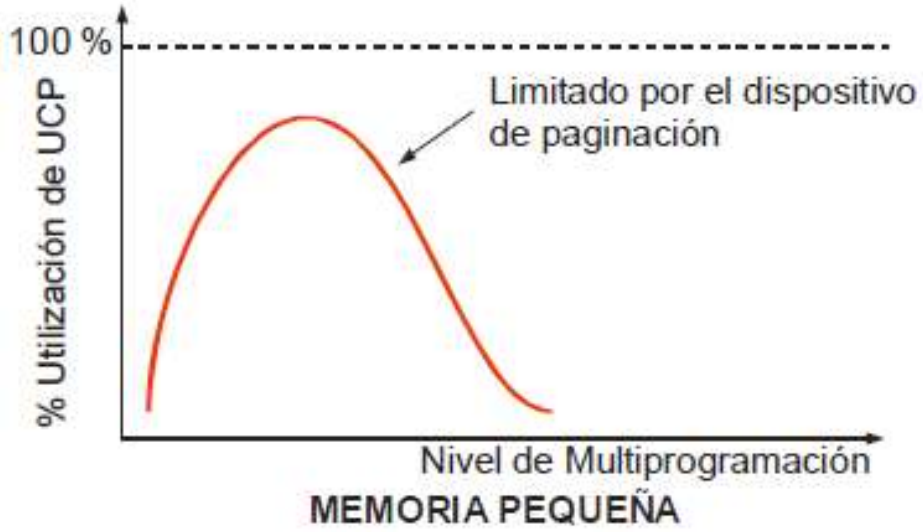
direccion	perms	offset	dev	nodo-i	
08048000-08049000	r-xp	00000000	08:11	630795	Código
08049000-0804a000	rw-p	00000000	08:11	630795	Datos sin VI
0804a000-08054000	rwxp	00000000	00:00	0	Datos dinámicos
40000000-4000a000	r-xp	00000000	08:01	30602	ld.so
4000a000-4000b000	rw-p	00009000	08:01	30602	
40010000-40028000	r-xp	00000000	08:01	30614	libm.so
40028000-40029000	rw-p	00017000	08:01	30614	
40029000-400ba000	r-xp	00000000	08:01	30606	libc.so
400ba000-400c2000	rw-p	00090000	08:01	30606	
400c2000-400ce000	rw-p	00000000	00:00	0	
bfffc000-c0000000	rwxp	ffffd000	00:00	0	Pila

Reemplazo de Paginas

- Objetivo: reducir la tasa de fallos.
- Algoritmos:
  - Óptimo.
  - FIFO.
  - NRU.
  - Segunda oportunidad.
  - Algoritmo del reloj.
  - LRU.
- Buffering de páginas.
- "Demonio" [*daemon*] de paginación.
  - Se activa cuando se necesitan páginas.
  - Objetivo: disponer de páginas libres.

¡¡¡MUY IMPORTANTE!!!

Hiperpaginación [trashing] o “vapuleo”



- Solución: reducir el grado de multiprogramación, suspendiendo uno o más procesos