Administración de Memoria

David Alejandro González Márquez

Clase disponible en: `https://github.com/fokerman/computingSystemsCourse`

Administración de Memoria

Los primeros sistemas eran **monotarea** y la administración de la memoria era muy simple.

Un rango de memoria correspondía al **sistema operativo**

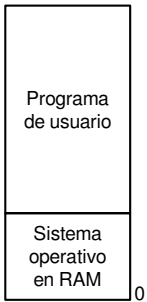
y otro rango se asignaba al **proceso en ejecución**.

Administración de Memoria

Los primeros sistemas eran **monotarea** y la administración de la memoria era muy simple.

Un rango de memoria correspondía al **sistema operativo**

y otro rango se asignaba al **proceso en ejecución**.

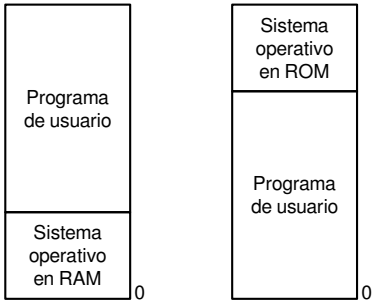


Administración de Memoria

Los primeros sistemas eran **monotarea** y la administración de la memoria era muy simple.

Un rango de memoria correspondía al **sistema operativo**

y otro rango se asignaba al **proceso en ejecución**.

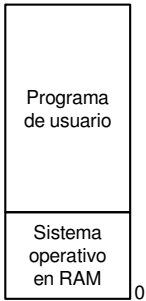


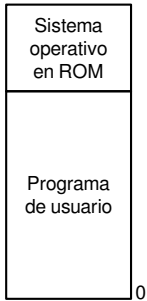
Administración de Memoria

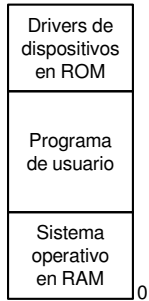
Los primeros sistemas eran **monotarea** y la administración de la memoria era muy simple.

Un rango de memoria correspondía al **sistema operativo**

y otro rango se asignaba al **proceso en ejecución**.







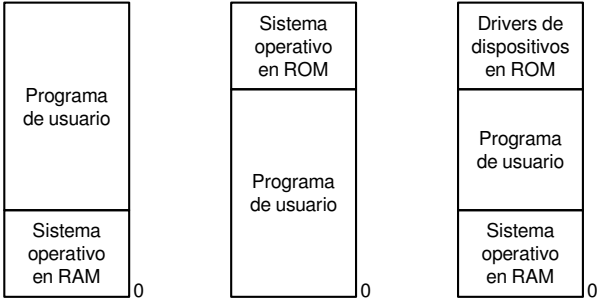
Es posible ubicar al Sistema Operativo en la memoria alta como en la memoria baja. Incluso, podemos tener parte del sistema en ROM mapeado a memoria.

Administración de Memoria

Los primeros sistemas eran **monotarea** y la administración de la memoria era muy simple.

Un rango de memoria correspondía al **sistema operativo**

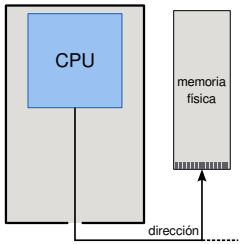
y otro rango se asignaba al **proceso en ejecución**.



Es posible ubicar al Sistema Operativo en la memoria alta como en la memoria baja. Incluso, podemos tener parte del sistema en ROM mapeado a memoria.

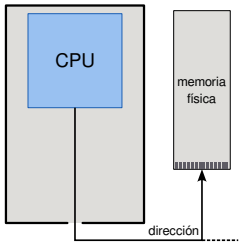
Para soportar múltiples tareas tenemos que contar con mecanismos para que el sistema operativo **administre la memoria** de cada tarea individualmente.

Soporte en hardware



Hasta el momento las direcciones de memoria a las que accede el CPU son las mismas que llegan a la memoria física.
Con este limitado soporte, el sistema operativo solo puede **asignar áreas de memoria** a los distintos procesos.

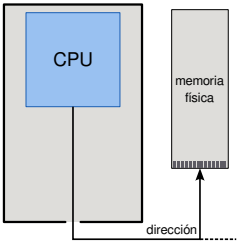
Soporte en hardware



Hasta el momento las direcciones de memoria a las que accede el CPU son las mismas que llegan a la memoria física.
Con este limitado soporte, el sistema operativo solo puede **asignar áreas de memoria** a los distintos procesos.

Problema de **reubicación**:
No conocemos donde vamos a cargar el programa en tiempo de generación de código.
Problema de **protección**:
No podemos impedir que un proceso escriba la memoria de otro proceso.

Soporte en hardware

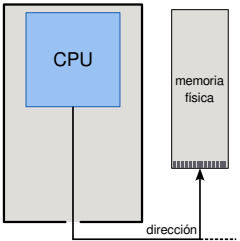


Hasta el momento las direcciones de memoria a las que accede el CPU son las mismas que llegan a la memoria física.
Con este limitado soporte, el sistema operativo solo puede **asignar áreas de memoria** a los distintos procesos.

Problema de **reubicación**:
No conocemos donde vamos a cargar el programa en tiempo de generación de código.
Problema de **protección**:
No podemos impedir que un proceso escriba la memoria de otro proceso.

Este tipo de soporte se limita a sistemas embebidos.

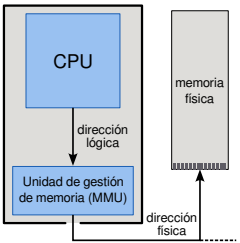
Soporte en hardware



Hasta el momento las direcciones de memoria a las que accede el CPU son las mismas que llegan a la memoria física.
Con este limitado soporte, el sistema operativo solo puede **asignar áreas de memoria** a los distintos procesos.

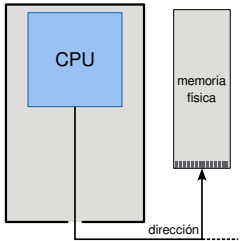
Problema de **reubicación**:
No conocemos donde vamos a cargar el programa en tiempo de generación de código.
Problema de **protección**:
No podemos impedir que un proceso escriba la memoria de otro proceso.

Este tipo de soporte se limita a sistemas embebidos.



Los procesadores avanzados cuentan con Unidad de Gestión de Memoria (MMU).
Esta se encarga de **traducir** las direcciones.
Las direcciones que ve el proceso se denominan **lógicas**.
Las direcciones que recibe la memoria se denominan **físicas**.

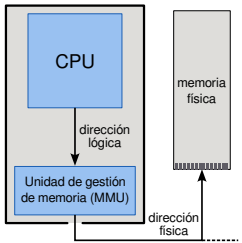
Soporte en hardware



Hasta el momento las direcciones de memoria a las que accede el CPU son las mismas que llegan a la memoria física.
Con este limitado soporte, el sistema operativo solo puede **asignar áreas de memoria** a los distintos procesos.

Problema de **reubicación**:
No conocemos donde vamos a cargar el programa en tiempo de generación de código.
Problema de **protección**:
No podemos impedir que un proceso escriba la memoria de otro proceso.

Este tipo de soporte se limita a sistemas embebidos.

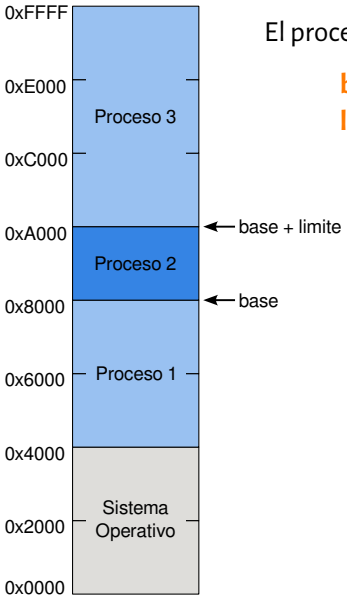


Los procesadores avanzados cuentan con Unidad de Gestión de Memoria (MMU).
Esta se encarga de **traducir** las direcciones.
Las direcciones que ve el proceso se denominan **lógicas**.
Las direcciones que recibe la memoria se denominan **físicas**.

Vamos a ver dos mecanismos de hardware:

Segmentación o *relocation register* y
Paginación

Gestión de memoria por Segmentos (*relocation register*)

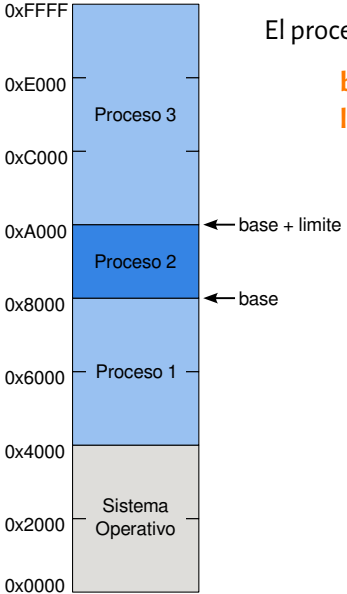


El procesador provee un mecanismo que consta de dos registros.

base: Indica donde comienza la memoria del proceso.

límite: Indica el tamaño de la memoria de proceso.

Gestión de memoria por Segmentos (*relocation register*)

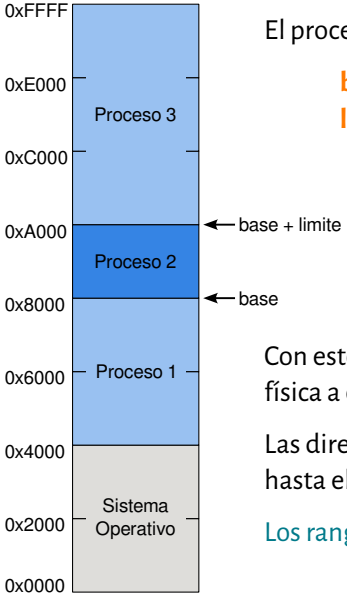


El procesador provee un mecanismo que consta de dos registros.

base: Indica donde comienza la memoria del proceso.
límite: Indica el tamaño de la memoria de proceso.

En el ejemplo el registro base vale `0x8000` y límite `0x2000`.
Las direcciones lógicas del Proceso 2
comienzan en `0x0000` y terminan en `0x1FFF`.
Estas se **mapean** a la memoria física entre `0x8000` y `0xBFFF`.

Gestión de memoria por Segmentos (*relocation register*)



El procesador provee un mecanismo que consta de dos registros.

base: Indica donde comienza la memoria del proceso.
límite: Indica el tamaño de la memoria de proceso.

En el ejemplo el registro base vale `0x8000` y límite `0x2000`.
Las direcciones lógicas del Proceso 2
comienzan en `0x0000` y terminan en `0x1FFF`.
Estas se **mapean** a la memoria física entre `0x8000` y `0xBFFF`.

Con este mecanismo podemos asignar cualquier rango de memoria física a cualquier proceso.

Las direcciones lógicas siempre comenzarán en cero y se extenderán hasta el tamaño de la memoria del proceso.

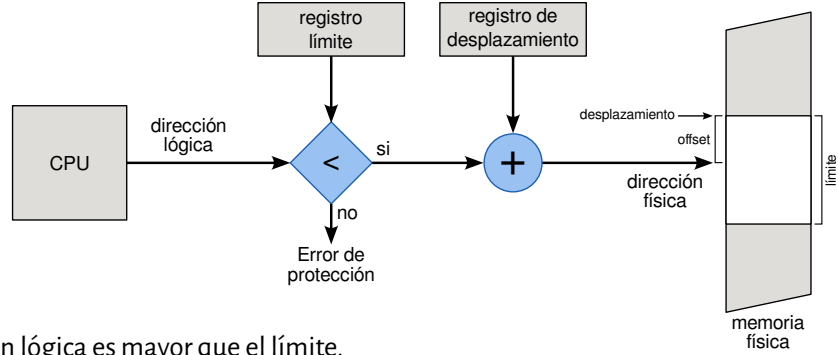
Los rangos de memoria asignados deben ser contiguos en memoria.

Gestión de memoria por Segmentación (*relocation register*)

El funcionamiento consiste en verificar que la dirección lógica que **no exceda el límite**.

Para luego **suma el registro de desplazamiento** a la dirección lógica,

obteniendo así la **dirección física**.

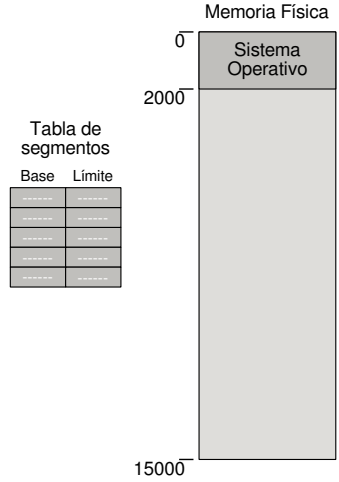


Si la dirección lógica es mayor que el límite,
entonces se genera un error de protección.

Gestión de memoria por Segmentación

Para implementar múltiples procesos con este mecanismo, se tiene una tabla de segmentos o de registros de desplazamiento y registros de límite.

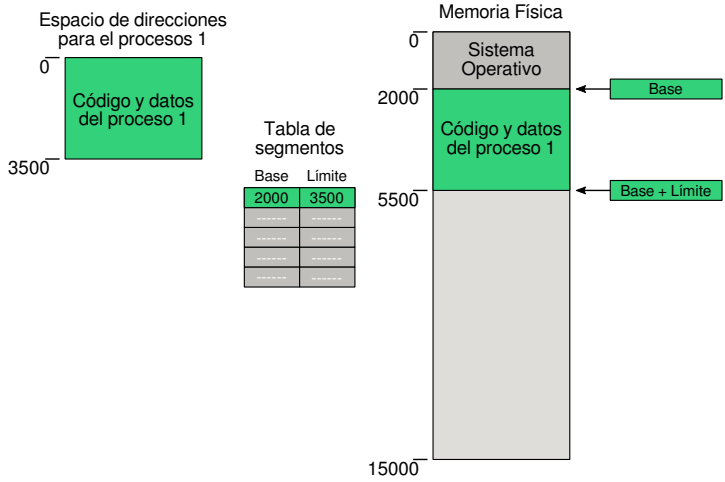
Esta tabla es administrada por el sistema operativo y carga los registros cada vez que una tarea distinta es ejecutada.



Gestión de memoria por Segmentación

Para implementar múltiples procesos con este mecanismo, se tiene una tabla de segmentos o de registros de desplazamiento y registros de límite.

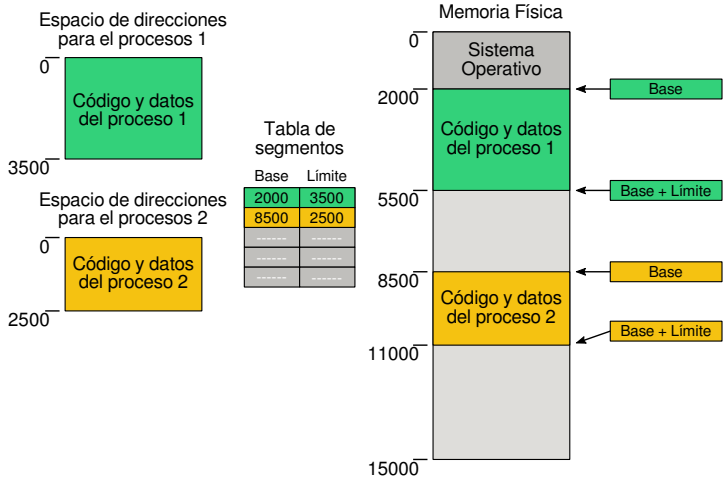
Esta tabla es administrada por el sistema operativo y carga los registros cada vez que una tarea distinta es ejecutada.



Gestión de memoria por Segmentación

Para implementar múltiples procesos con este mecanismo, se tiene una tabla de segmentos o de registros de desplazamiento y registros de límite.

Esta tabla es administrada por el sistema operativo y carga los registros cada vez que una tarea distinta es ejecutada.



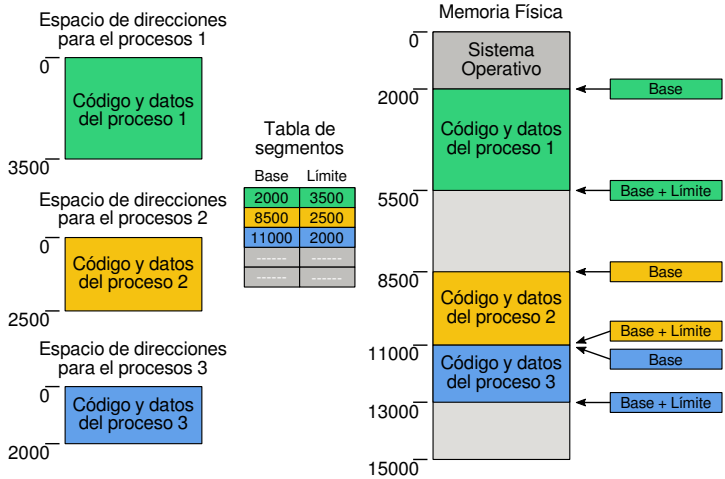
Gestión de memoria por Segmentación

Para implementar múltiples procesos con este mecanismo, se tiene una tabla de segmentos o de registros de desplazamiento y registros de límite.

Esta tabla es administrada por el sistema operativo y carga los registros cada vez que una tarea distinta es ejecutada.

¿Qué sucede si un proceso necesita más memoria?

¿Cómo se utiliza la memoria libre entre procesos?

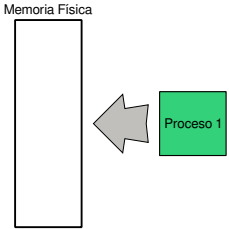


Gestión de memoria por Segmentación

Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:

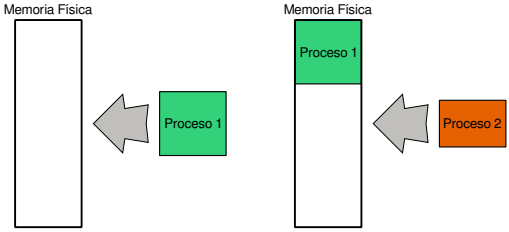
Gestión de memoria por Segmentación

Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:



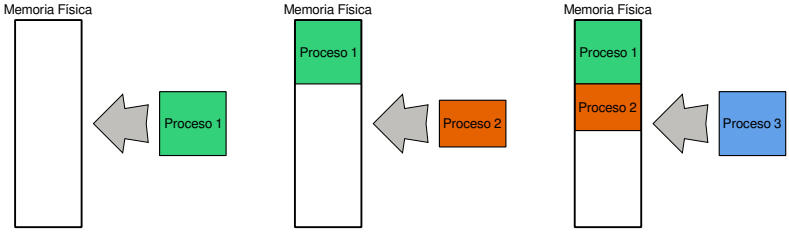
Gestión de memoria por Segmentación

Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:



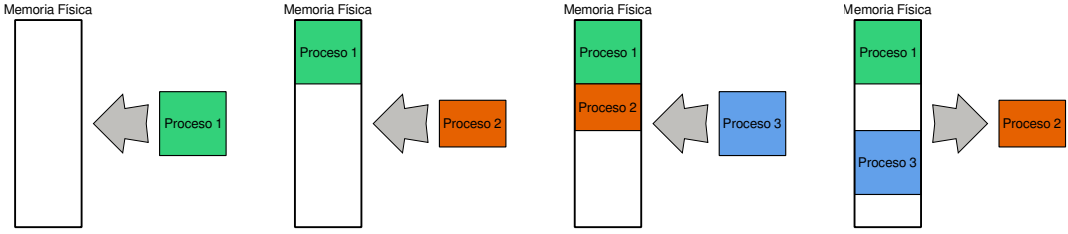
Gestión de memoria por Segmentación

Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:



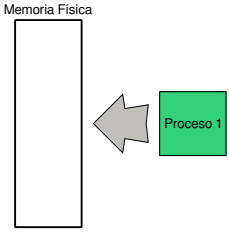
Gestión de memoria por Segmentación

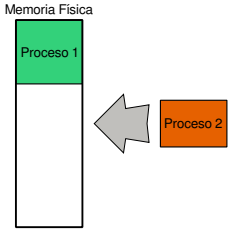
Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:

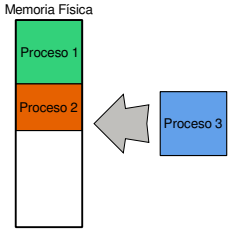


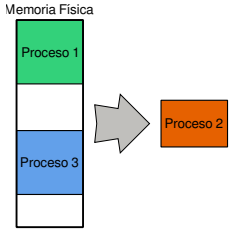
Gestión de memoria por Segmentación

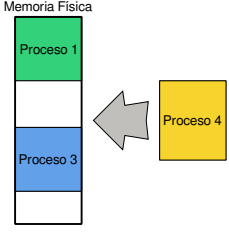
Con este mecanismo podemos ir asignando rangos de direccionamiento a procesos:











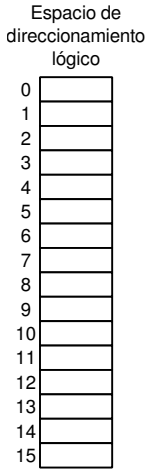
A pesar de tener espacio suficiente, no podemos cargar el nuevo proceso.
Decimos entonces que nuestra memoria está **fragmentada**.
Se puede dar:

- **fragmentación externa**: Cuando se da entre memoria de distintos procesos.
- **fragmentación interna**: Cuando se da dentro de un proceso.

Podemos solucionar este problema **compactando la memoria**.
Pero no resulta una solución practica por su **gran costo temporal**.

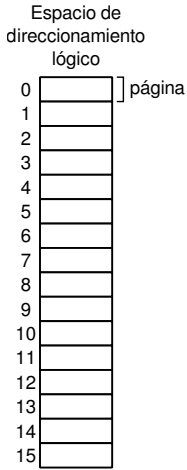
Gestión de memoria por Paginación

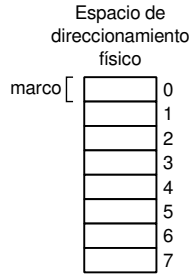
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.



Gestión de memoria por Paginación

El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.

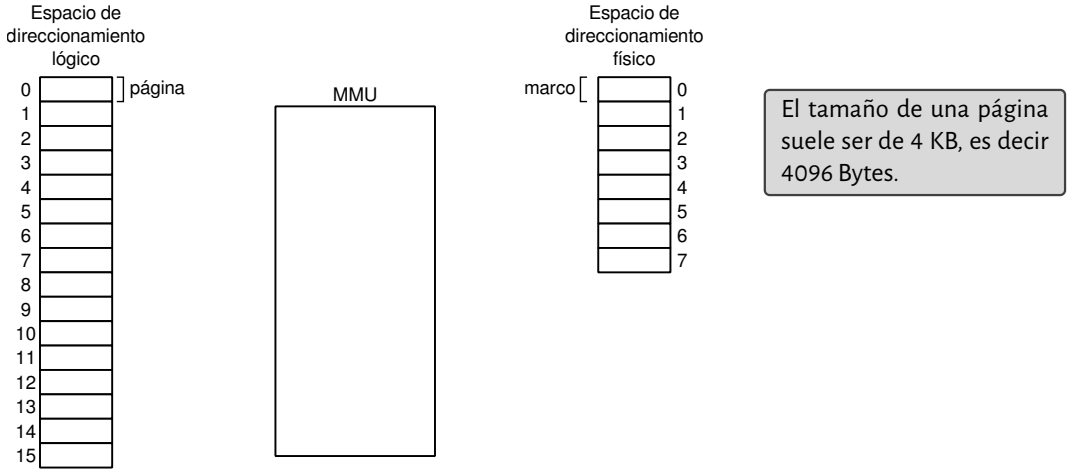




El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

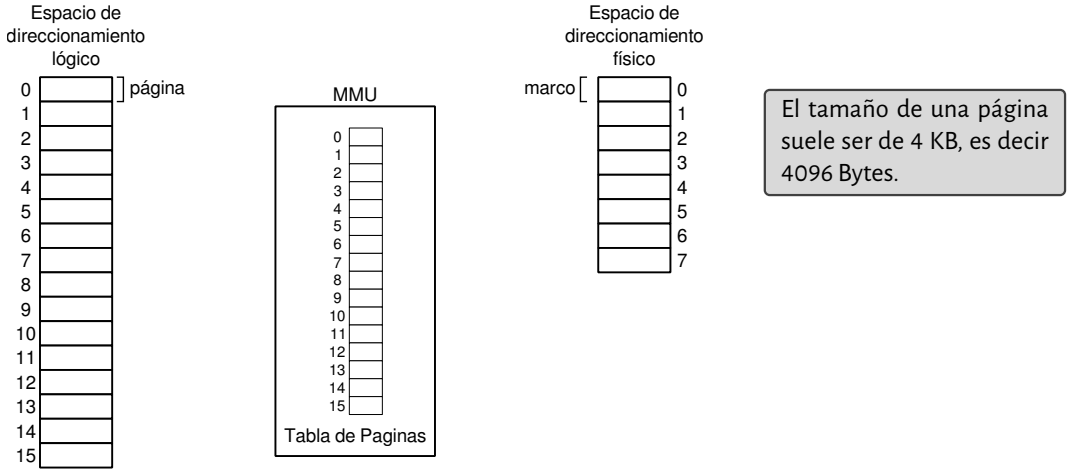
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

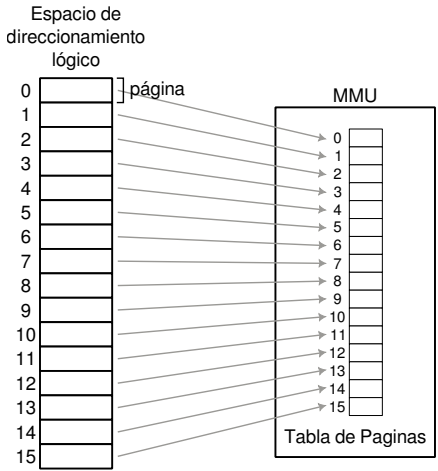
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

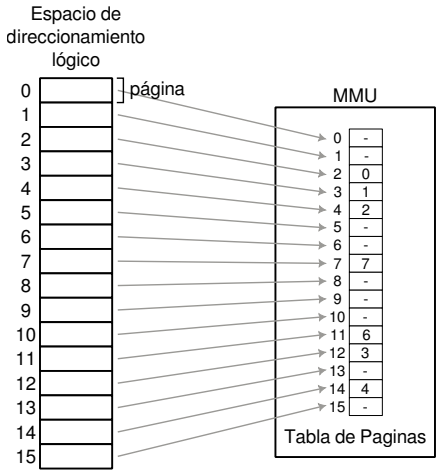
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

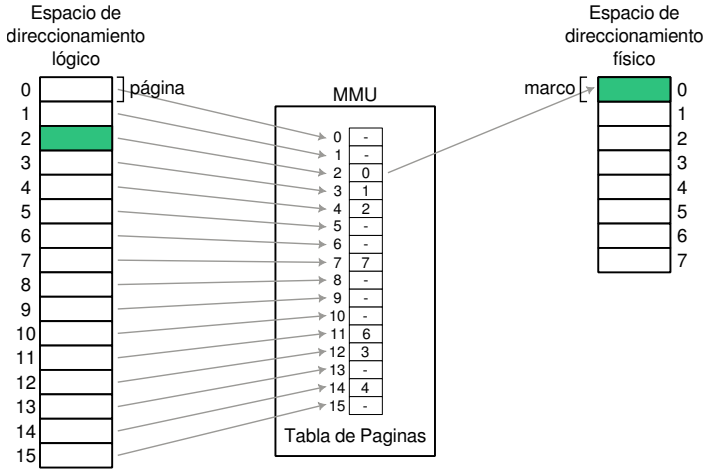
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

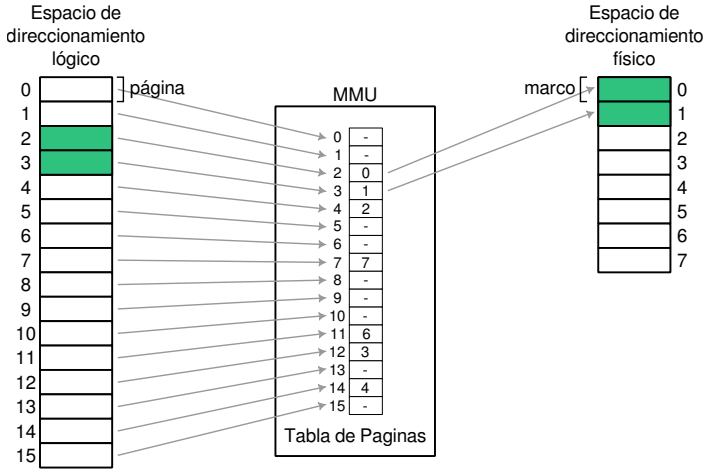
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

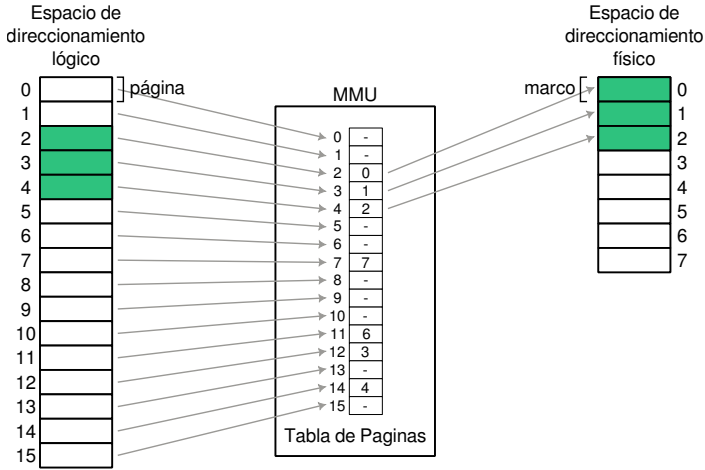
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

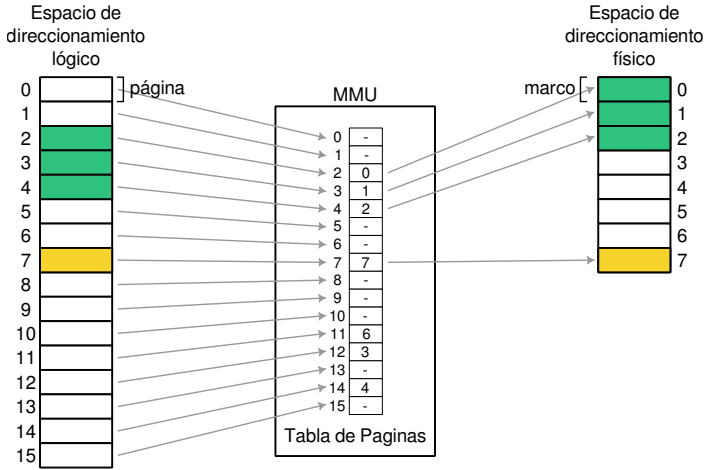
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

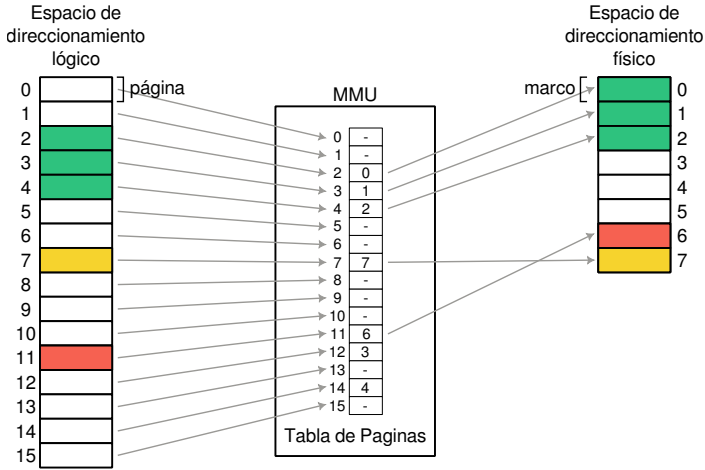
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

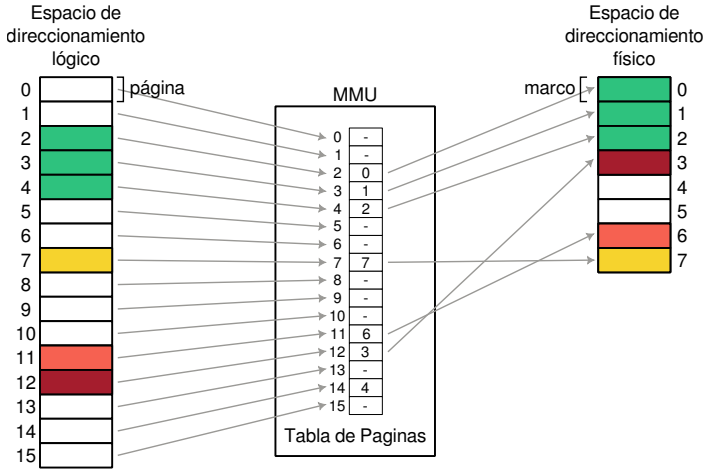
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

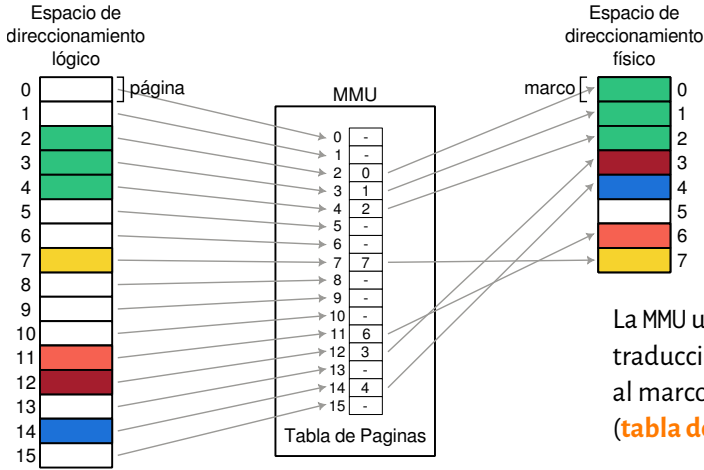
El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.



El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

Gestión de memoria por Paginación

El espacio de direccionamiento lógico se divide en **paginas** de tamaño fijo.
El espacio de direccionamiento físico se divide en **marcos** o **frames** del tamaño de las paginas.

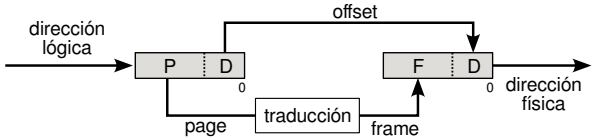


El tamaño de una página suele ser de 4 KB, es decir 4096 Bytes.

La MMU utiliza una tabla que contiene las traducciones para llegar de una página, al marco de página correspondiente (**tabla de páginas**).

Gestión de memoria por Paginación

En paginación, las direcciones lógicas se dividen en dos partes, el **índice de la página** (*page*) y el **desplazamiento** (*offset*).



La traducción de **page** a **frame** se realiza por medio de la tabla de páginas, la cual se encuentra **ubicada en memoria**, y es accedida por el procesador para todos los accesos a memoria.

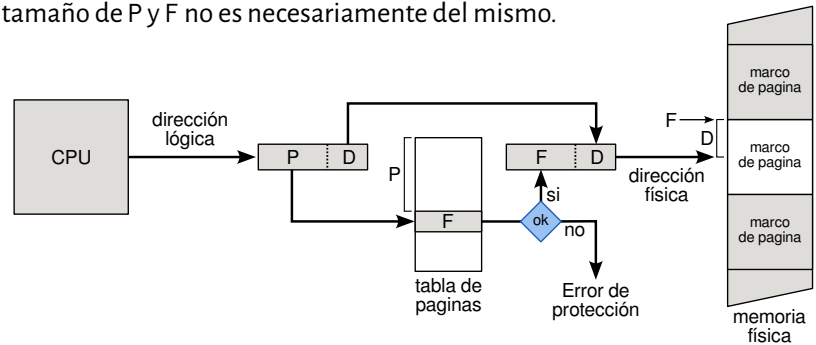
Es decir, por cada dirección de memoria que el proceso quiera leer o escribir, se debe utilizar la máquinaria de la MMU para traduccir la dirección lógica que ve el proceso a la dirección física que requiere el sistema.

Gestión de memoria por Paginación

Se utiliza como **desplazamiento** dentro de la tabla de páginas el valor de P (*page*).

Dentro de la tabla se toma el valor F (*frame*) y se construye la **dirección física**.

El tamaño de P y F no es necesariamente del mismo.

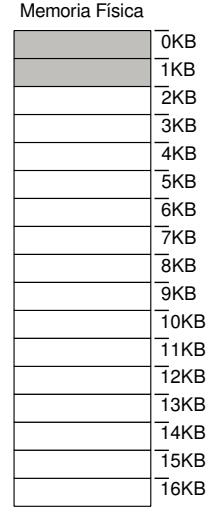


Si dentro de la tabla, no se encuentra un valor valido, se genera un error de protección.
Este error se conoce como **page-fault**.

Gestión de memoria por Paginación

Cada proceso tendrá su propia tabla de páginas en memoria.

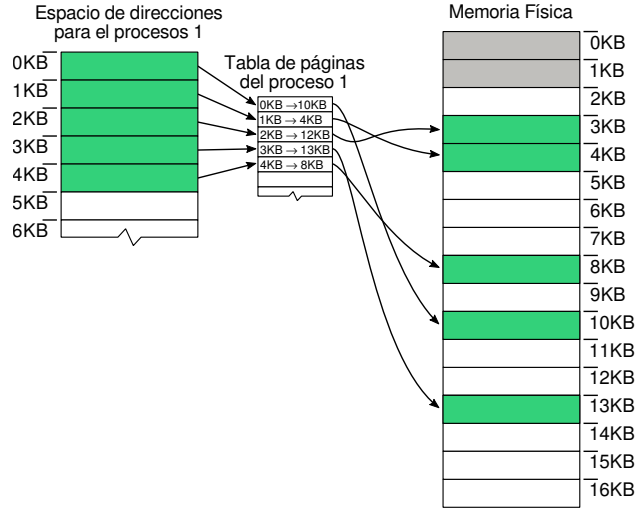
Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.



Gestión de memoria por Paginación

Cada proceso tendrá su propia tabla de páginas en memoria.

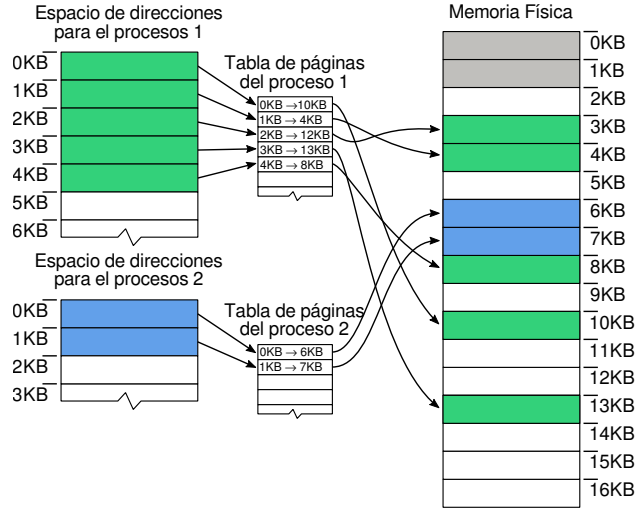
Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.



Gestión de memoria por Paginación

Cada proceso tendrá su propia tabla de páginas en memoria.

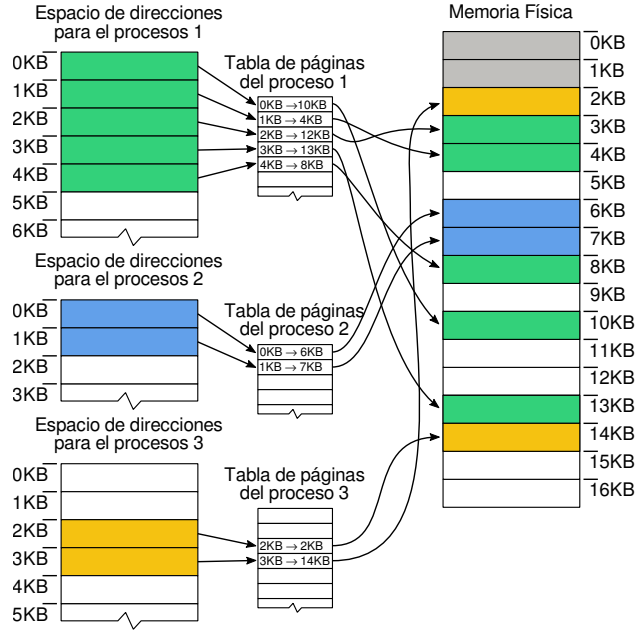
Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.



Gestión de memoria por Paginación

Cada proceso tendrá su propia tabla de páginas en memoria.

Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.

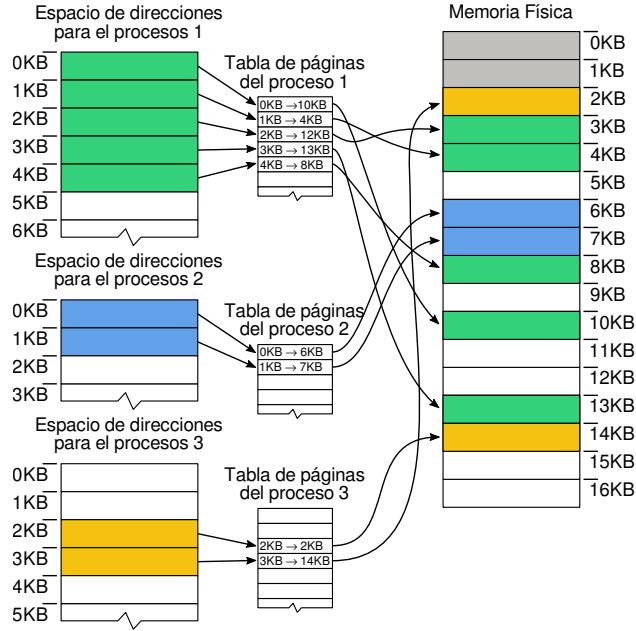


Gestión de memoria por Paginación

Cada proceso tendrá su propia tabla de páginas en memoria.

Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.

El Sistema Operativo utilizará información adicional para mantener registro de que páginas de memoria están asignadas realmente y cuales están solicitadas pero aun no asignadas.



Gestión de memoria por Paginación

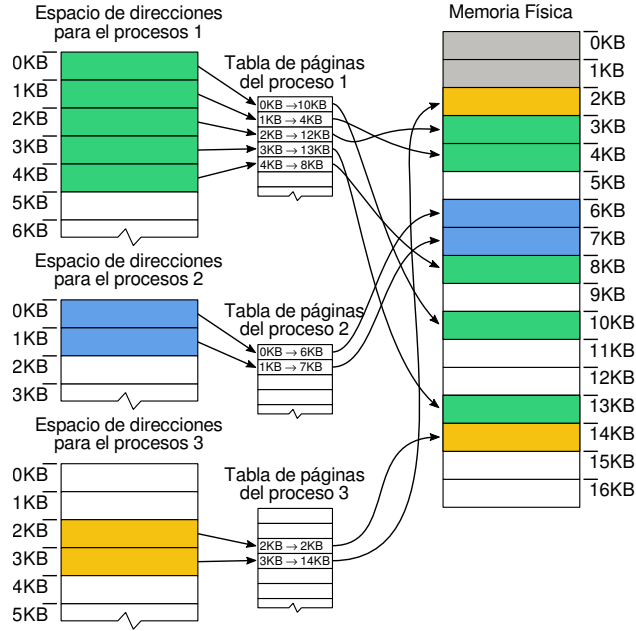
Cada proceso tendrá su propia tabla de páginas en memoria.

Asignando así páginas lógicas a marcos de página físicos según requiera cada proceso.

El Sistema Operativo utilizará información adicional para mantener registro de que páginas de memoria están asignadas realmente y cuales están solicitadas pero aun no asignadas.

Si bien este mecanismo es más complejo a nivel de hardware, resulta más flexible y simple de implementar a nivel de software.

Evitando además la fragmentación externa.

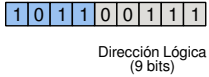


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

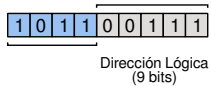


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

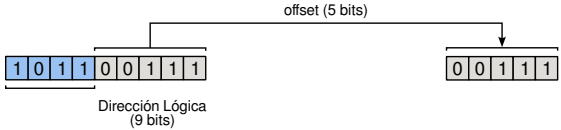


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

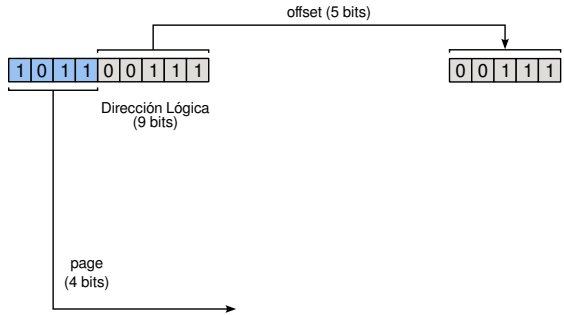


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

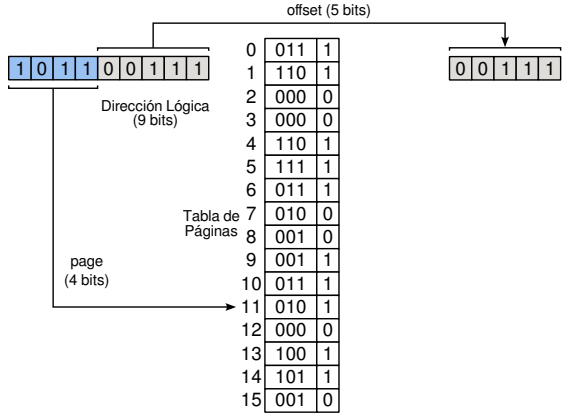


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

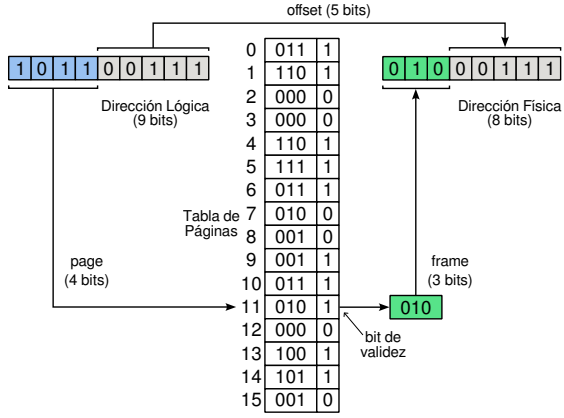


Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.



Gestión de memoria por Paginación

Cada entrada de una tabla de paginación consiste en un campo ***frame*** y bits de **atributos**.

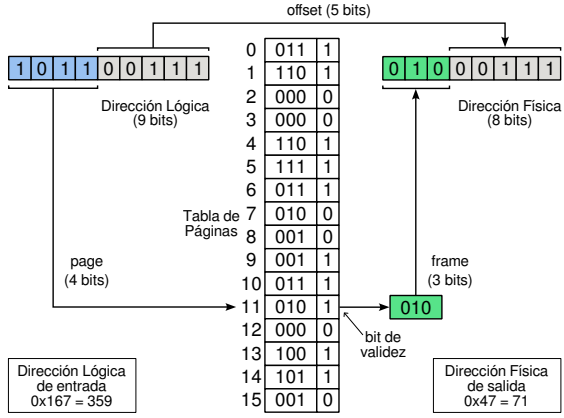
Los bits de atributos pueden ser de **protección**, de **permisos**, de **estado** y de **validez**.

En el ejemplo, la tabla tiene 3 bits de *frame* y 1 bit de validez. Si indica 1, el campo *frame* es válido.

La dirección lógica tiene 9 bits, de los cuales 5 corresponden al offset, dando un tamaño de página de 32 bytes.

Cada entrada de la tabla mínimamente ocupa un byte, por lo tanto nuestra tabla ocupa 16 bytes.

El espacio direccionable lógico es de 512 bytes, mientras que físicamente se cuenta con 256 bytes de memoria.



Gestión de memoria por Paginación

Si tenemos una memoria muy grande,
la tabla de paginas también será grande.

Si tenemos varios procesos con tablas grandes en memoria, estaremos invirtiendo mucho espacio en la administración de la memoria.

Para enfrentar este problema se construyen las tablas **multinivel**.

Gestión de memoria por Paginación

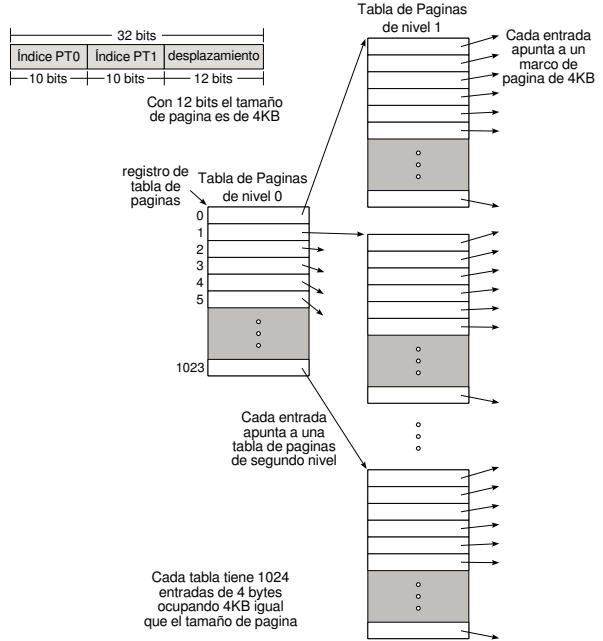
Si tenemos una memoria muy grande, **la tabla de paginas también será grande**.

Si tenemos varios procesos con tablas grandes en memoria, estaremos invirtiendo mucho espacio en la administración de la memoria.

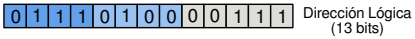
Para enfrentar este problema se construyen las tablas **multinivel**.

Estas consisten en varios niveles escalonados de tablas, para las cuales solo existirán punteros validos a las siguientes tablas si estas son necesarias.

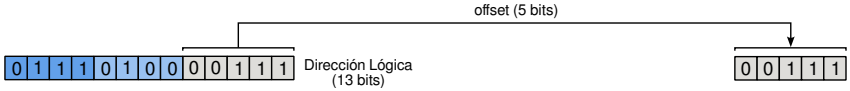
Permitiendo así ahorrar espacio y tener en memoria solamente la información útil.



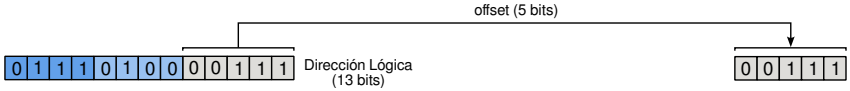
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles

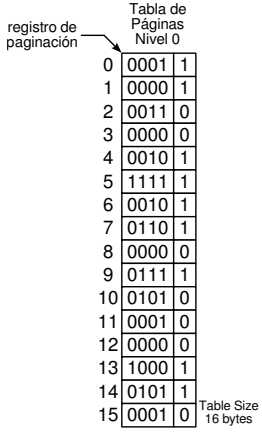


Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles

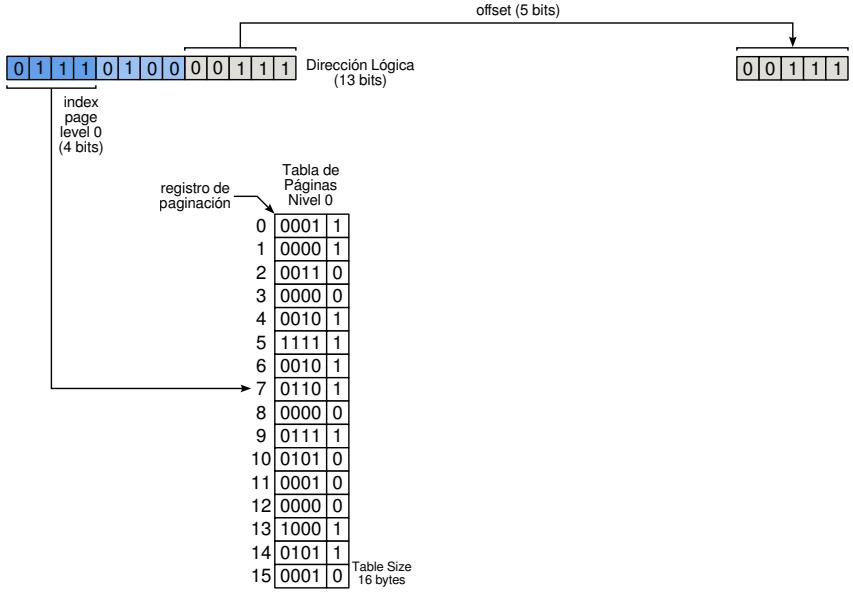


Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles

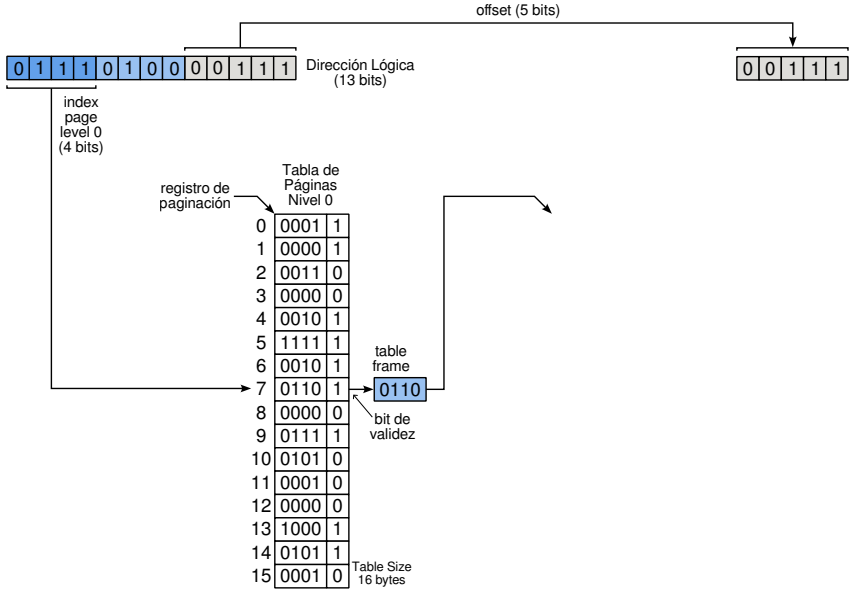




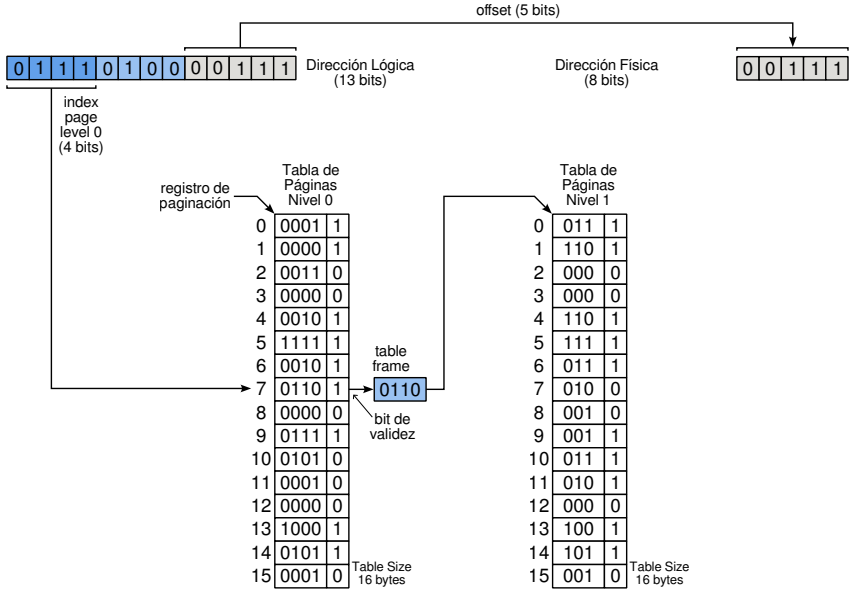
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



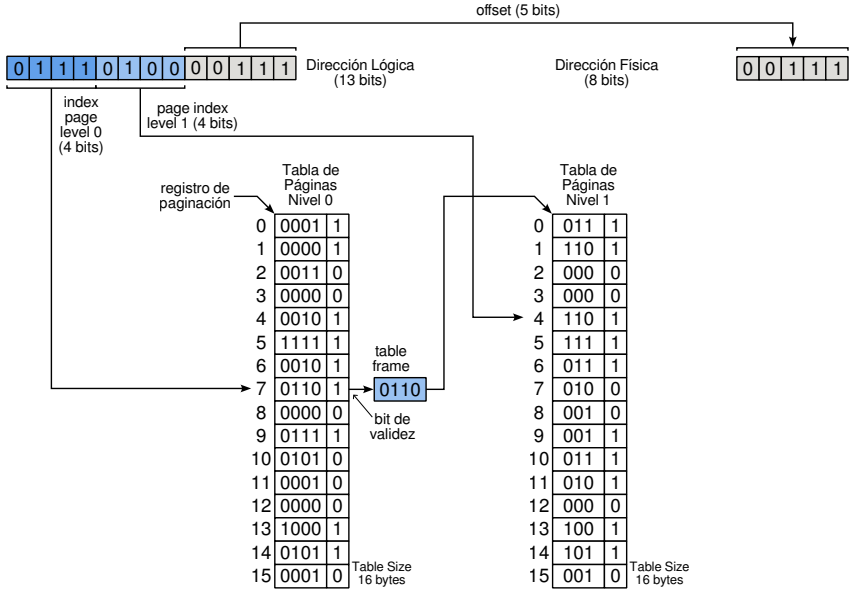
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



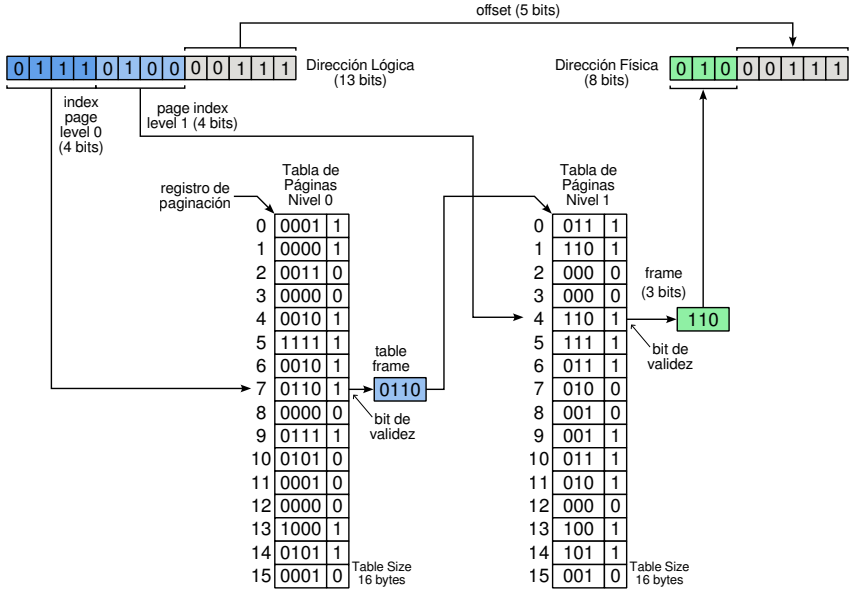
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



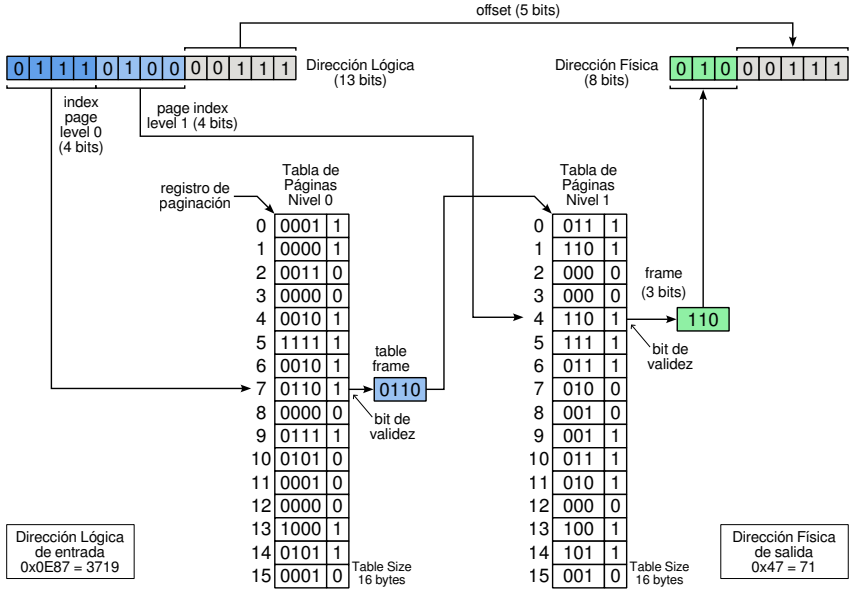
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



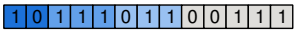
Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



Gestión de memoria por Paginación: Ejemplo de tabla de dos niveles



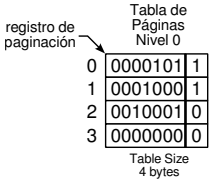
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



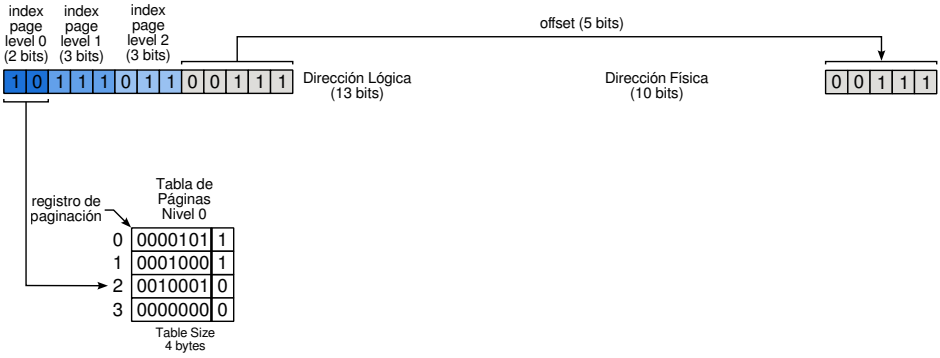
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles

Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles

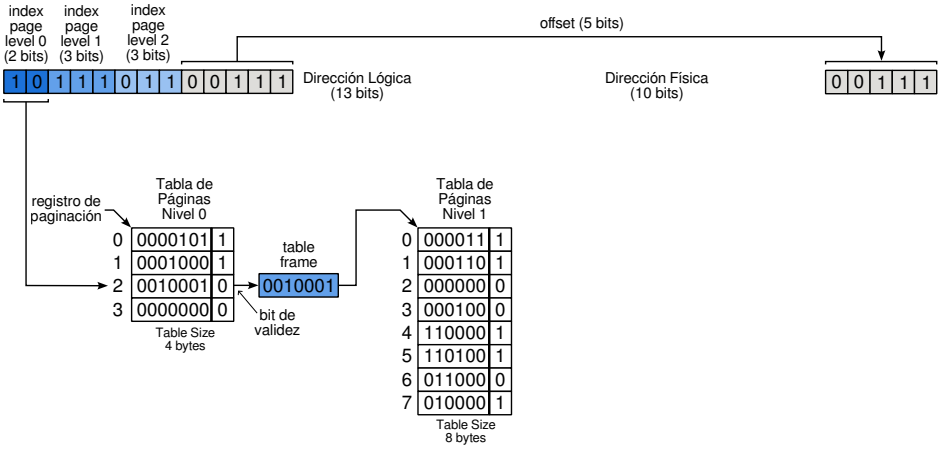
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



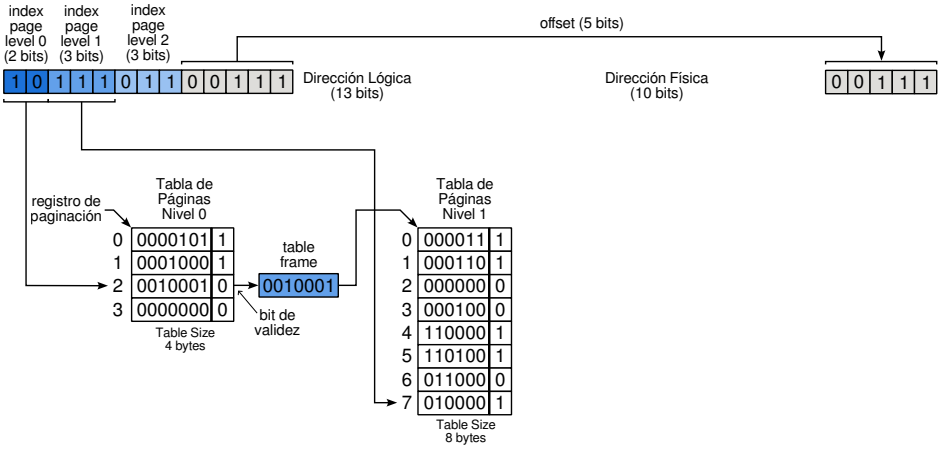
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



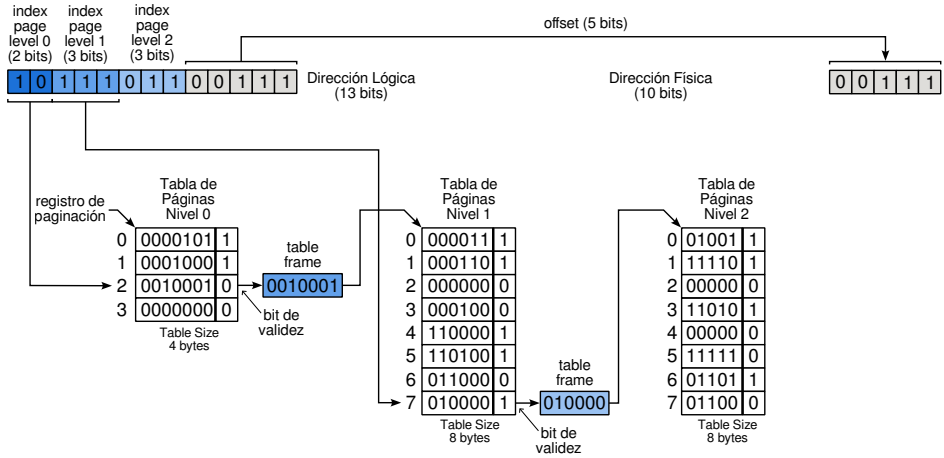
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



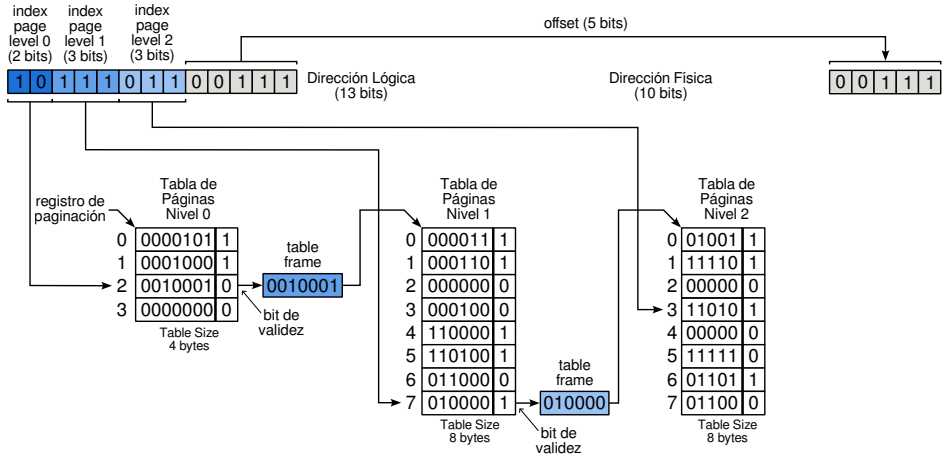
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



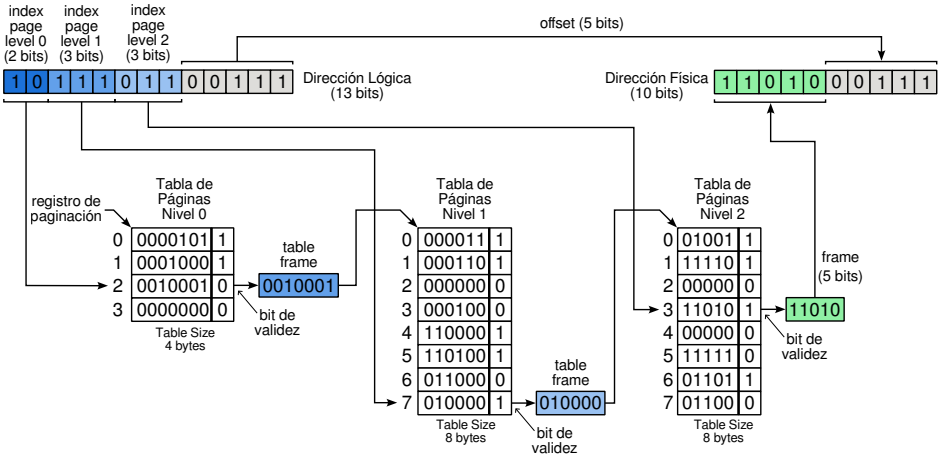
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



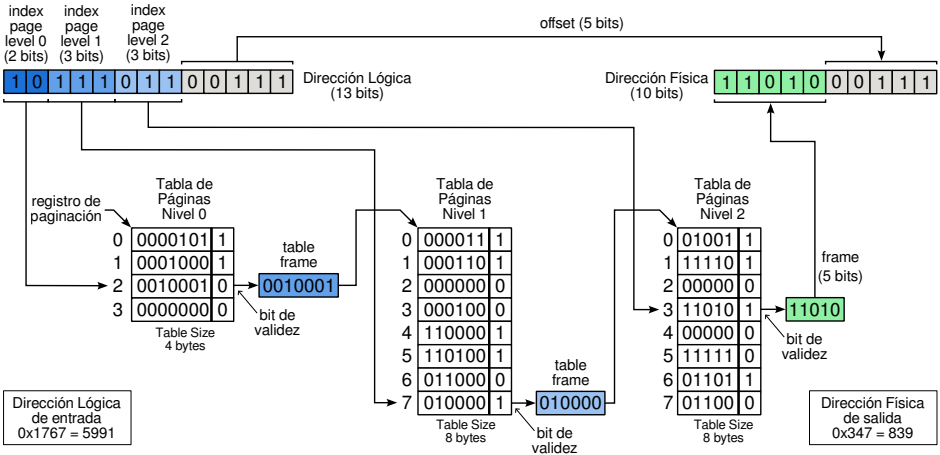
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



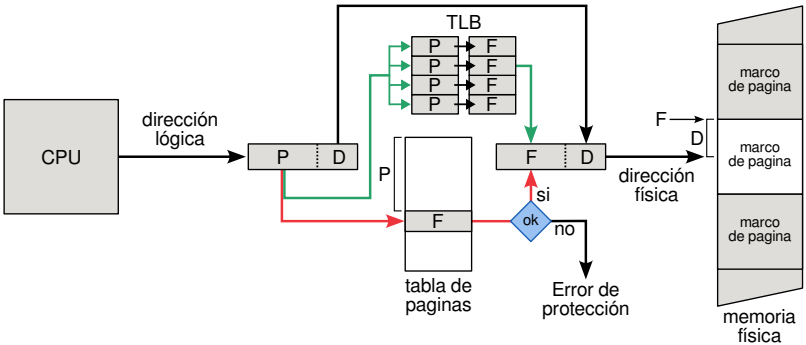
Gestión de memoria por Paginación: Ejemplo de tabla de tres niveles



Gestión de memoria por Paginación: TLB

La traducción de una dirección de memoria lleva tiempo.
Se requiere recorrer todo el árbol de paginación hasta llegar al marco de pagina buscado.

Para solucionar esto implementamos una memoria caché: **TLB (*Translation Lookaside Buffers*)**



Su tarea es encontrar el *frame* a partir de la *page*.
Su implementación es una cache completamente asociativa.

Estructuras de datos para la administración de memoria

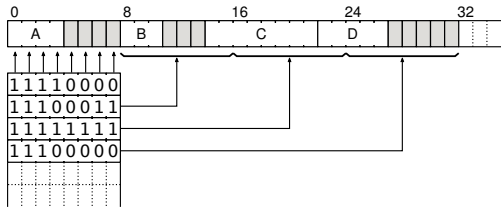
Para que es sistema operativo lleve registro de que áreas de memoria están en uso y cuales no, se utilizan estructuras de datos particulares.

Estructuras de datos para la administración de memoria

Para que es sistema operativo lleve registro de que áreas de memoria están en uso y cuales no, se utilizan estructuras de datos particulares.

En un **mapa de bits**, cada bit indica si un determinado rango de memoria esta en uso o esta libre.

Estructura de Mapa de bits



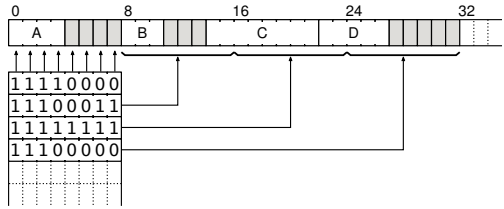
Estructuras de datos para la administración de memoria

Para que es sistema operativo lleve registro de que áreas de memoria están en uso y cuales no, se utilizan estructuras de datos particulares.

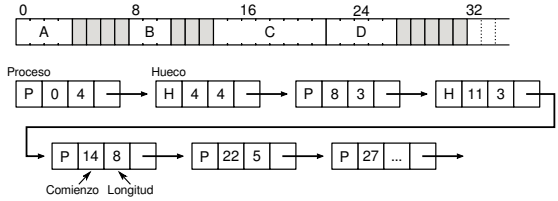
En un **mapa de bits**, cada bit indica si un determinado rango de memoria esta en uso o esta libre.

Una **lista enlazada**, deja registro en cada nodo, si se trata de un espacio libre o ocupado.
En esta estructura pueden existir nodos contiguos con áreas ocupadas, pero no libres, ya que estos últimos se colapsan en uno solo.

Estructura de Mapa de bits



Estructura de Lista enlazada



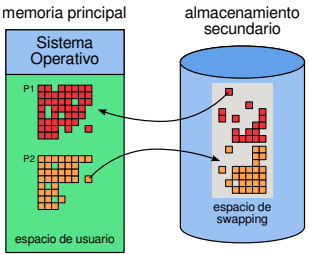
Memoria Virtual

Utilizando los mecanismos que provee el procesar para administrar memoria, podemos asignar un espacio de memoria independiente a cada proceso. Sin embargo, la memoria física es limitada, y no podemos tener todos los procesos simultaneamente en memoria.

Memoria Virtual

Utilizando los mecanismos que provee el procesar para administrar memoria, podemos asignar un espacio de memoria independiente a cada proceso. Sin embargo, la memoria física es limitada, y no podemos tener todos los procesos simultaneamente en memoria.

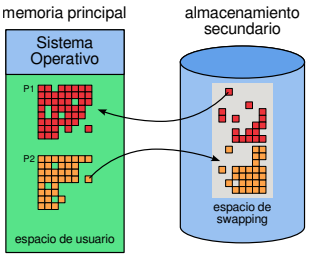
Para esto existen los **mecanismos de swapping**, que permiten guardar temporariamente en almacenamiento secundario las paginas poco utilizadas, mientras que las paginas de más demandadas están en memoria.



Memoria Virtual

Utilizando los mecanismos que provee el procesar para administrar memoria, podemos asignar un espacio de memoria independiente a cada proceso. Sin embargo, la memoria física es limitada, y no podemos tener todos los procesos simultaneamente en memoria.

Para esto existen los **mecanismos de swapping**, que permiten guardar temporariamente en almacenamiento secundario las paginas poco utilizadas, mientras que las paginas de más demandadas están en memoria.

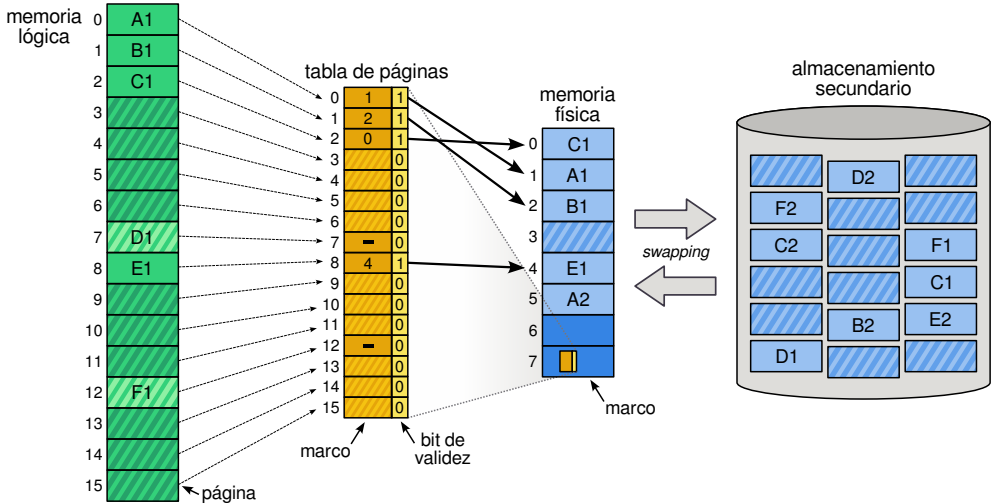


Conceptos clave en memoria virtual:

- Las direcciones lógicas o virtuales son las que ve el proceso en ejecución, y viven solamente dentro del procesador.
- Las direcciones físicas son las que llegan a la memoria principal.
- En almacenamiento secundario se guardan paginas de memoria que fueron remplazadas bajo alguna política de remplazos.

Si estamos remplazando paginas todo el tiempo, nuestro sistema puede entrar en *crashing*.

Memoria Virtual: Resumen



X	Página de memoria virtual mapeada en memoria física	X	Entrada en la tabla de páginas mapeada válida		Marco de página físico ocupado por el sistema operativo
X	Página de memoria virtual no mapeada	-	Entrada en la tabla de páginas invalida (lógicamente ocupada)	X	Marco de página físico ocupado por una página de aplicación
	Página de memoria virtual libre		Entrada en la tabla de páginas invalida		Marco de página físico libre

Interrupción de Page Fault

La interrupción de fallo de página se produce en tres situaciones.

- El **nivel de privilegios** no es suficiente para acceder.
- La acción a realizar (Read, Write, Execute) no se corresponde con los **permisos de la página**.
- La página **no está mapeada**.

Interrupción de Page Fault

La interrupción de fallo de página se produce en tres situaciones.

- El **nivel de privilegios** no es suficiente para acceder.
- La acción a realizar (Read, Write, Execute) no se corresponde con los **permisos de la página**.
- La página **no está mapeada**.

En cualquiera de los casos se pueden dar dos soluciones en la rutina de atención de la interrupción.

- El acceso es **incorrecto** y por lo tanto se debe matar al proceso en ejecución.
- El acceso es **correcto** y el sistema operativo debe resolver el problema.

Interrupción de Page Fault

La interrupción de fallo de página se produce en tres situaciones.

- El **nivel de privilegios** no es suficiente para acceder.
- La acción a realizar (Read, Write, Execute) no se corresponde con los **permisos de la página**.
- La página **no está mapeada**.

En cualquiera de los casos se pueden dar dos soluciones en la rutina de atención de la interrupción.

- El acceso es **incorrecto** y por lo tanto se debe matar al proceso en ejecución.
- El acceso es **correcto** y el sistema operativo debe resolver el problema.

Esta interrupción tiene un caracter especial, ya que funciona como una **trampa**.

Cuando se retorna de la rutina, se vuelve a la instrucción que produjo el error para que **continue ejecutando desde el mismo lugar**, pero ahora con la memoria correctamente mapeada.

Política de optimización Copy-On-Write

Cuando se ejecuta un `fork`, un proceso nuevo es creado.

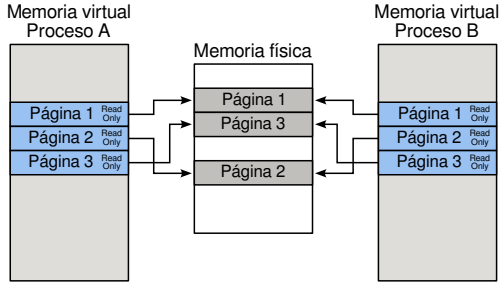
El nuevo proceso existirá en un espacio de memoria nuevo, y para esto se deben **copiar todas las páginas** del proceso padre, duplicando la memoria ocupada.

Política de optimización Copy-On-Write

Cuando se ejecuta un `fork`, un proceso nuevo es creado.

El nuevo proceso existirá en un espacio de memoria nuevo, y para esto se deben **copiar todas las páginas** del proceso padre, duplicando la memoria ocupada.

Para evitar copiar las páginas, estas se mapean como **solo lectura para ambos procesos**.



Política de optimización Copy-On-Write

Cuando se ejecuta un `fork`, un proceso nuevo es creado.

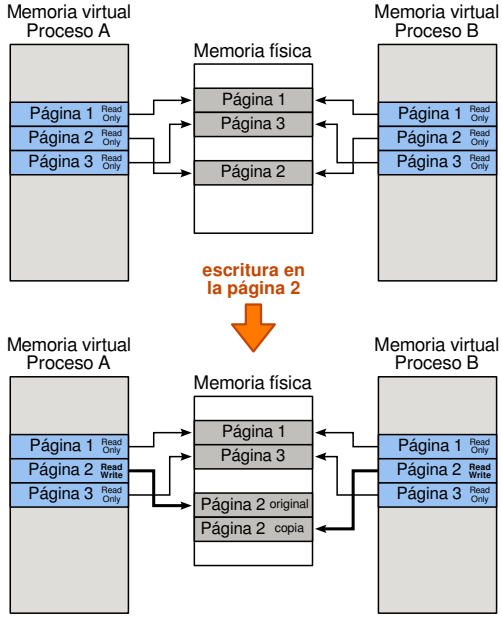
El nuevo proceso existirá en un espacio de memoria nuevo, y para esto se deben **copiar todas las páginas** del proceso padre, duplicando la memoria ocupada.

Para evitar copiar las páginas, estas se mapean como **solo lectura para ambos procesos**.

Cuando cualquiera de los procesos quiera escribir, se generará un fallo de página.

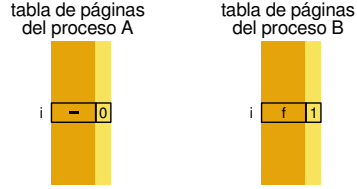
La rutina de atención de interrupciones actuará y determinará que se trata de **páginas físicas compartidas**. Duplicando entonces la memoria en dos páginas físicas distintas.

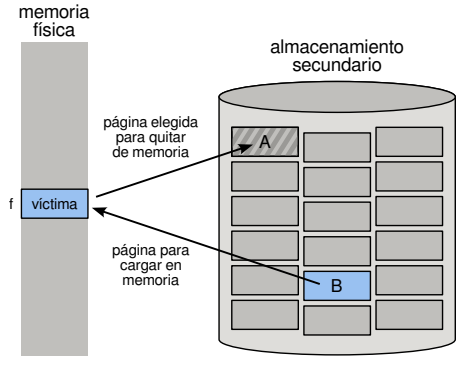
Se mapea una copia en ambos procesos.



Remplazo de páginas

Cuando no hay páginas libres en memoria física donde cargar los nuevos marcos de página.
Se debe seleccionar una página **víctima**.

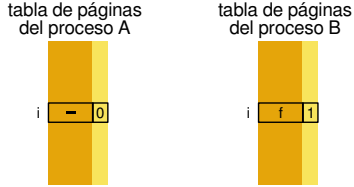


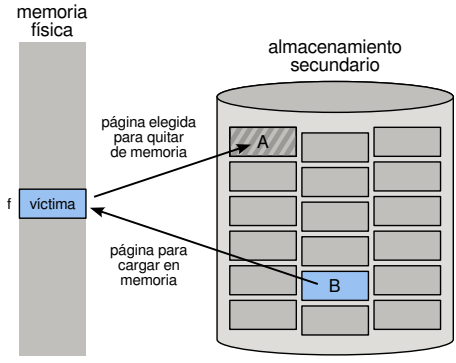


Remplazo de páginas

Cuando no hay páginas libres en memoria física donde cargar los nuevos marcos de página.
Se debe seleccionar una página **víctima**.

Esta página será desalojada de memoria y almacenada en almacenamiento secundario.





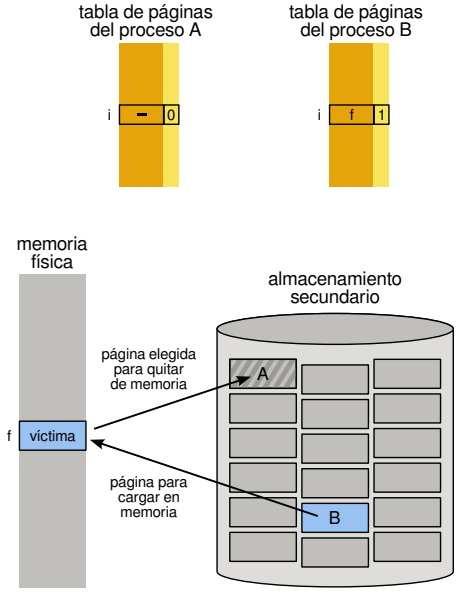
Remplazo de páginas

Cuando no hay páginas libres en memoria física donde cargar los nuevos marcos de página.
Se debe seleccionar una página **víctima**.

Esta página será desalojada de memoria y almacenada en almacenamiento secundario.

La nueva página será cargada en su lugar y se mapeará en la tabla de páginas de la tarea.

Los algoritmos de elección de páginas víctima pueden ser: First In First Out (FIFO), Least Recently Used (LRU), Least Frequenly Used (LFU), FIFO Second-chance, etc.



Remplazo de páginas

Cuando no hay páginas libres en memoria física donde cargar los nuevos marcos de página.
Se debe seleccionar una página **víctima**.

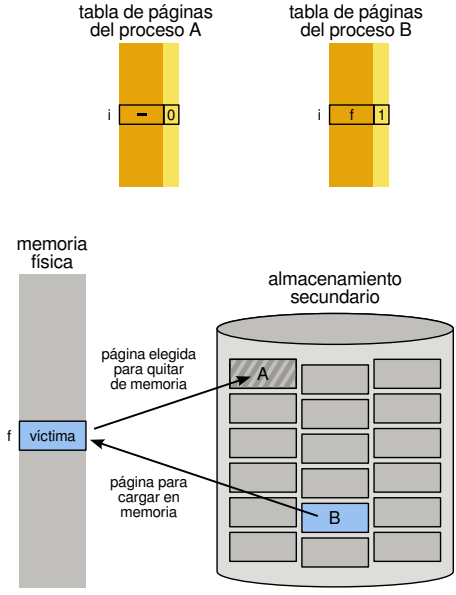
Esta página será desalojada de memoria y almacenada en almacenamiento secundario.

La nueva página será cargada en su lugar y se mapeará en la tabla de páginas de la tarea.

Los algoritmos de elección de páginas víctima pueden ser: First In First Out (FIFO), Least Recently Used (LRU), Least Frequenly Used (LFU), FIFO Second-chance, etc.

dirty-bit

La tabla de páginas contiene además un bit que indica si la página fue modificada. Esto permite optimizar el algoritmo de remplazo, copiando a memoria secundaria solo la página que fue modificada.



Bibliografía

- Tanenbaum, “Modern Operating Systems”, 4th Edition, 2015.
  - **Chapter 3 - Memory Management**, páginas 181-208

- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ma Edición, 2006.
  - **Capítulo 8 - Memoria principal**, páginas 243-267 y 279-288

Ejercicios

Con lo visto, ya pueden resolver todos los ejercicios de la Guía de Administración de Memoria.

¡Gracias!

Recuerden leer los comentarios adjuntos
en cada clase por aclaraciones.