Administración de Memoria

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

Administración de Memoria

- Base
- Alocación Contigua
- Segmentación
- Paginado
- Estructura de la Tabla de Páginas
- Intercambio (Swapping)

Objetivos

- Proveer con descripción detallada distintas formas de organizar el hardware de memoria.
- Discutir varias técnicas de administración de memoria, incluyendo paginación y segmentación.

Nota: *Estos temas son repaso de los vistos en las materias "Organización de Computadoras" y "Arquitectura".*

Base

- El programa debe en memoria y ubicado dentro de un proceso para ser *corrido*.
- La memoria principal y los registros son las únicas unidades de almacenaje accedidas directamente por la CPU.
- El acceso a los registros se hace en un pulso de reloj de CPU o menos.
- La memoria principal puede tomar muchos ciclos.
- El **Caché** se ubica entre la memoria principal y los registros de CPU.
- Se requiere protección de memoria.

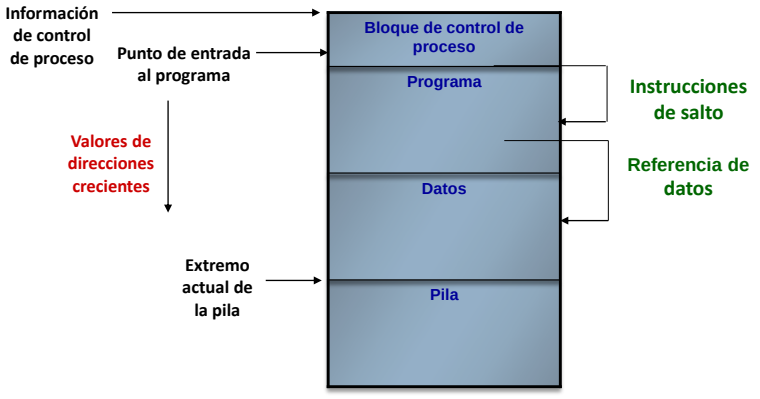
Gestión de la memoria

- Subdividir la memoria para acomodar múltiples procesos
- Es necesario asignar la memoria para asegurar una cantidad razonable de procesos listos que consuman el tiempo de procesador disponible

Requisitos de la gestión de la memoria

- Reubicación
  - Se des memoria principal se situará el programa cuando se ejecute
  - El proceso puede llevarse al disco y traerse de nuevo a la memoria principal en un área diferente (reubicado)
  - Deben traducirse las referencias de memoria encontradas en el código del programa en direcciones de memoria físicas

Requisitos de la gestión de la memoria



Requisitos de direccionamiento para un proceso

Requisitos de la gestión de la memoria

- **PROTECCIÓN**
  - Los procesos no deberían ser capaces de referenciar sin permiso posiciones de memoria principal de otro proceso
  - Es imposible comprobar las direcciones absolutas en el tiempo de compilación, deben comprobarse en el tiempo de ejecución
  - Es el procesador (hardware), en lugar del sistema operativo, el que debe satisfacer el requisito de protección de memoria. El sistema operativo no puede anticipar todas las referencias de memoria que un programa hará

Requisitos de la gestión de la memoria

- **COMPARTICIÓN**
  - Permite a varios procesos acceder a la misma porción de memoria principal
  - Es mejor permitir que cada proceso pueda acceder a la misma copia del programa en lugar de tener su propia copia separada

Requisitos de la gestión de la memoria

- **ORGANIZACIÓN LÓGICA**
  - Los programas están escritos en módulos
  - Los módulos se pueden escribir y compilar independientemente
  - Se pueden proporcionar diferentes grados de protección a los módulos (sólo lectura, sólo ejecución)
  - Se pueden compartir módulos entre los procesos

Requisitos de la gestión de la memoria

- **ORGANIZACIÓN FÍSICA**
  - La memoria principal disponible para un programa más sus datos podría ser insuficiente
    - La superposición (*overlaying*) permite asignar la misma región de memoria a varios módulos
  - El programador no conoce cuánto espacio estará disponible

Mapeo de Instrucciones y Datos a Direcciones de Memoria

El mapeo de instrucciones y datos a direcciones de memoria puede ocurrir en tres etapas diferentes.

- **Tiempo de Compilación** ➡ puede ser generado código absoluto.
- **Tiempo de Carga** ➡ código *reubicable*
- **Tiempo de Ejecución** ➡
  - el proceso puede ser movido durante su ejecución de un segmento de memoria a otro.
  - soporte de hardware para el mapeo de las direcciones (p.e., registros *base* - *límite*).

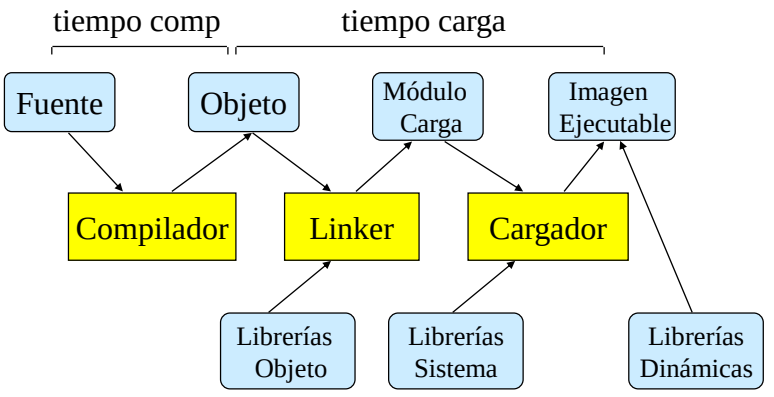
Carga Dinámica

- Las rutinas no son cargadas hasta que son llamadas.
- Mejor utilización del espacio de memoria; las rutinas no usadas no son cargadas.
- Es útil cuando hay que manejar grandes cantidades de código para casos poco frecuentes.
- No requiere un soporte especial del sistema operativo.

Enlace Dinámico

- El enlace es pospuesto hasta el tiempo de ejecución.
- Se usan pequeños pedazos de código, *stub*, para localizar las rutinas apropiadas de la librería residente en memoria.
- El stub se reemplaza a sí mismo con la dirección de la rutina y la ejecuta.
- El sistema operativo necesita verificar si la rutina está en las direcciones de memoria del proceso.

Mapeo de Direcciones



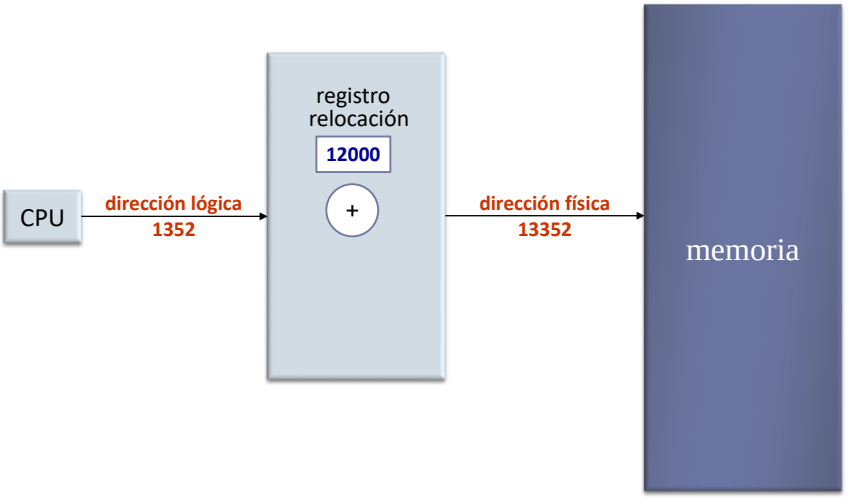
Espacio de Direcciones Lógico vs. Físico

- El concepto de *espacio de direcciones lógico* que está limitado a un espacio de *direcciones físicas* separado es central a la administración de la memoria.
  - *Dirección Lógicas* – generadas por la CPU; también llamadas ***direcciones virtuales***.
  - *Dirección Física* – dirección vista por la unidad de memoria.
- Las direcciones lógicas y físicas son las mismas en tiempo de compilación y en los esquemas de mapeo de direcciones en tiempo de carga; las direcciones lógicas (virtuales) y físicas difieren en el esquema de mapeo de direcciones en tiempo de ejecución.

Unidad de Administración de Memoria (MMU)

- Dispositivo hardware que mapea las direcciones virtuales a las físicas.
- En el esquema MMU, el valor en el registro de locación es agregado a cada dirección generada por el proceso del usuario en el momento que es presentada a la memoria.
- Los programas de usuario ven direcciones lógicas, nunca ven direcciones físicas reales.

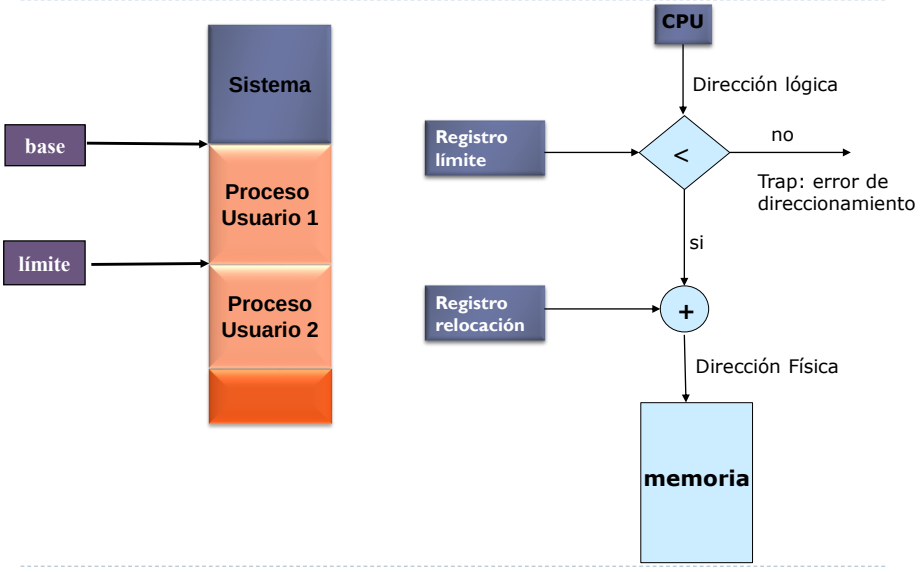
Unidad de Administración de Memoria (MMU)



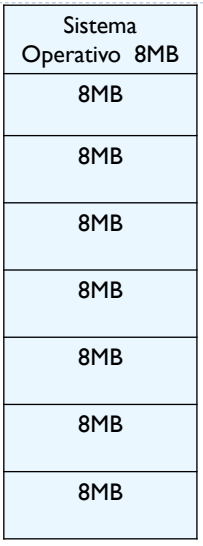
Alocación Contigua

- La memoria principal se divide, usualmente en dos particiones:
  - Parte residente del sistema operativo, generalmente en memoria baja con el vector de interrupciones.
  - Los procesos de usuario se mantienen en la parte alta de la memoria.
- Alocación en partición simple
  - Se usa un esquema de registro de relocación para proteger los procesos de usuario uno de otro, y cambio en el código y datos del SO.
  - El registro de relocación contiene el valor de la dirección física más pequeña; el registro límite contiene el rango de las direcciones lógicas – cada dirección lógica debe ser menor que el registro límite.

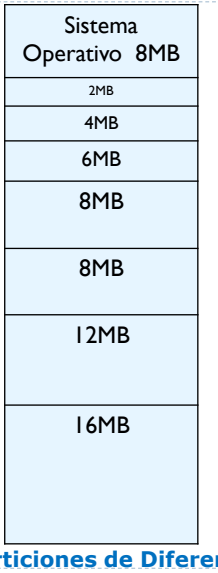
Alocación Contigua



Alocación Contigua – Particionamiento Fijo

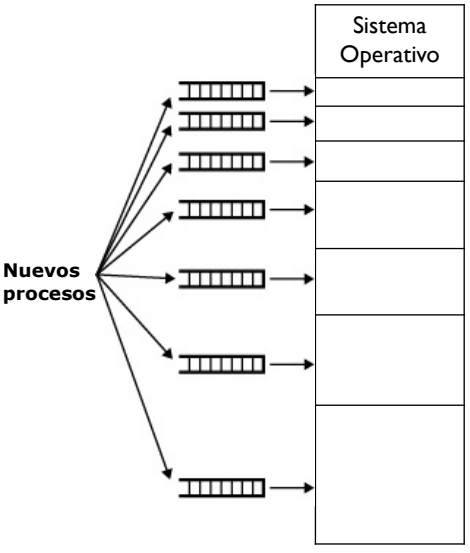


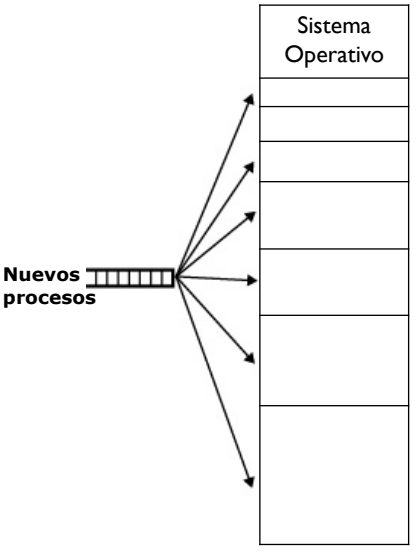
Particiones de Igual Tam.



Particiones de Diferentes Tam.

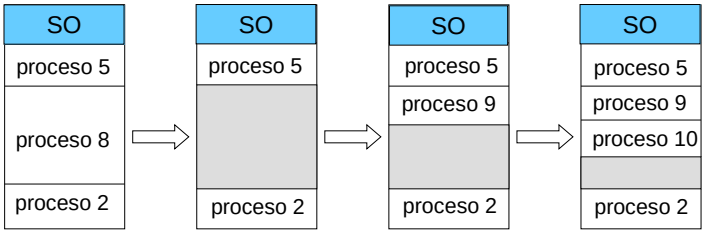
Alocación Contigua – Particionamiento Fijo





Alocación Contigua Dinámica

- Alocación en múltiple partición
  - *Agujero* – bloque de memoria disponible.
  - Cuando un proceso llega, es alocado en memoria en un agujero suficientemente grande para alojarlo.
  - El SO mantiene información sobre:
    a) particiones alojadas b) particiones libre (agujero)



Problema de Alocación Dinámica

Como satisfacer un requerimiento de tamaño *n* de una lista de agujeros libres

- **Primer lugar**: Aloca en el *primer* agujero lo suficientemente grande.
- **Próximo lugar**: Aloca en el próximo lugar luego de haber usado el primero.
- **Mejor lugar**: Aloca en el agujero *más chico* que lo puede alojar; debe buscar en la lista completa, salvo que sea ordenada por tamaño. Produce el agujero restante más chico.
- **Peor lugar**: Aloca el agujero más grande; debe buscar en la lista completa. Produce el agujero restante mas grande.

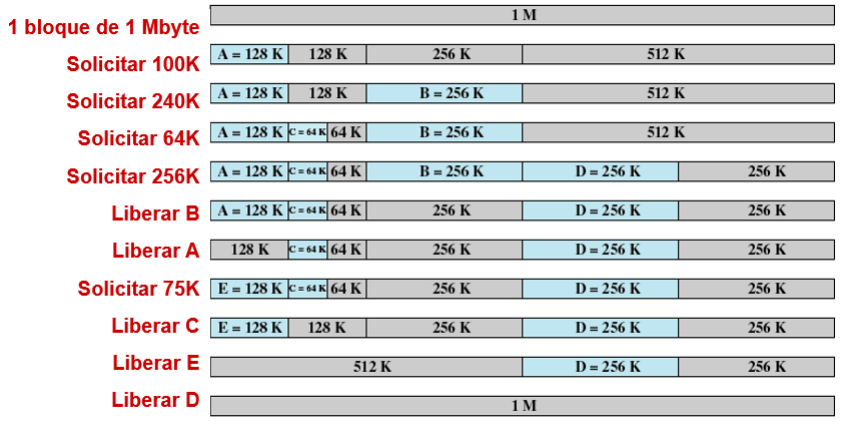
El *primer lugar* y el *mejor lugar* son mejores que el *peor lugar* en términos de velocidad y utilización del almacenamiento.

Buddy System (Alocación)

Es un sistema de alocación de memoria utilizado por algunos sistemas operativos.

- El espacio completo disponible es tratado como un bloque de tamaño 2^U
- Si un requerimiento de tamaño ***s*** tal que $2^{U-1} < s <= 2^U$, es alocado el bloque completo.
  - De otra manera el bloque es dividido en dos "compañeros" iguales.
  - El proceso continua hasta que es generado el bloque más pequeño mayor o igual que ***s***.

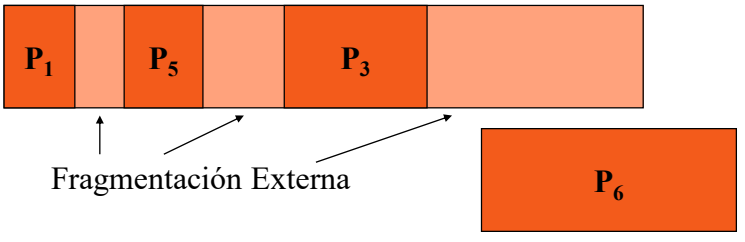
Buddy System (Alocación) - Ejemplo



Fragmentación

- **Fragmentación Externa** – existe espacio de memoria para satisfacer una demanda, pero no es contiguo.
- **Fragmentación Interna** – la memoria ocupada puede ser ligeramente más grande que la demandada; esta diferencia de tamaño es memoria interna de la partición, pero no es usada.
- Se reduce la fragmentación externa por compactación
  - Mueva el contenido de memoria de modo de dejar toda la memoria libre en un solo bloque.
  - La compactación es posible *solo* si la relocación es dinámica, y es hecha en tiempo de ejecución.
  - Problema de E/S
    - Dejar el job en memoria mientras está involucrado en una E/S.
    - Hacer E/S solo en los buffers del SO.

Fragmentación



Segmentación

- Esquema de administración de memoria que soporta la visión de usuario de la memoria.
- Un programa es una colección de segmentos. Un segmento es una unidad lógica como:

  programa principal,

  procedimiento,

  función,

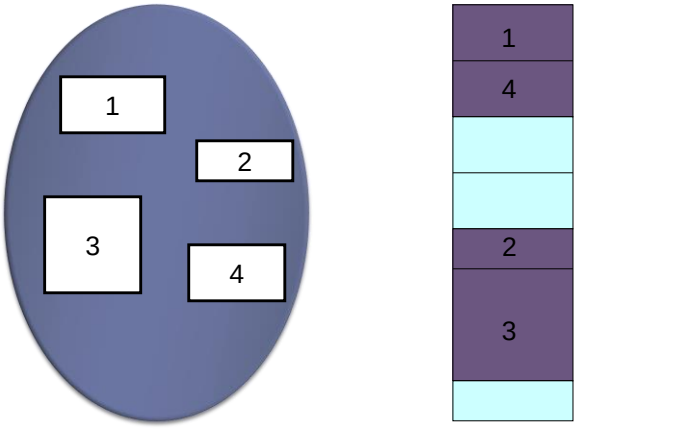
  variables locales, variables globales,

  bloque común,

  stack,

  tabla de símbolos, arreglos

Visión Lógica de la Segmentación



Espacio de usuario Espacio de memoria física

Arquitectura de Segmentación

- Dirección lógica consistente de dupla:

  < número-segmento, offset>,

- ***Tabla de Segmentos*** – mapean direcciones físicas en dos dimensiones; cada entrada en la tabla tiene:
  - ***base*** – contiene la dirección física inicial donde el segmento reside en memoria.
  - ***límite*** – especifica la longitud del segmento.
- ***registro base de la tabla de segmentos (STBR)*** apunta a la locación de la tabla de segmentos en memoria.
- ***Registro de longitud de la tabla de segmentos (STLR)*** indica el número de segmentos usados por el programa;

  el número de segmento ***s*** es legal si ***s*** < **STLR.**

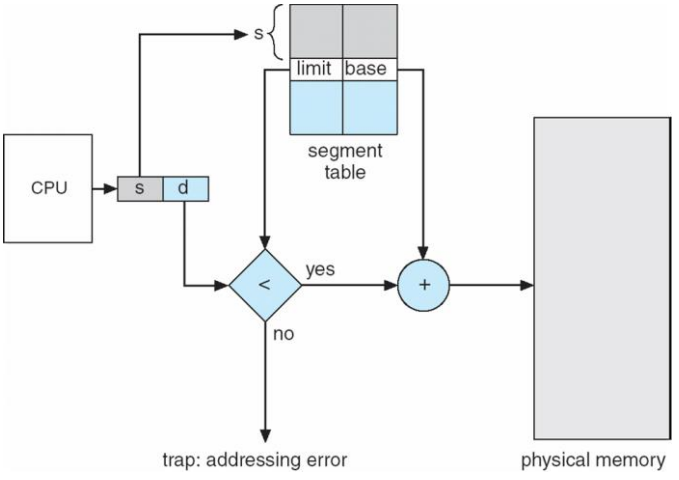
Arquitectura de Segmentación

- Relocación.
  - dinámica
  - por tabla de segmentos
- Compartir
  - segmentos compartidos
  - Igual número de segmento
- Alocación.
  - Primer lugar/mejor lugar
  - fragmentación externa

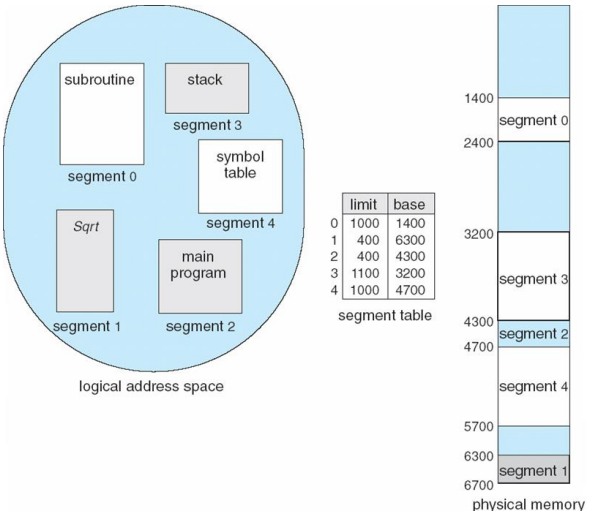
Arquitectura de Segmentación

- Protección. Con cada entrada en la tabla de segmentos se asocia:
  - bit de validación = 0 ⇒ segmento ilegal
  - privilegios read/write/execute
- Bits de protección asociados con segmentos; el código compartido ocurre a nivel de segmento.
- Dado que los segmentos varían en longitud, la alocación de memoria es un problema de alocación dinámica de almacenaje.
- Un ejemplo de segmentación se muestra en el siguiente diagrama

Hardware de Segmentación



Ejemplo de Segmentación



Paginado

- El espacio de direcciones puede no ser contiguo; el proceso es alojado en la memoria física donde haya lugar.
- Se divide a la memoria física en bloques de tamaño fijo llamados **cuadros** (el tamaño es potencia de 2, entre 512 bytes y 8192 bytes).
- Se divide a la memoria lógica en bloques de tamaño similar llamados **páginas**.
- Se guarda información de todos los cuadros libres.
- Para correr un programa de n páginas, se necesita encontrar *n* cuadros libres y cargar el programa.
- Se establece una **tabla de páginas** para traducir las direcciones lógicas a físicas.
- Fragmentación interna.

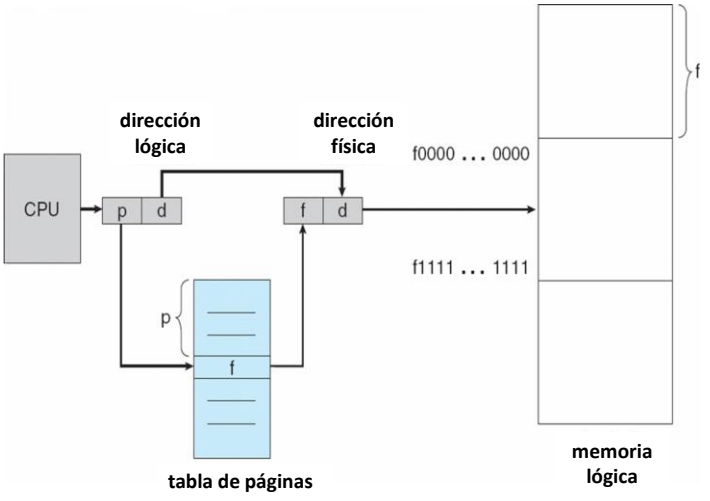
Esquema de Traducción de Direcciones

- La dirección generada por la CPU está dividida en:
  - *Número de página (**p**)* – usado como un índice en la *tabla de páginas* la cual contiene la dirección base de cada página en la memoria física.
  - *Desplazamiento en la página (**d**)* – combinado con la dirección base para definir dirección física de memoria que es enviada a la unidad de memoria.

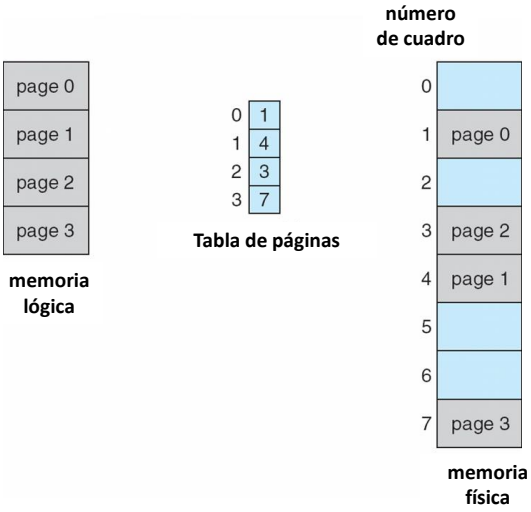
num página	desplazamiento
p	***d***
m - n	***n***

  - Para un espacio de direcciones 2^m y tamaño de página 2^n

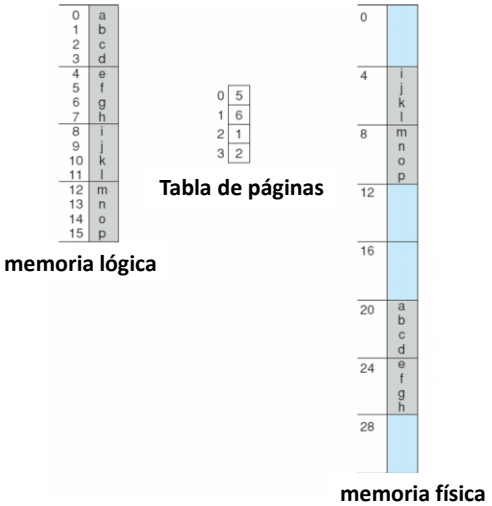
Hardware de Paginado



Modelo de Paginado de Memoria Lógica y Física



Ejemplo de Paginado



Memoria de 32 bytes y páginas de 4bytes

Implementación de la Tabla de Páginas

- La tabla de páginas es guardada en memoria principal.
- *El **registro base de tablas de páginas (PTBR)*** apunta a la tabla de páginas.
- *El **registro de longitud de la tabla de páginas (PRLR)*** indica el tamaño de la tabla de páginas.
- En este esquema cada acceso a instrucción/dato requiere dos accesos a memoria. Uno para la tabla de páginas y otro para la instrucción/dato.
- El problema del doble acceso puede ser resuelto por el uso de un cache hardware especial de adelantamiento llamado ***registro asociativo*** o ***translation look-aside buffers (TLBs)***

Registro Asociativo

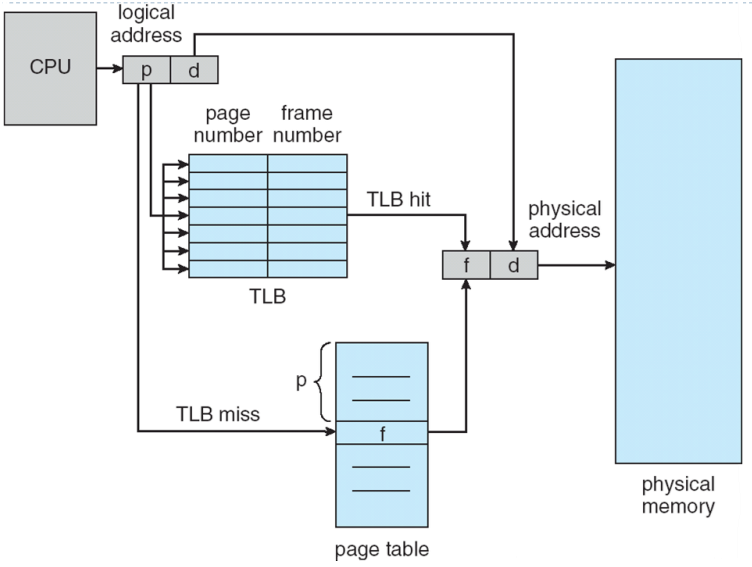
- Registro Asociativo – búsqueda paralela

página #	cuadro #

Traducción de dirección **(p, d)**

- Si **p** es un registro asociativo, tome el cuadro # .
- Sino tome el cuadro # desde la tabla de páginas en memoria.

Hardware de Paginado con TLB



Tiempo Efectivo de Acceso

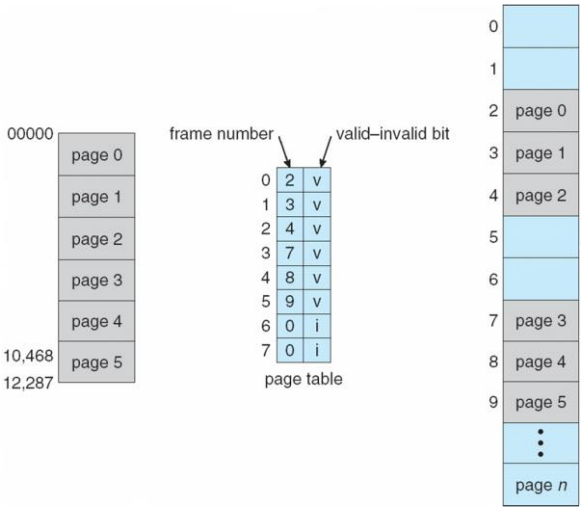
- Búsqueda Asociativa = ε unidad de tiempo
- Suponga que el ciclo de memoria es de 1 microsegundo
- Relación de acierto – porcentaje de veces que la página es encontrada en los registros asociativos; está relacionado con el número de registros asociativos.
- Relación de acierto = α
- Tiempo Efectivo de Acceso (TEA)

$$TEA = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$
$$= 2 + \varepsilon - \alpha$$

Protección de Memoria

- La protección de memoria se implementa asociando un bit con cada cuadro.
- Un bit de *válido-inválido* agregado a cada entrada en la tabla de páginas:
  - "**válido**" indica que la página asociada está en el espacio de direcciones lógicas del proceso, por lo tanto es una página legal.
  - "**inválido**" indica que la página no está en el espacio de direcciones lógicas del proceso.

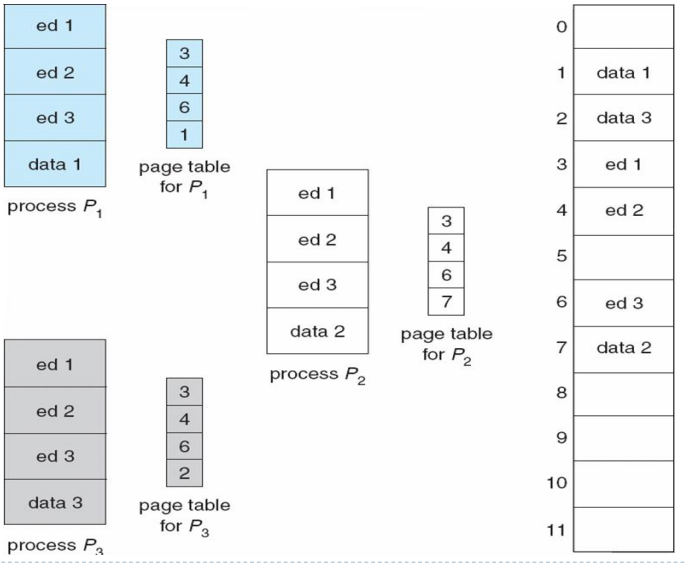
Bit de Válido (v) o Inválido (i) en una Tabla de Páginas



Páginas Compartidas

- Código compartido
  - Una copia de código de lectura solamente (reentrante) compartido entre procesos (p.e., editores de texto, compiladores, sistemas de ventanas).
  - El código compartido debe aparecer en la misma locación en el espacio de direcciones de todos los procesos.
- Código privado y datos
  - Cada proceso guarda una copia separada de código y datos.
  - Las páginas del código privado y datos puede aparecer en cualquier lugar del espacio de direcciones lógico.

Ejemplo de Páginas Compartidas



Tablas de Páginas con Hash

- Común en espacio de direcciones > 32 bits
- Un número de página virtual es pasada por hash en la tabla de páginas. Esta tabla de páginas contiene una cadena de elementos pasados por hash a la misma locación.
- Los números de páginas virtuales son comparados con lo que existe en estas cadenas. Si se encuentra se extrae el correpondiente cuadro físico.

Tablas de Páginas con Hash

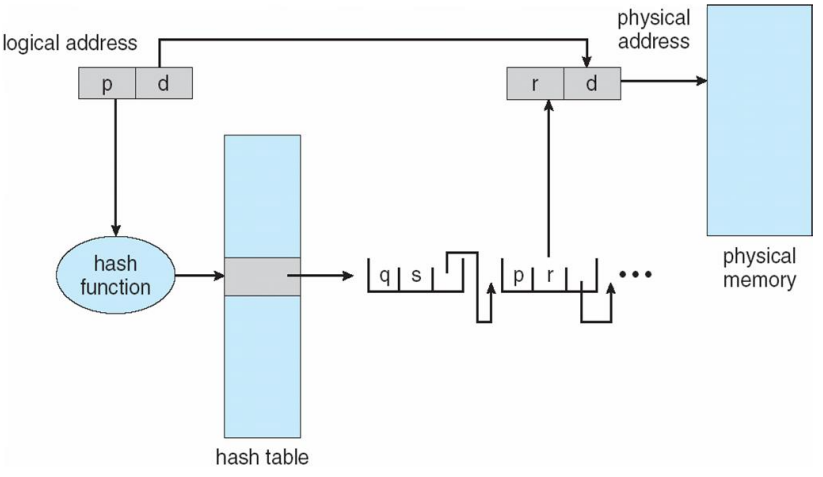
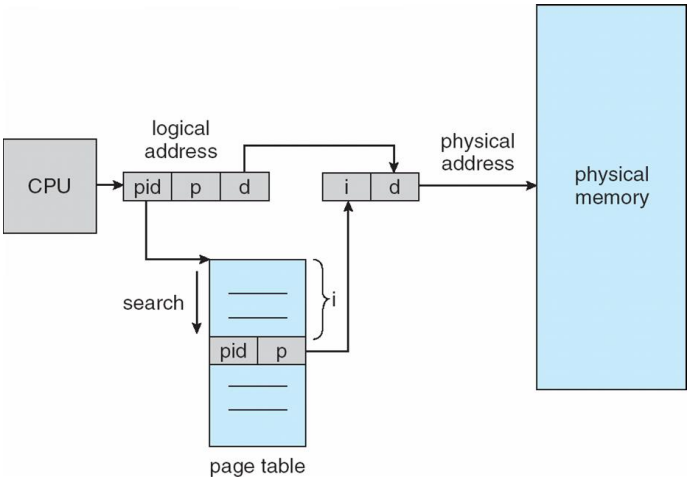


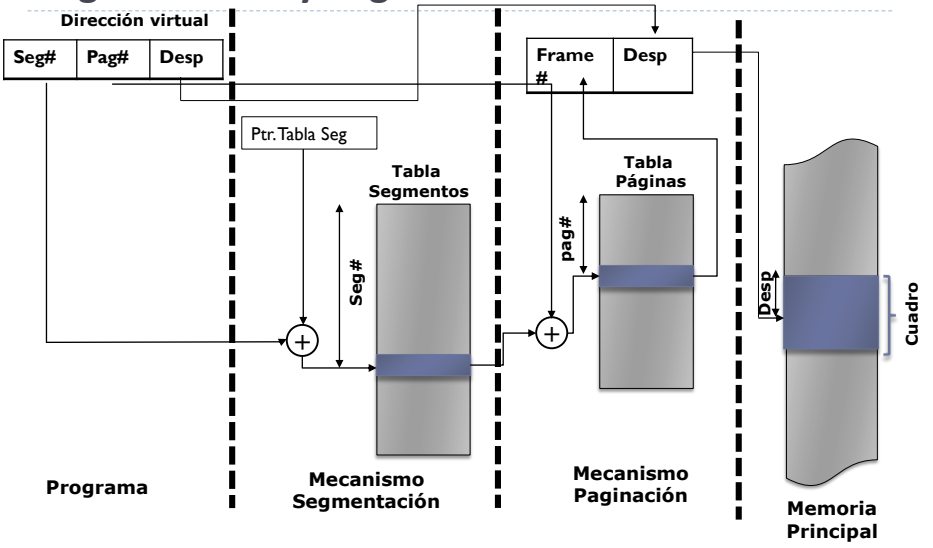
Tabla de Páginas Invertida

- Una entrada por cada página real de memoria.
- La entrada consiste de la dirección virtual de la página almacenada en la locación real de memoria, con información del proceso que es dueño de esa página.
- Decrementa la memoria necesaria para almacenar cada tabla de páginas, pero incrementa el tiempo necesario para buscar en la tabla cuando ocurre una referencia a una página.
- Se usa una tabla hash para limitar la búsqueda a una — o a lo sumo unas pocas — entrada a la tabla de páginas.

Arquitectura de la Tabla de Página Invertida



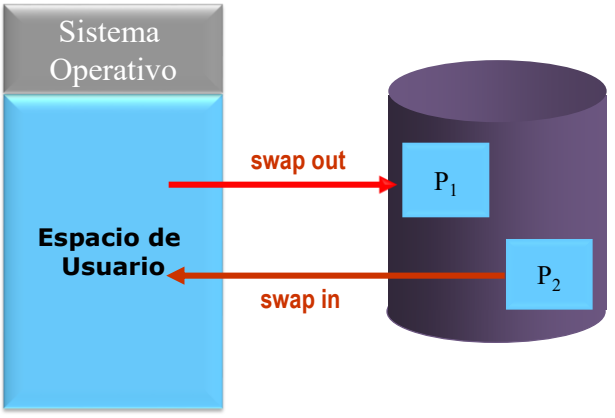
Segmentación y Paginación



Intercambio (Swapping)

- Un proceso puede ser intercambiado (*swapped)* temporariamente fuera de la memoria a un almacenamiento de respaldo (*backing store)*, y luego ser vuelto a la misma para continuar su ejecución.
- Backing store – espacio en disco lo suficientemente grande para acomodar copias de todas las imágenes de memoria de todos los usuarios.
- *Roll out, roll in* – variante del intercambio usado en algoritmos de planificación basados en prioridades.
- La mayor parte del tiempo de intercambio es *tiempo de transferencia* y es directamente proporcional a la cantidad de memoria intercambiada.

Visión Esquemática del Intercambio



Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "*Operating System Concepts*", 7ma Edición. 2009, 9na Edición 2012, 10ma Edición 2018.
- Stallings, W. *"Operating Systems: Internals and Design Principles"*, Prentice Hall, 7maEdición 2011; 8va. Edición 2014; 9na Edición 2018.