Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas

Sistemas Operativos

Unidad I:

Estructura de los sistemas operativos

Gestión de un sistema operativo

M. en C. Hermes Francisco Montes Casiano
hermes.upiicsa@gmail.com

Objetivos

Objetivos

1. Proporcionar una descripción de las diversas formas de organizar el hardware de memoria.
2. Analizar diversas técnicas de gestión de memoria, incluyendo la paginación y la segmentación.

Índice

Introducción

- La memoria es un componente crucial para la operación de un sistema de cómputo.
- La memoria está compuesta de una gran matriz de palabras o bytes, cada una con su propia dirección.
- La CPU extrae instrucciones de la memoria de acuerdo con el contrador de programa de la memoria.
- Dichas instrucciones pueden provocar operaciones adicionales de carga o de almacenamiento en direcciones de memoria específicas.

Introducción

- La memoria es un componente crucial para la operación de un sistema de cómputo.
- La memoria está compuesta de una gran matriz de palabras o bytes, cada una con su propia dirección.
- La CPU extrae instrucciones de la memoria de acuerdo con el contrador de programa de la memoria.

Ciclo de instrucción

1. Extraer una instrucción de la memora.
2. Decodificar instrucción y extraer operandos.
3. Ejecutar la instrucción.
4. Almacenar resultado en la memoria.

Introducción

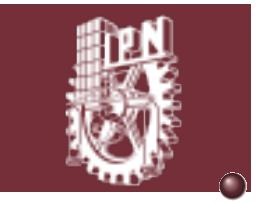
- La memoria es un componente crucial para la operación de un sistema de cómputo.
- La memoria está compuesta de una gran matriz de palabras o bytes, cada una con su propia dirección.
- La CPU extrae instrucciones de la memoria de acuerdo con el contrador de programa de la memoria.
- Dichas instrucciones pueden provocar operaciones adicionales de carga o de almacenamiento en direcciones de memoria específicas.

Instrucciones

La unidad de memoria sólo ve el flujo de direcciones de memoria y no sabe como se generan esas direcciones.

Hardware básico

- La memoria principal y los registros dentro del propio procesador son las únicas áreas de almacenamiento a las que la *CPU* puede acceder.
- Todas las instrucciones en ejecución y los datos utilizados por las instrucciones deberán encontrarse en almacenados en uno de los medios de almacenamiento de acceso directo.
- Si los datos no se encuentran en memora, deberán llevarse hasta allí antes de que ls *CPU* pueda operar con ellos.
- Puede accederse a los registros integrados en la *CPU* en único ciclo de reloj del procesador.
- El acceso a la memoria puede requerir más de un ciclo de reloj para poderse completar, el procesador necesitará detenerse.



Harware básico

Protección: registros base y límite

- Se debe garantizar una correcta operación que proteja el espacio de direcciones del sistema operativo y demás procesos.
- La protección debe ser proporcionada por el hardware.
- Cada proceso debe tener su espación de memoria separado.
- Protección mediante registros:
  1. **Registro base:** almacena la dirección físca legal más pequeña.
  2. **Registro límite:** epecífica el tamaño del rango.

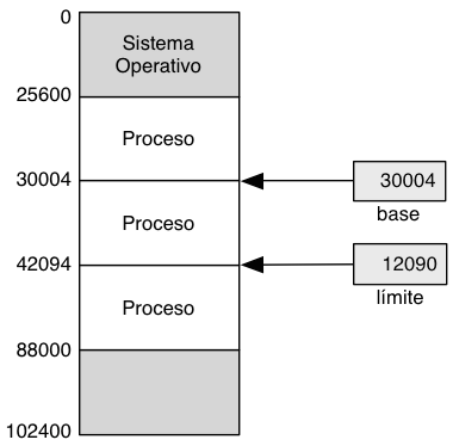


Figura: Registro base y registro límite.

Harware básico

Protección: registros base y límite

Protección mediante registros:

1. **Registro base:** almacena la dirección físca legal más pequeña.
2. **Registro límite:** epecífica el tamaño del rango.

- La protección del espacio de memoria se consigue haciendo que el *CPU* compare todas las direcciones con esos registros.
- Cualquier intento por parte de un proceso de acceder a la memoria de otro hará que se produzca una interrupción hacia el sistema operativo.

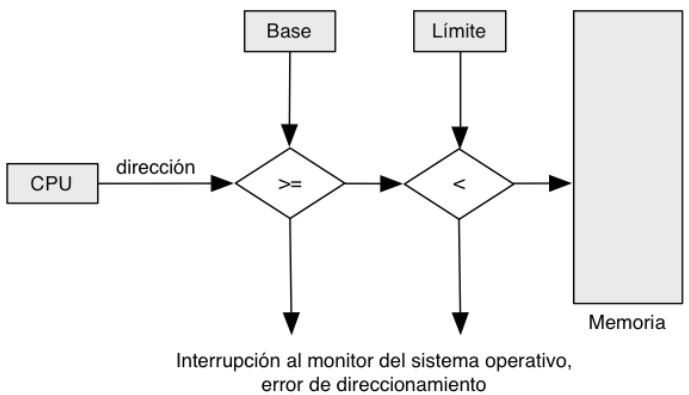


Figura: Protección hardware de las direcciones.

Harware básico

Protección: registros base y límite

- Se debe garantizar una correcta operación que proteja el espacio de direcciones del sistema operativo y demás procesos.
- La protección debe ser proporcionada por el hardware.
- Cada proceso debe tener su espación de memoria separado.
- Protecćon mediante registros:
  1. **Registro base:** almacena la dirección físca legal más pequeña.
  2. **Registro límite:** epecífica el tamaño del rango.
- La protección del espacio de memoria se consigue haciendo que el *CPU* compare todas las direcciones con esos registros.
- Cualquier intento por parte de un proceso de acceder a la memoria de otro hará que se produzca una interrupción hacia el sistema operativo.

Harware básico

Protección: registros base y límite

- Los registros base y límite sólo pueden ser cargador por el sistema operativo.
- El sistema operativo tiene acceso **no restringido** a la memoria tanto del sistema como de los demás procesos.
- Lo anterior permite al sistema operativo cargar programas en la memoria, volcarlos en caso de error, leer y modificar parámetros en llamadas al sistema, etc.

Reasignación de direcciones

- Los programas residen en disco duro como archivos binarios ejecutables.
- Para ejecutarse, un programa, deberá cargarse en memoria y colocarse dentro de un proceso.
- Los procesos del disco que estén esperando a ser cargados en memoria para su ejecución constituyen la **cola de entrada**.
- El procedimiento normal consiste en seleccionar uno de los procesos de la cola de entrada y cargarlo en memoria.
- Eventualmente, el proceso terminará su ejecución y su espacio de memoria será declarado como disponible.

Reasignación de direcciones

Ejecución de un proceso

- En la mayoría de los casos, un programa de usuario recorrerá varios pasos antes de ser ejecutado.
- A lo largo de esos pasos las direcciones sufren transformaciones:
  1. Las direcciones del programa fuente generalmente son **símbolicas**.
  2. Un compilador se encargará de asignar direcciones **reubicables** a las direcciones símbolicas.
  3. El editor de montaje se encargará de asignar direcciones **absolutas** a las direcciones reubicables.
- Cada operación de reasignación constituye una relación de un espacio de direcciones a otro.

Reasignación de direcciones

Ejecución de un proceso

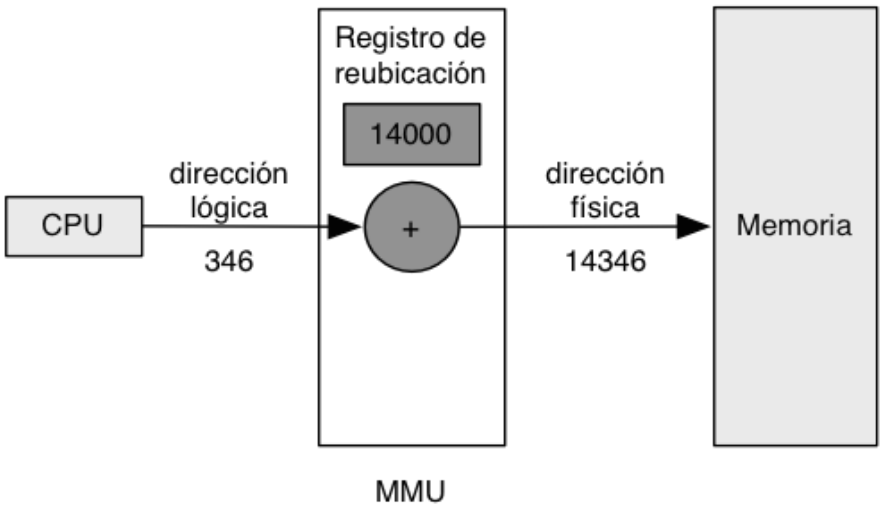


Figura: Pasos en el procesamiento de un programa de usuario.

Reasignación de direcciones

Ejecución de un proceso

- La reasignación de las instrucciones y los datos a direcciones de memoria puede realizarse en cualquiera de los siguientes pasos:
  1. **Tiempo de compilación:**
  2. **Tiempo de carga:**
  3. **Tiempo de ejecución:**

Reasignación de direcciones

Ejecución de un proceso

- La reasignación de las instrucciones y los datos a direcciones de memoria puede realizarse en cualquiera de los siguientes pasos:
  1. **Tiempo de compilación:**
     - Si al momento de compilar se sabe donde va a residir el proceso se pueden generar **direcciones absolutas**.
     - Si se se sabe que el proceso residira en una zona de memoria que comienza en la ubicación R, el código generado por el compilador comenzará en dicha ubicación y se extenderá a partir de ahí.
  2. **Tiempo de carga:**
  3. **Tiempo de ejecución:**

Reasignación de direcciones

Ejecución de un proceso

- La reasignación de las instrucciones y los datos a direcciones de memoria puede realizarse en cualquiera de los siguientes pasos:
  1. **Tiempo de compilación:**
  2. **Tiempo de carga:**
     - Si no se conoce en tiempo de compilación donde va a residir el proceso en memoria, el compilador deberá generar **código reubicable**.
     - En este caso, se retarda la asignación final hasta el momento de la carga.
  3. **Tiempo de ejecución:**

Reasignación de direcciones

Ejecución de un proceso

- La reasignación de las instrucciones y los datos a direcciones de memoria puede realizarse en cualquiera de los siguientes pasos:
  1. **Tiempo de compilación:**
  2. **Tiempo de carga:**
  3. **Tiempo de ejecución:**
     - Si el proceso puede desplazarse durante su ejecución desde un segmento de memoria a otro, entonces es necesario retardar la reasignación hasta el momento de la ejecución.
     - La mayoría de los sistemas operativos de propósito general utilizan este método.

Espacios de direcciones lógico y físico

Tipos de direcciones

- Una dirección generada por la CPU se denomina **dirección lógica** o **dirección virtual**.
- Una dirección física es una dirección vista por la unidad de memoria, es decir, la que se carga en el *registro de direcciones de memoria* de la memoria.

- Los métodos de reasignación de direcciones en tiempo de compilación y en tiempo de carga generan direcciones lógicas y físicas idénticas.
- El esquema de reasignación de direcciones en tiempo de ejecución hace que las direcciones lógicas y físicas difieran.

Espacios de direcciones lógico y físico

Definiciones

- El conjunto de todas las direcciones lógicas generadas por un proceso es lo que se denomina un **espacio de direcciones lógicas**.
- El conjunto de todas las direcciones físicas correspondientes a esas direcciones lógicas es un **espacio de direcciones físicas**.

- La correspondencia entre direcciones virtuales y físicas en tiempo de ejecución es establecida por el dispositivo hardware que se denomina **unidad de gestión de memoria (MMU)**.

Espacios de direcciones lógico y físico

Conversión de direcciones

- La correspondencia entre direcciones virtuales y físicas en tiempo de ejecución es establecida por el dispositivo hardware que se denomina **unidad de gestión de memoria (MMU)**.

Ejemplo de esquema MMU: registro base o de reubicación

El valor contenido en el registro de reubicación suma a todas las direcciones generadas por un proceso de usuario en el momento de enviarlas a memoria.

Espacios de direcciones lógico y físico

Conversión de direcciones

Ejemplo de esquema MMU: registro base o de reubicación

El valor contenido en el registro de reubicación suma a todas las direcciones generadas por un proceso de usuario en el momento de enviarlas a memoria.

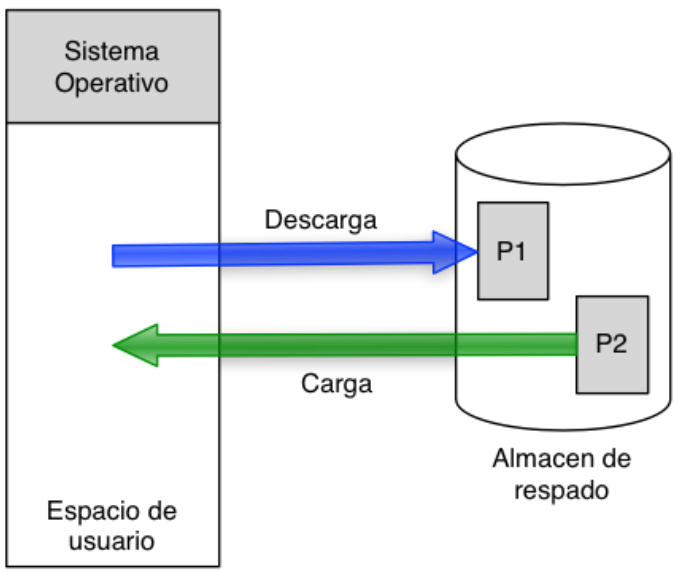


Figura: Reubicación dinámica mediante un registro de reubicación.

Espacios de direcciones lógico y físico

Conversión de direcciones

- El proceso de usuario nunca ve direcciones físicas reales.
- Un programa puede crear un puntero a la dirección de memoria *FF* y tratarla siempre como el número *FF*.
- Sólo cuando lo utiliza como dirección de memoria se producirá la reubicación en relación con el registro base.

Espacios de direcciones lógico y físico

Conversión de direcciones

Conversión

El proceso de usuario maneja direcciones lógicas y el harware de conversión de memoria convierte esas direcciones lógicas en direcciones físicas.

Espacios de direcciones

El concepto de un *espacio de direcciones lógicas* que se acopla a un *espacio de direcciones físicas* separado resulta crucial para una adecuada gestión de la memoria.

Carga dinámica

- El tamaño de un proceso está limitado al tamaño de la memoria física.
- Para obtener una mejor utilización del espacio de memoria se puede utilizar un esquema de **carga dinámica**.
- Con la carga dinámica una rutina no se carga hasta que se invoca.
- Todas las rutinas se mantienen en disco en formato de carga reubicable.
- La carga dinámica se realiza como sigue:
  1. El programa principal se carga en memoria y se ejecuta.
  2. Cuando una rutina necesita llamar a otra, la rutina que invoca comprueba si la otra rutina ya ha sido cargada.
  3. Si no es así, se invoca al cargador de montaje reubicable para que cargue en memoria la rutina deseada (actualiza la tabla de direcciones).

Carga dinámica

Ventaja

Una rutina no utilizada no se cargará nunca en memoria. Este método es muy útil cuando se necesitan grandes cantidades de código para gestionar casos que sólo ocurren de manera esporádica.

Montaje dinámico y bibliotecas compartidas

Montaje estático

Las bibliotecas del sistema se tratan como cualquier otro módulo objeto y son integradas por el cargador dentro de la imagen binaria del programa.

- En el montaje dinámico se incluye un *stub* dentro de la imagen binaria para cada referencia a una rutina de biblioteca.

Stub

Es un fragmento de código que indica como localizar la rutina adecuada de biblioteca residente en memoria o como cargar la biblioteca si esa rutina no está todavía presente.

Montaje dinámico y bibliotecas compartidas

- En el montaje dinámico se incluye un *stub* dentro de la imagen binaria para cada referencia a una rutina de biblioteca.
- Cuando se ejecuta el *stub*, éste comprueba si la rutina necesaria ya se encuentra en memoria.
- Si no se encuentra el programa carga la rutina en memoria.
- En cualquiera de los casos el *stub* se sustituye a si mismo por la dirección de la rutina y ejecuta la rutina.
- El mecanismo de **bibliotecas compartidas** permite que más de un proceso haga referencia a una misma biblioteca, sin que esta se cargue dentro del espacio de direcciones de cada proceso que la utiliza.

Montaje dinámico y bibliotecas compartidas

Carga dinámica vs. Montaje dinámico

A diferencia de la carga dinámica, el montaje dinámico requiere ayuda del sistema operativo.

El sistema operativo es la única entidad que puede comprobar su la rutina necesaria se encuentra dentro del espacio de direcciones de otro proceo y será tambien la única entidad que pueda permitir a múltiples procesos acceder a las mismas direcciones de memoria.

Índice

Intercambio

- Un proceso debe estar en memoria para ser ejecutado.
- Sin embargo, los procesos pueden ser intercabiados temporalmente, sacándolos de la memoria y almacenándolos en un almacén de respaldo.
- Posteriormente, dichos procesos se vuelven a llevar luego a la memoria para continuar su ejecución.
- Normalmente, un proceso descargado se volverá a cargar en el mismo espacio de memoria que ocupaba anteriormente (asignación en tiempo de ensamblado o carga).
- Esta restricción la dictaminará el método de reasignación de la direcciones.
- Si se está utilizando reasignación en tiempo de ejecución sí puede moverse un proceso a un espacio de memoria distinto.

Intercambio

- Los mecanismos de intercambio requieren un almacén de respaldo (disco duro).
- El sistema mantiene una **cola de procesos preparados** que consistirá en los procesos que se encuentren en memoria o en el almacén de respaldo y estén listos para ejecutarse.

Intercambio

Cada vez que el planificador de la CPU decide ejecutar un proceso, llama al despachador:

1. Verifica si el proceso se encuentra en memoria.
2. Si no es así, y si no hay ninguna región de memoria disponible, el despachador intercambia el proceso deseado por otro.
3. Recarga los registros y transfiere el control al proceso seleccionado.

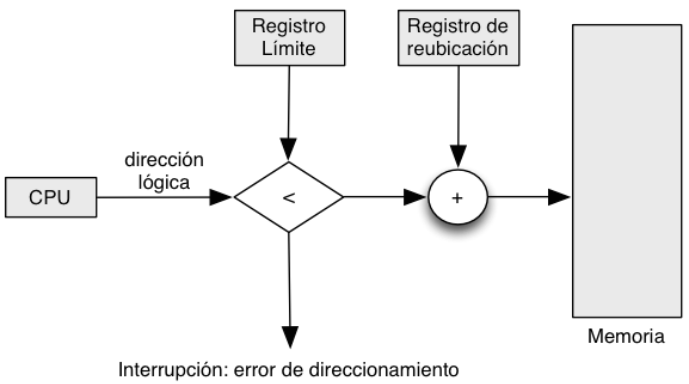


Figura: Intercambio de dos procesos utilizando un disco como almacén de respaldo.

Intercambio

Ejemplo

Ejemplo

Es necesario intercambiar un proceso de usuario que tiene de tamaño 10 MB y el almacén de respaldo es un disco duro con una velocidad de transferencia de 40 MB.

$$\frac{10000\ KB}{40000\ KB\ por\ segundo} = 250\ milisegundos \quad (1)$$

Asumiendo una latencia media de 8 milisegundos, el tiempo de intercambio será de 258 milisegundos.

Sin embargo, dado que el intercambio requiere de las operaciones de carga y descarga, el tiempo total necesario sería de $516ms$.

Intercambio

- Para conseguir un uso eficiente de la CPU, es necesario que el tiempo de ejecución de cada proceso se largo en relación con el tiempo de intercambio.
- La mayor parte del tiempo de intercambio es tiempo de transferencia.
- El tiempo de transferencia total es directamente proporcional a la *cantidad* de memoria intercambiada.
- Es útil saber cuando memoria utiliza un proceso, con la finalidad de intercambiar sólo está utilizando.

Intercambio

Factores que restringen el intercambio

Ejemplo

- Si se requiere intercambiar un proceso, se debe asegurar de que esté completamente inactivo.
- Es necesario prestar atención especial a todas las operaciones de E/S pendientes.
- Un proceso puede estar esperando por una operación de E/S en el momento en que quiera intercambiar para liberar memoria.

Intercambio

Factores que restringen el intercambio

Problema

- Si la E/S está accediendo asíncronamente a la memoria de usuario donde residen los *buffers* de E/S, el proceso no podrá ser intercambiado.
- Si se descargara el proceso P_1 y se cargara el proceso P_2, la operación de E/S podría entonces intentar utilizar la memoria que ahora pertenece al proceso P_2.

Intercambio

Factores que restringen el intercambio

Soluciones

Hay dos soluciones principalmente:

1. No descargar nunca un proceso que tenga actividades de E/S pendientes.
2. Ejecutar las operaciones de E/S únicamente con buffers del sistema operativo.

Intercambio

- En muchas versiones de UNIX, es intercambio esta normalmente desactivado.
- El intercambio se activa cuando se están ejecutando numerosos procesos y la cantidad de memoria utilizada excede cierto umbral.
- El intercambio se desactiva cuando la carga del sistema se reduce.

Índice

Introducción

- La memoria principal debe albergar tanto al sistema operativo como los diversos procesos de usuario.
- Es necesario asignar las distintas partes de la memoria principal de la forma más eficiente posible.
- La memoria usualmente está dividida en dos particiones: una para el sistema operativo y otra para los procesos.
- Dado que el vector de interrupciones se situa en la parte baja de la memoria, el sistema operativo suele situarse también en la parte baja de la memoria.
- En el esquema de **memoria contigua**, cada proceso está contenido en una única sección de memoria contigua.

Mapeo de memoria y protección

- Se puede proporcionar mapeo de memoria y protección mediante el uso de los registros de reubicación y límite.

  Reubicación: contiene el valor de la dirección física más pequeña.

  Límite: contiene el rango de las direcciones lógicas.
- La MMU convertirá la dirección lógica *dinámicamente* sumándole el valor contenido en el registro de reubicación.
- Cuado el planificador elije un proceso, el despachador carga en los registros de reubicación y de límite los valores correctos, como parte del proceso de cambio de contexto.

Mapeo de memoria y protección

- La MMU convertirá la dirección lógica *dinámicamente* sumándole el valor contenido en el registro de reubicación.
- Cuado el planificador elije un proceso, el despachador carga en los registros de reubicación y de límite los valores correctos, como parte del proceso de cambio de contexto.

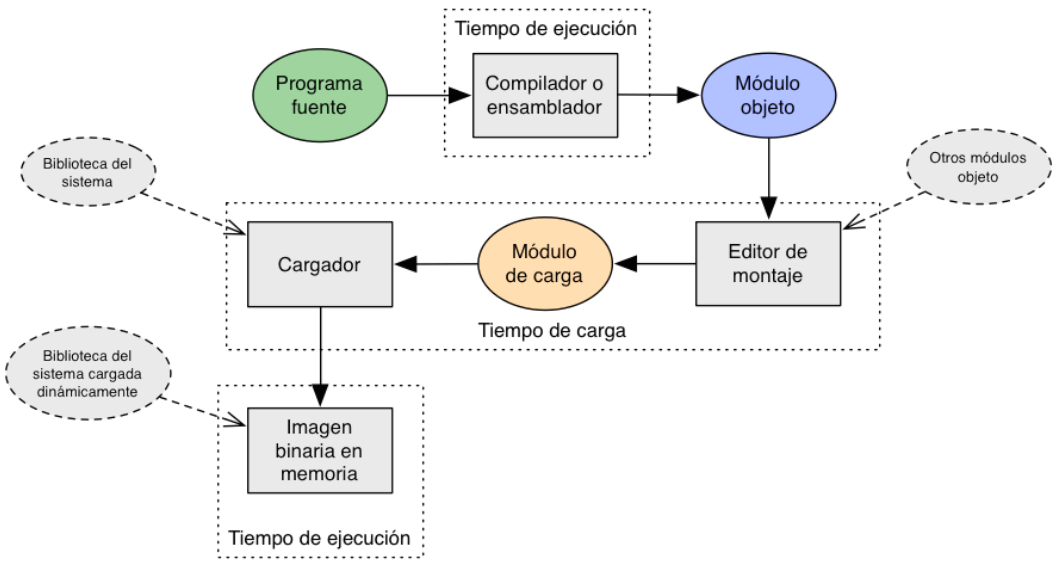


Figura: Pasos en el procesamiento de un programa.

Mapeo de memoria y protección

- La MMU convertirá la dirección lógica *dinámicamente* sumándole el valor contenido en el registro de reubicación.
- Cuado el planificador elije un proceso, el despachador carga en los registros de reubicación y de límite los valores correctos, como parte del proceso de cambio de contexto.

Protección

Puesto que todas las direcciones generadas por el CPU, este mecanismo permite proteger tanto al sistema operativo como a los demás procesos.

Índice

Introducción

Paginación

La **paginación** es un esquema de gestion de memoria que permite que el espacio de direcciones físicas de un proceso no sea contiguo.

- La paginación evita considerablemente el problema de encajar fragmentos de memoria de tamaño variable den el almacen de respaldo.
- El almacen de respando también sufre los problemas de fragmentación.
- El soporte para la paginación se gestionaba mediante hardware, actualmente se realiza mediante hardware y sistema operativo.

Método básico

Método básico

El método básico para implementar la paginación implica descomponer la memoria física en una serie de bloques de tamaño fijo denominados **marcos** y descomponer la memoria lógica en bloques del mismo tamaño denominados **páginas**.

- Cuando hay que ejecutar un proceso, sus páginas se cargan desde el almacen de respaldo en los marcos de memoria disponibles.
- El almacén de resplando está dividido en bloques de tamaño fijo que tienen el mismo tamaño que los marcos de memoria.

Método básico

- Toda dirección generada por la CPU está dividida en dos partes: un **número de página (p)** y un **desplazamiento de pagina (d)**.
- El número de página se utiliza como índice de una **tabla de páginas**.
- La tabla de páginas contiene la dirección base de cada página en memoria física.
- La dirección base se combina con el desplazamiento de página para definir la dirección de memoria física que se envía a la unidad de memoria.

Método básico

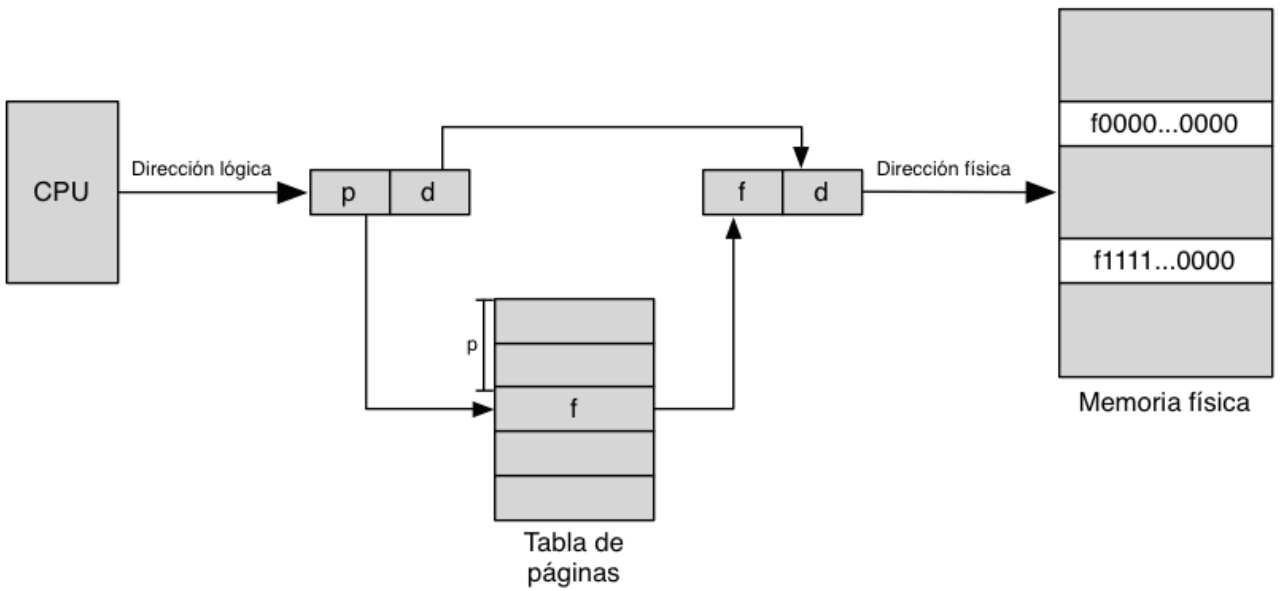


Figura: Hardware de paginación

Método básico

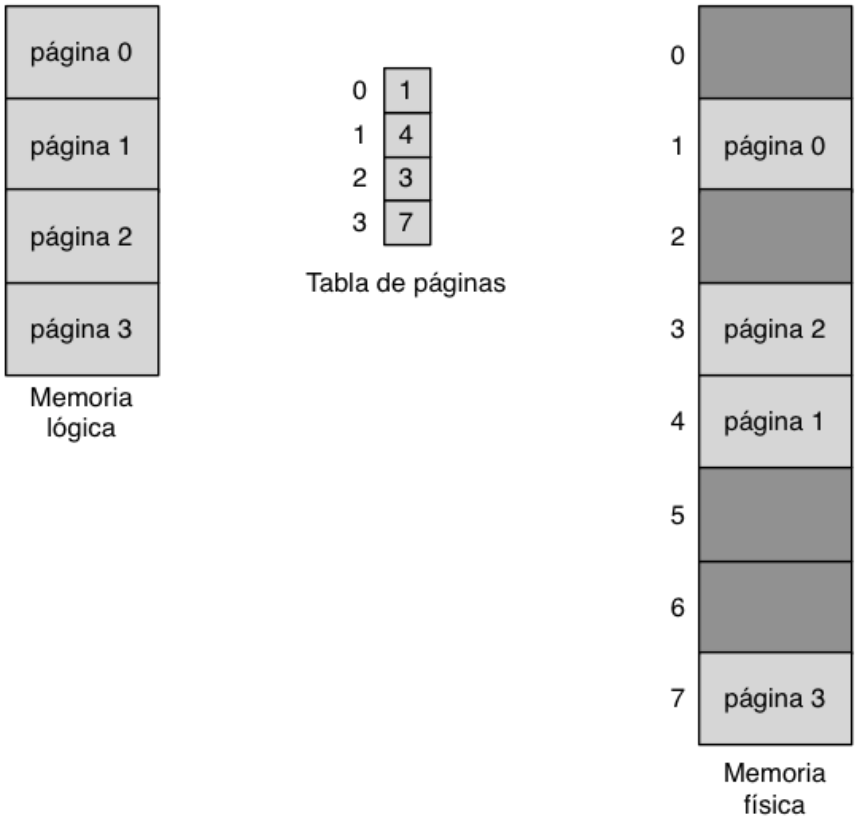


Figura: Hardware de paginación

Método básico

- El tamaño de página está definido por el hardware y normalmente es una potencia de 2.
- El tamaño de página varia entre 512 bytes y 16 MB por página, dependiendo de la arquitectura de la computadora.
- La selección de una potencia de 2 como tamaño de página facilita la traducción de direcciones lógicas a físicas.

Espacio de direcciones

Si el tamaño del espacio de direcciones lógicas es 2^n y el tamaño de página es 2^n unidades de direccionamiento, entonces los $m - n$ bits de mayor peso de cada dirección lógica designarán el numero de página, mientras que los n bits de menor peso indicarán el desplazamiento de página.

Método básico

Figura: Hardware de paginación

Método básico

Ejemplo

- Tamaño de página 4 bytes
- Memoria física de 32 bytes (8 páginas)
- La dirección lógica 0 representada en la página 0, desplazamiento 0.
- Realizando la indexación en la tabla de páginas, se observa que la página 0 se encuentra en el marco 5.
- La dirección lógica 0 se hace corresponder con la dirección $20 (= (5x4) + 0)$

Método básico

Ejemplo

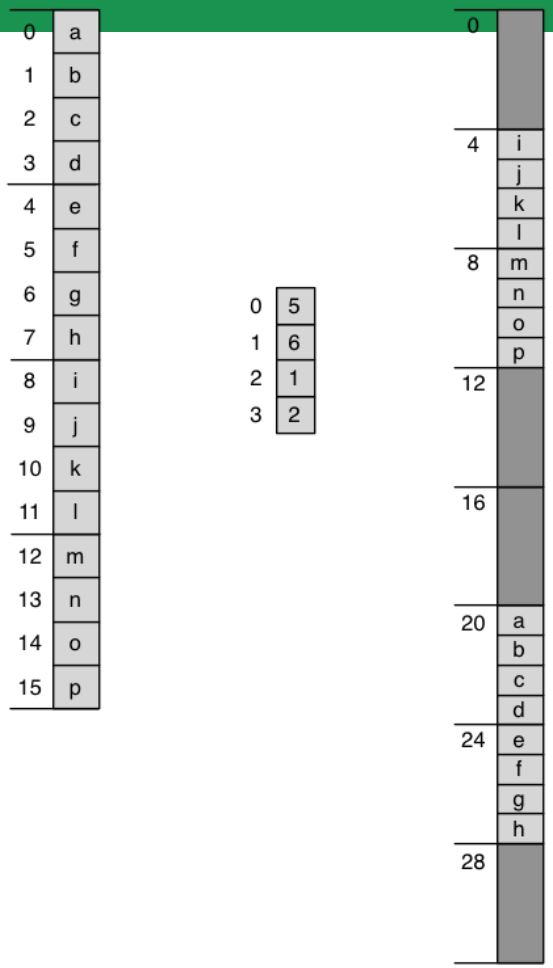


Figura: Hardware de paginación

Método básico

- Con un esquema de paginación no se tiene fragmentación externa, pero si fragmentación interna.
- Si el tamaño de los procesos es independiente del tamaño de las páginas, se puede esperar que la fragmentación interna se en promedio igual a media página por proceso.
- Hoy en día las páginas utilizadas se encuentran entre 4KB y 8KB.

Escenario

Si cada entrada de la tabla de páginas tiene 4 bytes de longitud. Una entrada de 32 bits puede apuntar a una de 2^{32} marcos de página físicos. Si el tamaño de marco es de 4KB, entonces un sistema con entradas de 4 bytes podrá direccionar 2^{44} bytes de memoria física.

Método básico

- Cuando un proceso llega al sistema para ejecutarlo, se examina su tamaño expresado en páginas.
- Cada página del proceso necesitará un marco.
- Si el proceso requiere n páginas, deberá haber disponibles al menos n marcos en memoria.
  1. Si hay disponibles n marcos, se les asignarán al proceso que acaba de llegar.
  2. La primera página del proceso se carga en uno de los marcos asignados y se incluye el número de marco en la tabla de páginas para este proceso.
  3. La siguiente página dr carga en otro marco y su número de marco se coloca en la tabla de páginas, y así sucesivamente.

Método básico

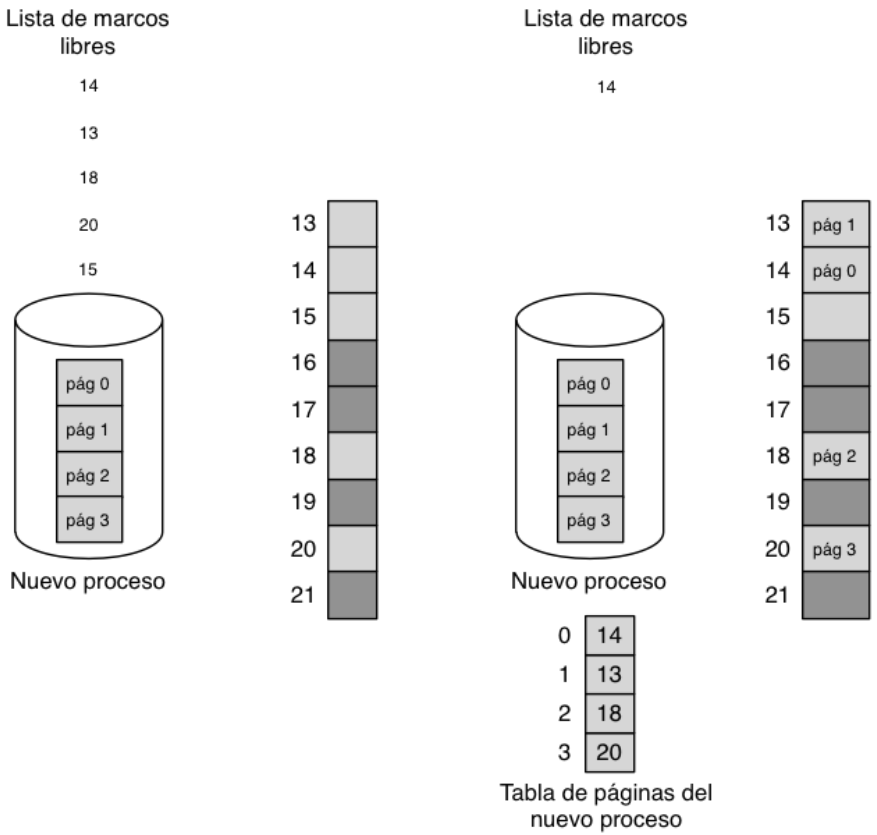


Figura: Marcos libres y asignados.

Soporte hardware

- La mayoria de los sistema operativos asignan una tabla de páginas para cada proceso.
- Almacenan un apuntador a la tabla de páginas en el bloque de control del proceso.
- La tabla de páginas se implementa se implementa como un conjunto de **registros** de dicados.
- Los registros se deben construir con lógica de muy alta velocidad, con el objetivo que la traducción de direcciones sea eficiente.

Soporte hardware

- Cada acceso a la memoria debe pasar a través del mapa de paginación.
- Las instrucciones para cargar o modificar los registros de la tabla de páginas son exclusivas del sistema operativo.
- Se utiliza un hardware especial de tamaño reducido y de rápido acceso denominado **buffer de consulta de traducción** (TLB).
- El número de entradas del TLB es de entre 64 y 1024.

Soporte hardware

- Su uso es el siguiente:
  1. El buffer TLB contiene sólo unas cuantas entradas de la tabla de páginas.
  2. Cuando se genera una dirección lógica, se presenta un número de página al TLB.
  3. Si se encuentra ese número de página, su número de marco estará inmediatamente disponible.
  4. Si el número de página no está disponible, es necesarió hacer referencia a la memoria para consultar la tabla de página.
- Una vez obtenido el número de marco, este se utiliza para acceder a la memoria.
- Se puede añadir el número de página y el numero de marco al TLB para los accesos futuros.

Soporte hardware

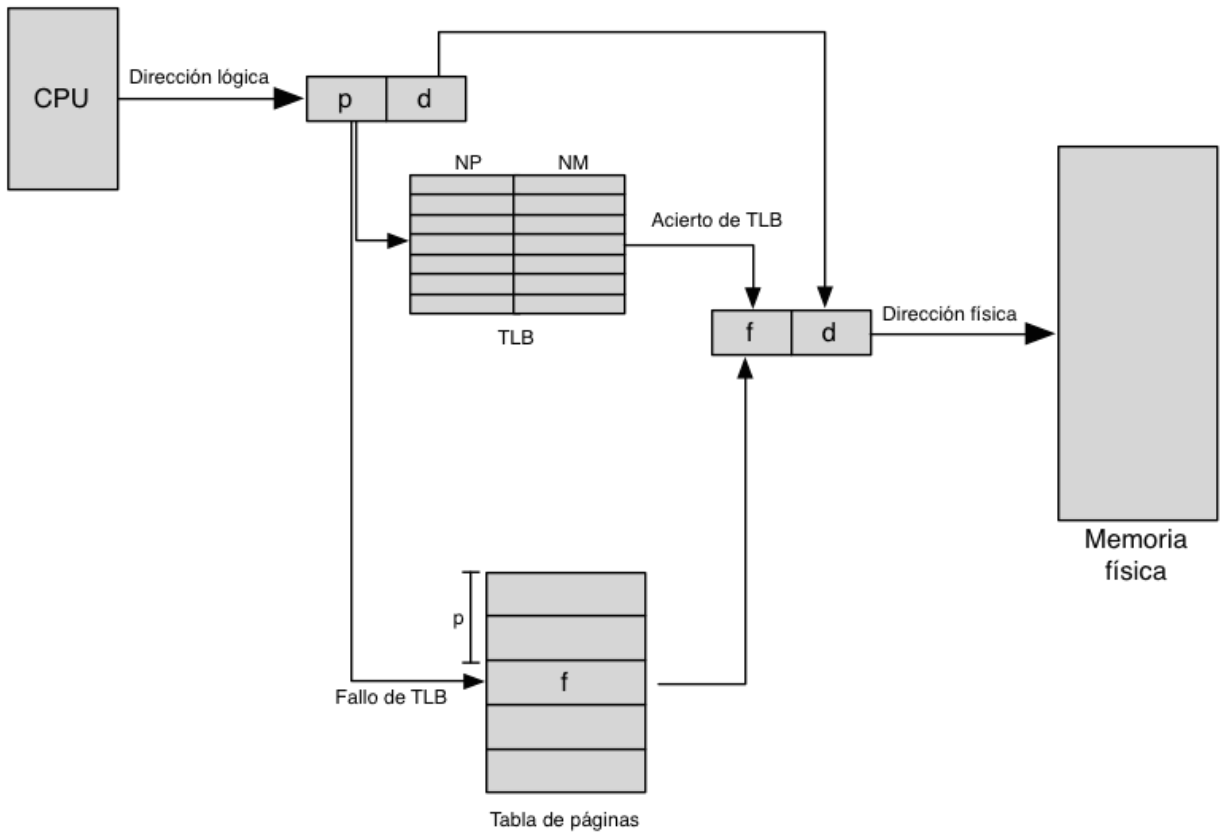


Figura: Hardware de paginación con TLB.

Protección

- La protección de memoria en un entorno paginado se consigue mediante una serie de bits de protección asociados con cada marco.
- Dichos bits se mantienen en la tabla de páginas.
- Uno de los bits puede definir a una página como de lectura-escritura o dolo de lectura.
- Toda referencia a la memoria pasa a través de la tabla de páginas con el fin de encontrar el número de marco correcto.
- Al mismo tiempo que se calcula la referencia se pueden comprobar los bits de protección.
- Cualquier intento anomalo provocará una interrupción hardware al sistema operativo.

Protección

- Se suele asociar un bit adicional con cada entrada de la tabla de páginas:
  - **Válido:** la página asociada se encontrará dentro del espacio de direcciones lógicas del proceso.
  - **Inválido:** la página no se encuentra dentro del espacio de direcciones lógicas del proceso.
- El sistema operativo configura dicho bit para permitir o denegar el acceso a dicha página.

Protección

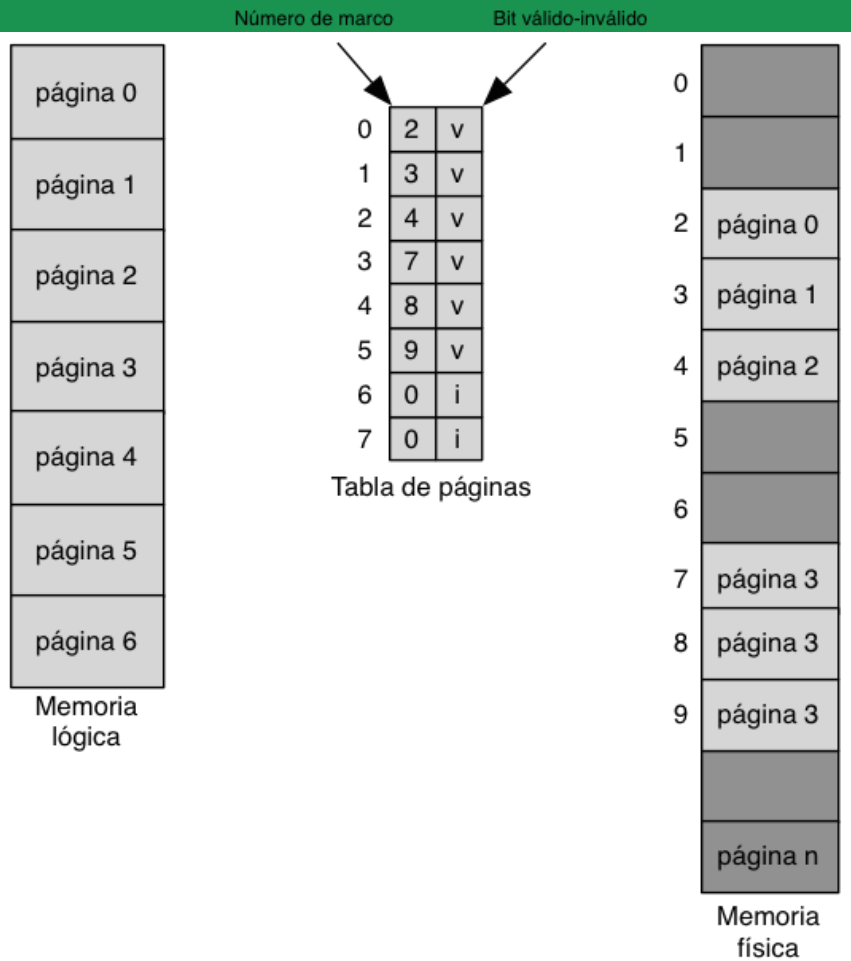


Figura: Bit válido-inválido.

Páginas compartidas

- Una ventaja de la paginacióm es la posibilidad de compartir código en común.
- Lo cual es muy útil en ambientes de tiempo compartido.

Ejemplo

Considere un sistema que de soporte a 40 usuarios, cada uno de los cuales esté ejecutando un editor de texto.
Si el editor de texto está compuesto por 150 KB de código y 50 KB de espacio de datos, se necesitarían 8000 KB para dar soporte a los 40 usuarios.

Páginas compartidas

- El código reentrante es código que no se automodifica durante el tiempo de ejecución.
- Dos o más procesos pueden utilizar el mismo código, si éste es reentrante.
- Cada proceso tendrá su propia copia de los registros y del almacenamiento de datos.

Páginas compartidas

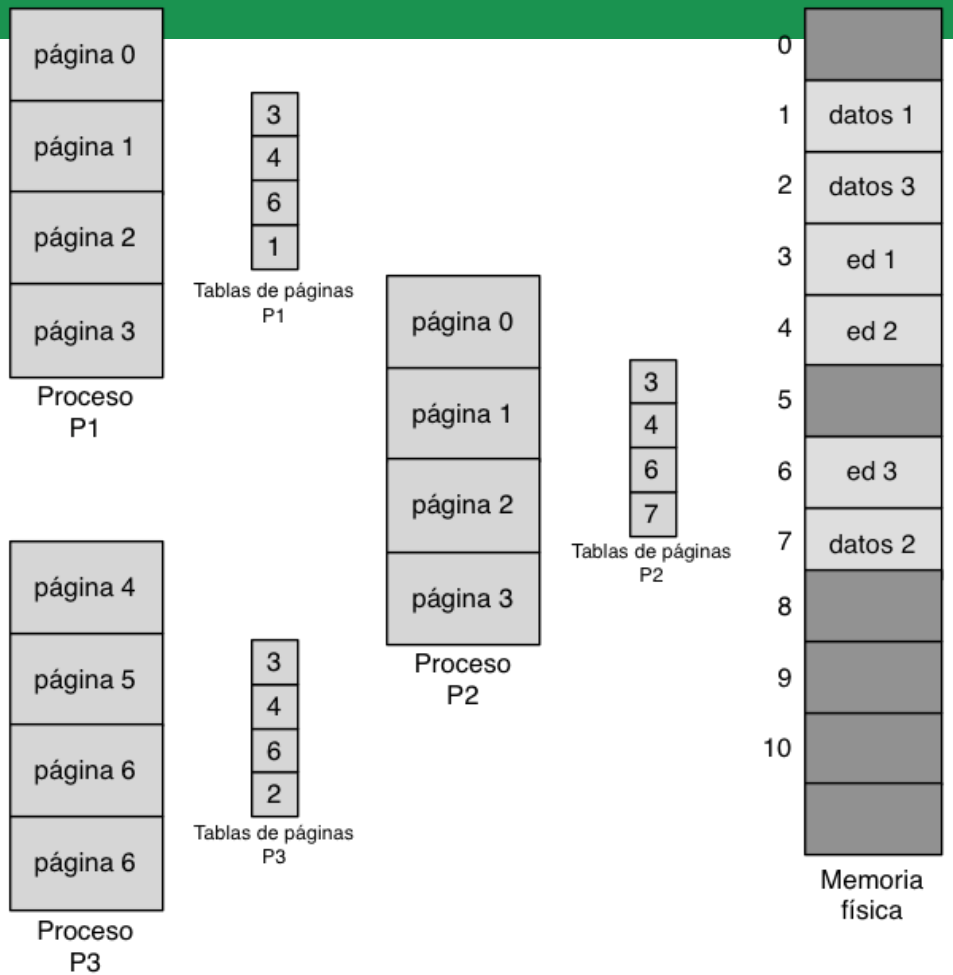


Figura: Páginas compartidas.

Índice

Paginación jerárquica

- La mayoría de los sistemas modernos soportan un gran espacio de direcciones lógico (2^{32} a 2^{64}).
- En dichos entornos la tabla de páginas puede llegar a ser muy grande.

Ejemplo

Considere un sistema con un espacio de direcciones lógico de direcciones de 32 bits. Si el tamaño de página en dicho sistema es de 4 KB (2^{12}), entonces la tabla de páginas puede estar compuesta por $\frac{2^{32}}{2^{12}}$ entradas.

- Una forma de minimizar la cantidad de memoria es utilizar un algoritmo en dos niveles.

Paginación jerárquica

- De modo que la dirección tendria la siguiente estructura:

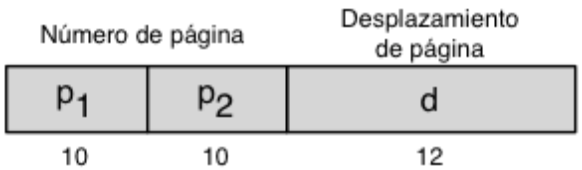


Figura: Estructura de la dirección

- Donde:
  - p_1: es un índice a la tabla de páginas externa.
  - p_2: es el desplazamiento dentro de la página de la página de la tabla de páginas externa.
  - d: es el desplazamiento dentro de la página.

Paginación jerárquica

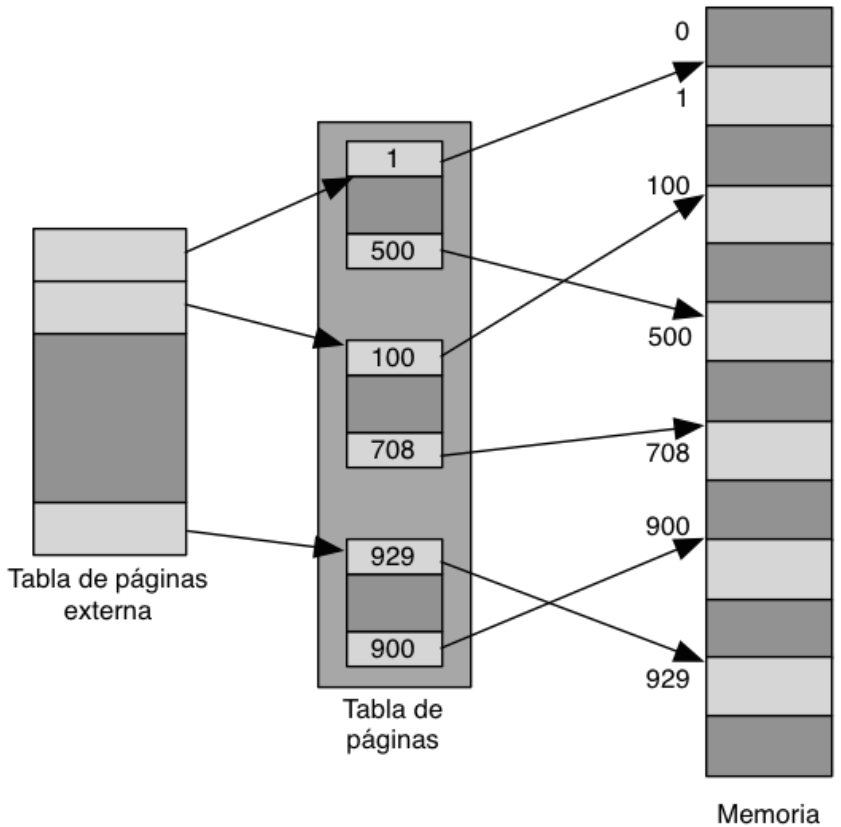


Figura: Tabla de páginas en dos niveles.

Paginación jerárquica

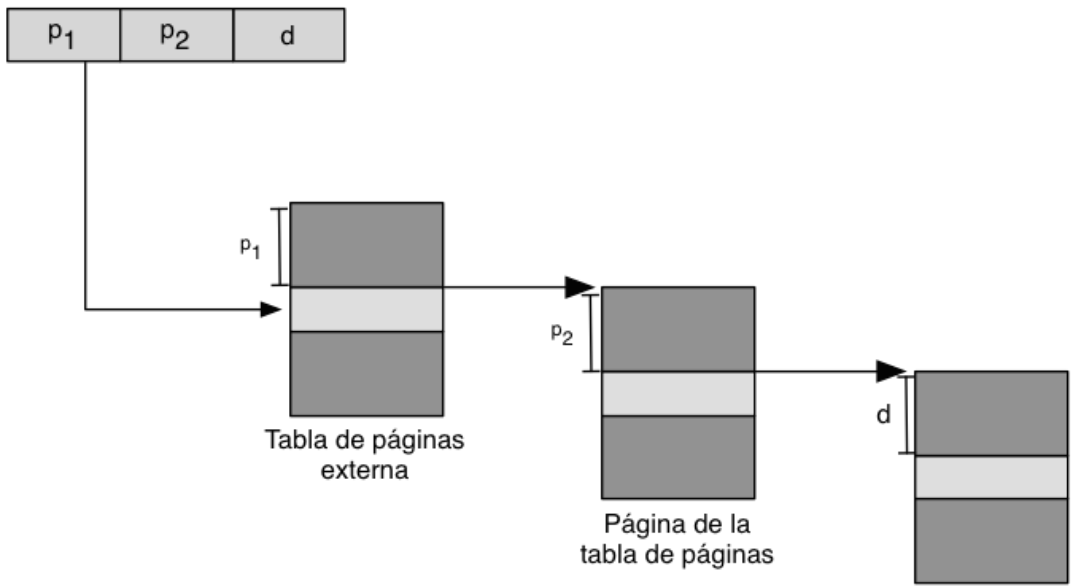


Figura: Tabla de páginas en dos niveles.

Tablas de páginas hash

- Una técnica común para gestionar los espacios de direcciones superiores a 32 bits sonsite en utilizar una **tabla hash** de páginas.
- Cada entrada de la tabla hash contiene una lista enlazada de los elementos que tienen como valor *hash* una misma ubicación.
- Cada elemento está compuesto de tres campos:
  1. El número de la página virtual
  2. El valor del marco de página mapeado
  3. Apuntador al siguiente elemento de la lista

Tablas de páginas hash

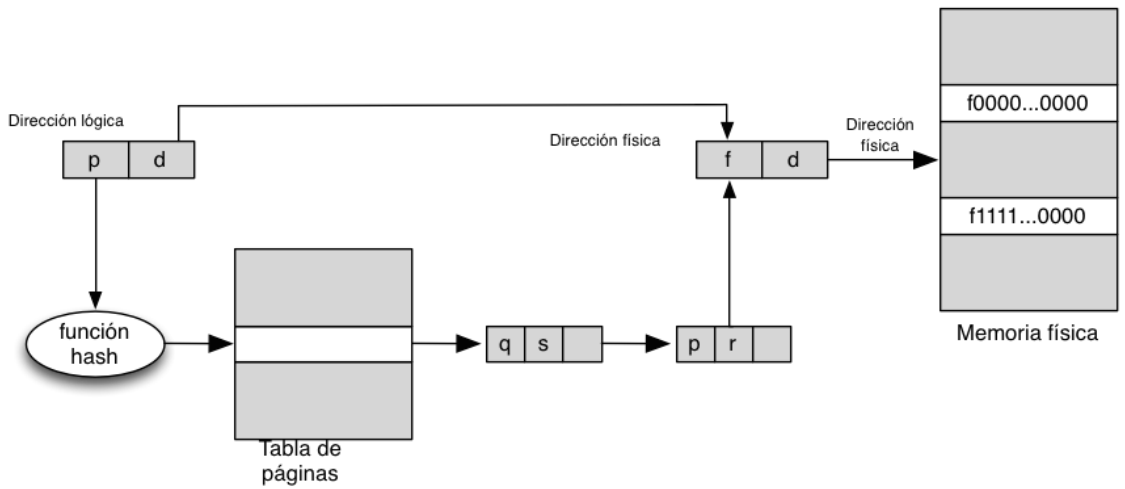


Figura: Tabla de páginas hash.

Tablas de páginas invertidas

- Usualmente, cada proceso tiene una tabla de páginas asociada.
- La tabla de páginas incluye una entrada por cada página que el proceso esté utilizando.
- El sistema operativo debe traducir cada referencia a una dirección física.

Desventaja

Cada tabla de páginas puede estar compuesta por millones de entradas y pueden ocupar gran cantidad de memoria física.

- Para resolver el problema se puede utilizar una tabla de páginas invertida.

Tablas de páginas invertidas

- Las tablas de páginas invertidas tiene una entrada por cada marco de memoria.
- Cada entrada está compuesta de la dirección virtual de la página almacenada en dicha ubicación de memoria real e incluye información del proceso que posee dicha página.

Ventaja

En el sistema habrá una única tabla de páginas y esa tabla sólo tendrá una entrada por cada página de memoria física.

- Las tablas de páginas invertidas requieren a menudo que se almacene un identificador del espacio de direcciones en cada entrada de la tabla de páginas.

Tablas de páginas invertidas

Traducción de direcciones

- Cada entrada de la tabla de páginas invertida es una pareja $< id - proceso, numero - pagina >$.
- La traducción de direcciones realiza como sigue:
  1. Cuando se produce una referencia a memoria, se presenta al subsistema de memoria una parte de la dirección virtual $< id - proceso, numero - pagina >$.
  2. Se explora la tabla de páginas invertida en busca de una correspondencia.
  3. Si se encuentra la correspondencia (ej. i), se generará la dirección física <i,desplazamiento>.
  4. Si no se encuentra la correspondencia, quierra decir que se ha realizado un intento de acceso ilegal a una dirección.

Índice

Método básico

Definición

La segmentación es un esquema de gestión de memoria que soporta la visión de la memoria que tienen los usuarios.

- Un espacio lógico de direcciones es una colección de segmentos y cada segmento tiene un nombre y una longitud.
- Las direcciones especifican tanto el nombre del segmento como el desplazamiento dentro de ese segmento.
- Por simplicidad de implementación, los segmentos están numerados y se hace referencia a ellos mediante el número de segmento, en lugar de utilizar el nombre del segmento.
- Una dirección lógica tiene la estructura: <número-segmento,desplazamiento>.

Método básico

Definición

La segmentación es un esquema de gestión de memoria que soporta la visión de la memoria que tienen los usuarios.

- Normalmente, el programa de usuario se compila y el compilador construye automaticamente los segmentos para reflejar el programa de entrada.
- Un compilador de C, podría generar segmentos separados para los siguientes elementos:
  1. El código
  2. Las variables globales
  3. El cúmulo de memoria
  4. Las pilas utilizadas por hilos independientes de ejecución
  5. La biblioteca C estándar

Hardware

- El mapeo de direcciones se realiza mediante la **tabla de segmentos**.
- Cada entrada de la tabla de segmentos tiene una *dirección base del segmento* y un *límite del segmento*.
  - *base:* contiene la dirección fisica inicial del lugar donde el segmento reside dentro de la memmoria
  - *límite:* especifica la longitud del segmento
- Una dirección lógica estará compuesta de dos partes:
  1. número de segmento (s): se utiliza como índice para la tabla de segmentos.
  2. desplazamiento (d): debe estar comprendido entre 0 y el límite del segmento, si no lo está se producirá una interrupción hacia el sistema operativo.
- Cuando un desplazamiento es legal, se le suma a la dirección base del segmento para generar la dirección de memoria física del byte deseado

Hardware

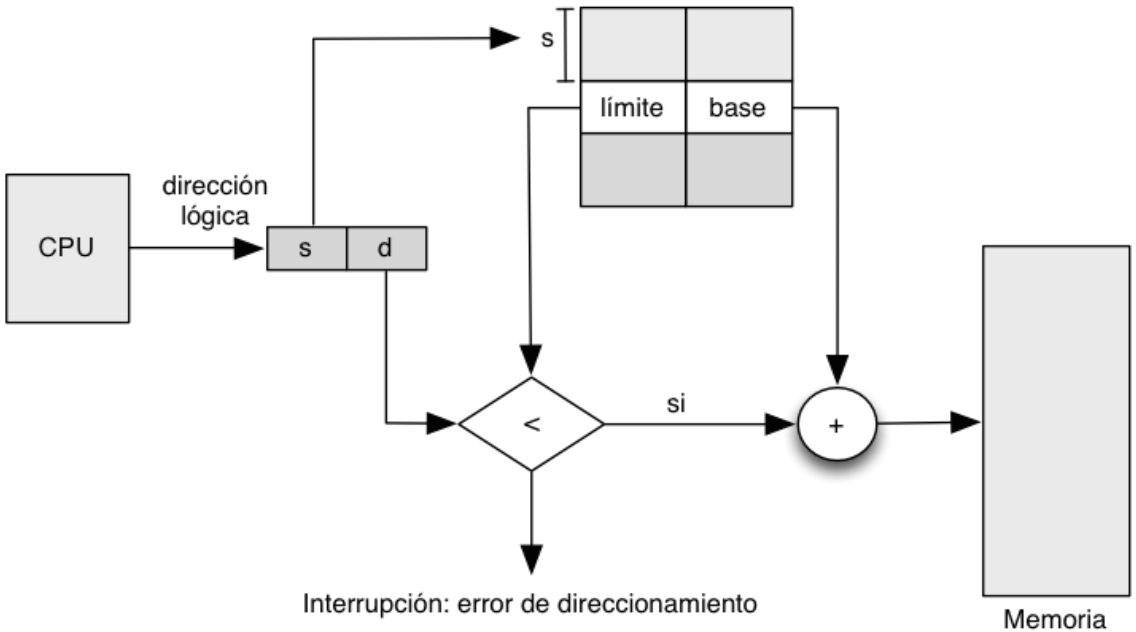


Figura: Hardware de segmentación.

Hardware

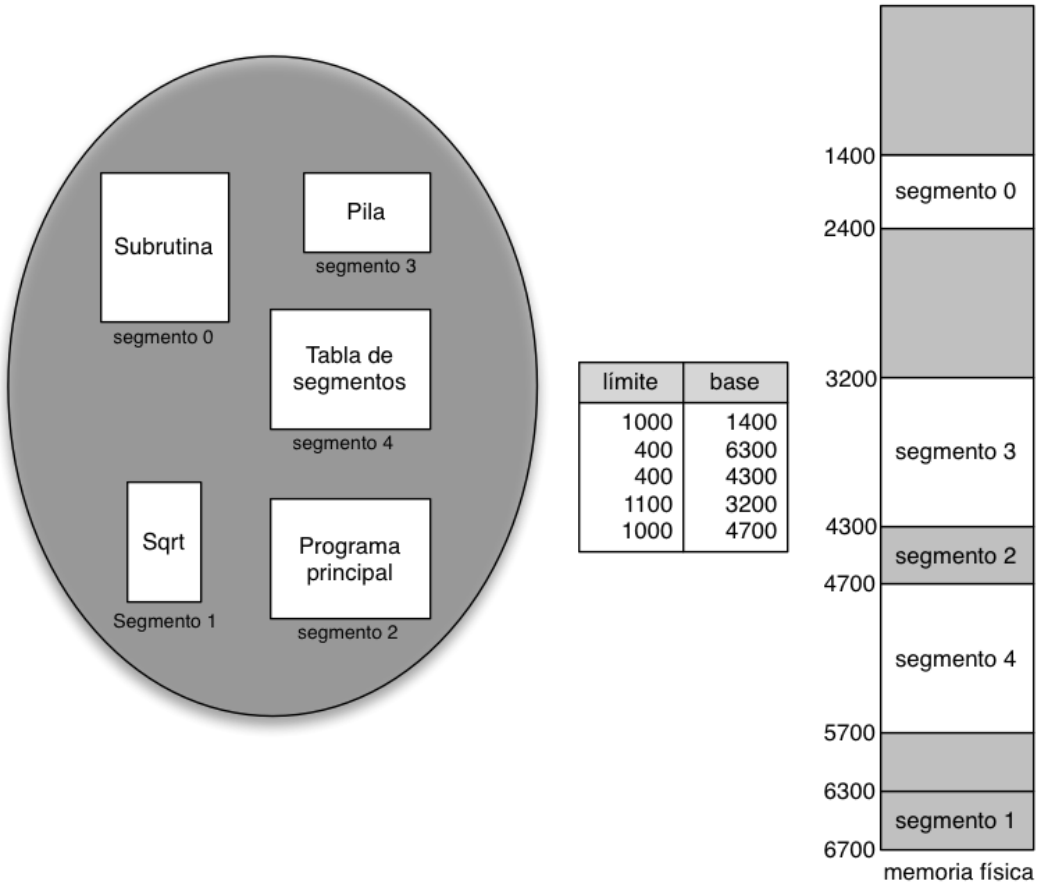


Figura: Ejemplo de segmentación