

4.3 Memoria virtual

- ❑ Tamaño de proceso, espacio para programa, pila, datos puede exceder la cantidad de memoria física disponible para él.
- ❑ El SO mantiene en memoria principal las partes del programa que se están usando y el resto en disco.

Paginación

- ❑ Instrucciones de máquina hacen acceso a memoria

```
move r1, [1000]
```

- ❑ En máquinas sin memoria virtual la dirección se coloca directamente en el bus de memoria.
- ❑ En máquinas con memoria virtual las direcciones pasan por la Unidad de Administración de Memoria que las transforma en direcciones de memoria física.
- ❑ Se entiende por **direcciones virtuales** a las que están presentes en los programas. Estas direcciones virtuales están dentro del espacio de **direcciones virtuales**.
- ❑ Se entiende por **direcciones reales** a las generadas por la MMU. Estas son direcciones físicas.
- ❑ El espacio de direcciones virtuales se divide en unidades llamadas **páginas**. Las unidades correspondientes en memoria física se denominan **marcos de página**.

- Ejemplo de páginas y marco de páginas

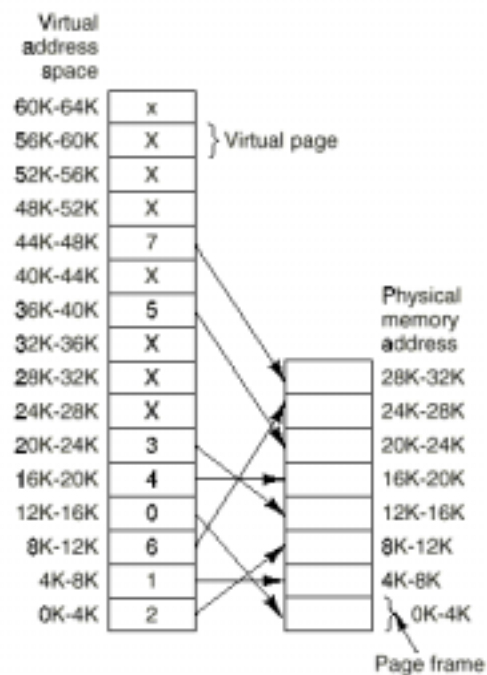


Figure 4-8. The relation between virtual addresses and physical memory addresses is given by the page table.

- Transformación entre direcciones virtuales y direcciones físicas

move r1,[0] => Dirección 0, página 0 (0k-4k), marco de página 2(8k-12k).

move r1,[8192] => move r1,[24576]

- Acceso a páginas sin correspondencia
 - Bit presente ausente. Registro cuyos bits indican la presencia o ausencia de una página en memoria real.
 - move r1, 32780 (byte 12 dentro de la página virtual 8, 32768)

- Se produce falla de página.
 - CPU pasa el control al SO
 - SO escoge un marco de página poco utilizado
 - Escritura en disco de marco de página
 - Página llamada se copia en marco de página vacío
 - Modificación del mapa:
 - Página que ya no tiene correspondencia
 - La que ahora tiene
 - Reinicia instrucción atrapada

- Funcionamiento interno de MMU

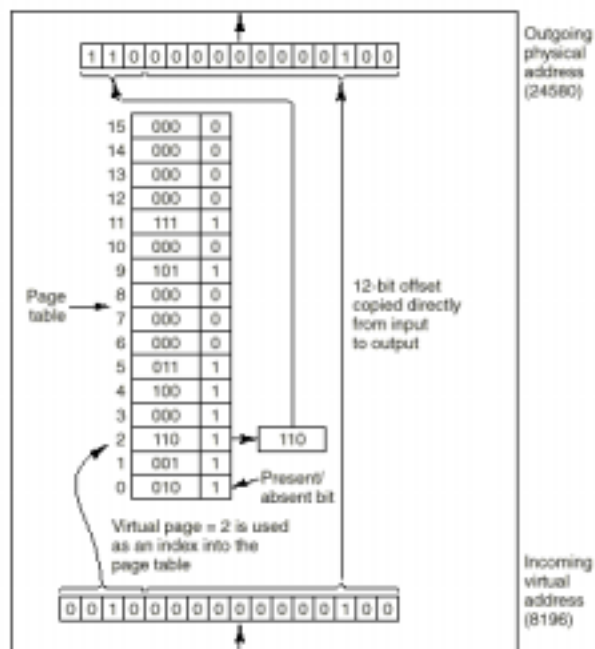


Figure 4-9. The internal operation of the MMU with 16 4K pages.

- Dirección virtual entrante => Mapa de MMU => dirección física saliente.

- Dirección virtual:
 - Número de página, 4 bits.
 - Distancia, 12 bits.

- Dirección física:
 - Número de marco: 3 bits.
 - Distancia, 12 bits.

Tablas de página

- ❑ El propósito de la tabla de páginas es transformar páginas virtuales en marcos de páginas reales.
- ❑ Cada proceso necesita su propia tabla de página.
- ❑ Problemas asociados
 - ❑ Tamaño de tabla de página
 - ❑ direcciones de 32 bits con 4k por página
 - ❑ 1 millón de paginas => 1 millón de entradas en la tabla
 - ❑ Velocidad de transformación
 - ❑ Dependiendo de la instrucción es necesario 1, 2, o más referencias a memoria
- ❑ **Una tabla de página**
 - ❑ Arreglo de registros de HW rápidos para contener tabla.
 - ❑ Una entrada por página virtual
 - ❑ No requiere referencias a memoria durante la transformación
 - ❑ Alto costo
 - ❑ Cambio de contexto implica carga de tabla de página
- ❑ Tabla de páginas en memoria principal.
 - ❑ Se necesita un registro de HW que apunte al comienzo de la tabla.
 - ❑ Se requieren referencias a memoria para leer las entradas de la tabla durante la ejecución de una instrucción.

- **Tablas de páginas multinivel**

- Por el problema de tener tablas de páginas enormes en memoria se usan tablas de páginas multinivel.

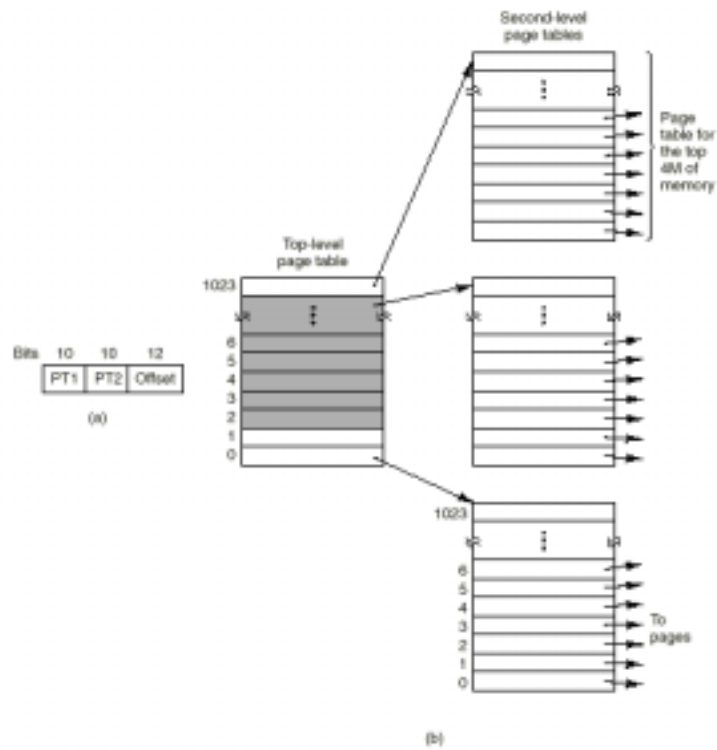


Figure 4-10. (a) A 32-bit address with two page table fields. (b) Two-level page tables.

- Sólo se mantienen en memoria las tablas de páginas que se utilizan.
- Tabla de mayor nivel tiene 3 entradas válidas: Programa, datos y pila.

- Detalle de una entrada de tabla

Habilitación Caché	Bits R	Bits M	Bits de Protección	Bit Presente	Número de Marco
--------------------	--------	--------	--------------------	--------------	-----------------

- Estructura depende de la máquina
- Contenidos:
 - Número de marco de página.
 - Bit presente ausente
 - Bits de protección
 - Indican que clases de accesos están permitidos.
 - 1bit: 0 indica lectura escritura. 1 indica sólo lectura.
 - 3 bits: Lectura, escritura y ejecución.
 - Modificación:
 - Indica si la página a sido modificada.
 - En caso de escritura de página bit a 1.
 - Referencia:
 - Se enciende cuando se hace referencia a una página.
 - Se usa para escoger una pagina no referida para desalojarla en caso de falla de página.
 - Caché:
 - Permite inhabilitar la colocación en caché de la página.
- **Buffers de consulta para traducción (TLB)**
- La programación estructurada hace que los programas hagan referencia a un número pequeño de páginas.
- TLB (buffer de consulta para traducción) o memoria asociativa: Pequeño dispositivo hardware para transformar las direcciones virtuales en físicas sin pasar por la tabla de páginas.
- Generalmente dentro de la MMU

- Cada entrada contiene información referente a una página

- Estructura de la memoria asociativa

Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	850	1	RW	14
1	851	1	RW	75

Figure 4-12. A TLB to speed up paging.

- Funcionamiento de TLB

- Verificación de número de página virtual está en el TLB(Se compara con todas las entradas en forma simultanea).
- Si hay coincidencia, el marco de página se toma desde el TLB sin recurrir a la tabla de páginas.
- Si número de página virtual está presente en TLB pero con conflicto de permiso hay falla de protección.
- Si no hay coincidencia:
 - Búsqueda en la tabla de páginas
 - Se desaloja una de las entradas de la TLB:
 - Se actualiza la entrada en tabla de página correspondiente a la entrada en TLB desalojada: Se copia bit modificado de entrada de TLB en tabla de página.
 - Sustitución de entrada desalojada en TLB por la entrada en tabla de página encontrada:
 - Se copia toda la entrada de tabla de página a entrada de TLB

- Enfoques de administración de TLB

- Administración de TLB (carga de entradas ante fallas) por HW de la MMU
- Cuando el tamaño del TLB es adecuado (64 entradas), es posible manejar eficientemente las tareas que tienen que ver con el funcionamiento del TLB por software.

❑ **Tablas de páginas invertidas**

- ❑ Tabla de páginas requiere una entrada por página virtual. Se busca el número de marco de página dependiendo del número de página virtual.
- ❑ El problema de tablas de páginas es su gran cantidad tamaño por elevada cantidad de entradas:
 - ❑ Espacio de direcciones de 64 bits => 2^{64} entradas
 - ❑ 4K por página => 10^{15} bytes por tabla de página
- ❑ Tabla de páginas invertidas:
 - ❑ Hay una entrada por cada marco de página (Reduce tamaño de tabla)
 - ❑ Dificultades en la traducción de dirección virtual a física. Ya no es posible usar el número de página virtual como índice de la tabla de páginas. Se debe buscar en toda la tabla el par n,p (número de proceso n, dirección virtual p). La búsqueda es lenta.
 - ❑ La búsqueda debe realizarse en cada referencia a memoria.
 - ❑ Solución: Uso de TLB de páginas invertidas.

4.4 Algoritmos de sustitución de páginas

- ❑ Ocurrencia de fallo de página
 - ❑ SO tiene que escoger la página que sacará de la memoria para entrar la nueva página.
 - ❑ Página eliminada fue modificada se debe reescribir en disco.
 - ❑ Si la página no fue modificada no será necesario escribirla a disco para actualizar su contenido.
 - ❑ Finalmente la página se sobrescribe en memoria.

- ❑ Es posible escoger una página al azar cuando hay fallo de página.
- ❑ Es mejor escoger una página no modificada para elevar el rendimiento.
- ❑ Si se escoge una página muy usada para eliminarla de memoria, es posible que se siga usando y que sea necesario traerla nuevamente, por lo que ocurrirá un fallo de página.

Algoritmo de sustitución de páginas óptimo

- ❑ Cada página podría rotularse con el número de instrucciones que se ejecutan antes que se haga referencia a ella. Para saber cual es la página que se ejecutará más tardíamente y ella reemplazarla.

- ❑ El algoritmo óptimo escoge la página que tenga el rótulo más alto.

- ❑ El algoritmo óptimo no se puede poner en práctica.
- ❑ En el momento de ocurrir un fallo de página no se puede saber cuando se hará referencia a cada una de las páginas.

- ❑ Es posible ejecutar el programa una vez y tomar nota del uso de páginas.
 - ❑ Con la información anterior implementar el algoritmo óptimo.
 - ❑ Se puede usar como medida de referencia para otros algoritmos.
 - ❑ Las referencias a páginas sólo son válidas para ese algoritmo.

Algoritmo de sustitución de páginas no recientemente usado NRU.

- ❑ La tabla de página contiene en sus campos información respecto al uso de ellas
- ❑ El campo R se enciende cada vez que se hace referencia a una página(lectura o escritura).
- ❑ M se enciende cuando se ha modificado una página.
- ❑ La actualización de los bits se debe realizar por hardware. Por software se pone a 0.

- ❑ Clases de página
 - 0: no referida, no modificada.
 - 1: no referida, modificada.
 - 2: referida, no modificada.
 - 3: referida, modificada.

- ❑ Para lograr clase 1, el SO debe borrar cada cierto tiempo el bit R.

- ❑ El algoritmo NRU elimina una página al azar de la clase no vacía que tiene el número más bajo.

Algoritmo FIFO (primera que entra, primera que sale).

- ❑ El sistema operativo mantiene una lista de todas las páginas que están en memoria.
- ❑ La página a la cabeza es la página más antigua y la que está a la cola la más reciente.
- ❑ Cuando hay falla de página se elimina la página cabeza y entra al final de la cola la página solicitada.
- ❑ Es posible que este algoritmo elimine una página antigua muy utilizada

Algoritmo de sustitución segunda oportunidad.

- Antes de eliminar la página más antigua se inspecciona el bit R
 - Si $R=0$, la página no ha sido referenciada últimamente. La página se elimina.
 - Si $R=1$, la página ha sido referenciada. Se pone bit R a 0 y se coloca la final de la cola.

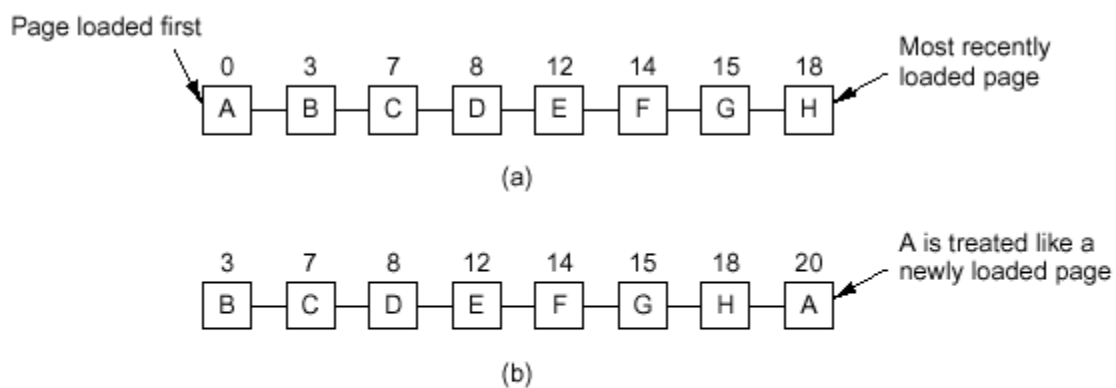
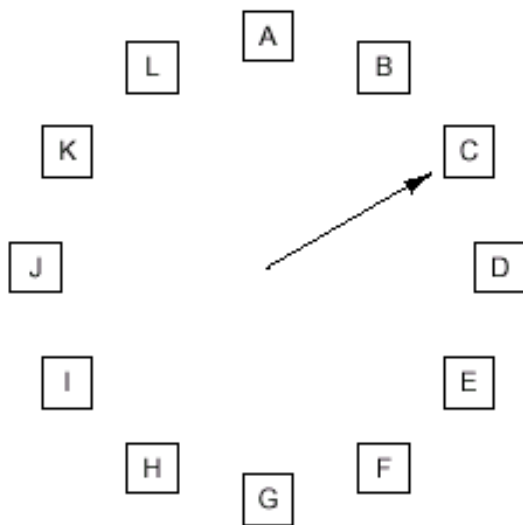


Figure 4-13. Operation of second chance. (a) Pages sorted in FIFO order. (b) Page list if a page fault occurs at time 20 and *A* has its *R* bit set.

- El algoritmo segunda oportunidad busca la página más vieja a la que no se ha hecho referencia desde el tiempo en que los bits *R* fueron puestos a cero.

Algoritmo de sustitución de páginas por reloj.

- ❑ Se mantiene una lista circular de páginas en memoria (disposición semejante a un reloj).
- ❑ Se tiene un apuntador a la página más vieja (semejante a una manecilla de reloj).
- ❑ Cuando hay fallo de página se examina el bit R de la página apuntada.
- ❑ Si $R=0$.
 - ❑ Se desaloja la página sustituyéndola por la que es referenciada.
 - ❑ Se apunta a la página siguiente (la avanza la manecilla).
- ❑ Si $R=1$
 - ❑ Se pone R a 0.
 - ❑ Se apunta a la siguiente página.



When a page fault occurs,
the page the hand is
pointing to is inspected.
The action taken depends
on the R bit:
R = 0: Evict the page
R = 1: Clear R and advance hand

Algoritmo de sustitución de páginas menos recientemente usada (LRU).

- ❑ Si una página ha sido muy referenciada por las últimas instrucciones es muy posible que se siga usando.
- ❑ Si una página no ha sido muy referenciada por las últimas instrucciones es muy posible que no se siga referenciando.
- ❑ LRU: Cuando hay una falla de página, se desaloja la que ha estado más tiempo sin usarse.

LRU Conceptual

- ❑ Mantención de lista enlazada con las páginas de memoria.
- ❑ La página más recientemente usada está en la cabeza de la lista.
- ❑ La página menos recientemente usada está en la cola de la lista.
- ❑ Problema: Se debe actualizar la lista con cada referencia a memoria.

Implementación HW de LRU mediante contador

- ❑ Se tiene un contador que se incrementa después de cada instrucción.
- ❑ Cada entrada a la tabla de página tiene un campo que puede contener al contador.
- ❑ Después de cada referencia a memoria el valor del contador se almacena en la entrada correspondiente de la página referenciada.
- ❑ Al ocurrir un fallo de página se examina la tabla y se elimina la del campo contador más bajo.

Implementación HW de LRU mediante matriz n x n

- Si hay n marcos de página. Se mantiene una matriz de “n x n” bits.
- Inicialmente todos los bits son 0’s.
- Cuando se hace referencia al marco k.
 - Se ponen a 1 los bits de la fila k.
 - Se ponen a 0 los bits de la columna k.
- La fila cuyo valor binario sea más bajo será la página menos recientemente utilizada.

Referencias de páginas: 0 1 2 3 2 1 0 3 2 3.

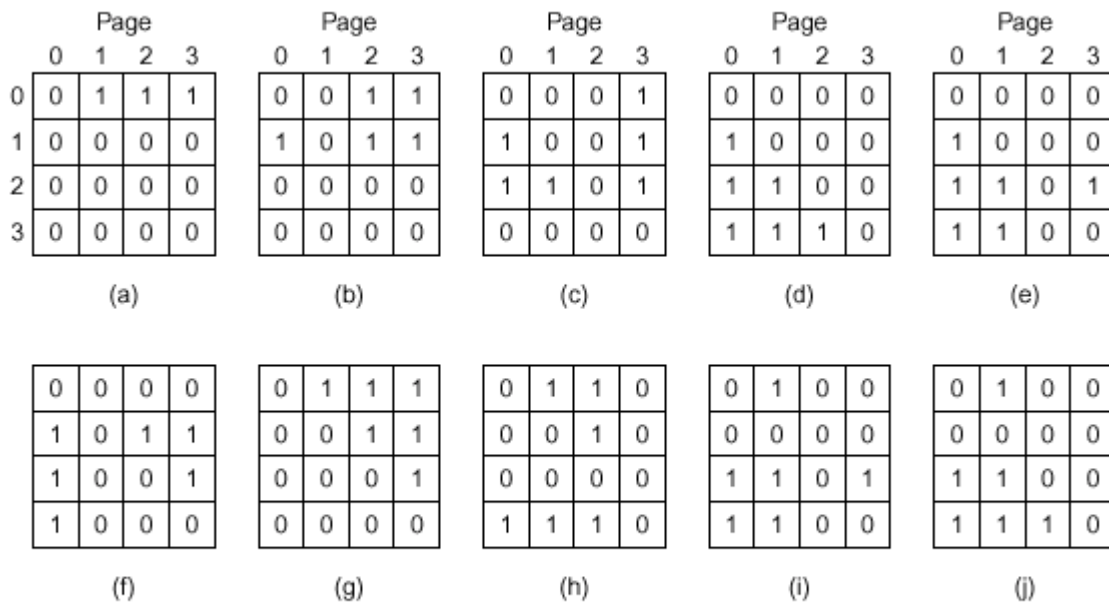
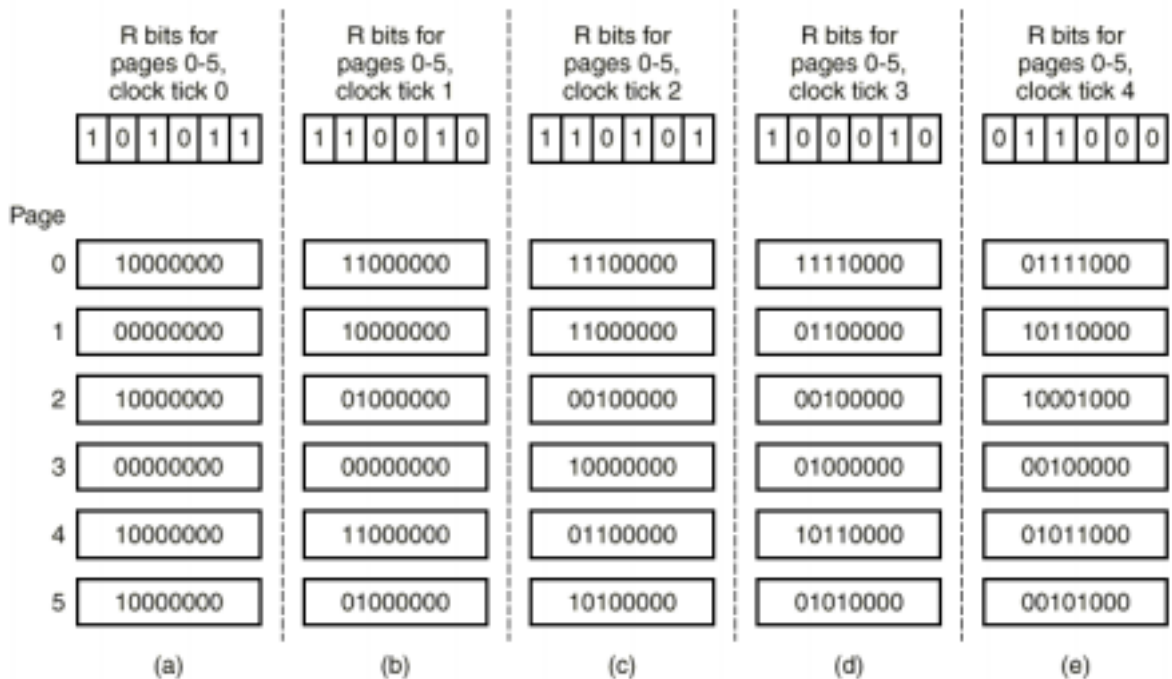


Figure 4-15. LRU using a matrix.

Simulación de algoritmo LRU en software

- Algoritmo LRU requiere hardware asociado.
- **Algoritmo NFU (no usada frecuentemente) solución software de LRU.**
 - Se requiere contador en software asociado a cada página
 - Cada cierto tiempo (en cada interrupción de reloj) se suma R a contador.
 - Contador "intenta" medir frecuencia de referencia a página.
 - NFU no olvida. Se reescriben páginas útiles.
- **Mejora de NFU. Algoritmo de maduración**
 - Todos los contadores binarios se desplazan a la derecha un bit antes de sumarles el bit R
 - Bit R se suma o "inserta" al bit de extrema izquierda.



4.5 Aspectos de diseño de sistemas con paginación

Modelo de conjunto de trabajo

- ❑ Paginación por demanda: Las páginas se cargan cuando son necesarias y no antes.
- ❑ Localidad de referencia: En una fase de ejecución de un proceso se hace referencia a un conjunto reducido de páginas.
- ❑ Conjunto de trabajo: Conjunto de páginas que usa actualmente el proceso.
- ❑ Modelo de conjunto de trabajo: Enfoque que sigue al conjunto de trabajo utilizado por cada proceso.
- ❑ Prepaginación: Carga de páginas antes que se necesiten.
- ❑ Implementación del conjunto de trabajo a través de maduración.
 - ❑ Cualquier página que tenga un 1 en los últimos n bits.
 - ❑ N de determina en forma experimental
- ❑ Algoritmo wsclock
 - ❑ Mejora al algoritmo de reloj.
 - ❑ Verificación de pertenencia al conjunto de trabajo si $R=0$.

Asignación local y global

- ❑ Además de escoger que página debe ser sustituida se debe considerar en que forma se debe repartir la memoria entre los procesos ejecutables
- ❑ Los procesos varían el tamaño de su conjunto de trabajo durante su vida.

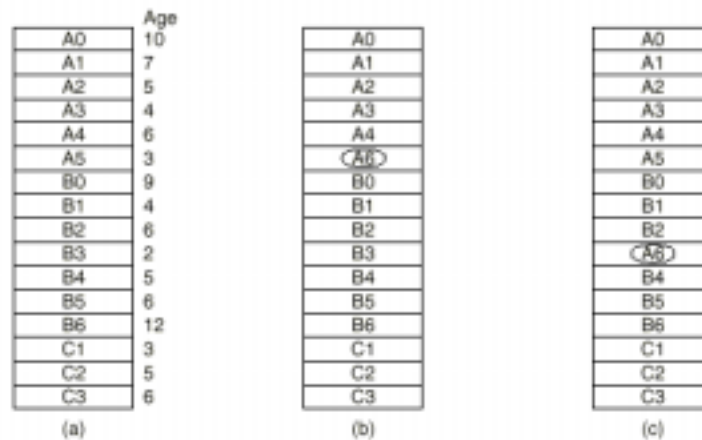


Figure 4-17. Local versus global page replacement. (a) Original configuration. (b) Local page replacement. (c) Global page replacement.

- ❑ Algoritmos de asignación local
 - ❑ Asignan a cada proceso un número fijo de marcos de memoria
 - ❑ Si aumenta el conjunto de trabajo aumentan los fallos de página.
 - ❑ Si disminuye el conjunto de trabajo se desperdician marcos de página.
- ❑ Algoritmos de asignación global
 - ❑ Reparten dinámicamente los marcos de páginas entre procesos ejecutables.
 - ❑ Varía número de marcos de página asignado a cada proceso.

- Enfoques de asignación de marcos
 - Examinar los bits de maduración: No considera cambios en el conjunto de trabajos entre ticks.
 - Existencia de algoritmo de asignación de marcos de páginas
 - El algoritmo debe determinar en número de procesos en ejecución.
 - Asignación equitativa
 - Asignación proporcional al tamaño.
 - Algoritmo de **frecuencia de fallas de páginas** PFF

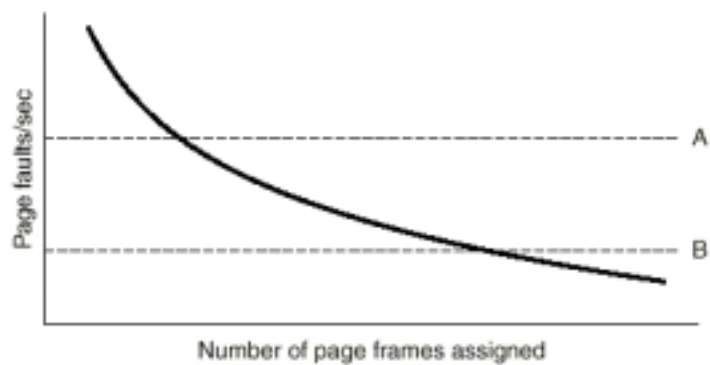


Figure 4-18. Page fault rate as a function of the number of page frames assigned.

- Tasa de falla muy alta: Implica asignación de marcos de páginas para reducción de fallas.
- Tasa de falla muy baja: Implica demasiados marcos de página asignados.
- PFF intenta mantener tasa de falla entre límites aceptables.
- Desalojamiento de procesos de memoria en caso de muchos procesos con altas tasas de falla.

Tamaño de página

- La determinación del tamaño de página óptimo requiere balancear varios factores opuestos.
- Tamaño de página pequeño => Fragmentación Interna: Espacio no ocupado por la última página del proceso.

n: número de proceso.

p: Tamaño de página

$$\text{Espacio desperdiciado} = n * (p/2)$$

- Tamaño de página pequeño:

Necesidad de muchas páginas por proceso => Tablas de páginas grandes.

- Diseño de tamaño de página

s: Tamaño promedio de proceso.

p: Tamaño de páginas en Bytes.

e: Ancho de Tabla de página (largo de la fila).

Número de páginas por proceso = s/p

Tamaño tabla de páginas = $s/p * e$

Fragmentación interna por proceso = $p/2$

Gasto extra de memoria por proceso = $se/p + p/2$

El objeto es minimizar el gasto extra

$$p = (2se)^{1/2}$$

4.6 Segmentación

- Paginación maneja un esquema de direcciones virtuales lineales.
- Manejo de espacios de direcciones virtuales independiente por crecimiento y decrecimiento de datos en memoria.

Manejo de distintas tablas por compilador

- 1.- Tabla que contenga texto fuente.
- 2.- Tabla que contenga nombre de variables y sus atributos.
- 3.- Tabla de contenga constantes enteras y punto flotante.
- 4.- Tabla que contenga análisis sintáctico del programa.
- 5.- Pila empleada para llamada a procedimiento.

- Asignación lineal de memoria.

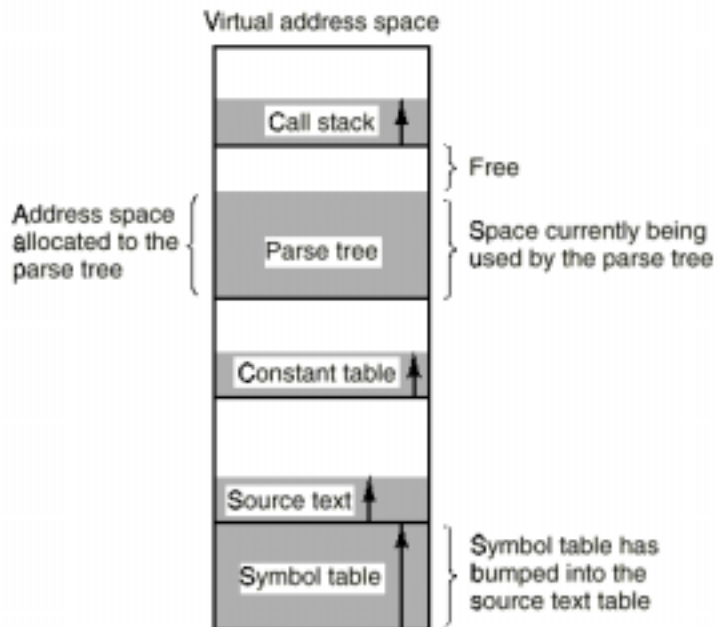


Figure 4-19. In a one-dimensional address space with growing tables, one table may bump into another.

❑ Segmentación

- ❑ Proveer a la máquina con muchos **segmentos** de espacios de memoria independientes
- ❑ Objetivo de la segmentación: Liberar al programador de la administración de tablas en expansión y contracción.
- ❑ Características de los segmentos
 - ❑ Segmento es una secuencia lineal de direcciones (desde 0 a un máximo)
 - ❑ Longitudes de segmentos pueden ser diferentes.
 - ❑ El tamaño del segmento puede variar durante la ejecución de los programas.
- ❑ Direccionamiento de memoria segmentada
 - ❑ Número de segmento.
 - ❑ Dirección dentro del segmento.

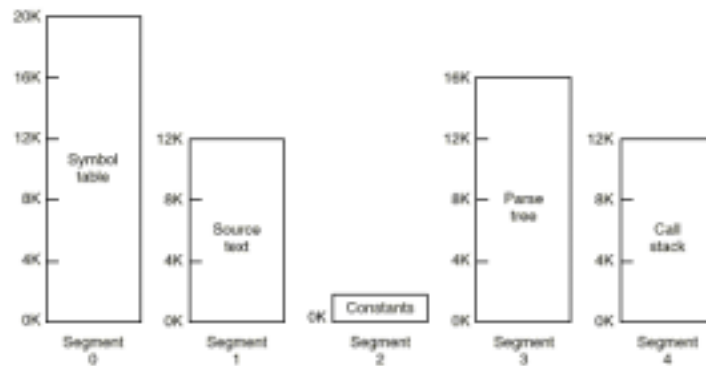


Figure 4-20. A segmented memory allows each table to grow or shrink independently of the other tables.

- Ventajas de la segmentación
 - Simplificar manejo de estructura de datos que crecen y se encojen.
 - Simplificación de enlazado de procedimientos cuando los procedimientos ocupan segmentos distintos.
 - No se afectan direcciones de inicio de otros procedimientos
 - Facilita compartir procedimientos o datos entre procesos. Segmento compartido.
 - Segmentos con diferentes permisos de protección.

- Fragmentación externa y compactación.

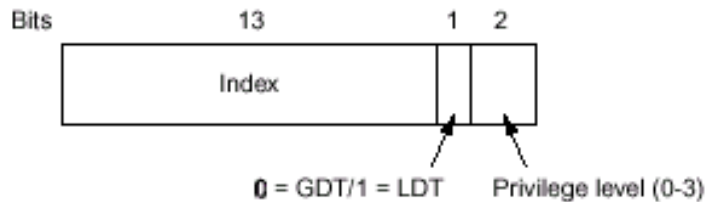
SEGMENTACIÓN CON PAGINACIÓN: PENTIUM INTEL

- **Multics y Pentium manejan la memoria virtual en presencia de segmentación y paginación.**
 - Multics:
 - 256 K segmentos
 - Cada segmento con 64 K palabras de 36 bits.
 - Pentium:
 - 16 K segmentos
 - Mil millones de palabras de 32 bits.
- **Manejo de memoria virtual de Pentium tiene como base 2 Tablas:**
 - **Tabla de descriptores Local (LDT):**
LDT describe segmentos que son locales a los programas.
Incluye segmento de código, segmento de datos, segmento de pila.
Una tabla LDT por programa.
 - **Tabla de descriptores Global (GDT):**
Describe segmentos compartidos por los programas o segmentos de sistema.
Incluye segmento para sistema operativo.
Una tabla para todos los programas.

- **Procedimiento de conversión de par (selector, distancia) a dirección física.**

1.- Un programa debe cargar un selector en un registro de segmento de la máquina

- Un selector es un número de 16 bits como se muestra en la figura



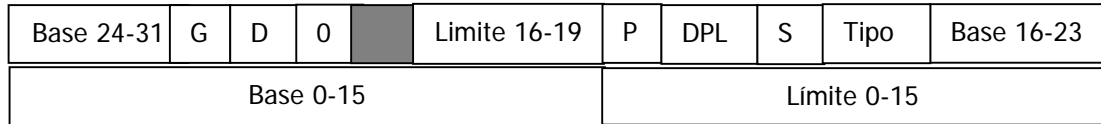
- Estructura de selector
 - 1 bit especifica la tabla al que pertenece el segmento.
 - 13 bits especifican el número de la entrada de tabla LDT o GDT.
 - 3 bits para nivel de privilegio.
- Pentium tiene 6 registros de segmentos
 - CS: Para código de segmento
 - DS: Para segmento de datos

2.- Ubicar la entrada correspondiente de la GDT o LDT y dar un apuntador directo al descriptor. Lo anterior se realiza el siguiente procedimiento.

- Se copia el selector del registro de segmento a un registro temporal interno
- Se ponen a 0 los tres bits menos significativos del registro temporal que contiene el selector
- Se suma la dirección de la tabla al contenido del registro temporal.

2.- Se extrae de la tabla LDT o GDT la entrada correspondiente (descriptor de segmento) al índice (index) y se almacena en registros internos que se manejan a nivel de microprograma.

- Un descriptor consiste en 8 bytes que incluye la dirección base del segmento, el tamaño, etc.



- Estructura del descriptor.

G: Tipo de límite.

0: Límite en bytes.

1: Límite en páginas.

D: Ancho palabras del segmento

0: Segmento de 16 bits.

1: Segmento de 32 bits.

P: Presencia o ausencia de segmento en memoria

0: Segmento ausente de la memoria.

1: Segmento presente en la memoria.

DPL: Nivel de privilegio.

S: Tipo de segmento

0: Sistema.

1: Aplicación.

Tipo: Hace referencia al tipo de segmento y protección.

3.- Verificación de existencia de segmento en memoria.

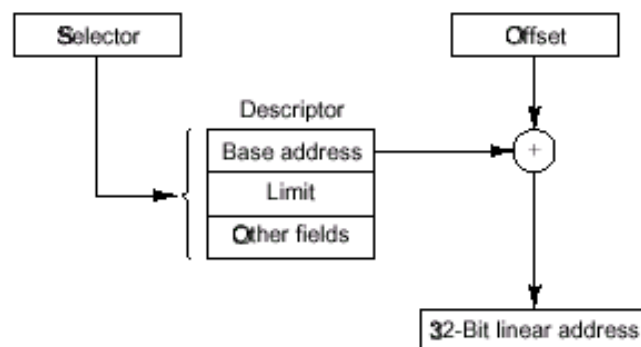
- Se analiza bit P

4.- Verificación si distancia dentro del segmento está dentro del límite del segmento.

- Se analiza Bit G y campo límite.
- Si $G = 0$, límite en bytes, con segmento de 1 Mbytes (2^{20})
- Si $G=1$, Tamaño de segmento en página. Si una página es 4Kbytes implica segmentos de 2^{32} bytes

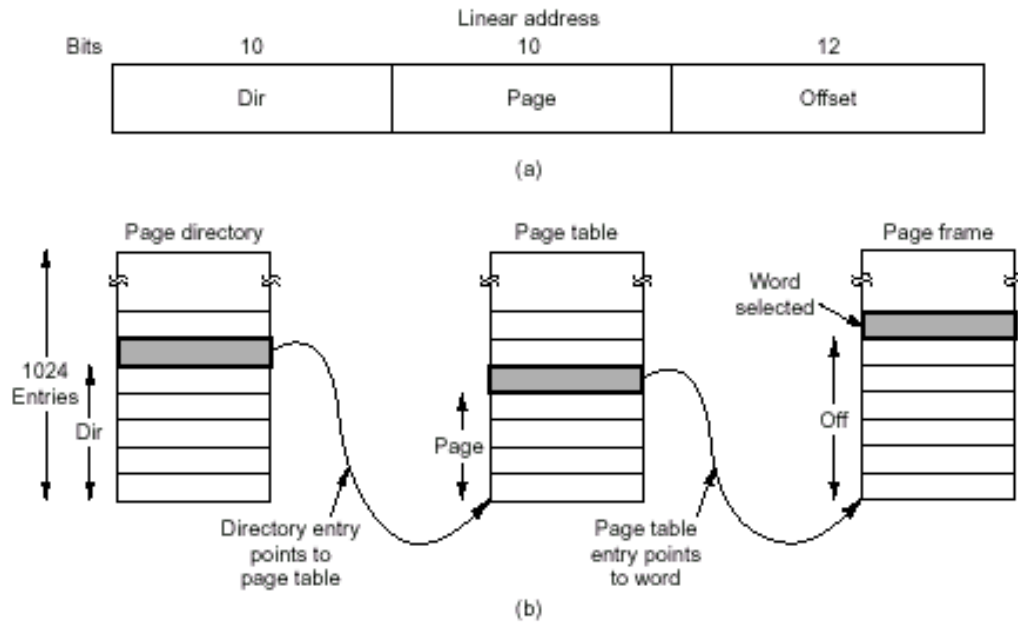
5.- Construcción de la **dirección lineal**

- Se suma el campo base de 32 bits del descriptor a la distancia según la siguiente figura



- Debido a que el campo base tiene 32 bits el segmento puede comenzar en cualquier lugar dentro de un espacio de 2^{32} palabras.
- Si no hay paginación la **dirección lineal** se interpreta como dirección física.

- Si hay paginación, la **dirección lineal** se interpreta como **dirección virtual**. Se usa un esquema de dos niveles para disminuir el tamaño de las tablas de páginas.



- **Uso de TLB**

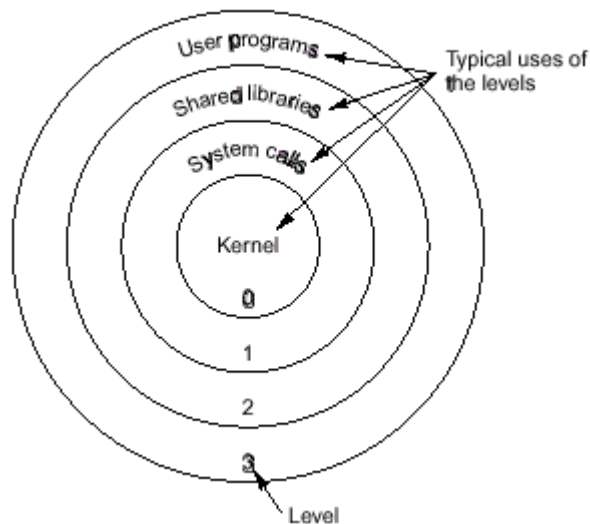
Pentium usa un TLB para transformar directamente las combinaciones (Dir-Página) en la dirección física del marco de página.

- **Observaciones**

- Si se usa solo paginación no tiene sentido un campo base distinto de 0. El campo base se usa para implementar segmentación pura.
- Si una aplicación sólo necesita segmentación todos los registros de segmento se llenan con un mismo selector, cuyo descriptor tiene base 0 y límite igual al máximo.

- **Protección**

- Pentium maneja 4 niveles de protección (0 más privilegiado, 3 menos privilegiado)
- En cada instante un programa en ejecución está en un nivel dado de protección.
- Cada segmento tiene un nivel de protección.
- No es problema si un programa usa segmentos de su propio nivel.
- Se pueden hacer llamadas a rutinas internivel si la llamada CALL usa en vez de una dirección un **selector**. Este selector designa un descriptor llamado **puerta de llamada** que da la dirección del procedimiento por invocar. Por lo tanto no es posible saltar a un lugar intermedio de código en niveles distintos.
- Las trampas y las interrupciones usan un mecanismo similar a las puertas de llamada. Hacen referencia a descriptores en vez de direcciones de memoria, y estos descriptores indican el procedimiento a ejecutar.
- El campo tipo del descriptor distingue en segmento de código, de datos, y de diferentes tipo de puertas de llamada.



SEGMENTACIÓN CON PAGINACIÓN EN MULTICS

CONCEPTOS

- Segmentación pura:
 - Existe un número de segmentos en memoria.
 - El segmento está completo en memoria.
 - Si existe fallo de segmento:
 - Se debe llevar el segmento faltante a la memoria.
 - Si no hay espacio para contener el nuevo segmento, se debe intercambiar un segmento a disco.
- Pueden existir segmentos muy grandes que sea imposible de mantener completos en memoria.
 - Se podría tener en memoria sólo las partes de los segmentos que estén utilizándose. Paginación de segmentos.

SEGMENTACIÓN PAGINADA EN MULTICS

- Usa segmentos paginados.
- Cada segmento es un espacio direcciones virtuales y se pagina.
- Cada proceso puede manejar hasta 2^{18} segmentos.
- Cada segmento tiene hasta 65536 palabras de 36 bits.
- Cada proceso maneja una tabla de segmentos.
- La tabla de segmentos tiene una entrada llamada descriptor de segmento.
- La tabla de segmentos se maneja en un segmento paginado.
- Cada descriptor contiene información si el segmento están en memoria principal.
- Si al menos alguna página del segmento está en memoria principal se considera que el segmento está en memoria.

- Una dirección virtual consta de dos campos:
 - Número de segmento
 - Dirección dentro del segmento
 - Número de Página
 - Desplazamiento de la página
- Al ocurrir una referencia a memoria se desarrolla el siguiente algoritmo
 - Se usa el número de segmento para encontrar el descriptor de segmento.
 - Se verifica si la tabla de página está en memoria.
 - Si está la tabla se le localiza.
 - Si no está en memoria ocurre una falla de segmento
 - También puede ocurrir una falla de protección.
 - Se examina la entrada de la tabla página correspondiente a la página virtual solicitada.
 - Si la página está en memoria se extrae la dirección de la base de la página en memoria (Se extrae marco de página).
 - Se agrega a la dirección del marco el desplazamiento dentro de la página virtual.
 - Si la página no está en memoria existe un fallo de página.

- Necesidad de uso de TLB:
 - Ejecución del algoritmo anterior se realiza por cada referencia a memoria. Ejecución de instrucciones lenta.
 - Se usa TLB para mejorar la velocidad de conversión de direcciones virtuales a reales.
 - Búsqueda en TLB
 - Se mantienen en el TLB la información de las páginas a que se ha hecho referencia últimamente (16 páginas usadas últimamente).
 - Se compara el número de segmento y número de página de la dirección virtual **en paralelo** con los campos correspondientes de todas las entradas
 - Si la página está en el TLB se obtiene la dirección real desde el TLB
 - Si la página no está en el TLB
 - Se encuentra la dirección en las tablas de memoria.
 - Se actualiza el TLB con la entrada requerida.
 - Se escoge una entrada a eliminar mediante algún algoritmo.
 - Se copia la información a la entrada de TLB