INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 2: MEMORIA

Motivación para Memoria Virtual

- Podemos pensar también que no todo el espacio de direcciones del proceso se necesita en todo momento.
    - Rutinas o librerías que se ejecutan una única vez (o nunca).
    - Partes del programa que no vuelven a ejecutarse.
    - Regiones de memoria alocadas dinámicamente y luego liberadas.
- No hay necesidad que la totalidad de la imagen del proceso sea cargada en memoria.

Como lo podemos trabajar…

- El SO puede traer a memoria las "piezas" de un proceso a medida que éste las necesita.
- Definiremos como "**Conjunto residente"** a la porción del espacio de direcciones del proceso que se encuentra en memoria RAM. (también como working set)
- Con el apoyo del HW:
    - Se detecta cuando se necesita una porción del proceso que no está en su Conjunto Residente.
    - Se debe cargar en memoria dicha porción para continuar con la ejecución.
- Ventajas:
    - Más procesos pueden ser mantenidos en memoria.
        - Solo se cargan algunas secciones de cada proceso.
        - Con más procesos en memoria RAM es más probable que existan más procesos Ready.
    - Un proceso puede ser mas grande que la memoria RAM (la cantidad requerida de RAM para el proceso total se excede).
        - El usuario no se debe preocupar por el tamaño de sus programas.
        - La limitación la impone el HW y el bus de direcciones.

¿Qué se necesita para aplicar la técnica de Memoria Virtual?

- El hardware debe soportar paginación por demanda (y/o segmentación por demanda),es decir que se meten paginas a medida que se necesiten.
- Un dispositivo de memoria secundaria (disco) que dé el apoyo para almacenar las secciones del proceso que no están en memoria principal (área de intercambio - swap).
- El SO debe ser capaz de manejar el movimiento de las páginas (o segmentos) entre la memoria principal y la secundaria.

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 2: MEMORIA

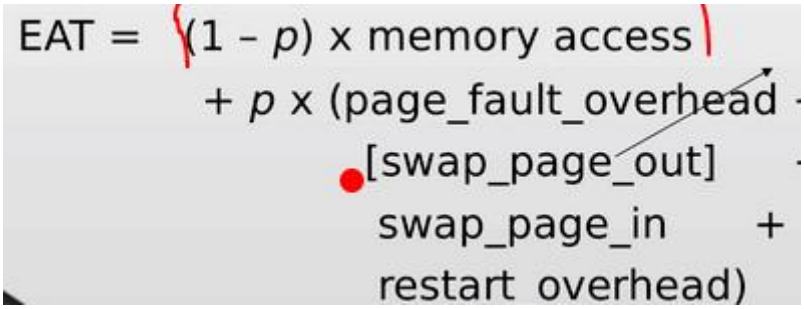
Memoria virtual con paginación

- ➔ Cada proceso tiene su tabla de paginas.
- ➔ Cada entrada en la tabla referencia al frame o marco en el que se encuentra la página en RAM.
- ➔ Cada entrada en la tabla de páginas tiene bits de control:
    - Bit V: indica si la página está en memoria. (cargada), el SO modifica el valor de este bit ya que es el encargado de subir o bajar paginas.
    - Bit M: indica si la página fue modificada. Si se modificó, en algún momento se deben reflejar los cambios en Memoria Secundaria (disco) para mantener una coherencia de datos. El bit modificado lo va cargando el hardware en sí (ya que este sabe sobre operaciones de escritura), el bit lo usa el SO.

Fallo de páginas(page fault)

- Ocurre cuando el proceso intenta usar una dirección que está en una página que no se encuentra en RAM. Bit V = 0.
    - La pagina no se encuentra en su conjunto residente.
- El HW detecta la situación y genera un trap al S.O
- El S.O podrá colocar al proceso en estado de “Blocked” (espera) mientras gestiona que la pagina que se necesite se cargue.
    - La carga consiste en:
        - El S.O busca un marco libre en la memoria y genera una operación de E/S al disco para copiar en dicho marco la página del proceso que se necesita usar.
        - El SO puede asignarle la CPU a otro proceso mientras se completa la E/S
            - La E/S avisará por interrupción cuando finalize.
    - Cuando la E/S finaliza, se notifica al SO y este:
        - Actualiza la tabla de páginas del proceso.
            - Coloca el Bit V en 1 en la pagina correspondiente.
            - Coloca la dirección base del frame donde se colocó la pagina.
        - El proceso que generó el fallo de página vuelve al estado de Ready.
        - Cuando el proceso, se ejecute se volverá a ejecutar la instrucción que provocó el fallo.
- **PERFORMANCE:**
    - Si los page faults son excesivos, la performance del sistema decae.
    - Tasa de page faults entre 0 y 1. $0 <= p <= 1$

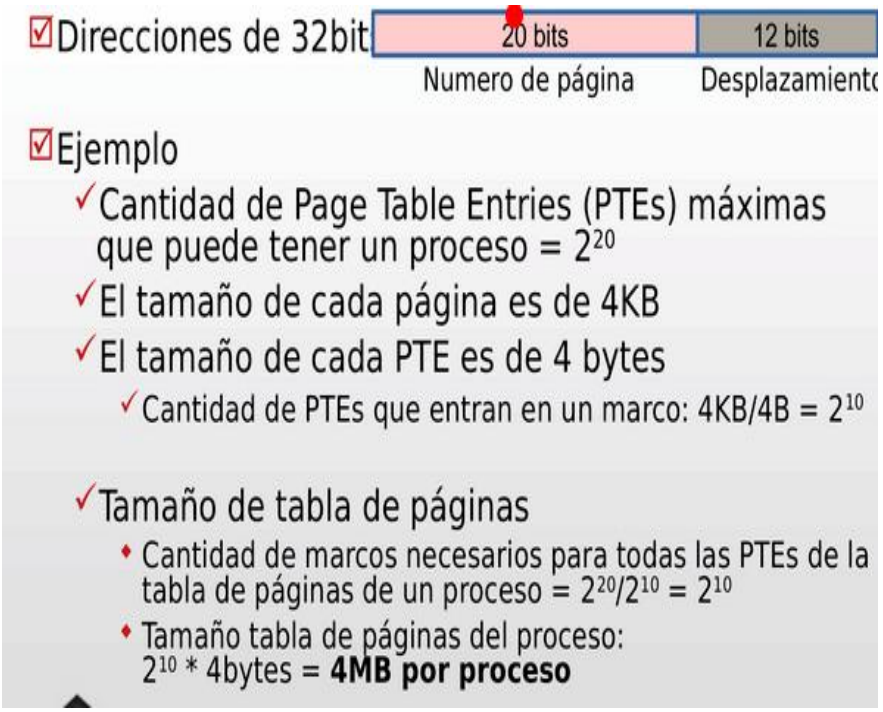
- Si p=0 no hay page faults.
- Si p=1, en cada acceso a memoria hay un page fault (caos).
- P tiene que tender a 0, nunca va a ser 0 por que la técnica se basa en el fallo de pagina.

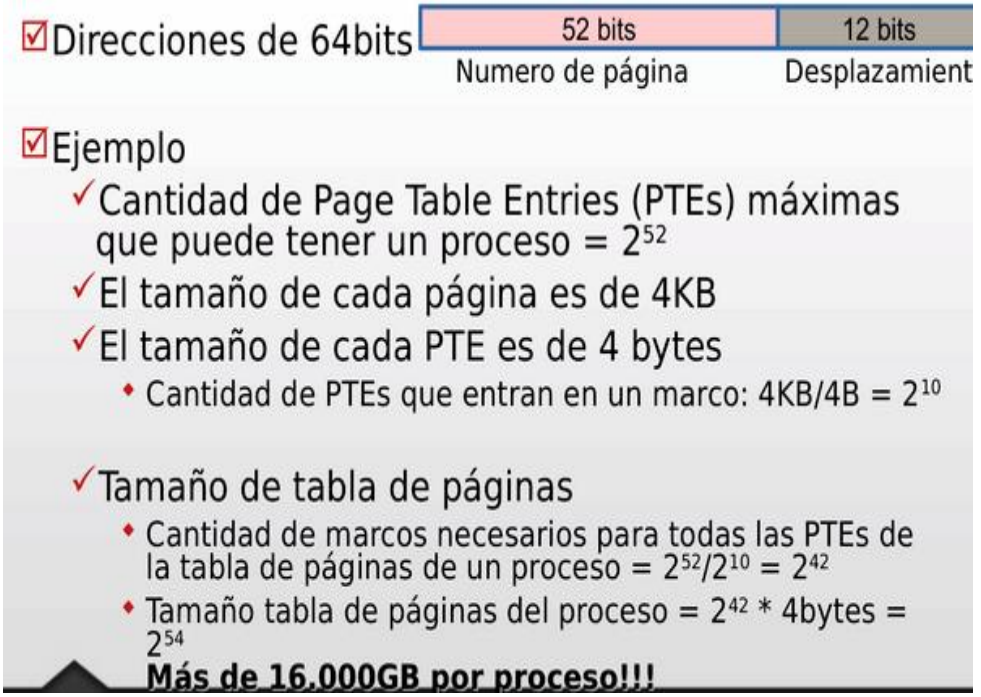
- Effective Access time (EAT) consiste en el tiempo efectivo de acceso:
  - 

  - Si no hay un frame libre, habrá que descargar otro para lograr espacio para la nueva pagina (swap_page_out)

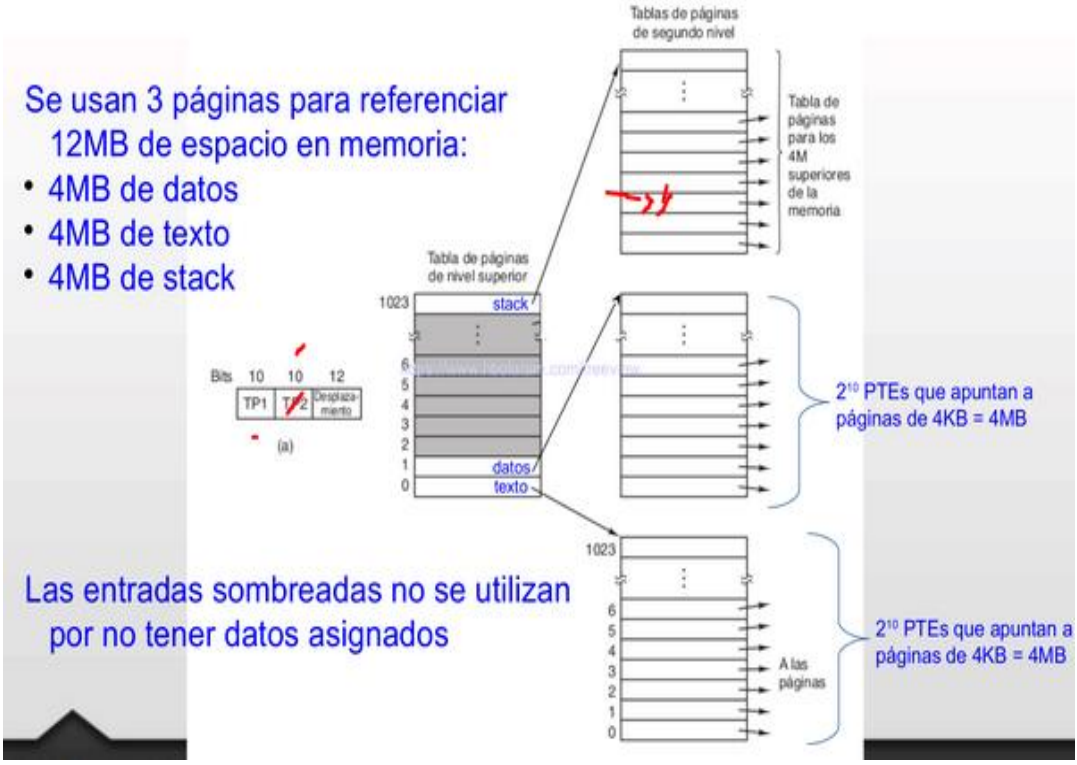
Tabla de paginas

- Cada proceso tiene su tabla de páginas.
- El tamaño de la tabla de páginas depende del espacio de direcciones del proceso.
- Puede alcanzar un tamaño considerable.
- Formas de organizar:
  - Tabla de 1 nivel: tabla única lineal
  - Tabla de 2 niveles: o más, multinivel
  - Tabla invertida: Hashing
- La forma de organizar la tabla de paginas depende del HW subyacente.

- 

- 

  - Eso para tablas de 1 nivel.
  - Existen las tablas de 2 niveles.



-
- El propósito de la tabla de páginas multinivel es dividir la tabla de páginas lineal en múltiples tablas de páginas.
- Cada tabla de paginas suele tener el mismo tamaño pero se busca que tengan un menor numero de paginas por tabla.
- La idea general es que cada tabla sea más pequeña.
- Se busca que la tabla de paginas no ocupe demasiada RAM.
- Además solo se carga una parcialidad de la tabla de paginas (la que se necesite usar).
- Existe un esquema de direccionamientos indirectos.
- Beneficios?
  - Las tablas de segundo nivel (o más) se pueden llevar a memoria secundaria, liberando RAM.
- Desventaja?
  - Más de un acceso a memoria para obtener un dato.
- ----------

- **Tabla invertida**:
  - Utilizada en arquitecturas donde el espacio de direcciones es muy grande.
    - Las tablas de paginas ocuparían muchos niveles y la traducción es costosa.
    - Por esa razón se usa ésta técnica.

        - Por ejemplo, si el espacio de direcciones es de 2^64 bytes, con páginas de 4KB, necesitamos una tabla de páginas con 2^52 entradas.
        - Si cada entrada es de 8 bytes, la tabla es de más de 30 millones de Gigabytes.
    - Hay una entrada por cada marco de página en la memoria real. Es la visión inversa a la que veníamos viendo.
    - Hay una sola tabla para todo el sistema.
    - El espacio de direcciones de la tabla se refiere al espacio físico de la RAM (todo lo que esté cargado), en vez del espacio de direcciones virtuales de un proceso.
    - El numero de página es transformado en un valor de HASH.
    - El HASH se usa como índice de la tabla invertida para encontrar el marco asociado.
    - Se define un mecanismo de encadenamiento para solucionar colisiones (el hash da igual para dos direcciones virtuales).
  - Solo se mantienen las entradas de tablas de pagina (PTE) de páginas presentes en memoria ram.
    - La tabla invertida se organiza como tabla hash en RAM.
      - Se busca indexadamente por número de página virtual.
      - Si está presente en tabla, se extrae el marco de página y sus protecciones.
      - Si no está presente en tabla, corresponde a un page fault.

Tamaño de la pagina

➔ Pequeño
  - Menor fragmentación interna (menos probabilidad de que queden espacios libres que sean muy grandes).
  - Más paginas requeridas por proceso → tablas de paginas más grandes.
  - Más paginas pueden residir en memoria.

➔ Grande
  - Mayor fragmentación interna.
  - La memoria secundaria está diseñada para transferir grandes bloques de datos más eficientemente → es más rápido mover páginas hacia la memoria ram.

➔ Relación con la E/S

- Vel de transferencia: 2 MB/s
- Latencia: 8 ms
- Búsqueda: 20.
- Búsqueda y latencia tienen que ver con el cabezal del disco.

➔ Pagina de 512 bytes.

- 1 pagina → total: 28,2 ms
- Solo 0,2 ms de transferencia (1%) → transferencia real (sin búsqueda ni latencia).
- 2 paginas → 56,4ms

➔ Pagina de 1024 bytes

- Total: 28,4 ms
- Solo 0,4 ms de transferencia.

➔ Vemos que a mayor tamaño de página se transfiere mejor

Translation Lookasid Buffer (TLB)

- Cada referencia en el espacio virtual puede causar 2 (o más) accesos a la memoría física RAM.
  - Uno (o más) para obtener la entrada en tabla de páginas.
  - Uno para obtener los datos.
- Para solucionar este problema, una memoria cache de alta velocidad es usada para almacenar entradas de páginas.
  - TLB.
- Contiene las entradas de la tabla de páginas que fueron usadas más recientemente.
- Dada una dirección virtual, el procesador examina la TLB.
  - Si la entrada de la tabla de paginas se encuentra en la TLB (hit), es obtenido el frame y armada la dirección física.
  - Si la entrada no es encontrada en la TLB (miss), el número de página es usado como índice en la tabla de paginas del procesos.
- Se controla si la pagina está en la memoria
  - Si no está, se genera un Page Fault.
- La TLB es actualizada para incluir la nueva entrada.
- El cambio de contexto genera la invalidación de las entradas de la TLB.

Asignación de Marcos:

- ¿Cuántas paginas de un proceso se pueden encontrar en memoria?
  - Tamaño del Conjunto Residente.
- Asignación dinámica
  - El número de marcos para cada proceso es variable, los procesos de alta prioridad pueden usar muchas páginas (y esto puede dejar que procesos con menos prioridad tengan menos páginas disponibles).
- Asignación fija:
  - Número fijo de marcos para cada proceso.
- Asignación equitativa: Ejemplo: si tengo 100 frames y 5 procesos, 20 frames para cada proceso, como desventaja: cada proceso puede requerir mas o menos frames, no es eficiente.
- Asignación proporcional: Asignado acorde al tamaño del proceso.
  - $s_i =$ size of process p_i
  - $S = \sum s_i$
  - $m =$ total number of frames
  - $a_i =$ allocation for $p_i = \frac{s_i}{S} \times m$

  $m = 64$
  $s_i = 10$
  $s_2 = 127$
  $a_1 = \frac{10}{137} \times 64 \approx 5$
  $a_2 = \frac{127}{137} \times 64 \approx 59$
- El SO hace juego entre asignación fija y dinámica.
  - Primero al proceso se le asigna un numero fijo de frames.
  - Si necesita más se le asigna más.

INTRODUCCIÓN A LOS SISTEMAS OPERATIVOS – TEMA 2: MEMORIA

Reemplazo de páginas:

- ¿Qué sucede si ocurre un fallo de página y todos los marcos están ocupados → "Se debe seleccionar una pagina víctima" (la cual será sacrificada xd).
- ¿Cuál sería un reemplazo optimo?
  - Que la pagina a ser removida no sea referenciada en un futuro próximo.
  - Es perfecto, pero sólo teórico, ósea, inaplicable.
- Por eso el sistema operativo trata: la mayoría de los reemplazos predicen el comportamiento futuro mirando el comportamiento pasado.

Alcance del reemplazo

- Reemplazo global
  - El fallo de página de un proceso puede reemplazar la página de **cualquier** proceso.
  - El SO no controla la tasa de page-faults de cada proceso.
  - Puede tomar frames de otro proceso aumentando la cantidad de frames asignados a él.
  - Un proceso de alta prioridad podría tomar los frames de un proceso de menor prioridad.
- Reemplazo local
  - El fallo de página de un proceso solo puede reemplazar sus propias páginas – de su conjunto residente.
  - No cambia la cantidad de frames asignados.
  - El SO puede determinar cual es la tasa de page-faults de cada proceso.
  - Un proceso puede tener frames asignados que no usa, y no pueden ser usados por otros procesos.

Algoritmos de reemplazo

- Optimo: es sólo teórico.
- FIFO: el más sencillo
- LRU (Least Recently Used): requiere soporte del hardware para mantener timestamps de acceso a las páginas. Favorece a las páginas menos recientemente accedidas.

- 2da chance: un avance del FIFO tradicional que beneficia a las páginas más referenciadas.
- NRU (Non Recently Used):
    - Utiliza Bits R y M.
    - Favorece a las páginas que fueron usadas recientemente.