Practica 5 PN

1)

Dirección Lógica:

Es una dirección que enmascara o abstrae una dirección física, la cual referencia a una localidad de memoria. Esta luego debe ser traducida a dirección física.

Dirección Física:

Es una dirección real en la que se accede efectivamente a memoria. Representa una dirección absoluta de memoria principal

2)

A)

Particiones Fijas:

Consta en dividir la memoria en particiones de tamaño fijo → tamaños iguales o diferentes

Alojan un único proceso.

Ventajas:

- Facilidad de implementación
- No genera fragmentación externa
- Previsibilidad de distribución de memoria

Desventajas:

- Fragmentación interna
- Utilización ineficiente de la memoria

Particiones Dinámicas:

Las particiones varían en tamaño y número. Cada partición se genera de forma dinámica del tamaño justo que necesita cada proceso.

Ventajas:

- No genera fragmentación interna
- Mayor flexibilidad
- Mejor utilización de memoria

Desventajas:

- Fragmentación externa
- Posible necesidad de técnicas de compactación
- Sobrecarga de tiempo buscando espacio a asignar

B)

Particiones Fijas:

Cosas que el S.O debe saber:

- Método para buscar espacio libre
  - Listar particiones libres
  - Algoritmos de asignación
- Control de procesos en ejecución
- Información específica de cada particion
  - Estado
  - tamaño
  - Dirección base

Particiones dinámicas:

- Método para buscar espacio libre
  - Listar particiones libres
  - Algoritmos de asignación (normalmente peor ajuste)
- Tamaño de cada partición dinámica
- Tamaño de los procesos
- Control de fragmentación externa
- Información específica de cada particion
  - Estado

- Tamaño
- Dirección base

C)

3)

Particiones de igual tamaño:

Ventajas:

- Simplicidad de gestión: El sistema operativo puede gestionar facilmente las particiones, ya que todas tienen el mismo tamaño.
- Facilidad para determinar el espacio libre: Como todas las particiones tienen el mismo tamaño, es fácil verificar cuales de todas están libres, y cuales ocupadas.
- Previsibilidad: Al ser todas las particiones del mismo tamaño, es posible predecir cuanta memoria se utilizará y cuantos procesos podrán estár en en la memoria que tenemos disponible

Desventajas:

- Fragmentación interna: Al ser todas del mismo tamaño, como no todos los procesos ocuparan la totalidad asignada, tendremos desperdicio de espacio.
- Utilización ineficiente de la memoria: Se desperdicia demasiada memoria.

Particiones de diferente tamaño

Ventajas:

- Mejor aprovechamiento de la memoria: Al ser variables, el sistema operativo asigna lo justo y necesario para cada proceso. Reduciendo la fragmentación interna.
- Flexibilidad: Este enfoque es más flexible que el anterior. Ya que se pueden asignar particiones del tamaño que requiere y necesita el proceso.

Desventajas:

- Tamaño rígido: SI los tamaños de las particiones no se ajustan bien a alas características de los procesos, habrá desperdicio de memoria.
- Fragmentación externa: Aunque hay menos fragmentación interna, puede ocurrir fragmentación externa si un proceso grande no cabe en ninguna

partición, incluso si la suma de los espacios libres es suficiente

4)

Fragmentación interna:

- Se produce en el esquema de particiones fijas
- Es interna a la localidad asignada
- Es la porción de la localidad que queda sin utilizar

Fragmentación externa:

- Se produce en el esquema de particiones dinámicas
- Son huecos que van quedando en la memoria a medida que los procesos finalizan
- AL no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero no la podemos utilizar
- Solución → compactación

Compactación:

Reorganiza los bloques de memoria ocupados para consolidar el espacio libre en un único bloque contiguo

Pasos a seguir:

- Identificar fragmentos libres
- Reubicar procesos
- Actualizar direcciones
- Consolidación de espacio libre

5)

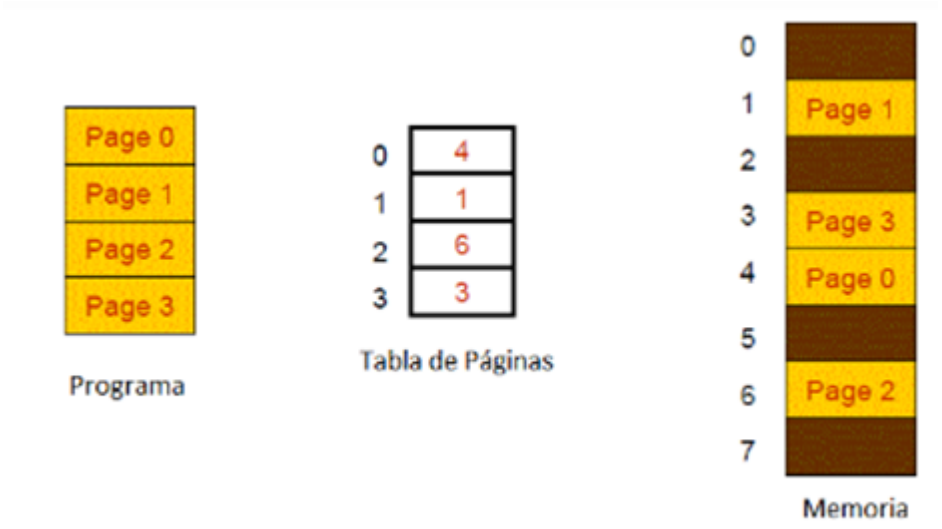
A)

La memoria se divide en porciones de igual tamaño llamadas marco

El espacio de direcciones de procesos se divide en porciones de igual tamaño llamadas páginas

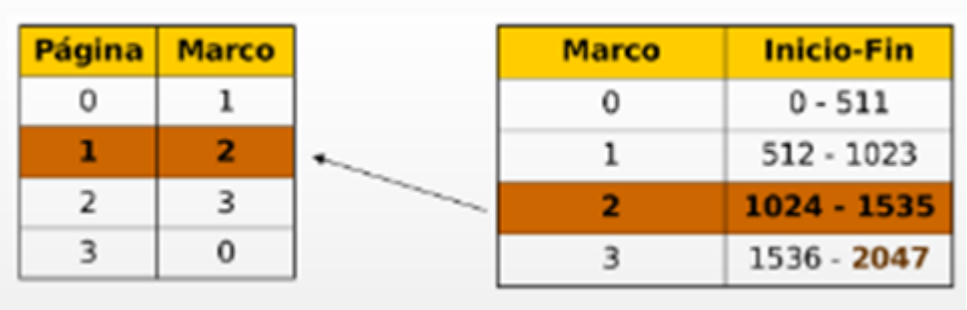
Tamaño de página = tamaño marco = 512 bytes (generalmente)

B)



El SO mantiene una tabla de paginas para cada proceso, la cual contiene el marco donde se encuentra cada página

C)



Ejemplo:

Tamaño de página → 515 Bytes

- Cada dirección de memoria referencia a 1 Byte
- Los marco en memoria principal se encuentran desde la

dirección física 0

- Tenemos un proceso con un tamaño de 2000 Bytes y con la

siguiente tabla de páginas

Si tenemos una dirección virtual, por ejemplo 580:

- Para averiguar el número de página hacemos 580 div 512 = 1.

Luego esta dirección corresponde a la página 1 que se

encuentra en el marco 2

- Para averiguar el desplazamiento hacemos 580 mod 512 = 68
- La dirección física es 1024 + 68 = 1092

D)

Produce fragmentación interna. Ya que al ser todas las páginas de un tamaño fijo, aquellos procesos que no sean lo suficientemente grandes para poder “completar” una página, generan fragmentación

interna.

6)

Similitudes:

- División de la memoria física: ambos métodos dividen la memoria física en bloques de tamaño fijo. En paginación lo llamamos páginas y en particiones fijas son particiones.
- Desperdicio de espacio: Ambos enfoques pueden tener fragmentación interna.

Diferencias:

- En paginación, un proceso puede ocupar varias páginas dispersas por memoria. En cambio, en particiones fijas cada proceso ocupa una única partición.
- El acceso a memoria, en paginación requiere una tabla de páginas en la cual se traducen las direcciones de virtuales a físicas. En particiones, es mas directo ya que los procesos se asignan a particiones específicas

9)

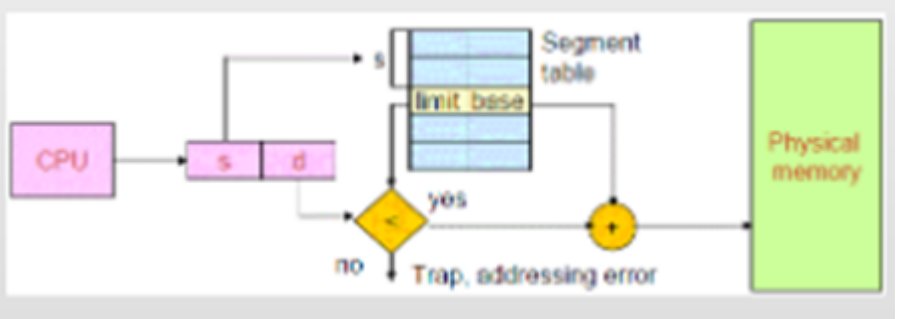
A y b)

La segmentación básicamente la podemos ver como una mejora de la paginación (no hay F.I sino externa)

Ahora la tabla de segmentos además de tener la dirección de inicio del mismo, tiene la longitud o el límite del segmento.

Las direcciones lógicas constan de dos partes → número de segmento s y un desplazamiento dentro del segmento

C)



10)

Similitudes:

- Mejor aprovechamiento de espacio: debido a que ambos le asignan al proceso el espacio que requiere
- Asignación dinámica de memoria
- Ambos producen fragmentación externa

Diferencias:

- La segmentación permite al programa compartir segmentos comunes entre procesos(ej. Uso de librerías) mientras que las particiones no
- Los segmentos son varios para un único proceso. Mientras que la partición es 1 por procesos

11)

Similitudes

1. **Gestión de memoria:** Ambas técnicas permiten asignar memoria a los procesos de forma más eficiente en comparación con sistemas de partición fija.
2. **Separación lógica y física:** Tanto la paginación como la segmentación separan la dirección lógica del programa de la dirección física en memoria.
3. **Uso de tablas:** En ambos casos, el sistema operativo utiliza tablas (tabla de páginas o tabla de segmentos) para traducir direcciones lógicas a físicas.

4. **Fragmentación interna:** En ambas técnicas puede existir fragmentación interna, aunque con diferentes causas.

Aspecto	Paginación	Segmentación
Unidad de división	Divide la memoria en bloques de tamaño fijo llamados **páginas**.	Divide la memoria en segmentos lógicos de tamaño variable (ej., código, datos, pila).
Fragmentación	Puede haber fragmentación interna debido a que las páginas tienen tamaño fijo.	Puede haber fragmentación externa porque los segmentos tienen tamaño variable.
Propósito	Optimizar el uso físico de la memoria dividiéndola en bloques iguales.	Organizar la memoria según las necesidades lógicas del programa.
Tabla usada	Usa una **tabla de páginas**, donde cada entrada apunta a un marco en memoria física.	Usa una **tabla de segmentos**, donde cada entrada define el inicio y el límite de un segmento.
Traducción de direcciones	La dirección lógica incluye un número de página y un desplazamiento dentro de la página.	La dirección lógica incluye un número de segmento y un desplazamiento dentro del segmento.
Acceso a la memoria	Más eficiente en términos de hardware porque el tamaño fijo simplifica la gestión.	Más compleja, ya que el tamaño variable requiere cálculos adicionales y control.
Fragmentación interna/externa	Solo interna.	Solo externa.

Conclusión

La paginación se enfoca en la eficiencia del espacio físico mediante el uso de bloques uniformes, mientras que la segmentación organiza la memoria basándose en las estructuras lógicas del programa. En algunos sistemas, se combinan ambas técnicas para aprovechar sus ventajas (segmentación con paginación).

12)

i)

(2,1,1)

1521

ii)

(1,3,15)

575

iii)

(3,1,10)

5130

iv)

(2,3,5)

1505

13)

A)

- Mas procesos pueden ser mantenidos en memoria
    - Solo son cargadas algunas de las secciones de cada proceso
- Facilita la multiprogramación permitiendo que más procesos se ejecuten simultáneamente distribuyendo dinámicamente memoria física entre ellos
- Un proceso puede ser más grande que la memoria principal

B)

- El hardware debe soportar paginación por demanda(y/o segmentación por demanda)
- Un dispositivo de memoria secundaria (disco) que dé el apoyo para almacenar las secciones del proceso que no estén en memoria principal (área de intercambio)
- EL SO debe ser capaz de manejar el movimiento de las páginas/segmentos entre la memoria principal y la secundaria

C)

Se le adiciona un bit de referencia para saber si recientemente fue usada la página. Es necesario para cuando ocurre un fallo de página. En el caso de que este ocurra, se selecciona una página víctima, la cual será la menos recientemente usada.

14)

A)

Cada vez que hay que alocar una página en un marco, se produce un fallo de página. En otras palabras, ocurre cuando se necesita una página, y esta no está cargada en memoria principal.

b)

El hardware es quien detecta si ocurre un fallo de página.

1. C) Al ocurrir un fallo de página, a traves de alguno de los diferentes algoritmos, se debe seleccionar una página “víctima” que no haya sido referenciada en el último tiempo para liberar dicho marco y cargar la página que se necesita en memoria principal.

15)

A)

4 giga bytes

b)

512 kb = 512 X 1024 = 524,288 bytes

Número max de páginas = Espacio virtual local / tamaño de pag = 2^32 / 524,288

Num max de pag = 8.192

Se pueden tener 8.192 páginas

C)

Se obtiene dividiendo la memoria real por el tamaño de página

256 X 1024 X 1024 = 268,435,456

268,435,456 / 524,288 = 512

Se pueden tener 512 marcos

D)

8.192 × 2^11 = 8.192 × 2048 = 16.777.216

16)

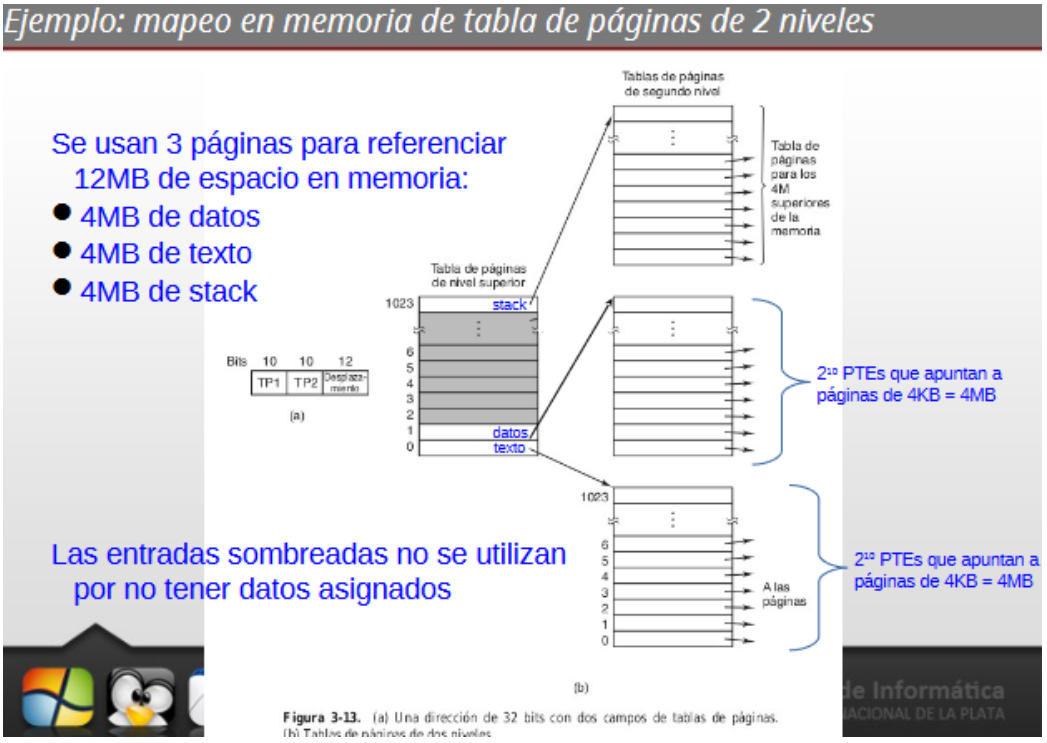
Tabla de páginas de 1 nivel:

- Direcciones de 32 bits (20 bits para num de pag, 12 para desplazamiento)
  - Ejemplo
    - Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 220 (1.048.576)
    - El tamaño de cada página es de 4KB (212)
    - El tamaño de cada PTE es de 4 bytes
      - Cantidad de PTEs que entran en un marco: 4KB/4B = 210
    - Tamaño de tabla de páginas
      - Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = 220/210 = 210
      - Tamaño tabla de páginas del proceso:
        210 ∗ 4bytes = 4MB por proceso
- Direcciones de 64 bits (52 para num de pag, 12 para desplazamiento)
  - Ejemplo
    - Cantidad de Page Table Entries (PTEs) máximas que puede tener un proceso = 252
    - El tamaño de cada página es de 4KB
    - El tamaño de cada PTE es de 4 bytes
      - Cantidad de PTEs que entran en un marco: 4KB/4B = 210
    - Tamaño de tabla de páginas
      - Cantidad de marcos necesarios para todas las PTEs de la tabla de páginas de un proceso = 252/210 = 242
      - Tamaño tabla de páginas del proceso = 242 ∗ 4bytes = 254
        Más de 16.000GB por proceso!!!

Tabla de páginas 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 páginas suele tener el mismo tamaño pero se busca que tengan un menor número de páginas por tabla

- La idea general es que cada tabla sea mas pequeña
- Se busca que la tabla de páginas no ocupe demasiada memoria RAM
- Además solo se carga una parcialidad de la tabla de páginas (solo lo que se necesite resolver)
- Existe un esquema de direccionamientos indirectos



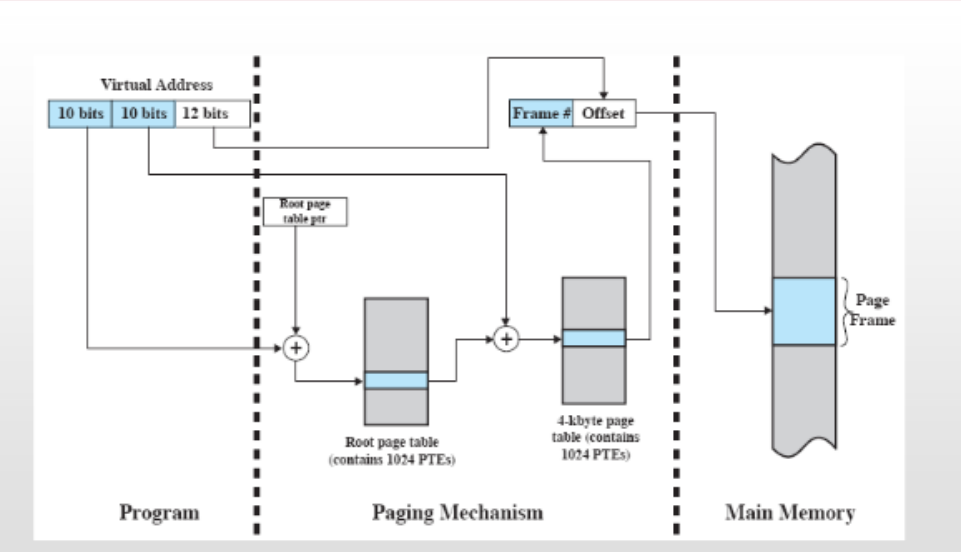
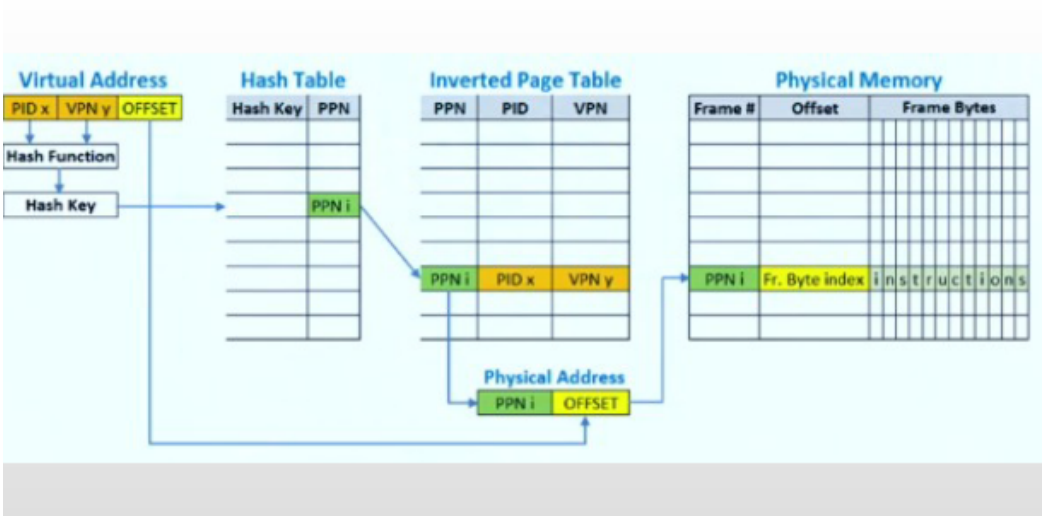


Tabla de páginas invertida:

- Utilizada en arquitecturas donde el espacio de direcciones es muy grande
    - Las tablas de páginas ocuparían muchos niveles y la traducción sería costosa
    - Por esta razón se adopta la técnica
- Por ejemplo, si el espacio de direcciones es de 264 bytes, con páginas de 4 KB, necesitamos una tabla de páginas con 252 entradas
- Si cada entrada es de 8 bytes, la tabla es de más de 30 millones de Gigabyes (30 PB)
- 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 refiera al espacio físico de la RAM, en vez del espacio de direcciones virtuales de un proceso
- Usada en PowerPC, UltraSPARC, y IA-64
- El número 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 (cuando el hash da igual para 2 direcciones virtuales)
- Sólo se mantienen los PTEs de páginas presentes en memoria física
    - La tabla invertida es organizada como tabla hash en memoria principal
    - 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 fallo de página



17)

A)

Dir virtual = 1052

1052 div 512 = 2

1052 mod 512 = 28

Corresponde a pagina 2. Ocurre PF porque la página no está cargada en ningún marco de memoria

B)

Dir virtual 2221

2221 div 512 = 4

2221 mod 512 = 173

Corresponde a pagina 4. Ocurre PF porque la página no está cargada en ningún marco de memoria

c)

Dir virtual = 5499

5499 div 512 = 10

Página inválida

D)

Dir virtual = 3101

3101 div 512 = 6

Página inválida

18)

Tamaño de página pequeño:

Ventajas:

- Menor fragmentación interna: Con páginas más pequeñas, la cantidad de memoria no utilizada dentro de cada página (fragmentación interna) es menor, ya que las páginas se ajustan mejor a los datos que contienen. Esto es particularmente útil cuando las aplicaciones tienen muchas áreas de datos pequeñas.
- Mejor aprovechamiento de la memoria: El sistema puede cargar solo las partes necesarias de un programa en la memoria, lo que mejora el uso global de la memoria.
- Mayor flexibilidad en la asignación de memoria: Las aplicaciones o procesos que requieren menos memoria pueden ser gestionados de manera más eficiente.

Desventajas:

- Mayor sobrecarga de gestión de páginas: Con un tamaño de página pequeño, el sistema operativo debe manejar una mayor cantidad de páginas, lo que genera un aumento en la tabla de páginas y en la necesidad de realizar más operaciones para gestionar las páginas (acceso a memoria, cambios de contexto, etc.).
- Mayor número de fallos de página: Si las páginas son muy pequeñas, es más probable que se produzcan fallos de página debido a la mayor frecuencia con que se deben cargar nuevas páginas desde el disco.
- Mayor uso de recursos de hardware: La mayor cantidad de páginas requiere más entradas en las tablas de páginas, lo que consume más memoria y potencia de procesamiento.

Tamaño de página grande:

Ventajas:

- Menor sobrecarga en la gestión de páginas: Con páginas más grandes, hay menos entradas en las tablas de páginas, lo que reduce la sobrecarga de gestión y mejora la eficiencia del sistema operativo.
- Menor número de fallos de página: Las páginas grandes permiten que se cargue más datos en cada acceso, lo que puede reducir el número de fallos de página, especialmente cuando los programas acceden a grandes bloques de memoria de manera secuencial.
- Mejor rendimiento en sistemas con acceso secuencial a grandes bloques de memoria: En aplicaciones que trabajan con grandes volúmenes de datos, como bases de datos o procesamiento de archivos grandes, las páginas grandes pueden mejorar la eficiencia al reducir las interrupciones por fallos de página.

Desventajas:

- Mayor fragmentación interna: Si el programa no utiliza toda la memoria de la página, se desperdicia memoria dentro de cada página, lo que puede generar una mayor fragmentación interna.
- Mayor uso de memoria en cachés y tablas de páginas: Aunque hay menos páginas, cada una es más grande, lo que puede llevar a un uso menos eficiente de la memoria, especialmente si el programa usa solo una pequeña parte de la página.

19)

A)

La asignación de marco se puede realizar de dos modos:

- Asignación Fija: a cada proceso se le asigna una cantidad

arbitraria de marco. A su vez para el reparto se puede usar:

- Reparto equitativo: se asigna la misma cantidad de marcos a

cada proceso → m div p

- Reparto proporcional: se asignan marco en base a la

necesidad que tiene cada proceso → Vp . m / Vt

- Asignación dinámica: los procesos se van cargando en forma

dinámica de acuerdo a la cantidad de marcos que necesiten

B)

Equitativo:

En el equitativo le corresponde la misma cantidad a cada proceso, y serían 10 por proceso.

Proporcional:

Proceso 1:

15 . 40 / 63 = 9.52 = 9

Proceso 2:

20 . 40 / 63 = 12.69 = 13

Proceso 3:

20 . 40 / 63 = 12.69 = 13

Proceso 4:

8 . 40 / 63 = 5.07 = 5

A mi criterio, el proporcional es más eficiente, ya que reparte según lo que requiere cada proceso. En ambos sigue habiendo probabilidad de fallo de página, pero no terminará asignando más paginas de las que un proceso requiere.

20)

1. FIFO

- Este algoritmo reemplaza la página que lleva más tiempo en memoria, sin considerar su uso reciente o futuro.
- **Problema**: Es susceptible a la *anomalía de Belady*, donde aumentar los marcos de memoria puede incrementar los fallos de página.
- Generalmente, tiene la mayor tasa de fallos de los algoritmos mencionados.
- Cuando ocurre un PF el SO selecciona el primero de al cola

2. Segunda Chance

- Variante de FIFO que da una "segunda chance" a las páginas si han sido referenciadas recientemente (usando un bit de referencia).
- Reduce los fallos respecto a FIFO al considerar el uso reciente, pero no es tan eficiente como los algoritmos más avanzados.
- Cuando ocurre un PF el SO selecciona aquel mas cercano al tope de la cola que no esté marcado con el bit de usado

3. LRU

- Reemplaza la página que no ha sido usada durante más tiempo.
- Se basa en la suposición de que las páginas usadas recientemente probablemente se usarán de nuevo pronto.
- Tasa de fallos significativamente más baja que FIFO o Segunda Oportunidad.
- Cuando ocurre un PF el SO reemplaza aquel menos recientemente usado

4. Óptimo (OPT)

- Este es el algoritmo ideal, ya que siempre reemplaza la página que no será utilizada durante el período de tiempo más largo en el futuro.
- Es teórico y no implementable en la práctica, ya que requiere conocimiento previo del acceso futuro a las páginas.
- Tiene la menor tasa de fallos de todos los algoritmos.
- Cuando ocurre un PF selecciona la página cuyo próxima referencia se encuentra más lejana a la actual

C)

El sistema operativo reserva uno o varios marco para la descarga asincrónica de paginas

- Cuando es necesario descargar una página modificada:
    - La pagina que provoco el fallo se coloca en un frame designado a la descarga asincrónica

    - El SO envía la orden de descargar asincrónicamente la pagina modificada mientras continua la ejecución de otro proceso
    - El frame de descarga asincrónica pasa a ser el que contenía a la página victima que ya se descargó correctamente

21)

A)

Reemplazo global: el fallo de página de un proceso puede

reemplazar la página de cualquier proceso.

- Reemplazo local: el fallo de pagina de un proceso solo puede

reemplazar sus propias páginas.

B)

Es posible utilizar la política de "Asignación Fija" de marcos junto con la política de "Reemplazo Global" de la siguiente manera:

- Asignación Fija de Marcos asegura que cada proceso tenga un número determinado de marcos asignados, pero el reemplazo de las páginas dentro de esos marcos no se limita a un solo proceso. Es decir, los marcos asignados a un proceso pueden ser "reemplazados" por páginas de otros procesos (esto sería el "reemplazo global"), pero solo dentro de los marcos que originalmente le fueron asignados al proceso.
- Selección de una página víctima: Si un proceso genera un fallo de página y no tiene suficiente espacio para cargar una nueva página, el sistema puede elegir una página víctima de otro proceso, pero esta página será colocada dentro de los marcos ya asignados a ese proceso en lugar de asignar nuevos marcos al proceso.

25)

Thrashing (hiperpaginación)

- Decimos que un sistema está en thrashing cuando pasa más tiempo paginando que ejecutando procesos

- Si un proceso cuenta con todos los frames que necesita, no habría thrashing. Salvo excepciones como la anomalía de Belady

C)

- **Tasa de fallos de página alta:**
  - Una cantidad excesiva de fallos de página en relación con la cantidad de accesos a memoria indica que los procesos no tienen suficientes páginas en memoria.
- **Bajo uso de la CPU:**
  - Durante el thrashing, la CPU pasa mucho tiempo inactiva porque los procesos están esperando que se completen las operaciones de intercambio de páginas.
- **Desempeño general del sistema:**
  - El SO puede notar un descenso significativo en el rendimiento del sistema debido al tiempo excesivo que se gasta en el disco.
- **Análisis de las colas de swap:**
  - Si el sistema observa una actividad constante y alta en las colas de intercambio, esto puede ser un indicador de thrashing.

D)

1. Controlar el grado de multiprogramación

- **Reducir la cantidad de procesos en ejecución**:
  - El SO puede suspender temporalmente algunos procesos y colocarlos en estado de espera (swapping), liberando marcos de memoria para los procesos más activos.
  - De esta manera, los procesos restantes tienen suficiente memoria para trabajar sin generar constantes fallos de página.

---

2. Ajustar dinámicamente los marcos asignados a los procesos

- Implementar políticas como el **Modelo de Trabajo** (*Working Set Model*):

    - Este modelo asegura que a cada proceso se le asignen los marcos necesarios para contener sus páginas más utilizadas en un período de tiempo reciente, evitando faltas de página excesivas.
- Usar **Paginación con Control de Faltas**:
    - Incrementar los marcos asignados a un proceso si este genera muchas faltas de página, o reducirlos si su tasa de faltas disminuye.

---

3. Mejorar la planificación del uso de memoria

- **Políticas de reemplazo de páginas eficientes**:
    - Usar algoritmos avanzados como *Least Recently Used (LRU)* o *Clock* para garantizar que se reemplacen páginas menos utilizadas, reduciendo la probabilidad de fallos de página.

---

4. Limitar el uso de memoria por proceso

- Implementar límites de memoria estrictos para procesos individuales:
    - Evitar que un solo proceso consuma de manera excesiva los recursos de memoria, dejando insuficientes marcos para otros procesos.

---

5. Incrementar el tamaño de memoria física disponible

- Si el hardware lo permite, el SO puede recomendar añadir más memoria RAM al sistema para reducir la necesidad de operaciones de intercambio (*swapping*).

---

6. Monitoreo y predicción de cargas

- Usar herramientas de monitoreo para prever y ajustar el uso de memoria antes de que ocurra el thrashing:
    - Detectar patrones de uso excesivo de memoria o aumentos repentinos en las tasas de fallos de página.

---

7. Ajuste del sistema de prioridades

- Asignar mayor prioridad a procesos críticos o más activos y menor prioridad a aquellos en espera, para reducir la competencia por recursos de memoria.