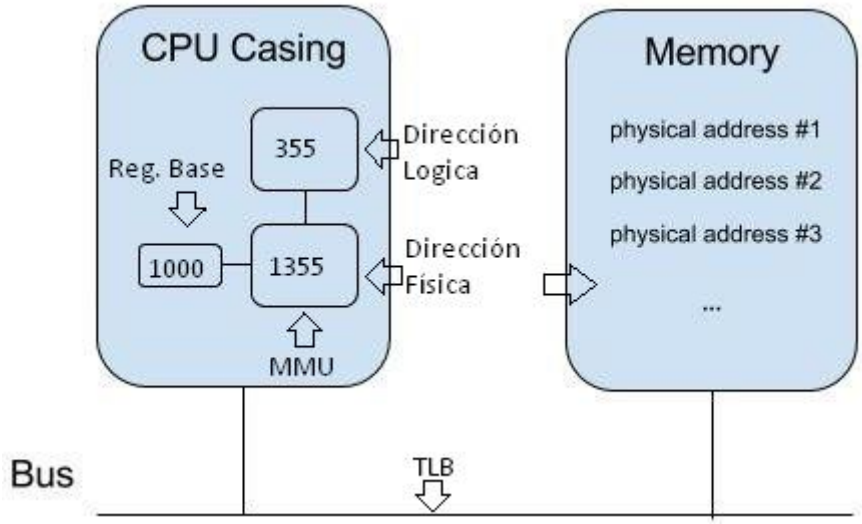
TRABAJO PRÁCTICO 6

EJERCICIO 6.1

Una dirección lógica es una dirección que enmascara o abstrae una dirección física.
Una dirección física hace referencia directamente a una dirección de la memoria.
El CPU genera una dirección lógica que es traducida a una dirección física por el MMU.
La conversión de dirección lógica a física implica sumarle la dirección base del proceso solicitante.

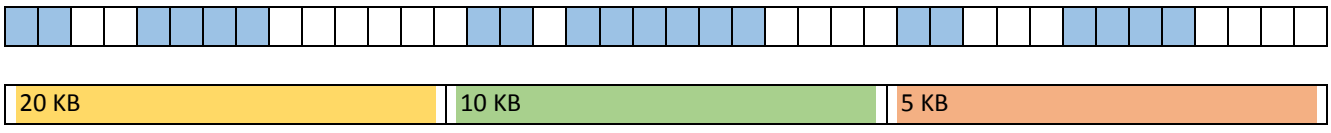


EJERCICIO 6.2

Swapping es un mecanismo en el que un proceso puede ser transferido temporalmente fuera de la memoria principal a un almacenamiento secundario para dejar ese espacio de memoria disponible para otros procesos. Después de un tiempo, el sistema transfiere nuevamente el proceso de la memoria secundaria a la memoria principal.
A pesar de que el swapping afecta la performance ayuda a la ejecución de múltiples y grandes procesos en paralelo.
Los SO modernos no utilizan swapping porque es demasiado lento y existen mejores alternativas (paginación).

2 MB/s == 2048 KB/s
2048 KB -> 1000 ms
200 KB -> 200*1000/2048 = 97,6 ms
Tiempo total 97,6 ms + 15 ms = 112,6 ms

EJERCICIO 6.3

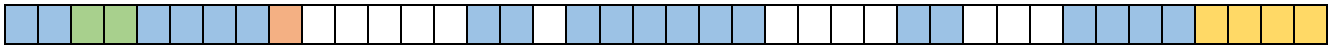


A)
First – fit

B)
Best – fit

C)
Worst – Fit

D)



EJERCICIO 6.4

Segmento	Base	Longitud (bytes)
0	660	248
1	1752	422
2	222	198
3	996	604

A)

0,198

660 + 198 = 858

B)

2,156

222 + 156 = 378

C)

1.530

1752 + 530 = 2282 (ilegal, produce un trap)

D)

3.444

996 + 444 = 1440

E)

0.222

660 + 222 = 882

EJERCICIO 6.5

La memoria principal se divide en un conjunto de frames de igual tamaño. Cada proceso se divide en una serie de páginas del tamaño de los frames. Un proceso se carga en los frames que requiera (todas las páginas), no necesariamente contiguos.

Evita la fragmentación externa, aunque existirá fragmentación interna pequeña.

El SO mantiene una tabla de páginas para cada proceso, que contiene la lista de frames para cada página.

Una dirección de memoria es un número de página y un desplazamiento dentro de la página.

Los bits de desplazamiento de una dirección lógica dependerán de la cantidad de frames por página. Si la cantidad de frames es mayor a la cantidad de bits potencia 2, será imposible acceder a una región de la página. Por lo tanto, la cantidad de frames por página será una potencia de 2, para equipararse con los bits de desplazamiento de la dirección lógica.

EJERCICIO 6.6

A medida que los procesos son cargados y eliminados de la memoria, el espacio de memoria libre se parte en pequeñas piezas. La fragmentación externa se da cuando el tamaño total de espacio libre existente es suficiente para satisfacer una demanda, pero no es contiguo; la memoria está fragmentada en un gran número de pequeños agujeros.

En algunos casos, los datos utilizados para llevar registro de estos agujeros consumen más memoria que el propio agujero. Una solución para este caso, es incluir el pequeño agujero dentro del espacio de memoria del proceso contiguo anterior. Esto evitará la necesidad de registrar el pequeño agujero, pero existirá una porción de memoria inutilizada dentro de una partición (fragmentación interna).

Un esquema segmentado puede sufrir ambas fragmentaciones, aunque en mayor medida sufre fragmentación externa.

Un esquema paginado no sufre fragmentación externa, ya que todos los bloques son del mismo tamaño. En cambio, cada proceso tendrá una página cuyo espacio de memoria no estará siendo completamente utilizado (fragmentación interna).

Si un proceso de 135748 bytes debe ser almacenado en una memoria con un tamaño de página de 4 KB (4096 bytes), se producirá una fragmentación interna de 3516 bytes.
El tamaño de fragmentación interna dependerá de la relación entre el tamaño de página y el tamaño promedio de los procesos. En general, la fragmentación se reducirá cuando el tamaño de páginas sea pequeño y el tamaño de procesos sea grande.

EJERCICIO 6.7

A)

1014 $= 2^{10}$
64 $= 2^{6}$
Bits en dirección lógica = 16

B)

32 $= 2^{5}$
Bits en dirección física = 15

EJERCICIO 6.8

El TLB es un conjunto de registros asociativos construido con una memoria de alta velocidad. Cada registro consiste en dos partes: una clave y un valor. Cuando se le presenta un ítem al registro asociativo, lo compara con todas las claves simultáneamente. Si el ítem es encontrado, el valor correspondiente es retornado. El TLB contiene solo algunos de los registros de la tabla de páginas. Cuando una dirección lógica es generada por el CPU, se consulta si existe la clave correspondiente en el TLB. Si está presente, el número de marco estará disponible inmediatamente para generar la dirección física. Si no está presente, se deberá consultar la tabla de páginas en memoria. El registro obtenido de la tabla de memoria se agregará al TLB (si no hay más espacio se aplicara un algoritmo de remplazo) para que esté disponible en la próxima búsqueda.
El TLB es propio de cada proceso, por lo que en cada cambio de contexto deberá ser borrado.

EJERCICIO 6.9

Si, cuando hacen referencia a una página correspondiente a código compartido.

EJERCICIO 6.10

Espacio direccionado lógico: 4 GB = 2^{32} bytes
Tamaño de Página = 4 KB = 2^{12} bytes
Cantidad de páginas = 32 – 12 = 20 => 2^{20} páginas

A)

Si para direccionar el primer nivel se utilizan 8 bits, serán necesarios 12 bits para direccionar el segundo nivel

B)

Si para direccionar el primer nivel se utilizan 8 bits el tamaño de la tabla de páginas de primer nivel será de 2^{8} * 32 = 256 * 32 = 8192

C)

2^{12} * 32 = 4096 * 32 = 131072

D)

EJERCICIO 6.11

Sistema de 32 – bit

Tamaño de página de 4 KB = 2^{12} bytes
Cada nivel tiene el tamaño de una página

A)

I)
2^{12}

II)
2^{12}

III)
4 GB / 4 KB = 1 GB = 2^{20} frames

IV)

EJERCICIO 6.12

A)
150 ns + 150 ns = 300 ns

B)
(0.85 * 150) + 5 + (0.15 * 300) = 177.5 ns

C)
El uso de registros asociativos mejora notablemente la eficiencia de acceso a memoria

EJERCICIO 6.13

Una dirección en un sistema de paginación es un número de página lógica y un desplazamiento. La página física se encuentra buscando una tabla basada en el número de página lógica para producir un número de página física. Dado que el sistema operativo controla el contenido de esta tabla, puede limitar un proceso para acceder sólo a las páginas físicas asignadas al proceso. No hay manera de que un proceso se refiera a una página que no posee porque la página no estará en la tabla de páginas. Para permitir dicho acceso, el sistema operativo simplemente tiene que permitir que las entradas para memoria no procesada se agreguen a la tabla de páginas del proceso. Esto es útil cuando dos o más procesos necesitan intercambiar datos: simplemente leen y escriben en las mismas direcciones (que pueden estar en direcciones lógicas variables). Esto hace que la comunicación entre procesos sea muy eficiente.

EJERCICIO 6.14

La paginación obliga a que las páginas de un programa estén en memoria constantemente, lo que provoca problemas al intentar mantener en ejecución varios programas con gran cantidad de código. La memoria virtual permite mover datos de la memoria principal a un almacenamiento secundario. La paginación permite que parte del código de un programa esté en memoria principal y el resto en memoria virtual.

EJERCICIO 6.15

Un sistema de paginación por demanda es similar a un sistema de paginación con swapping. Los procesos se ubican en la memoria secundaria. Cuando deseamos ejecutar un proceso lo trasladamos a memoria. En lugar de trasladar todo el proceso a memoria, utilizamos un lazy swaper: no intercambia una página a memoria a menos que sea necesaria.

EJERCICIO 6.16

1. Trap al SO
2. Se guardan los registros de usuario y el estado del proceso
3. Se determina si la interrupción fue un page fault
4. Se verifica que la pagina referenciada sea legal y determina la locación de la pagina en el disco

5. Emite una lectura desde el disco a un marco libre
    a. Espera en una cola por el dispositivo hasta que la solicitud de lectura sea atendida
    b. Espera el tiempo de búsqueda o latencia del dispositivo
    c. Comienza la trasferencia de la pagina a el marco libre
6. Mientras espera, asigna el CPU a otro usuario
7. Interrupción desde el disco
8. Guarda los registros y estado del proceso
9. Determina que la interrupción fue desde el disco
10. Corrige la tabla de paginas y otras tablas para mostrar que la pagina deseada está ahora en memoria
11. Espera que el CPU se asigne a el proceso de nuevo
12. Recupera los registros del usuario y el estado del procesador y la nueva tabla de páginas y reanuda la instrucción interrumpida.

EJERCICIO 6.17

Es una política de optimización utilizada en programación. Si múltiples procesos piden recursos que inicialmente son indistinguibles (iguales), se les devuelven punteros al mismo recurso; en el momento en que un proceso intenta modificar su "copia" del recurso, se crea una copia auténtica para prevenir que los cambios producidos por dicho proceso sean visibles por todos los demás. Todo ocurre de forma transparente para los procesos. La principal ventaja de este método es que no se crea ninguna copia adicional del recurso si ningún proceso llega a realizar modificaciones.
Cuando un proceso crea una copia de sí mismo, las páginas cargadas en memoria que puedan ser modificadas por dicho proceso o su copia se marcan como copy-on-write. Cuando un proceso modifica la memoria, el núcleo del sistema operativo interviene en la operación y crea una copia de forma que los cambios en la memoria ocupada por un proceso no son visibles por el otro.

EJERCICIO 6.18

Valid
Cuando este bit está activado, indica que la pagina asociada esta en el espacio de direccionamiento lógico del proceso. En caso contrario, no está.
Modify
Indica si alguna palabra de la página fue modificada
Reference
Indica si alguna palabra de la página fue referenciada

EJERCICIO 6.19

A)

El CPU genera una dirección lógica compuesta por la Direccion de Pagina Virtual y el offset (suponiendo que se utiliza una sola tabla de páginas). Para obtener la dirección física, se debe obtener la dirección del marco correspondiente a la página y sumarle el desplazamiento.

B)

I)

1.052
1 => 7
7 * 1024 + 052 = 722’

II)

Page Fault

III)

5.499
5 => 0

0 * 1024 + 499 = 499

EJERCICIO 6.20

Si el tamaño de página es de 4 KB se necesitan 12 bits para el desplazamiento, 1 bite para el segmento y 3 bits para la página

A)
00010100 01010111
001010- 0100 01010111

B)
11100100 11111111
000101 - 0100 11111111

C)
11110100 11000111
¿Segment fault?

D)
00110010 11000111
100110 - 0010 11000111
Page fault (es invalido)

EJERCICIO 6.21

A)
FIFO

Proc	0	9	0	1	8	1	8	7	8	7	1	2	8	2	7	8	2	3	8	3
F1	0	0	0	0	8	8	8	8	8	8	8	8	8	8	8	8	8	3	3	3
F2		9	9	9	9	9	9	7	7	7	7	7	7	7	7	7	7	7	8	8
F3				1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
PF	1	2		3	4			5				6						7	8	

B)
LRU

Proc	0	9	0	1	8	1	8	7	8	7	1	2	8	2	7	8	2	3	8	3
F1	0	0	0	0	0	0	0	7	7	7	7	7	8	8	8	8	8	8	8	8
F2		9	9	9	8	8	8	8	8	8	8	2	2	2	2	2	2	2	2	2
F3				1	1	1	1	1	1	1	1	1	1	1	7	7	7	3	3	3
PF	1	2		3	4			5				6	7		8			9		

C)
Óptimo

Proc	0	9	0	1	8	1	8	7	8	7	1	2	8	2	7	8	2	3	8	3
F1	0	0	0	0	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
F2		9	9	9	9	9	9	7	7	7	7	7	7	7	7	7	7	3	3	3
F3				1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
PF	1	2		3	4			5				6						7		

EJERCICIO 6.22

A)

LRU

P	1	0	2	2	1	7	6	7	0	1	2	0	3	0	4	5	1	5	2	4	5	6	7	6	7	2	4	2	7	3	3	2	3
F1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	2	2	2	2	2	2	2	2
F2		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7
F3			2	2	2	2	6	6	6	6	2	2	2	2	2	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4
F4						7	7	7	7	7	7	7	3	3	3	3	1	1	1	1	1	6	6	6	6	6	6	6	6	3	3	3	3
PF	1	2	3			4	5				6		7		8	9	0		1			2	3			4	5			6			

Hit- ratio: (33 – 16) / 33 = 14 / 33 = 0.424

B)

P	1	0	2	2	1	7	6	7	0	1	2	0	3	0	4	5	1	5	2	4	5	6	7	6	7	2	4	2	7	3	3	2	3
F1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	4	4	4	4	4	4	4	6	6	6	6	6	6	6	6	6	6	2	2
F2		0	0	0	0	0	0	0	0	1	1	1	1	1	1	5	5	5	5	5	5	5	7	7	7	7	7	7	7	7	7	7	7
F3			2	2	2	2	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4
F4						7	7	7	7	7	7	7	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3
PF	1	2	3				4			5		6	7		8	9	0		1			2	3				4			5		6	

Hit- ratio: (33 – 16) / 33 = 14 / 33 = 0.424

C)

Los hit-ratio son iguales

EJERCICIO 6.23

El *trashing* (hiperpaginacion) se produce cuando un proceso pasa más tiempo paginando que ejecutándose.
El trashing puede ser causado por programas o cargas de trabajo que presentan una localidad de referencia insuficiente: si el working set de un programa no puede mantenerse efectivamente en memoria principal, puede ocurrir un intercambio constante de datos (trashing).
Si varios procesos están realizando un swap constante, el rendimiento del CPU se reduce, ya que estos procesos están a la espera de que finalice el swap correspondiente. Al reducir el procesamiento del CPU, el sistema decide integrar más procesos a ejecución, provocando que el trashing aumente de forma cíclica.
Una solución por parte del sistema para eliminar el trashing es ofrecerle a un proceso todas las páginas que necesita. Para suponer este dato se utilizan algoritmos de reemplazo de página (working-set, pff).
La solución por hardware para evitar trashing es aumentar el tamaño de memoria principal, reducir la cantidad de procesos en ejecución o modificar el tamaño de partición de intercambio.

EJERCICIO 6.24

Para hacer el ejercicio, supongo que los request y releases de meoria están expresados en kilobytes

1024 kb															

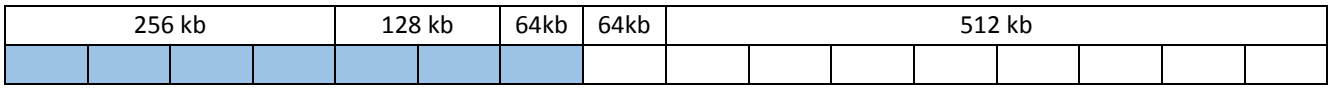
Request 240 kb

256 kb				256 kb				512 kb							

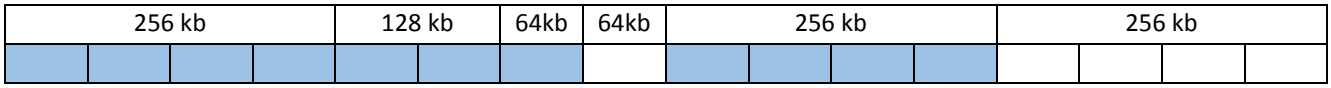
Request 12 kb

256 kb				128 kb		128 kb		512 kb							

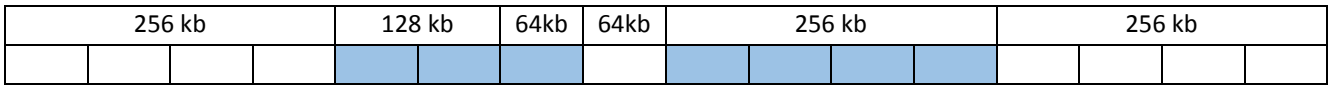
Request 60 kb



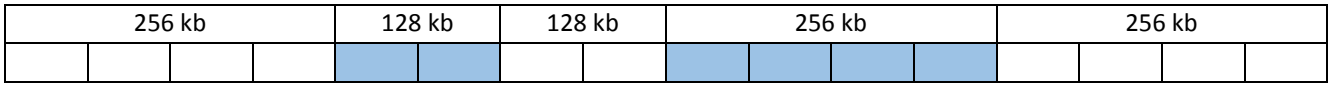
Request 130 kb



Release 240 kb



Release 60 kb



Release 120 kb