Sistemas Operativos
1° Rec 2° Parcial 1C2022 - Resolución

<u>Aclaración</u>: La mayoría de las preguntas o ejercicios no tienen una única solución. Por lo tanto, si una solución particular no es similar a la expuesta aquí, no significa necesariamente que la misma sea incorrecta. Ante cualquier duda, consultar con el/la docente del curso.

Teoría

1. La dirección lógica se divide en NRO SEG | NRO PAG | OFFSET. La MMU primero lee la entrada correspondiente al número de segmento en su tabla de segmentos para chequear permisos y que su tamaño sea mayor al desplazamiento. Luego se lee la entrada correspondiente al número de página en la tabla de páginas de ese segmento (1 acceso a memoria) para obtener el número de frame de esa página. La DF entonces queda: NRO FRAME | OFFSET.
   Introduciendo una TLB, nos podríamos ahorrar el acceso a memoria para leer la tabla de páginas del segmento ya que previamente se chequea que su traducción ya esté cacheada en la misma.
2.
   a. Verdadero. El tamaño ocupado en memoria principal será menor, ya que gran parte de las entradas estarán en el área de swap y serán cargadas ante PF.
   b. Falso. Para evitar el thrashing se podría aumentar la cantidad de marcos asignados a cada proceso. Hacer lo contrario contribuiría a empeorar la situación.
3. En los FS de tipo FAT, al no haber una estructura FCB, la metadata relevante al archivo que normalmente estaría en su FCB se guarda en su entrada de directorio. Esto se contradice con el concepto de hardlink, ya que si un mismo archivo tiene más de una entrada de directorio, su metadata estaría duplicada, por lo que el FS debería tener cuidado de no generar inconsistencias entre las mismas. Por esto, FAT no implementa hardlinks.
4. Existen dos tipos de tabla: una global, donde se incluye información sobre los archivos abiertos por cualquier proceso y campos generales (como el contador de aperturas e información del archivo en memoria) y a su vez, una tabla por proceso, donde se guarda un puntero a la tabla global y campos por cada proceso (como el puntero de lectura y el modo de apertura). Cuando un archivo se abre por primera vez, es escrito en la tabla global y en la del proceso. Cuando el mismo archivo se abre por otro proceso, se escribe sólo en la del proceso y se actualiza el contador de aperturas en la global.
5. Mapear archivos a memoria consiste en incrementar el espacio de direcciones del proceso para que estas nuevas páginas hagan referencia a bloques de un archivo. Al haber un page fault por acceder a una página, se cargará ese bloque en un frame de memoria para que el proceso lo pueda acceder como si fuera parte su imagen sin tener que recurrir a syscalls. Normalmente estas nuevas páginas se consideran parte del heap del proceso ya que luego podrían ser liberadas. El código del proceso es un buen ejemplo de algo que puede ser considerado un archivo mapeado a memoria.

Práctica

1. a)
   Como el FS actual tiene un solo puntero indirecto simple (el más indirecto), su tamaño máximo teórico es un poco más de lo que pueda direccionar con el mismo.
   Tam_máx_archivo =- ptrs_x_bloque * tam_bloque
   256 KiB =- ptrs_x_bloque * 1KiB
   Ptrs_x_bloque = 256

   Tam_puntero = tam_bloque/ptrs_x_bloque
   Tam_puntero = 1 KiB / 256
   Tam_puntero = **4 Bytes** = 32 bits

   b)
   Ptrs_x_bloque = tam_bloque / tam_puntero
   Ptrs_x_bloque = 4 KiB / 4 Bytes = 1024 = 2^10
   Tam_máx_archivo =- (2^10)^2 * 2^12 B = 2^32 B = **4 GiB**

   c)
   - La **fragmentación interna** en el nuevo esquema será mayor ya que el tamaño de bloque es mayor (4 KiB > 1 KiB)
   - Para leer archivos de 20 KiB inicialmente debía leer **20 bloques de datos + 1 bloque de punteros**. En el nuevo esquema solamente necesita leer **5 bloques de datos**.

2.

Lo primero es determinar la cantidad de bits para #pag / offset.
Si vemos la TLB, veremos que las páginas necesitan 16 bits (4 caracteres hexadecimales). Entonces, se necesitarán 8 bits para cada nivel de TP.
A1012222h:
- A101 #pag, 2222 offset.
- Buscamos primero en TLB. Se encuentra esta página del proceso A en la segunda entrada, obteniendo el frame Ah (10d)
- Reemplazamos la DL con el frame y nos queda **DF: A2222h**

030110A1h:
- 0301 #pag, 10A1 offset.
- No se encuentra en la TLB (la entrada 3 corresponde al proceso B).
- Buscamos en TP 1 nivel los primeros 8 bits: 03h. Corresponde a la página 3 de 2do nivel, presente en RAM.
- Como la página anterior está presente, no se produce PF
- Se accede luego en TP 2do nivel a 01h. La página no está presente. Se da un PF y se carga en el frame 21 con la página referenciada, añadiendo una entrada en la TLB.
- Finalmente, sabiendo que el frame es el 21d (15h), nos queda **DF: 1510A1h**

03030000h:
- 0303 #pag, 0000 offset.
- No se encuentra en la TLB.
- Buscamos en TP 1 nivel los primeros 8 bits: 03h. Corresponde a la página 3 de 2do nivel, presente.
- Se accede luego en TP 2do nivel a 03h. La página también está presente en el frame 11d (Bh). Añadimos una nueva entrada en la TLB.
- Finalmente, sabiendo que el frame es el 11d (Bh), nos queda **DF: B0000h**

Frag interna: como se usan 16 bits para el offset, entonces las páginas/frames son de 64 KiB. El máximo desperdicio es 64 KiB - 1 Byte.
Eliminar columna: cambiaría en que al volver a ejecutar el proceso A, se habría tenido que borrar toda la TLB (ya que sino las traducciones hubiesen sido erróneas, apuntando a frames de otros procesos).
Estado final TLB:

Pag	Marco	Proceso
1020	11	A
A101	C	A
0301	AA	B
1111	10	A
0022	4	C
0301	**15**	**A**
0303	**B**	**A**

3.

últ referencia ->seg 1 pág 1 (> TUR y P =1)
DL 40132124h -> sabemos que es segmento 1 pág 1 y que la dirección lógica se divide en NRO SEG | NRO Pág | OFFSET
paso primeros 4 hexa a binario
01| 00 0000 0001 |0011 -> con esos cortes coincide con el seg y pág
=> 2 bits para nro de segmento y 10 bits para nro de pág
Con eso podemos saber que 20 bits se usan para el offset (32-2-10)

DL 40004321h -> 01|00 0000 0000 -> segmento 1 , página 0 PF
Se reemplaza la pág 2 del seg 0 (< TUR). Es necesario acceder a la tabla de págs tanto para buscar la pág que no está presente como luego para actualizarla (tanto la de seg 0 como la del 1). De disco hay que traer a la pág 0 del seg 1.
Se asigna el frame 7 => DF 704321h

DL 80144324h -> 10|00 0000 0001 -> seg 2 pág 1 -> está presente en el frame 8 :)
Hay acceso a la tabla de página para obtener el frame y luego el acceso en sí mismo
DF -> 844324h

01009302h -> 00| 00 0001 0000 -> seg 0 pág 16 -> PF
Reemplaza a pág 1 seg 0 por < TUR
se asigna frame 5
Es necesario acceder a la tabla de págs tanto para buscar la pág que no está presente como luego para actualizarla (tabla del seg 0). De disco hay que traer a la pág 16 del seg 0.
DF -> 509302h