| APELLIDOS | SOLUCIONES | NOMBRE | | Grupo |
|---|---|---|---|---|
| DNI | | Firma | | |

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 8 cuestiones, cuya valoración se indica en cada una de ellas.**

---

**1.** Un sistema de archivos organizado en bloques de 512 Bytes, con punteros a bloque de 2Bytes tiene un tiempo medio de acceso a bloque de 2 mseg. Suponga que en memoria se encuentra la información que contiene el puntero al primer bloque de datos de un archivo de 32KBytes y que el tiempo de acceso a Memoria principal es despreciable. Indique de forma justificada el tiempo medio necesario para acceder a la lectura del Byte 4650 de dicho archivo para cada uno de los siguientes métodos de asignación de bloque:

**(1,0 punto = 0,4 + 0,3 + 0,3)**

**1**

**a)** Asignación Enlazada

En asignación enlazada cada bloque de datos contiene en sus últimos Bytes el puntero al bloque siguiente por lo que de los 512 bytes de cada bloque sólo quedan disponibles 510 para datos. El número de bloque que contiene el byte 4650 es el **bloque 9**, ya que:
(4650 / 510) = cociente 9, resto 60
Es necesario leer 10 bloques secuencialmente de dicho archivo, del 0 al 9, por tanto el tiempo requerido es de: **10 * 2 = 20 mseg**

**b)** Asignación indexada con dos niveles de indexación

En asignación indexada con dos niveles cada archivo tiene al menos dos bloques que contienen punteros a bloques de datos, uno por cada nivel. Los bloques de segundo nivel contiene punteros a bloques de datos. Hay pues que realizar dos lecturas de bloque de punteros antes de poder leer el bloque de datos por lo que el tiempo requerido es de:
2 mseg (lectura bloque de primer nivel) + 2 mseg (lectura bloque de segundo nivel) + 2 mseg (lectura del bloque de datos del Byte solicitado)= 2+2+2= **6 mseg**

**c)** FAT (*File allocation Table*). Considere que la tabla con todos sus punteros a bloque se encuentra en memoria principal y que el tiempo de acceso a memoria es despreciable

Si la FAT o tabla de punteros está en memoria entonces la localización de la ubicación del bloque que contiene el Byte 4650 se hace sin acceder a disco, en un tiempo mucho menor que el tiempo de acceso a disco, por lo que el tiempo requerido es de **2 mseg aproximadamente**, ya que sólo es necesario el tiempo de lectura del bloque de datos.

**2.**Un proceso debe escribir el mensaje *"parent message"* en el archivo *"messages.txt"* y a continuación crear un hijo que escriba el mensaje *"child message"* en el mismo archivo. El proceso padre debe esperar la finalización del proceso hijo para leer TODO el contenido del archivo *"messages.txt"* a través de su entrada estándar y escribirlo en su salida estándar. Además el proceso padre debe finalizar cerrando todos los descriptores de archivos abiertos. Complete el programa en código C del apartado a) con las primitivas POSIX necesarias, una en cada línea con número subrayado, para que realice dichas acciones. **NOTA**: Utilice open(), read(), write(), close() y dup2() cuando lo necesite. No utilice la llamada al sistema lseek().

**(1,2 puntos = 0,8 + 0,4)**

**2 a)**

```
1   #include <all_needed>
2   #define SIZE 50
3   int main( int argc, char **argv ){
4     int fd1, fd2, nbytes;
5     char buffer[SIZE];
6     mode_t fd_mode = S_IRWXU;     // file permissions
7     char *parent_message = "parent message \n";
8     char *child_message = "child messages \n";
9     fd1 = open( "messages.txt", O_TRUNC|O_CREAT|O_RDWR, fd_mode); //complete open()
10    write( fd1 , parent_message, strlen(parent_message));      //complete write()
11    if (fork() == 0) { // child
12     write(fd1, child_message, strlen(child_message));
13     close(fd1);
14      exit(0);  }
15    wait(NULL);c
16    fd2 = open("messages.txt", O_RDONLY);
17    dup2(fd2, STDIN_FILENO);
18    while(1) {
19      nbytes = read( 0, buffer, strlen(child_message));       //complete read()
20      if (nbytes > 0)
21        write(1, buffer, nbytes);
22      else
23        break;
24     }
25   close(fd1);
26   close(fd2);
27  }
```

**b)** Rellene la tabla de descriptores de archivos del proceso hijo tras ejecutar la línea 12 y del proceso padre tras la línea 20. Las tablas deben ser correctas con los requisitos y la implementación de la sección a)

| Tabla del Proceso hijo en 12 | |
|---|---|
| 0 | STDIN |
| 1 | STDOUT |
| 2 | STDERR |
| 3 | "messages.txt" |
| 4 | |
| 5 | |

| Tabla del Proceso padre en 20 | |
|---|---|
| 0 | "messages.txt" |
| 1 | STDOUT |
| 2 | STDERR |
| 3 | "messages.txt" |
| 4 | "messages.txt" |
| 5 | |

**3.** El programa */usr/bin/passwd* permite cambiar las contraseñas de los usuarios. Para su ejecución este programa necesita leer y escribir los archivos */etc/passwd* y */etc/shadow*. El programa */usr/bin/chage* permite cambiar el tiempo de expiración de una contraseña (guardado en */etc/shadow*), lo que implica leer el archivo */etc/passwd* y leer y escribir en el archivo */etc/shadow*. Considere el siguiente contenido (parcial) de estos dos directorios: **(1,2 puntos = 0,8 + 0,4)**

**directorio */usr/bin*:**

```
i-nodo permisos enlaces  usuario  grupo  tamaño    fecha      nombre
655364 drwxr-xr-x  2     root    root   36864 nov 18 14:02 .
655363 drwxr-xr-x 12     root    root    4096 jul 20  2018 ..
656249 -rwxr-sr-x  1     root    shadow 71816 mar 22  2019 chage
658048 lrwxrwxrwx  1     root    root       5 may 20  2019 gcc -> gcc-7
657101 lrwxrwxrwx  1     root    root      22 may  8  2019 gcc-7 -> x86_64-linux-gnu-gcc-7
657839 -rwsr-xr-x  1     root    root   59640 mar 22  2019 passwd
655397 -rwxr-xr-x  2     root    root 2097720 nov 19  2018 perl
655397 -rwxr-xr-x  2     root    root 2097720 nov 19  2018 perl5.26.1
657867 -rwxr-xr-x  1     root    root 1010624 may  8  2019 x86_64-linux-gnu-gcc-7
```

**directorio */etc:***

```
808275 -rw-r--r--  1     root    root         1812 jul 16  2018 passwd
800878 -rw-r-----  1     root    shadow       1041 jul 20  2018 shadow
```

**3** **a)** Indique si funcionaría sin error la ejecución por el usuario especificado de las siguientes órdenes. En caso de éxito justifique cuáles son los permisos que se van comprobando y, en caso de error, cuál es el permiso que falla y por qué.

| (UID, GID) | ORDEN | ÉXITO | JUSTIFICACIÓN |
|---|---|---|---|
| (eva, fso) | `/usr/bin/passwd` | Sí | El archivo /usr/bin/passwd tiene activo el bit SETUID, por lo que, el proceso passwd, se ejecuta como (root, fso). Los archivos /etc/passwd y /etc/shadow tienen permisos de lectura y escritura para root. |
| (eva, fso) | `chage -M7 eva`<br><br>(cambio de la expiración a 7 días para el usuario eva) | No | El archivo chage tiene activo el bit SETGID, por lo que, s el proceso chage e ejecuta como (eva, shadow). El archivo /etc/passwd tiene permiso de lectura en la tripleta *other*. Pero el archivo /etc/shadow no se puede escribir, porque solo tiene permiso de escritura para root. |
| (eva, fso) | `gcc --help`<br><br>(obtener ayuda del programa gcc) | Sí | gcc conduce mediante un doble enlace simbólico al ejecutable x86_64-linux-gnu-gcc-7. Los permisos que se verifican son los de este último, que dan permiso de ejecución a *other*, por lo que el usuario eva puede ejecutarlo. |

**b)** Para el directorio */usr/bin* justifique número de enlaces de sus archivos "." y "perl"

| Nombre | Enlaces | JUSTIFICACIÓN |
|---|---|---|
| . | 2 | Este directorio tiene **2 enlaces, el "/usr/bin"** en su padre y el propio **"/usr/bin/."**. No tiene ningún subdirectorio que lo referencie con ".." |
| perl | 2 | Este archivo tiene 2 enlaces: la propia entrada **"/usr/bin/perl**" y la entrada **"/usr/bin/perl5.26.1"** que tienen el mismo nodo-i. |

**4.** Un disco con capacidad de 64 MBytes, se formatea en una versión de MINIX, cuyas especificaciones son:

- El bloque de arranque y el superbloque ocupan 1 bloque cada uno
- Tamaño del nodo-i de 32 Bytes (7 punteros directos, 1 indirecto,1 doble indirecto).
- Punteros a zona de 16 bits
- Entrada de directorio de 16 Bytes: 2 Bytes para nodo-i, 14 Bytes para el nombre
- 1 zona = 1 bloque = 1 KByte

**(1,6 puntos = 0,8 + 0,8)**

**4** **a)** Se formatea reservando espacio en la cabecera para un total de 16384 nodos-i (16 K nodos-i). Calcule el número de bloques que ocupa cada elemento de la cabecera y el área de datos. Justifique su respuesta

| Arranque | Super bloque | Mapa de bits de Nodos-i | Mapa de bits de Zonas | Nodos- i | Zonas de datos |
|---|---|---|---|---|---|

1 Bloque de arranque → bloque 0
1 Bloque de Super bloque → bloque 1
2 Bloques para el Mapa de Nodos-i → bloque 2 y 3
16 K de nodos-i requieren 16K bits; 16Kbits/ 1KByte= 16Kbits/8Kbits= **2 bloques Mapa nodos-i**
8 Bloques para el Mapa de Zonas → bloque 4, 5, 6, 7, 8, 9, 10 y 11.
Partición de 64 MBytes= 64MBytes/1KByte= 64K zonas → 64Kbits/8Kbits= **8 Bloques Mapa zonas**
512 Bloques para los Nodos-i→ bloque 12 al bloque 523
Nodos-i = 16 K nodos-i; 16Knodos-i * 32 Bytes = 512KBytes para almacenar los nodos-i
512KBytes/1KByte =**512 bloques para almacenar nodos-i**
La zona de datos comienza en el bloque 524
La cabecera ocupa: 1+1+2+8+512=**524 bloques**
La partición es de 64MBytes → 65536 Bloques; 65536-524 de la cabecera= **65012 zonas de datos**

**b)** Suponga que partiendo del disco vacío, se va ocupando el disco de acuerdo a las siguientes acciones:

- Creación del directorio */ (raíz)*
- Creación del archivo regular ***/fso*** con un tamaño de 514KBytes
- Creación del directorio ***/Examen***
- Creación del directorio ***/Examen/Final***
- Creación del archivo regular ***/Examen/Final/Cursoactual*** con un tamaño de 800 KBytes
- Creación del enlace físico ***/EFinal*** al archivo regular ***/Examen/Final/Cursoactual***

Tras estas acciones, **indique de forma razonada** para dicho sistema de archivos los siguientes valores:

**-** Tamaño en Bytes del directorio / *(raíz)*

El directorio / tiene 5 entradas que son:

- . y .. (dos entradas)
- entrada de directorio del archivo /fso,
- entrada de /EFinal
- entrada del directorio /Examen

Cada entrada ocupa 16 Bytes, por tanto el directorio **ocupa (5 x 16Bytes) 80 Bytes**

-Número de i-nodos ocupados en dicho sistema de archivos

**Se ocupan 5 i-nodo**s, uno por cada uno de los archivos o directorios creados, excepto el enlace físico que comparte i-nodo.

1 Nodo-i→ Directorio /
1 Nodo-i → /fso
1 Nodo-i → /Examen
1 Nodo-i → /Examen/Final
1 Nodo-i → /Examen/Final/Cursoactual

- Número de bloques de la zona de datos ocupados con referencias a otros bloques

Los nodos-i tienen 7 ptos directos directo y el tamaño de bloque es de 1KByte, por lo tanto sólo se utilizarán bloques con referencias a bloques, en archivos con tamaños superiores a 7 KBytes.
Un bloque tiene un tamaño de 1KByte y los punteros a bloque/zona tienen un tamaño de 2 Bytes → en cada bloque se pueden almacenar hasta 512 punteros a zona.

En este sistema de archivos, los únicos archivos que utilizan bloques con referencias a otros bloques son:

1) Archivo /fso con un tamaño de 514KBytes → necesita referenciar 514 bloques de datos. Por lo tanto, utilizará 7 punteros directos y el puntero SI, que apunta a **un bloque** con 507 referencias a otros bloques.
2) Archivo /Examen/Final/CursoActual con un tamaño de 800KBytes → necesita referenciar 800 bloques de datos. Utilizará 7 punteros directos, el puntero SI que apunta a **un bloque** con 512 referencias a otros bloques, y el puntero DI que utilizará **dos bloques** más con referencias a otros bloques. El bloque del primer nivel tendrá un única referencia a un bloque de segundo nivel que contendrá 800 - (512+7)= 281 referencias a otros bloques. Por lo tanto, 3 bloques con referencias a bloques.

En total, se utilizan **4 bloques con referencias a otros bloques**

**5**. Un sistema gestiona una memoria principal de 1000KB con asignación contigua con particiones variables y SIN compactación y cuya ocupación inicial a considerar es:

0 1000KB-1

| SO 100KB | HUECO 150KB | P1 200KB | HUECO 100KB | P2 100KB | HUECO 350KB |
|---|---|---|---|---|---|

El algoritmo de asignación, ubica siempre los procesos dentro de un hueco ajustándose a las direcciones más bajas (izquierda), dejando libre las direcciones más altas del hueco. Indique las bases de los proceso P3 (100KBytes), P4 (400KBytes) y P5 (200KBytes) para cada uno de los algoritmos de asignación de huecos: Mejor Ajuste (Best Fit), Primer Ajuste (First Fit) y Peor Ajuste (Worst Fit). Tenga en cuenta la secuencia de eventos propuesta y el estado inicial de la memoria para cada caso. Si un proceso no cabe indique NO CABE y continúe con el siguiente.

**(1,2 puntos = 0,9 + 0,3)**

| EVENTO | **BEST FIT** | **FIRST FIT** | **WORST FIT** |
|---|---|---|---|
| LLEGA P3<br>¿BASE P3? | 450K | 100K | 650K |
| TERMINA P1 | ---- | ---- | ---- |
| LLEGA P4<br>¿BASE P4? | NO CABE | NO CABE | 100K |
| TERMINA P2 | ---- | ---- | ---- |
| LLEGA P5<br>BASE P5? | 100K | 200K | 750K |

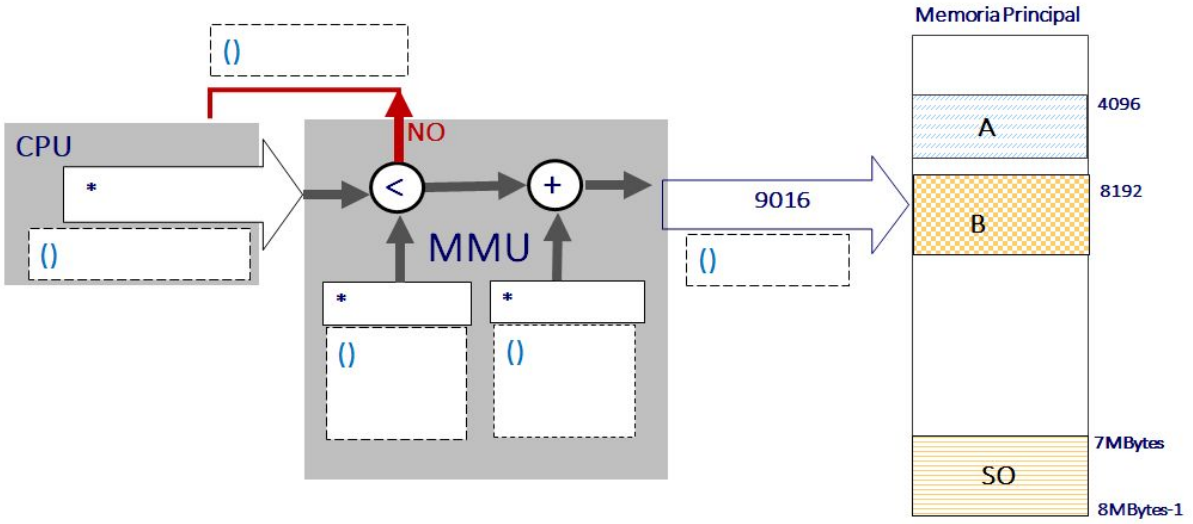
b) Indique para el algoritmo de Worst Fit, los huecos que han quedado (**dirección de inicio** y **tamaño**) al finalizar la secuencia de eventos anterior, y el tipo de fragmentación que se produce:

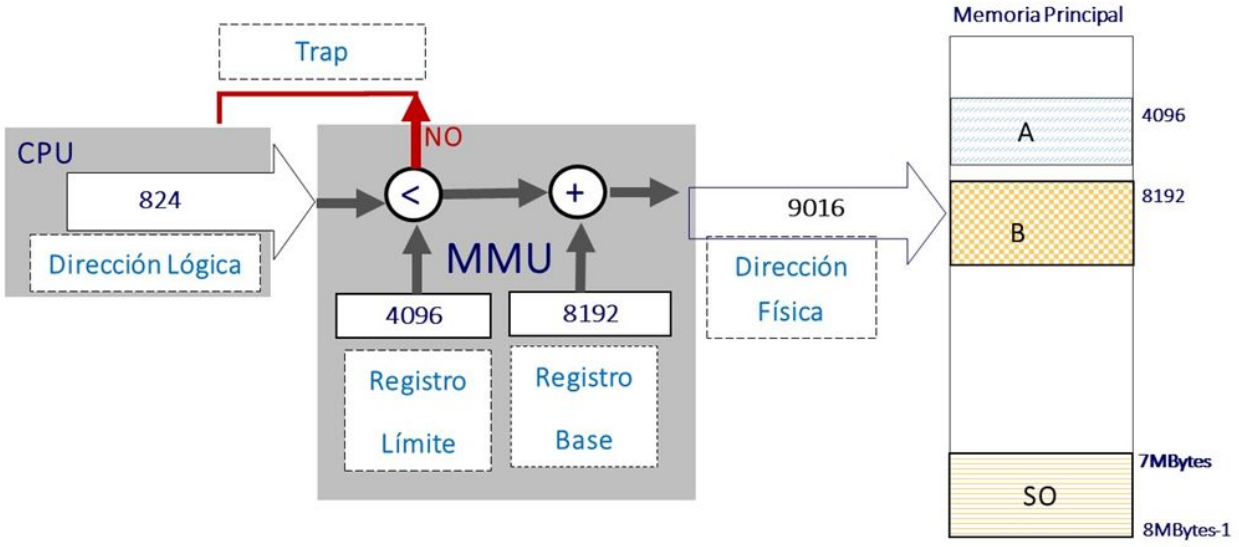
En Worst Fit han quedado dos huecos: un hueco de tamaño 150KB a partir de la dirección 500K y hueco de tamaño 50KB a partir de la dirección 950K.
Se trata de Fragmentación externa, ya que el tamaño de la partición se ajusta al tamaño del bloque dejando huecos en memoria cuando los procesos finalizan. Estos huecos pueden no ser lo suficientemente grandes para ubicar un proceso contiguo en memoria pero que tras la compactación el proceso podría ser ubicado.

**6.** Considere un sistema con una memoria principal de 8MBytes en la que se encuentran ubicados dos procesos A (de tamaño 2KBytes) y B (de tamaño 4KBytes) a partir de las direcciones que se indican en la figura.

**(0,8 puntos )**

**6** Suponga que en este instante se está ejecutando el proceso B y se accede a la dirección de Memoria principal 9016. Dado que el sistema dispone de una MMU básica y asignación contigua de memoria, rellene los valores en los recuadros marcados con * para cada uno de los elemento, de manera que se pueda llevar a cabo dicho acceso. Indique también el nombre de cada elemento en los recuadros próximos a ellos marcados con ()




Dirección Lógica se obtiene → Dirección Física -Registro base = 9016-8192= 824
Registro Límite corresponde al tamaño del proceso B → 4096
Registro Base corresponde a la dirección de memoria a partir de la cual se encuentra ubicado el proceso B → 8192

**7.** Una familia de procesadores de Intel con 32 bits de dirección, trabaja con una arquitectura de memoria paginada con dos niveles de paginación y páginas de 4KBytes. Para ambos niveles de paginación, cada entrada de la tabla de páginas (descriptor) ocupa 4 Bytes y cada tabla de páginas ocupa 4KBytes. Para este sistema indique de forma justificada:

**(1.2 puntos = 0,4 x 3)**

**7**

**a)** Número de páginas que puede llegar a tener, en total, un proceso

**Número máximo de páginas es de $2^{20}$ → 1 Mpáginas**

Direcciones lógicas de 32bits:

- Páginas de 4KBytes = $2^2$ $2^{10}$ → 12 bits para desplazamiento
- 32 bits - 12 bits = 20 bits para números de páginas

**b)** Tamaño en Bytes de la tabla de páginas de primer nivel y número máximo de entradas (descriptores) que puede llegar a tener dicha tabla

**Una única Página de primer nivel con 1024 descriptores y un tamaño de 4KBytes**

Direcciones lógicas de 32bits:

- Páginas de 4KBytes = $2^2$ $2^{10}$ → 12 bits para desplazamiento
- Cada tabla de páginas ocupa 4KBytes y descriptores de 4Bytes → 4KBytes/4Bytes = 1024 descriptores en cada página
- 1024 descriptores → 10 bits de la dirección lógica para el primer nivel

b31----------b22 b21----------b12 b11----------b0

| 1er nivel (10 bits) | 2º nivel (10 bits) | Desplazamiento (12 bits) |
|---|---|---|

**c)** Para un programa que ocupa 100MBytes, indique de forma razonada el número de descriptores de cada nivel y el espacio que consumen sus tablas de páginas

Un programa de 100MBytes → 100MBytes/4KBytes=25 Kpáginas → **25 Kdescriptores**
Cada tabla de páginas ocupa 4KBytes y contiene 1024 descriptores → **25 tablas de páginas de segundo nivel con 1024 descriptores cada una + una tabla de páginas de primer nivel con sólo 25 descriptores**

**8.** Sea un sistema con paginación por demanda, páginas de 4KBytes y direcciones lógicas y físicas de 24 bits. En dicho sistema se están ejecutando dos procesos A y B, a los que el sistema asigna 5 marcos que comparten con una política de reemplazamiento **de segunda oportunidad con ámbito global.** Ambos procesos tienen un tamaño de 4 páginas (de la 0 a la 3). La relación de los marcos asignados con las páginas de los procesos se muestra en la siguiente tabla:

**(1,8 puntos = 0,4 + 0,4 + 1,0)**

| Marco | Proceso: página | Tiempo de Carga | Tiempo. último acceso | Bit de Referencia | Bit de Validez |
|---|---|---|---|---|---|
| 0 | A:0 | 1 | 6 | 1 | 1 |
| 1 | B:1 | 2 | 15 | 1 | 1 |
| 2 | A:1 | 7 | 7 | 1 | 1 |
| 3 | B:2 | 8 | 12 | 1 | 1 |
| 4 | A:3 | 12 | 14 | 1 | 1 |

**8**

**a)** Sabiendo que hasta el instante t= 15 sólo se han producido fallos de página sin reemplazo, indique el contenido de las entradas de la **tabla de páginas del proceso B**, para cada uno de sus descriptores, no es necesario poner los bits de permisos, ni los tiempos.

| N º de Página | Marco | Referencia | Validez |
|---|---|---|---|
| 0 | - | - | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 3 | 1 | 1 |
| 3 | - | - | 0 |

La tabla de páginas de un proceso contiene el número de marco de aquellas páginas de dicho proceso que están en Memoria Principal

**b)** A partir del instante t= 16 la CPU emite las siguiente direcciones lógicas:

A:0x002345, A:0x002346, B:0x001B72, B:0x000B32, B:0x000B33,
A:0x000111, A:0x001222, A:0x000111, B:0x002ABC, B:0x002ABD

Suponga que se maneja un esquema de área activa, con un tamaño de ventana de 4. Determine las áreas activas de los procesos A y B tras completarse el último acceso.

Las páginas correspondientes (en hexadecimal) a las direcciones lógicas emitidas son:
**A:002, A:002**, B:001, B:000, B:000, **A:000, A:001, A:000**, B:002, B002
Con una ventana de 4 en t=16 el AreaActiva vendrá dada por:
AreaActiva(A)= {0, 1, 2}
AreaActiva(B)= {0, 2}

¿Puede producirse hiperpaginación?. Justifique la respuesta
La suma de los tamaños de las dos áreas activas es 5 que coincide con los marcos disponibles por tanto no se producirá hiperpaginación.
TamañoAreaActiva = Tamaño (AreaActiva(A))+ Tamaño (AreaActiva(B)) = 2+3=5
TamañoAreaActica=Cantidad de Marcos → No hay peligro de hiperpaginación

**c)** Calcule la serie de referencias correspondientes a la secuencia de direcciones del apartado b) :
La serie de referencias se obtiene (en hexadecimal) a partir de las páginas correspondientes a direcciones lógicas emitidas eliminando las repeticiones consecutivas a una misma página:
**A:002,** A:002, **B:001**, **B:000**, B:000, **A:000, A:001, A:000**, **B:00**2, B002
Por tanto la serie de referencia es:
**A:002, B:001, B:000, A:000, A:001, A:000, B:002**

Aplique el algoritmo de segunda oportunidad con ámbito global para dicha serie de referencias. Rellene la siguiente tabla (las columnas que sean necesarias, máximo de 8), con la evolución de los marcos de memoria de los procesos A y B a partir del instante t=16. La primera columna corresponde al instante t=15, en el que el proceso B accedió a su página 1. Indique en cada casilla de la tabla el proceso, la página y el valor de bit R tras realizar el acceso y cuando haya reemplazo marque la casilla elegida como víctima.
Según el enunciado hasta este instante no se han producido fallos de página y, por tanto no se ha aplicado todavía el algoritmos de reemplazo. Para aplicar el algoritmo de segunda oportunidad y elegir víctima, es necesario utilizar un puntero que nos indique cúal es la siguiente víctima a considerar.En este caso el puntero apuntará a la primera página que se haya cargado en memoria -> A:2.

| marco | B:1 (R) | A:2 | B:1 | B:0 | A:0 | A:1 | A:0 | B:2 |
|---|---|---|---|---|---|---|---|---|
| 0 | A:0 (1) | A:2 (1) | A:2 (1) | A:2 (1) | A:2 (1) | ->A:2 (1) | ->A:2 (1) | A:2 (0) |
| 1 | B:1 (1) | ->B:1 (0) | ->B:1 (1) | B:1 (0) | B:1 (0) | B:1 (0) | B:1 (0) | B:2 (1) |
| 2 | A:1 (1) | A:1 (0) | A:1 (0) | B:0 (1) | B:0 (1) | B:0 (1) | B:0 (1) | ->B:0 (1) |
| 3 | B:2 (1) | B:2 (0) | B:2 (0) | ->B:2 (0) | A:0 (1) | A:0 (1) | A:0 (1) | A:0 (1) |
| 4 | A:3 (1) | A:3 (0) | A:3 (0) | A:3 (0) | ->A:3 (0) | A:1 (1) | A:1 (1) | A:1 (1) |

Indique el número de reemplazos: Se han producido fallo de página todos ellos con reemplazo. Se han realizado 5 reemplazos.