Trabajo práctica N° 2.

1. **Responder sintéticamente**

    a. Un sistema operativo es una capa de software que se encarga de administrar los recursos de hardware y actúa como intermediario entre el usuario y el sistema.

    b. Para que un sistema operativo cumpla sus objetivos de gestionar el hardware, se necesitan principalmente: CPU para ejecutar instrucciones, memoria RAM para el acceso rápido a datos, placa base para conectar todos los componentes, dispositivos de almacenamiento para guardar datos a largo plazo y periféricos de entrada/salida para la interacción con el usuario.

    c. Los componentes de un sistema operativo son: el núcleo (kernel), el shell (interfaz de texto que permite la comunicación del usuario con el kernel) y el file system (sistema de archivos para la organización y gestión de los datos).

    d. Mecanismo por el cual un programa de usuario solicita un servicio al sistema operativo, el cual se ejecuta en un modo privilegiado con acceso a recursos de hardware. Para implementar una system call, se crea un número para la misma, se define la función de la lógica de la llamada en el kernel y se la registra en la tabla de system calls del sistema operativo.

    e. Un programa es un conjunto estático de instrucciones almacenadas en el disco o la memoria, mientras que un proceso es una abstracción utilizada por el sistema operativo para representar un programa en ejecución.

    f. El kernel debe contar con cierta información mínima sobre un proceso, la cual comprende: el identificador de proceso (PID), su estado, el espacio de direcciones, la prioridad, los permisos y la información de su proceso padre para gestionarlo de forma más eficiente. Toda esta información es almacenada en la estructura de datos conocida como Process Control Block (PCB).

    g. Los algoritmos de planificación tienen como objetivo optimizar el uso del CPU y mejorar así el rendimiento general del sistema.

    h. Un algoritmo de planificación apropiativo puede interrumpir la ejecución de un proceso para darle paso a la ejecución de otro, mientras que un algoritmo de planificación no apropiativo no puede interrumpir la ejecución de un proceso y

por lo tanto la CPU es acaparada por el mismo hasta que termina de ejecutarse o se bloquea por alguna operación de entrada/salida.

i.

- Short term scheduler: selecciona procesos de la cola de listos para asignarle la CPU y ponerlos en estado de ejecución. También gestiona los cambios de contexto y garantiza el uso constante y eficiente de la CPU.

- Long term scheduler: decide que procesos se admiten desde la cola de trabajo (en el disco) a la cola de listos (en memoria principal), cargándolos para su ejecución. Busca un equilibrio entre procesos CPU-bound y procesos I/O-bound para optimizar el rendimiento general del sistema.

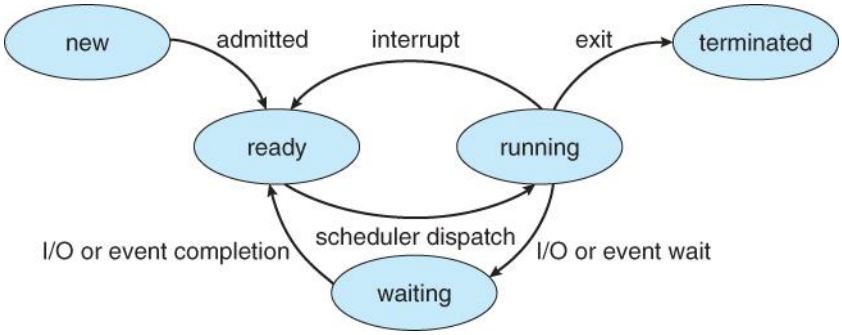
- Medium term scheduler: gestiona la transición de procesos entre la memoria principal y la memoria secundaria. Reduce el grado de multiprogramación, mueve procesos suspendidos para su reanudación y mejora la utilización de recursos y el tiempo de respuesta del sistema.

j.

- Dispatcher: transfiere el control de la CPU al proceso seleccionado por el short term scheduler.
- Loader: carga y prepara los programas para su ejecución, localizando el archivo ejecutable, asignando memoria, cargando el programa en esa memoria y resolviendo dependencias como las bibliotecas del sistema.

k. Un proceso es CPU-bound si su tiempo de ejecución está limitado principalmente por la velocidad de la CPU, mientras que un proceso es I/O-bound si su tiempo de ejecución está limitado por la velocidad del subsistema de entrada/salida.

l. New, Ready, Running, Waiting y Terminated.



- New: el proceso no ha sido cargado aún en la memoria principal, pero hay intenciones de ejecución por parte del usuario. Se crea la PCB.

- Ready: el proceso tiene todos los recursos necesarios para ejecutarse y está esperando que el sistema operativo le asigne la CPU para comenzar o continuar su ejecución.

- Running: el proceso está utilizando activamente la CPU y ejecutando sus instrucciones.

- Waiting: el proceso necesita un recurso que no está disponible (como una operación e/s) o está esperando que ocurra un evento externo, como recibir una señal.

- Terminated: el proceso ha finalizado su ejecución y está siendo retirado del sistema.

m. -

n.

- Tiempo de retorno (TR): es el intervalo de tiempo que comprende desde que un proceso llega al sistema hasta que finaliza su ejecución.

- Tiempo de espera (TE): es el tiempo acumulado que un proceso se encuentra en el sistema sin utilizar la CPU (incluidos los tiempos donde está ejecutando una operación de e/s).

o.

- Tiempo promedio de retorno (TPR): indica cuanto tiempo tarda, en promedio, un lote de procesos en completar todas sus operaciones y salir del sistema.

- Tiempo promedio de espera (TPE): tiempo promedio que los procesos de un lote están en el sistema sin hacer uso de la CPU.

p. El tiempo de respuesta es el intervalo entre el momento en que se presenta una solicitud o un proceso a un sistema y el momento en que se genera la primera respuesta.

2.

- **FCFS (**First Come First Served): algoritmo de scheduling no apropiativo que ejecuta los procesos en el orden exacto en el que llegan. Si llegaran P1 (necesita 5 unidades de tiempo), P2 (necesita 9 unidades de tiempo) y P3 (necesita 3 unidades de tiempo), P1 es el primero en llegar a la cola y por lo tanto el primero en ejecutarse durante 5 unidades de tiempo, cuando termina y se libera la CPU se ejecuta P2 por 9 unidades de tiempo y cuando termina, se ejecuta P3 por 3 unidades de tiempo.

  Este algoritmo cuenta con un parámetro clave para su funcionamiento:

  - Tiempo de llegada de los procesos a la cola de listos

  FCFS es simple y adecuado para sistemas por lotes y procesos de tiempos de ráfaga cortos. Sus ventajas consisten en ser fácil de implementar y entender. Por otro lado, sus desventajas consisten en posibles tiempos de espera promedio altos, especialmente si un proceso largo llega primero a la cola.

- **SJF** (Shortest Job First): algoritmo de scheduling en el que el planificador selecciona el proceso disponible con menor tiempo de ráfaga para ejecutarse. Si dos o más procesos coinciden en el tiempo de ráfaga, se ejecuta el primero que haya llegado a la cola de listos. Si llegan P1 (tiempo de llegada 0, ráfaga 6), P2 (tiempo de llegada 2, ráfaga 9) y P3 (tiempo de llegada 4, ráfaga 3), el primero en ejecutarse es P1 ya que al llegar a la cola la CPU está libre. Mientras P1 se ejecuta, llegan P2 y P3 a la cola de listos. Cuando P1 finaliza su

ejecución, el siguiente proceso que se elige para ejecutar es P3 debido a que es el que menor tiempo de ráfaga posee, aunque haya llegado más tarde que P2.

Este algoritmo tiene algunos parámetros de funcionamiento:

- Tiempo de ráfaga del proceso (duración de su ejecución en la CPU)
- Tiempo de llegada del proceso a la cola de listos

SJF es mejor para minimizar el tiempo promedio de espera si se conocen las duraciones de los procesos. Aunque por otro lado es difícil de implementar en la práctica porque requiere conocer de antemano la duración de cada proceso, la cual no siempre es predecible.

- **Round Robin**: algoritmo de scheduling que asigna tiempo de CPU a los procesos de manera equitativa en una cola circular, donde cada proceso se ejecuta durante un "quantum". Supongamos que llegan P1 (4 unidades de ráfaga), P2 (6 unidades de ráfaga) y P3 (2 unidades de ráfaga) y un quantum de tiempo de 2 unidades. P1 comienza a ejecutarse y luego de 2 unidades de tiempo se interrumpe, entonces comienza a ejecutarse P2 durante 2 unidades de tiempo, luego se ejecuta P3 durante 2 unidades de tiempo. Como la cola es circular y P1 tenía un tiempo de ráfaga de 4 unidades no terminó de ejecutarse, por lo que vuelve a ser asignado a la CPU y se ejecuta durante 2 unidades más, completando su ejecución. Así sigue la cola de manera circular con los otros procesos hasta finalizar todos.

  Este algoritmo cuenta con un parámetro clave para su funcionamiento:

  - Quantum de tiempo (el intervalo de tiempo fijo que se asigna a cada proceso para ejecutarse en la CPU)

  Round Robin es ideal para sistemas de tiempo compartido e interactivos, garantizando equidad, un buen tiempo de respuesta y evitando la inanición de procesos. Por otro lado, dentro de sus desventajas podemos mencionar que su eficacia depende de la elección correcta del quantum, ya que si es demasiado pequeño aumenta la sobrecarga y si es demasiado grande se asemeja a FCFS.

- **Prioridades:** elige el proceso que será ejecutado basándose en un valor de prioridad asignado. Supongamos que llegan P1 (prioridad 2, tiempo de llegada 0, ráfaga 5), P2 (prioridad 1, tiempo de llegada 2, ráfaga 3) y P3

(prioridad 3, tiempo de llegada 4, ráfaga 4). Se asigna la CPU a P1, llega P2 con prioridad más alta y entonces se interrumpe la ejecución de P1 y se le asigna la CPU a P2, quedando P1 con un tiempo restante de ejecución de 3 unidades. Cuando P2 finaliza, se elige nuevamente a P1 ya que su prioridad es más alta que la de P3 y al finalizar su ejecución, se ejecuta P3.

Este algoritmo cuenta con algunos parámetros de funcionamiento:

- Prioridad del proceso
- Naturaleza de la prioridad (estática o dinámica)
- Preemptivo o No preemptivo (si un proceso de mayor prioridad que llega a la cola puede o no interrumpir al que está actualmente ejecutándose)

El algoritmo de prioridades es útil cuando ciertos procesos son más críticos que otros, aunque debe usarse con cuidado para evitar la inanición de procesos de baja prioridad (que nunca lleguen a ejecutarse).

3.

c. En el caso de este lote de procesos, el algoritmo con mayor eficiencia es el SJF.

4.

c. En el caso de este lote de procesos, el algoritmo con mayor eficiencia es el SJF.

d. Si el quantum del algoritmo RR es demasiado chico, el TPR y el TPE son altos.

e. El algoritmo RR con un valor de quantum alto se utilizará en sistemas donde predominen los procesos largos y sea aceptable un mayor tiempo de respuesta. Esto implica una menor sobrecarga por cambios de contexto, mejorando la eficiencia de la CPU para procesos extensos. Por lo otro lado, el tiempo de respuesta para procesos cortos e interactivos será más lento, habrá un mayor tiempo promedio de espera y un posible comportamiento similar a FCFS.

5.

b. En el caso de que los procesos sean cortos, el SRTF asegura un tiempo promedio de retorno y tiempo promedio de espera más bajos en comparación con otros algoritmos de planificación.

6. -

7.

    a. La inanición es cuando un proceso no puede acceder a la CPU para ejecutarse porque otros procesos de mayor prioridad están haciendo uso del recurso de forma continua.

    b. La inanición puede ser generada por el algoritmo de planificación por prioridades o por el SJF (Shortest Job First).

    c. Sí, para evitar la inanición se implementa la técnica de envejecimiento, la cual incrementa gradualmente la prioridad de los procesos que llevan mucho tiempo en espera, garantizando de esta manera que eventualmente obtengan el recurso necesario para ejecutarse.

8. Los incisos son prácticos.

9.

    a. Las principales desventajas del RR con procesos CPU-bound y I/O-bound incluyen una alta sobrecarga de cambios de contexto que reduce la eficiencia, tiempos de respuesta y espera prolongados para trabajos largos y una subutilización de la CPU si el quantum es muy grande. La planificación Round Robin no optimiza el tiempo de espera ni de respuesta, ya que no distingue entre procesos largos y cortos, tratando a todos por igual.

    b. Las principales desventajas del SRTF con procesos CPU-bound y I/O-bound son la posibilidad de inanición y la dificultad para predecir la duración del proceso, lo que limita su aplicabilidad práctica y puede generar ineficiencias en la planificación.

10. Los incisos son prácticos.

11. Sí, es probable que el quantum de un proceso nunca llegue a 0, porque en el algoritmo VRR el contador solo se decrementa mientras el proceso está en ejecución. Si el proceso de bloquea antes de agotar su quantum, el tiempo restante se guarda y se reutiliza al desbloquearse. Si al desbloquearse el proceso se bloqueara nuevamente antes de consumir las unidades de tiempo restantes, el quantum nunca llegaría a 0.

12.

    a. -

13.

    a. En el caso de los procesos interactivos, la mejor opción para administrar las colas es el algoritmo de planificación Round Robin o Virtual Round Robin, ya que garantizan que los procesos reciban CPU con frecuencia y respondan rápido a la entrada del usuario. Por otro lado, en el caso de los procesos batch, las mejores opciones son los algoritmos de planificación FCFS o SJF debido a que reducen el tiempo promedio de espera y maximizan el rendimiento del procesador, no la rapidez de respuesta.

    b. Para planificar las dos colas (procesos interactivos y procesos batch) implementaría un algoritmo de planificación por colas multinivel con prioridad fija. De esta manera, la cola de procesos interactivos tendrá mayor prioridad (ya que requieren una respuesta rápida) y la cola de procesos batch tendrá menor prioridad y solo será atendida cuando la cola de interactivos esté vacía.

       Con esta implementación, el algoritmo de prioridades determina sobre qué cola elegir un proceso, garantizando buena respuesta para los interactivos sin descuidar la ejecución de los batch cuando el sistema está libre.

14.
15.
16. Todos estos ejercicios son prácticos.
17.
18.

19.

a. “hola”, salida del comando ls.
b. “hola”, salida del comando ps, salida del comando ls.
c. “anda a rendir el primer parcial de promo”, “estoy comenzando el examen”, salida del comado ps, “¿cómo te fue?”

20. -

21. -

22. La MMU (Memory Management Unit) es un componente del hardware del procesador que gestiona la memoria virtual traduciendo direcciones virtuales a direcciones físicas y aplicando protecciones de memoria.

23. El espacio de direcciones es un conjunto de direcciones virtuales que el sistema operativo asigna a cada proceso, permitiéndole almacenar y acceder a sus datos y código como si tuviera acceso a un espacio de memoria contiguo y privado.

24. Las direcciones virtuales son representaciones lógicas de las celdas de memoria RAM que utiliza la CPU, las cuales son traducidas por la MMU para poder acceder al espacio físico de la memoria al que hacen referencia.

25.

a. **Particiones fijas**: la memoria es dividida en bloques de tamaño fijo definidos previamente a iniciar el sistema. Cada proceso es cargado en una partición de tamaño igual o menor al suyo. Es fácil de implementar, evita que los procesos interfieran entre sí y garantiza una cantidad mínima de memoria para cada proceso. Por otro lado, esta técnica genera fragmentación interna (el tamaño del proceso es menor que la partición, dejando memoria sin usar), es muy rígida porque limita la cantidad de procesos que se pueden ejecutar simultáneamente y desperdicia memoria si los procesos son pequeños

   Particiones dinámicas: el tamaño de las particiones varía y se crea según la necesidad del proceso al momento de cargarlo. Es eficiente debido a que asigna solo la memoria necesaria, es flexible ya que se adapta a procesos de diferentes tamaños, mejorando la multiprogramación y no produce fragmentación interna. Sin embargo, esta técnica genera fragmentación externa (se crean pequeños huecos inútiles de memoria libre entre las particiones), para evitar la fragmentación el sistema lleva a cabo la tarea de

compactar los pequeños huecos, lo cual consume tiempo y recursos del procesador y además posee otro nivel de complejidad.

b. -
c. -

26. Las particiones fijas con tamaño fijo ofrecen simplicidad y fácil implementación, pero generan fragmentación interna y rigidez. En cambio, las particiones fijas con tamaño variable aportan más flexibilidad, reduciendo la fragmentación interna.

27.

a. La fragmentación interna es espacio de memoria asignado a un proceso que no se utiliza, mientras que la fragmentación externa es un espacio de memoria libre que no es lo suficientemente grande ni contiguo para alojar nuevos procesos.

b. Una de las posibles técnicas a implementar para subsanar el problema de la fragmentación externa es la paginación, la cual divide la memoria física y el espacio de direcciones de los procesos en bloques de tamaño fijo llamados “marcos” y “páginas”, respectivamente. De esta manera las páginas pueden ser alojadas en marcos no contiguos.

28.

a. **Best fit**: al recibir un proceso, busca en toda la memoria disponible el hueco libre que sea de menor tamaño pero lo suficientemente grande para alojar el proceso.

b. **Worst fit:** busca la partición de memoria más grande disponible para asignarla al proceso entrante, con el objetivo de dejar un fragmento grande de espacio libre.

c. **First fit:** asigna al proceso entrante el primer bloque libre buscado secuencialmente desde el principio que sea lo suficientemente grande para alojarlo.

d. **Next fit:** busca en la memoria el primer hueco disponible que pueda alojar al proceso, pero con la particularidad de que la próxima búsqueda no comienza desde el principio, sino desde la última posición asignada.

29.

a. La segmentación divide la memoria en bloques lógicos de tamaño variable (segmentos), reflejando la estructura del programa (código, datos, pila). El sistema operativo utiliza una tabla de segmentos, con información de la dirección base y límite de cada segmento dentro del espacio de direcciones del proceso.

b. -

c. -

d. Sí, en la segmentación se puede generar fragmentación externa debido a que cuando los segmentos de un proceso son eliminados generan huecos de memoria que pueden no ser lo suficientemente grandes para satisfacer las necesidades de nuevos procesos.

e. No hay una única mejor técnica, siempre depende del objetivo que se quiera alcanzar. Sin embargo, First Fit y Best Fit suelen ofrecer mejor rendimiento y utilización de la memoria, mientras que Worst Fit es más lento y puede generar fragmentación externa y Next Fit es más rápido que First Fit pero corre el riesgo de pasar por alto bloques útiles.

30.

Segmento	Dir. Base	Tamaño
0	102	12500
1	28699	24300
2	68010	15855
3	80001	400

a. **0000:9001**

   - **Dirección límite:** 102+12500 = 12602
   - **Conversión:** 102+9001 = 9103

b. **0001:24301**

   - **Dirección límite:** 28699+24300 = 52999
   - **Conversión:** 28699+24301 = 53000. Acceso erróneo a memoria, la dirección se encuentra fuera del rango del segmento.

c. **0002:5678**

   - **Dirección límite:** 68010+15855 = 83865
   - **Conversión:** 68010+5678 = 73688

d. **0001:18976**

   - **Dirección límite:** 28699+24300 = 52999
   - **Conversión:** 28699+18976 = 47675

e. **0003:0**

   - **Dirección límite:** 80001+400 = 80401
   - **Conversión:** 80001+0 = 80001

31.

a. La paginación es una técnica de asignación de memoria que divide la memoria física y la memoria virtual en bloques iguales de tamaño fijo, llamados “marcos” y “páginas”, respectivamente. El sistema operativo mantiene una tabla de páginas y la MMU use el número de página como índice en la tabla para encontrar el marco físico donde se encuentra alojada la misma
b. Consultar en teoría.
c. --

d. Sí, en este esquema se puede producir fragmentación interna, ya que los procesos pueden no completar la última página que se les asigna. Por otro lado, la fragmentación externa no ocurre debido a que los marcos utilizados pueden no ser contiguos.

32.

- **Tamaño de página:** 512 bytes.
- Cada dirección de memoria referencia 1 byte.
- Se tiene una memoria de 10240 bytes.
- Los marcos se enumeran desde 0 y comienzan en la dirección física 0.
- Suponga un proceso P1 con un tamaño lógico de 2000 bytes con la siguiente tabla de páginas.

Página	Marco
0	3
1	5
2	2
3	6

a.

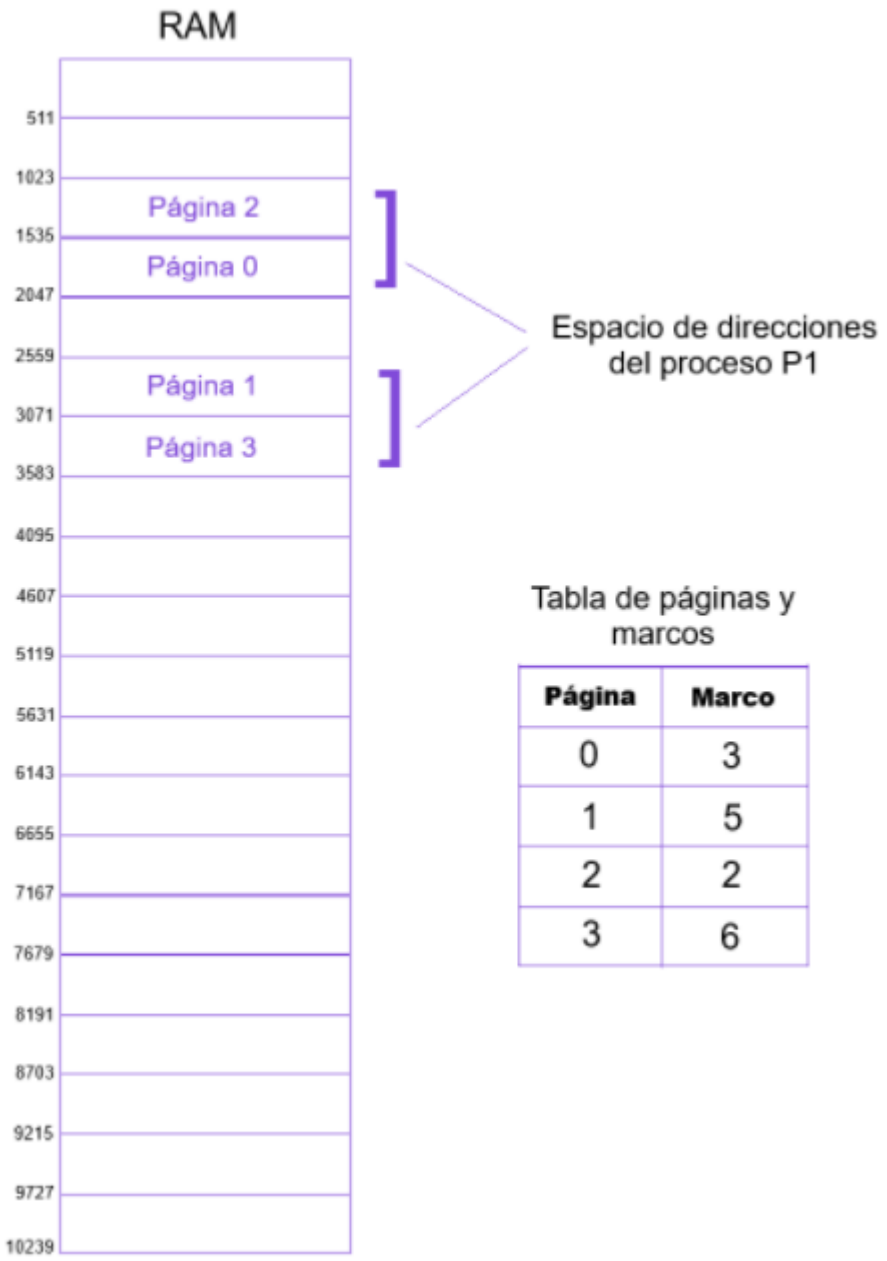


Tabla de páginas y marcos

Página	Marco
0	3
1	5
2	2
3	6

b.

i. **Nro página:** 35 DIV 512 = 0

Marco: 3

Valor base del marco: 3*512 = 1536

Desplazamiento: 35 MOD 512 = 35

Dirección física: 1536+35 = 1571

Sí, la dirección lógica 35 pertenece al espacio de direcciones del proceso P1 y corresponde a la dirección física 1571 del marco 3.

ii. **Nro página:** 512 DIV 512 = 1

Marco: 5

Valor base del marco: 5*512 = 2560

Desplazamiento: 512 MOD 512 = 0

Dirección física: 2560+0 = 2560

Sí, la dirección lógica 512 pertenece al espacio de direcciones del proceso P1 y corresponde a la dirección física 2560 del marco 5.

iii. **Nro página:** 2051 DIV 512 = 4

No, la dirección lógica 2051 no pertenece al espacio de direcciones del proceso P1, ya que la página 4 no existe para dicho proceso.

iv. **Nro página:** 0 DIV 512 = 0
Marco: 3
Valor base del marco: 3*512 = 1536
Desplazamiento: 0
Dirección física: 1536+0 = 1536

Sí, la dirección lógica 0 pertenece al espacio de direcciones del proceso P1 y corresponde a la dirección física 1536 del marco 3.

v. **Nro página:** 1325 DIV 512 = 2
Marco: 2
Valor base del marco: 2*512 = 1024
Desplazamiento: 1325 MOD 512 = 301
Dirección física: 1024+301 = 1325

Sí, la dirección lógica 1325 pertenece al espacio de direcciones del proceso P1 y corresponde a la dirección física 1325 del marco 2.

vi. **Nro página:** 602 DIV 512 = 1
Marco: 5
Valor base del marco: 5*512 = 2560
Desplazamiento: 602 MOD 512 = 90
Dirección física: 2560+90 = 2650

Sí, la dirección lógica 2560 pertenece al espacio de direcciones del proceso P1 y corresponde a la dirección física 2650 del marco 5.

Para calcular el residuo en la calculadora tomo la parte entera del cociente entre los valores, lo multiplico por el divisor y ese resultado se lo resto al dividendo.

Por ejemplo: **35 MOD 512**

1. 35/512 = 0
2. 0 x 512 = 0
3. 35 - 0 = 35.

c.

I. **Nro marco:** 509 DIV 512 = 0

La dirección física 509 no se corresponde con ninguna dirección lógica del proceso P1, ya que ninguna página del proceso ocupa el marco 0 de la memoria principal.

II. **Nro marco:** 1500 DIV 512 = 2
Nro página: 2 (utilizo la tabla)
Valor base de página: 2*512 = 1024
Desplazamiento: 1500 MOD 512 = 476
Dirección lógica: 1024+476 = 1500

La dirección física 1500 se corresponde a la dirección lógica 1500 del proceso P1.

III. **Nro marco:** 0 DIV 512 = 0

La dirección física 0 no se corresponde con ninguna dirección lógica del proceso P1, ya que ninguna página del proceso ocupa el marco 0 de la memoria principal.

IV. **Nro marco:** 3215 DIV 512 = 6
Nro página: 3
Valor base de página: 3*512 = 1536
Desplazamiento: 3215 MOD 512 = 143
Dirección lógica: 1536+143 = 1679

La dirección física 3215 se corresponde con la dirección lógica 1679 del proceso P1.

V. **Nro marco:** 1024 DIV 512 = 2
Nro página: 2
Valor base de página: 2*512 = 1024
Desplazamiento: 1024 MOD 512 = 0
Dirección lógica: 1024+0 = 1024

La dirección física 1024 se corresponde con la dirección lógica 1024 del proceso P1.

VI. **Nro marco:** 2000 DIV 512 = 3
Nro página: 0
Valor base de página: 0*512 = 0
Desplazamiento: 2000 MOD 512 = 464
Dirección lógica: 0+464 = 464

La dirección física 2000 se corresponde con la dirección lógica 464 del proceso P1.

33.

- **Tamaño de página:** 2048 bytes.
- Cada dirección hace referencia 1 byte.
- Se posee la siguiente tabla de páginas:

Página	Marco
0	16
1	13
2	9
3	2
4	0

I. **Número página:** 5120 DIV 2048 = 2
Marco: 9
Valor base del marco: 9*2048 = 18432
Desplazamiento: 5120 MOD 2048 = 1024
Dirección física: 18432+1024 = 19456

II. **Número página:** 3242 DIV 2048 = 1
Marco: 13
Valor base del marco: 13*2048 = 26624
Desplazamiento: 3242 MOD 2048 = 1194
Dirección física: 26624+1194 = 27818

III. **Número página:** 1578 DIV 2048 = 0
Marco: 16
Valor base del marco: 16*2048 = 32768
Desplazamiento: 1578 MOD 2048 = 1578
Dirección física: 32768+1578 = 34346

IV. **Número página:** 2048 DIV 2048 = 1
Marco: 13
Valor base del marco: 13*2048 = 26624
Desplazamiento: 2048 MOD 2048 = 0
Dirección física: 26624+0 = 26624

V. **Número página:** 8191 DIV 2048 = 3
Marco: 2
Valor base del marco: 2*2048 = 4096
Desplazamiento: 8191 MOD 2048 = 2047
Dirección física: 4096+2047 = 6143

34.

a. Un tamaño de página pequeño reduce la fragmentación interna pero incrementa el número de fallos de página, lo que puede aumentar la sobrecarga del sistema y el tiempo de acceso inicial a datos.

b. Un tamaño de página grande ofrece menor cantidad de fallos de página pero aumenta la fragmentación interna y requiere tablas de páginas más grandes para administrar las mismas.

35.

a. **Tabla de 1 nivel:** cada entrada de la tabla mapea una página lógica de un proceso a una página física de la memoria principal. La dirección virtual de un proceso se divide en un número de página y un desplazamiento.}

b. **Tabla de 2 niveles:** divide la tabla de páginas en sub-tablas. La tabla de nivel 1 apunta a las tablas de páginas del nivel inferior (2 o más), que a su vez contienen entradas de tabla de páginas finales que apuntan a los marcos en memoria.

c. **Tabla de páginas invertidas:** gestiona la memoria haciendo que cada entrada de la tabla corresponda a un marco de memoria física, no a una página virtual por proceso. Para traducir una dirección lógica, el sistema operativo busca la página virtual y el PID correspondiente dentro de la tabla invertida, que luego indica el número de marco físico donde se encuentra esa página.

36.

a.

- **Tamaño dirección:** 32bits
- **Tamaño página:** 2048 bytes.

Como cada dirección dentro de una página identifica un byte, el número de bytes por página determina cuantos bits se usan para el desplazamiento.

- **Bits para desplazamiento:** log2 (2048) = 11

Al tener la cantidad de bits que se utilizan en la dirección para determinar el desplazamiento, los restamos del valor total de bits y obtenemos la cantidad que se utilizará para identificar la página.

- **Bits para el número de página:** 32-11 = 21

b. El tamaño lógico máximo de un proceso será de 2^{32} bytes.

c. El tamaño máximo de páginas que puede tener el proceso P1 es 2^{21}

d.

1. 1GiB = 2^{30} bytes
   Por lo tanto, 4GiB = $4*2^{30} = 2^2 * 2^{30} = 2^{32}$ bytes.

2. 2^{32} (tamaño total de la memoria) / 2^{11} (tamaño de un frame, el cual equivale al de una página) = 2^{21} aplicando división de potencias de igual base (se restan exponentes).

e. **Cantidad de páginas:** 51358 (tamaño del código) / 2048 (tamaño de página) = 26. La última página tendrá mucha fragmentación interna debido a que no se ocupa totalmente su tamaño.

f. **Cantidad de páginas:** 68131 (tamaño de los datos) / 2048 (tamaño de página) = 34. La última página también contará con fragmentación interna debido a que no se ocupa totalmente su tamaño.

g.

- **Bytes asignados para código:** 53248
- **Bytes realmente ocupados por el código:** 51358
- **Bytes de fragmentación interna:** 53248-51358 = 1890.

- **Bytes asignados para datos:** 69632
- **Bytes realmente ocupados por los datos:** 68131
- **Bytes de fragmentación interna:** 69632-68131 = 1501

En total se generaron 3391 bytes de fragmentación interna.

h.

- **Bits para página:** 20
- **Tamaño del PTE:** 1024 bytes = 2^{10}
- **Tamaño mínimo TB del P1:** $2^{20} * 2^{10} = 2^{30}$