Examen Julio de 2012

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones). Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva y cada parte del problema de teórico en una hoja nueva.**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un mínimo de 60 puntos.**

Finalización

- **El examen dura 4 horas.**

Problema 1 (33 puntos)

1. Describa que entiende por un sistema con multiprocesamiento simétrico (*SMP – Symetric Multi-Processing*).
2. Cómo se implementa generalmente la protección de CPU?
3. Qué estado tiene que tener un proceso para poder ser seleccionado por el planificador (*scheduler*) del sistema operativo?.
4. Describa el planificador de colas multinivel con retroalimentación (*Multi-Level Feedback Queue*).
5. Dos hilos que pertenecen a diferentes procesos qué cosas comparten? Justifique.
6. Describa que entiende por memoria virtual.
7. Para qué es utilizado la pila (*stack*) y *heap* de un proceso?
8. Describa como se representan los datos de un archivo en un sistema tipo:
   i. FAT (*File Allocation Table*).
   ii. I-Nodo (*Index Node*).
9. Para qué sirve el servicio de *buffering* que brinda el subsistema de Entrada/Salida.

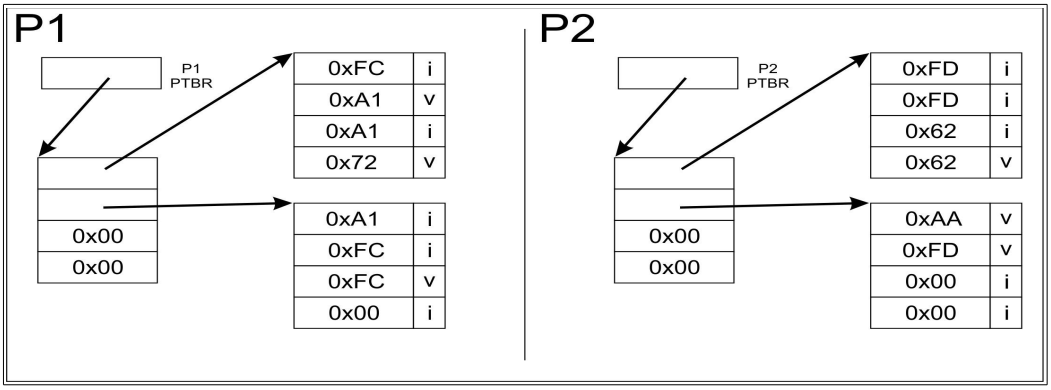
Problema 2 (33 puntos)

Sea un sistema operativo con un planificador round-robin con quantum de 2 unidades de ejecución en un sistema computacional mono-procesador. El sistema utiliza: memoria virtual con direcciones virtuales de 16 bits, paginación por demanda con asignación de marcos local de 3 marcos por proceso, algoritmo de reemplazo LRU (*Least Recently Used*) y un esquema de traducción de dos niveles.

Los procesos de este sistema pueden ejecutar 3 tipos de instrucciones: accesos a memoria (M), acceso a Entrada/Salida (I) y operaciones con registros (Op). Ejecutar una instrucción consume una unidad de tiempo de ejecución. Las operaciones de acceso a memoria reciben como parámetro la dirección virtual a la cual se accede, ej. M(X, Y, Z) representa: *X* referencia sobre la tabla de primer nivel, *Y* referencia sobre la tabla de segundo nivel, *Z* desplazamiento. Si un proceso intenta acceder a una página cuya entrada es inválida consumirá 2 unidades de tiempo más (aparte de la unidad consumida por la ejecución de la instrucción).

Las operaciones de E/S reciben como parámetro la cantidad de unidades de ejecución que el proceso esperará para que culmine la operación. Ej.: I(5) el proceso ejecuta la instrucción de E/S y luego espera por 5 unidades de tiempo de ejecución para que se complete el pedido de E/S.

Sean la siguientes tablas de páginas para los procesos P1 y P2:



Notas:

- La cantidad de entradas en las tablas de página de primer y segundo nivel es de 4 y se numeran como 0, 1, 2 y 3.
- Las direcciones 0x00 representan memoria virtual no utilizada por el proceso.

Se pide:

1. Determine el tamaño en bytes de las páginas y marcos. 3Pts.
2. Determine el espacio virtual utilizado y la memoria residente en bytes para los procesos P1 y P2 según las tablas de páginas presentadas en la figura. 5Pts.
3. Sea la siguiente secuencia de operaciones por parte de los procesos P1 y P2; y suponiendo que en el primer instante de tiempo se le asigna el procesador al proceso P1 (ambos procesos inician en t=0). 22Pts.

P1	I(2)	M(0,3,10)	M(0,1,56)	Op	M(1,2,10)	M(1,0,8)	M(0,1,28)	Op	M(0,3,28)	M(0,3,12)
P2	M(0,3,3)	M(1,0,10)	M(1,1,10)	Op	Op	M(0,1,28)	M(0,3,3)	M(0,1,34)	M(0,1,35)	M(0,2,15)

   a) Muestre un esquema que para cada instante de tiempo indique la operación que se esta ejecutando y el estado de las tablas de página luego de la ejecución de dicha operación.

   b) Determine cuántos fallos de página se producen.

4. Indique que sucede si el proceso P2 realiza la operación M(2, 0, 43). 3Pts.

1)

Las direcciones virtuales son de 16 bits y se componen de dos niveles de tablas de páginas.

Como la tabla de páginas de primer y segundo nivel son de 4 entradas, entonces se necesitan 2 bits para direccionar en cada nivel y quedan 12 bits para el desplazamiento sobre la página.

De esa forma, el tamaño de la página es de 2^{12} = 4096 bytes.

Por definición los marcos son del mismo tamaño que las páginas en paginación bajo demanda.

2)

El espacio virtual utilizado por P1 es de 7 páginas (28Kb - 28672 bytes), mientras que la memoria residente (espacio en memoria física utilizado por el proceso) es de 3 marcos (12Kb - 12288 bytes).

El espacio virtual utilizado por P1 es de 6 páginas (24Kb - 24576 bytes), mientras que la memoria residente (espacio en memoria física utilizado por el proceso) es de 3 marcos (12Kb - 12288 bytes).

3.a)

t0) P1 ejecuta I(2)

El estado de las tablas de pagina no es modificado dado que no existen accesos a memoria.

El estado del LRU no se modifica dado que no existen accesos a memoria.

P1 pasa ha estado de bloqueado

t1) P2 ejecuta M(0,3,3)

El estado de la tabla de paginas de P2 no es modificado dado que se accede a una pagina valida.

El estado del LRU del proceso P2 ahora tiene como pagina mas recientemente accedida a la pagina 0|3

LRU2 = {0|3, X, Y}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

t2) P2 ejecuta M(1,0,10)

El estado de la tabla de paginas de P2 no es modificado dado que se accede a una pagina valida.

LRU2 = {1|0, 0|3, X}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T3) En este instante de tiempo P1 finaliza su entrada/salida (es decir pasa de estado bloqueado a lista para ejecutar) y a P2 se le finaliza su quantum, se decidió darle prioridad al proceso que viene de entrada salida.

P1 ejecuta M(0,3,10)

El estado de la tabla de paginas de P1 no es modificado dado que se accede a una pagina valida.

LRU1 = {0|3, X, Y}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T4) P1 ejecuta M(0,1,56)

El estado de la tabla de paginas de P1 no es modificado dado que se accede a una pagina valida.

LRU1 = {0|1, 0|3, Y}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que

no existen accesos a memoria por parte del proceso P2.

T5) P2 ejecuta M(1, 1, 10)

El estado de la tabla de paginas de P2 no es modificado dado que se accede a una pagina valida.

LRU2 = {1|1, 1|0, 0|3}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T6) P2 ejecuta Op

El estado de la tabla de paginas de P2 no es modificado dado que P2 no realiza accesos a memoria, por el mismo motivo no cambia el estado del LRU.

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T7) P1 ejecuta Op

El estado de la tabla de paginas de P1 no es modificado dado que P1 no realiza accesos a memoria, por el mismo motivo no cambia el estado del LRU.

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T8) P1 ejecuta M(1, 2, 10)

El estado de la tabla de paginas de P1 no es modificado dado que se accede a una pagina valida.

LRU1 = {1|2, 0|1, 0|3}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T9) P2 ejecuta Op

El estado de la tabla de paginas de P2 no es modificado dado que P2 no realiza accesos a memoria, por el mismo motivo no cambia el estado del LRU.

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T10) P2 ejecuta M(0, 1, 28)

Aquí se produce un fallo de pagina y se comienza con el proceso de swap la tabla de pagina de P2 y el LRU de P2 no serán modificados hasta la finalización de la operación (luego de consumidas 2 unidades de tiempo mas).

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T11) P1 ejecuta M(1, 0, 8)

Aquí se produce un fallo de pagina y se comienza con el proceso de swap la tabla de paginas de P1 y el LRU de P1 no serán modificados hasta la finalización de la operación (luego de consumidas 2 unidades de tiempo mas).

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T12) P1 ejecuta M(1, 0, 8) (primera de las dos unidades de tiempo extras necesarias para completar el acceso a una pagina invalida). La tabla de paginas y el LRU del P1 no serán modificados hasta la finalización de la operación.

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T13) P2 ejecuta M(0, 1, 28) (primera de las dos unidades de tiempo extras necesarias para completar el acceso a una pagina invalida). La tabla de paginas y el LRU del P2 no serán modificados hasta la finalización de la operación.

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T14) P2 ejecuta M(0, 1, 28) (segunda unidad de tiempo extras necesarias para completar el acceso a una pagina invalida).

El nuevo estado de la tabla de paginas del proceso P2 es:

```
   |------|---|             |------|---|
   | 0xFD | i |             | 0xAA | v |
   | 0x62 | v |             | 0xFD | v |
   | 0x62 | i |             | 0x00 | i |
   | 0x62 | i |             | 0x00 | i |
   |------|---|             |------|---|
```

LRU2 = {0|1, 1|1, 1|0}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T15) P1 ejecuta M(1, 0, 8) (segunda unidad de tiempo extras necesarias para completar el acceso a una pagina invalida).

El nuevo estado de la tabla de paginas del proceso P1 es:

```
   |------|---|             |------|---|
   | 0xFC | i |             | 0x72 | v |
   | 0xA1 | v |             | 0xFC | i |
   | 0xA1 | i |             | 0xFC | v |
   | 0x72 | i |             | 0x00 | i |
   |------|---|             |------|---|
```

LRU1 = {1|0, 1|2, 0|1}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T16) P1 ejecuta M(0, 1, 28).

El estado de la tabla de paginas de P1 no es modificado dado que P1 accede a una pagina valida,

LRU1 = {0|1, 1|0, 1|2}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T17) P2 ejecuta M(0, 3, 3).

Aquí se produce un fallo de pagina y se comienza con el proceso de swap la tabla de pagina de P2 y el LRU de P2 no serán modificados hasta la finalización de la operación (luego de consumidas 2 unidades de tiempo mas).

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T18) P2 ejecuta M(0, 3, 3) (primera de las dos unidades de tiempo extras necesarias para completar el acceso a una pagina invalida). La tabla de paginas y el LRU del P2 no serán modificados hasta la finalización de la operación.

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T19) P1 ejecuta Op

El estado de la tabla de paginas de P1 no es modificado dado que P1 no realiza accesos a memoria, por el mismo motivo no cambia el estado del LRU.

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T20) P1 ejecuta M(0, 3, 28)

Aquí se produce un fallo de pagina y se comienza con el proceso de swap la tabla de paginas de P1 y el LRU de P1 no serán modificados hasta la finalización de la operación (luego de consumidas 2 unidades de tiempo mas).

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que

no existen accesos a memoria por parte del proceso P2.

T21) P2 ejecuta M(0, 3, 3) (segunda unidad de tiempo extras necesarias para completar el acceso a una pagina invalida).

El nuevo estado de la tabla de paginas del proceso P2 es:

0xFD	i		0xAA	**i**
0x62	v		0xFD	v
0x62	i		0x00	i
0xAA	**v**		0x00	i

LRU2 = {0|3, 0|1, 1|1}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T22) P2 ejecuta M(0, 1, 34).

El estado de la tabla de paginas de P2 no es modificado dado que P1 accede a una pagina valida,

LRU2 = {0|1, 0|3, 1|1}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T23) P1 ejecuta M(0, 3, 28) (primera de las dos unidades de tiempo extras necesarias para completar el acceso a una pagina invalida). La tabla de paginas y el LRU del P1 no serán modificados hasta la finalización de la operación.

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T24) P1 ejecuta M(0, 3, 28) (segunda unidad de tiempo extras necesarias para completar el acceso a una pagina invalida).

El nuevo estado de la tabla de paginas del proceso P1 es:

0xFC	i		0x72	v
0xA1	v		0xFC	i
0xA1	i		0xFC	**i**
0xFC	**v**		0x00	i

LRU1 = {0|3, 0|1, 1|0}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T25) P2 ejecuta M(0, 1, 35).

El estado de la tabla de paginas de P2 no es modificado dado que P1 accede a una pagina valida,

LRU2 = {0|1, 0|3, 1|1}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T26) P2 ejecuta M(0, 2, 15)

Aquí se produce un fallo de pagina y se comienza con el proceso de swap la tabla de pagina de P2 y el LRU de P2 no serán modificados hasta la finalización de la operación (luego de consumidas 2 unidades de tiempo mas).

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T27) P1 ejecuta M(0, 3, 12).

El estado de la tabla de paginas de P1 no es modificado dado que P1 accede a una pagina valida,

LRU1 = {0|3, 0|1, 1|0}

El estado de la tabla de paginas del proceso P2, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P2.

T28) P2 ejecuta M(0, 2, 15) (primera de las dos unidades de tiempo extras necesarias para completar el acceso a una pagina invalida). La tabla de paginas y el LRU del P2 no serán modificados hasta la finalización de la operación.

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

T29) P2 ejecuta M(0, 2, 15) (segunda unidad de tiempo extras necesarias para completar el acceso a una pagina invalida).

El nuevo estado de la tabla de paginas del proceso P2 es:

0xFD	i		0xAA	i
0x62	v		0xFD	**i**
0xFD	**v**		0x00	i
0xAA	v		0x00	i

LRU2 = {0|2, 0|1, 0|3}

El estado de la tabla de paginas del proceso P1, así como el estado del LRU no se modifica dado que no existen accesos a memoria por parte del proceso P1.

3.b)

El proceso P1 realiza 2 fallos de pagina.

El proceso P2 realiza 3 fallos de pagina.

En total existen 5 fallos de pagina.

4)

Esto es una dirección virtual válida, pero el proceso no tiene asignado el espacio virtual al que quiere acceder por lo que el sistema dará un fallo de direccionamiento. Un ejemplo de este tipo de error es cuando se direcciona fuera de un arreglo.

Problema 3 (34 puntos)

Se desea modelar usando monitores el control de acceso a la entrada de un estadio olímpico. Los asistentes llegan al estadio y forman fila frente a una de las 10 entradas. En la puerta de cada entrada hay Controlador que verifica el ticket de cada asistente antes de dejarlo o no pasar.

El Asistente debe ir a la entrada con la cola más corta. Los asistentes son atendidos por orden de llegada con la excepción de las futuras mamás que tendrán prioridad sobre todos los demás asistentes en cualquier entrada que les haya tocado.

Al llegar a la entrada el Controlador de dicha entrada verificará el ticket y dejará pasar a la persona en caso de no encontrar problemas.

En caso de encontrar un problema llamará al Supervisor para que determine la validez del ticket.

El supervisor recibirá los pedidos de validación y registrará el ticket en una planilla. Una vez que el supervisor recibe el pedido de validación el controlador seguirá atendiendo al siguiente en la entrada y el asistente involucrado en la validación quedará a la espera de que el supervisor valide la entrada.

Cuando el supervisor recibe 3 pedidos procederá a realizar la validación en el orden en que llegaron. La validación por parte del supervisor podrá tener como resultado la entrada del asistente o el rechazo del mismo.

Se dispone de las siguientes funciones:

- **miTicket(): ticket** Ejecutado por el asistente retorna su ticket.
- **soyFuturaMama(): boolean** Indica si el asistente es una futura mamá.
- **verificarTicket(ticket): boolean** Indica si el ticket es válido.
- **registrarProblema(ticket)** Registra un problema de una entrada que se pasa como parámetro.
- **validarTicket(ticket): boolean** Valida el ticket y decide si el asistente puede entrar o no.
- **entrar()** Ejecutado por los asistentes para ingresar al estadio.

Notas:

- Se deben modelar los procesos asociados a los Asistentes, los Controladores, y el Supervisor (no se permiten tareas auxiliares).

```
-------------------------------------------------------------------------
Monitor Colas
      var   colas: array [1..10] of integer

      Procedure entrar(var entrada: integer)
      var   mascorta: integer;
      begin
            mascorta := 1;
            for i:= 2 to 10 do
                  if colas[i] < colas[mascorta] then
                        mascorta = i;
                  end
            end
```

```
            colas[mascorta] := colas[mascorta] + 1;
            entrada := mascorta
      end

      Procedure salir(entrada: integer)
      begin
            colas[entrada] := colas[entrada] - 1;
      end

      begin
            for i:= 1 to 10
                  colas[i] := 0;
            end
      end
end Colas

Monitor Entrada
      var   controlador, asistente, mama: condition;
            ok: boolean;
            ticketsAsistentes: lista(Ticket);
            ticketsMamas: lista(Ticket);

      Procedure entrar (prioridad: boolean, ticket: Ticket, var entra: boolean)
      begin
            if prioridad then
                  ticketsMamas.add(ticket);
            else
                  ticketsAsistentes.add(ticket);
            end

            controlador.signal();

            if prioridad then
                  mama.wait();
            else
                  asistente.wait();
            end

            entra := ok;
      end

      Procedure controlar(var ticket: Ticket, var esMama: boolean)
      begin
            if ticketsMamas .size() = 0 and ticketsAsistentes .size() = 0 then
                  controlador.wait();
            end

            if ticketsMamas .size() > 0 then
                  ticket := ticketsMamas.first();
                  esMama := true;
            else
                  ticket := ticketsAsistentes.first();
                  esMama := false;
            end
      end

      Procedure terminar (entra: boolean, esMama: boolean)
      begin
            ok := entra
            if esMama then
                  mama.signal();
```

```
            else
                  asistente.signal();
            end
      end
end Entrada

Entradas := array [1..10] of Entrada;

Monitor Supervisor
      var   tickets: lista(Ticket);
            ok: boolean;
            supervisor, asistente: condition;

      Procedure validar(ticket: Ticket, var vale: boolean)
      begin
            tickets.add(ticket);
            supervisor.signal();
            asistente.wait();
            vale := ok
      end

      Procedure supervisar(var ticket: Ticket)
      begin
            if tickets.size() = 0 then
                  supervisor.wait();
            end
            ticket := tickets.first();
      end

      Procedure verificar(vale: boolean)
      begin
            ok := vale;
            asistente.signal();
      end
end Supervisor

Procedure asistente()
var   entrada: integer;
      ok: boolean;
begin
      Colas.entrar(entrada);
      Entradas[entrada].entrar(soyFuturaMama(), miTicket(), ok);
      Colas.salir(entrada);
      if ok then
            entrar();
      else
            Supervisor.validar(ticket, ok);
            if ok then
                  entrar();
            end
      end
end
```

```
Procedure controlador(int i)
var   ticket: Ticket;
      ok, mama: boolean;
begin
      while true do
            Entradas[i].controlar(ticket, mama);
            ok = verificarTicket(ticket);
            Entrdas[i].terminar(ok, mama);
      end
end

Procedure supervisor()
var   ticket: Ticket;
      registrados: integer;
      tickets: array[1..3] of Ticket;
begin
      registrados := 1;
      while true do
            Supervisor.supervisar(ticket);
            registrarProblema(ticket);
            tickets[registrados] = ticket;
            registrados := registrados + 1;
            if registrados > 3 then
                  for i:= 1 to 3 do
                        ok := verificarTicket(tickets[i]);
                        Supervisor.verificar(ok);
                  end
                  registrados := 1;
            end
      end
end

cobegin
      controlador(1);
      controlador(2);
      controlador(3);
      controlador(4);
      controlador(5);
      controlador(6);
```

```
      controlador(7);
      controlador(8);
      controlador(9);
      controlador(10);
      supervisor();
coend
```