Segundo Parcial Julio 2014

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida de los puntos del parcial.

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.**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos según el orden de hojas.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 15 minutos del parcial.**

Material

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

Finalización

- **El parcial dura 3 horas.**
- **Al momento de finalizar el parcial no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del parcial.**

Problema 1 (10 puntos)

1. Explique cómo funciona el método de asignación indexada para los sistemas de archivos.
2. En un sistema operativo con memoria virtual, explique los pasos que debe realizar el sistema operativo ante un fallo de página.
3. Describa brevemente ventajas y desventajas del uso de la encripción simétrica frente a la asimétrica y cite algún ejemplo del uso de ambas técnicas.
4. Explique el sistema RAID 0+1 y compárelo con los sistemas RAID 0 y RAID 1
5. En el contexto de Virtualización, explique el mecanismo de *traducción binaria* e indique en qué tipos de hipervisor puede ser utilizado.

Problema 2 (22 puntos) (8, 4, 10)

Sea un sistema operativo que utiliza un gestor de memoria, el cual implementa memoria virtual utilizando un modelo de paginación bajo demanda. Las direcciones virtuales son de 40 bits y la traducción se realiza a través de 3 niveles de tabla de página. El tamaño de los marcos (*frames*) de la memoria es de 4KB (4096 bytes) y se utilizan 56 bits (7 bytes) para identificar a cada uno de ellos.

Notas:

- La tabla correspondiente al primer nivel utiliza dos páginas y las mismas siempre están en memoria.
- Las tablas de página del segundo y tercer nivel tienen el mismo tamaño.
- El sistema utiliza 6 bits de control para cada entrada de la tabla de paginas.
- En las tablas de página se almacenan los identificadores de marcos y la información de control.
- Las direcciones a partir de la dirección virtual 0 se utilizan para almacenar el área de código, el área de datos globales y el área de memoria dinámica (heap).
- El resto de las direcciones se destinan al espacio de pila (stack), que se almacena comenzando en la dirección virtual más alta y creciendo hacia las direcciones bajas.

Se pide (justifique cada respuesta):

1) Determine cuantos bits son utilizados para el desplazamiento (*offset*) y para cada uno de los niveles de la tabla de pagina.
2) Indique el funcionamiento del componente de hardware que realiza la traducción de direcciones virtuales a direcciones físicas y del componente de hardware que mejora el rendimiento de dichas traducciones.
3) Asumiendo que tenemos un proceso que requiere de 4GB (2^32 bytes) para almacenar su código, datos globales en memoria y datos dinámicos, y que la memoria requerida por su pila es de 32MB (2^25 bytes). Determinar:
   a) ¿Cuántas tablas de segundo y tercer nivel utiliza el proceso?
   b) ¿Qué índices se ocupan de la tabla de primer nivel?

Solución:

1)Son necesarios 12 bits para direccionar el offset.

Se usan 7 bytes para direccionar cada marco en memoria y se necesita 1 byte adicional para control, en total 8 bytes por entrada de tabla.

Si la tabla de primer nivel utiliza 2 páginas (8 KB), entonces se tiene que en total hay 8KB / 8 bytes = 1024 (2^10) entradas de primer nivel. Se necesitan 10 bits para direccionar la tabla de primer nivel, eso nos deja 40-12-10=18 bits para las tablas de segundo y tercer nivel.
Por lo tanto como las tablas de segundo y tercer nivel tienen la misma cantidad de entradas se tienen 9 bits direccionables para cada una, 2^9 = 512 entradas cada una.

2) MMU y TLB. Describir funcionamiento

3)

a)

Se tienen 2^32 / 2^12 = 2^20 páginas utilizadas desde la dirección 0 de la memoria virtual.

Como cada tabla del tercer nivel tiene 512 entradas, preciso 2^20 / 2^9 = 2^11 tablas del tercer nivel. Cada tabla del segundo nivel tiene 512 entradas, por lo tanto preciso 2^11/2^9 = 4 tablas del segundo nivel.

En el caso de la memoria utilizada para la pila, se utilizan 2^25 / 2^12 = 2^13 páginas.

Nuevamente se calcula la cantidad de tablas del tercer nivel utilizadas por la pila: 2^13 / 2^9 = 16 tablas del tercer nivel. Por lo tanto preciso una única tabla del segundo nivel para la pila.

En total se utilizan 2048 + 16 = 2064 tablas de 3er nivel y 4+1 = 5 tablas de 2do nivel.

b)

Puesto que el espacio de código, datos globales y dinámicos utiliza el rango de direcciones que va desde la direccion 0 hasta 2^32-1, y dado que cada entrada de la tabla de primer nivel direcciona 2^30 bytes, se deduce que se utilizan las entradas 0, 1, 2 y 3 que corresponden a las primeras 4 tablas del segundo nivel.

Para el espacio de pila (que comienza desde la última dirección virtual y “crece” hacia abajo) se utiliza la última entrada de la tabla de páginas del primer nivel, es decir la 1023.

Problema 3 (30 puntos)

La FIFA luego del mundial decide implementar mecanismos automatizados de ayuda a los jueces. Para esto dispone la instalación de un sensor en la ropa de cada jugador y un sensor en la pelota. Estos sensores enviarán información periódicamente al proceso Recolector.

El proceso Recolector recibirá la info de cada jugador y de la pelota y enviará esta información empaquetada en un array de 23 posiciones, donde las primeras 11 posiciones corresponden a los jugadores del primer equipo, las siguientes 11 al segundo equipo y por último la información de la pelota.
Una vez que el Recolector recibe la información de un jugador o pelota **NO DEBE** recibir información del mismo jugador o pelota antes de enviar el paquete en forma completa.

El proceso Concentrador además de recibir los paquetes del recolector y guardarlos en la base de datos, recibe consultas del Juez, las cuales deberán tener **prioridad** frente a la recepción de paquetes.

El Juez estará constantemente mirando jugadas, haciendo consultas y tomando acciones.

Se dispone de las siguientes funciones:

- `obtener_posición(): Posicion`.
  Ejecutada por el sensor para obtener su posición.
- `armar_paquete (array[1..23] of Posición): Paquete`.
  Ejecutada por el Recolector para armar el paquete.
- `anotar_paquete(Paquete)`.
  Ejecutada por el Concentrador para guardar el paquete en la base de datos.
- `consulta(Tipo_consulta):Info`.
  Ejecutada por el Concentrador para conseguir la información solicitada por el Juez.
- `mirar_jugada()`.
  Ejecutada por el Juez.
- `decidir_consulta():Tipo_consulta`.
  Ejecutada por el Juez para decidir que consulta hacer.
- `tomar_accion(Info)`.
  Ejecutada por el juez con la información obtenida del Concentrador.

Se pide:
Implementar en ADA las tareas Sensor, Recolector, Concentrador y Juez.
Aclaración: **NO** se permite realizar tareas auxiliares

Solución:

```
Task Juez is
end task;

Task body Juez is
      var consulta;
      var info;
begin
      loop
            mirar_jugada();
            consulta = decidir_consulta();
            Concentrador.Consulta(consulta, info);
            tomar_accion(info);
      end loop
end task;

Task Concentrador is
      entry Consulta(in consulta: Tipo_Consulta, out info: Info_consulta);
      entry RecibirPaquete(in paq: Paquete);
End task;

Task body Concentrador is
var paquete: Paquete;
begin
      loop
        select
            when Consulta'count = 0 =>
              Accept RecibirPaquete(in paq: Paquete)
                  paquete := paq;
              end;
              anotar_paquete(paquete);
            or
              Accept Consulta(in consulta: Tipo_Consulta, info: Info_consulta)
                  info := consulta(consulta);
              end;
      end loop
End task;

Task Recolector is
      entry PasarPosicion[1..23](in pos : Posicion);
End task;

Task body Recolector is
      var posiciones: array[1..23] of Posicion;
      var pos : Posicion;
begin
      for(sensor in [1..23])
            Sensores[sensor].PosArray(sensor);
      end for;
      loop
            for(sensor in [1..23])
                  Accept PasarPosicion[sensor](pos)
                        posiciones[sensor] := pos;
                  end;
            end for;
            Concentrador.RecibirPaquete(armar_paquete(posiciones));
      end loop
End task;
```

```
Task Type Sensor is
      entry PosArray(in posArray : Integer);
End task;

Task body Sensor is
      var pos;
      var posArray;
begin
      Accept PosArray(in posA : Integer)
            posArray := posA;
      end;
      loop
            pos := obtener_posicion();
            Recolector.PasarPosicion[posArray](pos);
      end loop
End task;

Var Sensores : Array[1..23] of Sensor;
```