Segundo Parcial Julio 2015

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 15 minutos.**
- **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. Describa el mecanismo de *Segmentación* indicando el soporte en hardware necesario para su implementación.
2. Enumere y describa brevemente cada una de las capas que componen la implementación de un sistema de archivos.
3. Describa brevemente tres métodos de autenticación de usuarios.
4. Describa los registros asociados a un puerto de E/S, indicando para cada uno si son de lectura y/o escritura.
5. Explique el concepto de *emulación de plataforma* e indique ventajas y desventajas.

Problema 2 (21 puntos) (2,3,3,3,5,5)

Se tiene una arquitectura de 48 bits con páginas de tamaño 2 MB y con direcciones virtuales de 48 bits. Se sabe que esta arquitectura trabaja con un esquema de memoria paginado multinivel de 3 niveles.

Se pide:

a) Determine el tamaño de los marcos del sistema.

b) En general, mencione ventajas y desventajas de trabajar con páginas de tamaño grande.

c) Determine la cantidad de bits necesarios para direccionar el desplazamiento dentro de una página.

d) Determine la cantidad de entradas de las tablas de cada nivel, sabiendo que las tablas de todos los niveles tienen exactamente el mismo tamaño.

e) Asumiendo que tenemos un proceso que requiere de 2 GB (2^31 bytes) para almacenar su código, datos globales en memoria y datos dinámicos, y que la memoria requerida por su pila es de 64 MB (2^26 bytes).

Determinar:

i) ¿Cuántas tablas de segundo y tercer nivel utiliza el proceso?

ii) ¿Qué indices se ocupan de la tabla de primer nivel?.

f) Realice un diagrama que muestre cómo se realiza la traducción de la siguiente dirección virtual:

10 | 30 | 68 | 14

Solución

a) El tamaño de los marcos es 2 MB, igual al de las páginas.

b) Ventajas:

- Vuelve más efectiva la TLB. Aumentar el tamaño de las páginas implica que un proceso requerirá menos páginas para almacenar sus datos en memoria. Esto reduce la cantidad de entradas que ocupa cada proceso en la TLB, disminuyendo la presión sobre la TLB y disminuyendo los caché misses debidos a reemplazo de entradas.

- Reduce el tiempo requerido para la traducción de memoria virtual a memoria física. Permite reducir la cantidad de niveles de jerarquía en las tablas de páginas multinivel, con lo cual en caso de un TLB miss se deben recorrer menos tablas para encontrar la traducción.

Desventajas:

- El sistema sufrirá de mayor fragmentación interna. Debido al gran tamaño de página, los procesos que requieran poca memoria terminarán desperdiciando un alto porcentaje del total de su memoria asignada.

c) Se tienen páginas de 2 MB, es decir $2x2^{10}x2^{10} = 2^{21}$ bytes. Entonces para direccionar el desplazamiento dentro de una página se necesitan 21 bits.

d) Se sabe que las direcciones virtuales son de 48 bits, de los cuales 21 bits son para el desplazamiento. Entonces nos quedan 48 – 21 = 27 bits para las tres tablas de página. Como todas las tablas tienen exactamente el mismo tamaño, entonces la cantidad de entradas de cada tabla será 2^9.

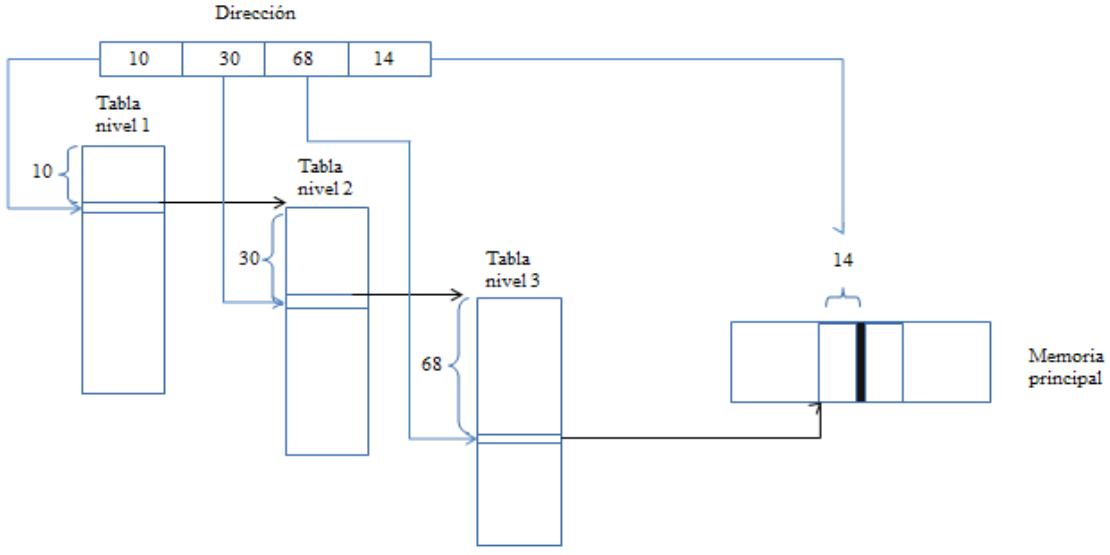
e.i) Para los segmentos de código y datos, el proceso requiere $2^{31} / 2^{21} = 2^{10}$ páginas. Cómo cada tabla de 3er nivel puede direccionar hasta 2^9 páginas, entonces se necesitarán 2 tablas de 3er nivel. Para direccionar esas 2 tablas de 3er nivel basta con 1 única tabla de 2do nivel.

Para el segmento de pila serán necesarias $2^{26} / 2^{21} = 2^5$ páginas, por lo que basta con 1 única tabla de 3er nivel para referenciar las páginas y 1 única tabla de 2do nivel para referenciar las tablas de 3er nivel.

En total serán necesarias 3 tablas de 3er nivel y 2 tablas de 2do nivel.

e.ii) En la tabla de primer nivel solamente se ocuparán 2 índices, uno por cada tabla de 2do nivel. La tabla de 2do nivel correspondiente a los segmentos de código y datos ocupará el índice 0, mientras que la correspondiente al segmento de pila ocupará el índice 2^9-1

f) Diagrama que muestra cómo se realiza la traducción de la dirección virtual: 10 | 30 | 68 | 14



Problema 3 (31 puntos)

Se desea modelar un sistema de metro donde cada estación consta de un “molinete” que permite controlar la entrada de pasajeros al andén de espera. Cada estación tiene un andén con capacidad para 60 pasajeros. Por razones de seguridad cuando hay un tren en la estación no se permite el ingreso de nuevos pasajeros al andén.

El sistema consta de 8 estaciones en un circuito circular y una sola locomotora con sus vagones que recorre el circuito constantemente. El tren tiene capacidad para 50 personas. En cada estación el tren permitirá el ascenso y descenso de pasajeros pero deberá partir cuando se cumpla una de estas circunstancias: pasaron 3 minutos, no hay más lugar para nuevos pasajeros ni pasajeros para bajar, o pasaron 15 segundos sin que nadie suba ni baje.

Los pasajeros, una vez que suben al tren, eligen la estación de bajada y luego solicitan bajar en dicha estación. Los pasajeros bajan del tren por el lado derecho y suben por el lado izquierdo por lo que al salir no pasan por el molinete ni ocupan lugar en el andén.

En cada estación hay 7 inspectores que verifican el boleto de los pasajeros que descienden del tren. Los pasajeros presionarán un botón al salir que les indicará el número de inspector que lo inspeccionará o 0 si no corresponde la inspección.

Se pide: implementar las tareas tren, molinete, inspector y pasajero usando ADA.

No se permite usar tareas auxiliares. El inspector tiene como restricción que no puede aceptar más de una entrada.

Se cuenta con las siguientes funciones auxiliares:

- `vidaAntesDeViajar(): 1..8` Ejecutada por los pasajeros antes de llegar a la estación, retorna la estación de subida.
- `destino(): 1..8` Ejecutada por los pasajeros para elegir la estación de bajada.
- `botón: 0..7` Ejecutada por el pasajero para obtener el número de inspector.
- `tiempo(): integer` Retorna la cantidad de segundos transcurridos desde la hora 0 (el tren solo funciona de 6 a 23 hs.)
- `inspeccionar()` Ejecutada por el inspector para verificar el boleto del pasajero.
- `cambiar_estación()` Ejecutada por el tren para ir de una estación a otra.

Solución:

```
Task Tren is
  entry subir[1..8]();
  entry bajar[1..8]();
End Tren;

Task body Tren is
var e, cant, tot, ti, espera:integer;
var seguir: boolean;
begin
 loop
  tot := 0
  for e = 1..8
    molinetes[e].vino_tren();
    ti = tiempo() + 3*60;
    cant := 0;
    seguir := false;
    espera := 15;
    loop while not seguir
      select
        when tot < 50 AND bajar[e]'count = 0 =>
          ACCEPT subir[e]();
          cant := cant + 1;
          tot := tot + 1;
        or ACCEPT bajar[e]();
          tot := tot - 1;
        or DELAY espera
          seguir := true;
      endselect;
      espera = ti - tiempo();
      if espera > 15
        espera := 15;
      endif
      if tot = 50 or espera <= 0
        seguir := true;
      endif
    endloop
    molinetes[e].subieron(cant);
    cambiar_estacion();
  endfor;
 endloop;
End Tren;

Task Type Inspector is
 entry inspeccionar();
End Inspector;

Task body Inspector is
 loop
      ACCEPT inspeccionar() do
            inspeccionar();
      END;
 endloop
End Inspector;
```

```
Task Type Molinete is
  entry entrar();
  entry subieron(subieron: IN INTEGER);
  entry vino_tren();
End Molinete;

Task body Molinete is
var cant, m_subieron : integer;
begin
 cant := 0;
 loop
  select
    WHEN cant < 60 AND vino_tren'count == 0 =>
      ACCEPT entrar();
      cant := cant + 1;
    OR ACCEPT vino_tren();
       ACCEPT subieron(subieron) do
            m_subieron := subieron;
       END;
       cant := cant - m_subieron;
  end;
 endloop
End Molinete

Task Type Pasajero is
End Pasajero;

Task body Pasajero is
var e, i, d:integer;
begin
  e := vidaAntesDeViajar();
  molinetes[e].entrar();
  tren.subir[e]();
  d:=destino();
  tren.bajar[d]();
  i:=boton();
  if i > 0
    inspectores[d, i].inspeccionar();
  endif;
End Pasajero


Var molinetes : Array[1..8] of Molinete;
Var inspectores: Array[1..8, 1..7] of Inspector;
```