UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA
INGENIERÍA EN INFORMÁTICA. ARQUITECTURA DE COMPUTADORES II
20 septiembre de 2006

Para la realización del presente examen se dispondrá de **2 horas 30 minutos. NO** se podrán utilizar libros ni apuntes. Entregar cada ejercicio en hojas separadas.

Problema 1. 1 Punto.

¿En qué consiste la fase de **mapeo** en el proceso de paralelización de una aplicación? ¿Qué consideraciones son necesarias realizar en esta fase?

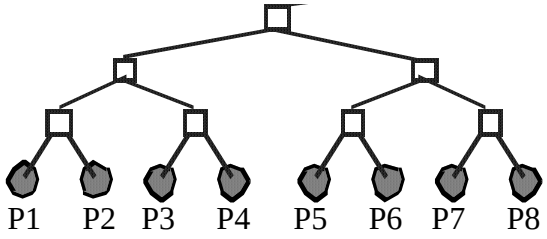
Problema 2. 1 Punto.

¿Qué tipos de contención existen en una arquitectura paralela? ¿Cómo se pueden solucionar?

Problema 3. 3 Puntos.

Sea un multiprocesador de memoria distribuida con las siguientes características:

- 8 procesadores.
- Una red de topología en árbol con 8 nodos (P1, …, P8) de cálculo y 7 *switches*. Cada nodo de cálculo cuenta con un procesador, una memoria local y un adaptador de red.



- La red tiene las siguientes características:
  - No existe restricción en el tamaño de paquete (cualquier conjunto de datos puede ser enviado en un único paquete).
  - Protocolo de encaminamiento es por conmutación de circuitos **en donde cada *switch* da soporte a una única conexión simultánea y no tiene *buffers* internos**.
  - *Routing delay* (retardo de encaminamiento del *switch*): 10 ms.
  - Retardo de envío y recepción del adaptador de red: 0 ms.
  - El ancho de banda es tan alto que se puede suponer despreciable el tiempo de transmisión.

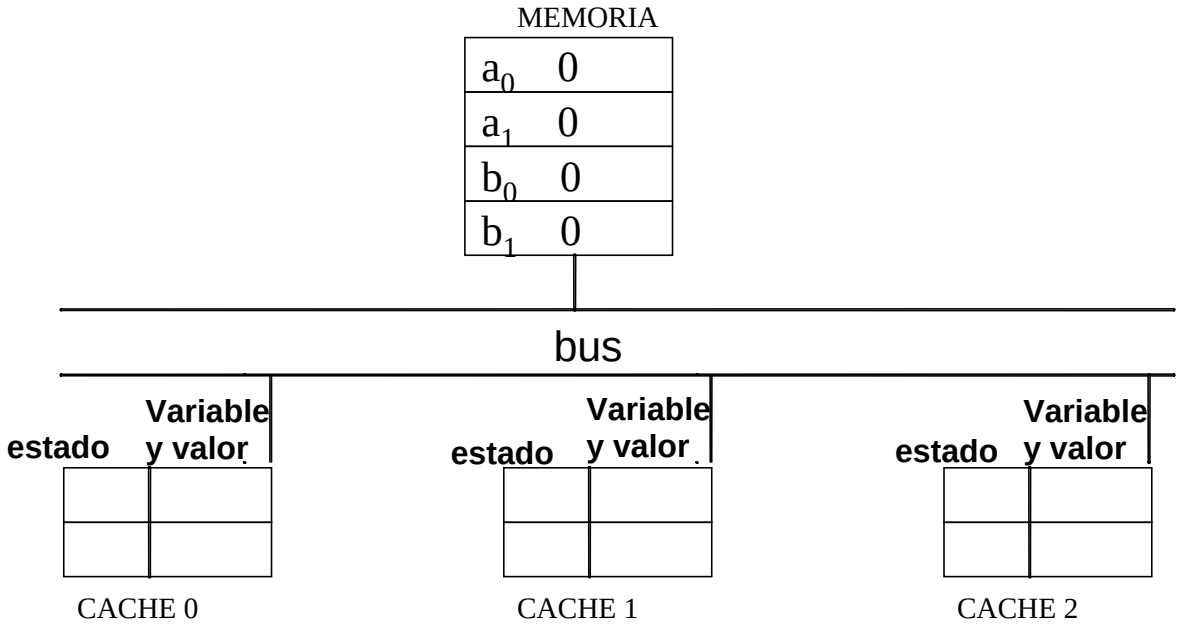
Se quiere utilizar la arquitectura para procesar los distintos fotogramas (*frames*) de un vídeo digital con el objeto de aplicar un efecto de suavizado sobre cada uno de ellos. El coste de procesado de cada fotograma es de 0.1ms. El vídeo consta de 8.000 fotogramas que están inicialmente distribuidos en partes iguales (en bloques) sobre los nodos de cálculo. Después de ser procesados, es necesario acumular todos los fotogramas en el nodo de cálculo P8 (para tener así la secuencia de vídeo completa).

Se pide:

1. Asumiendo una cantidad ilimitada de memoria en cada nodo de cálculo, elaborar la estrategia de orquestación entre los procesos **más adecuada** (de modo que consiga minimizar el tiempo de ejecución asignado cada proceso a un procesador) para resolver el problema. Describe cómo se realizaría la ejecución paralela.
2. Calcula razonadamente la aceleración (*speedup*) obtenida. ¿cómo mejorarías la eficiencia de la arquitectura sin cambiar la topología de la red?

Problema 4. 2 Puntos.

Un computador paralelo consta de una memoria compartida que tiene cuatro bloques, cada bloque consta de cuatro bytes y en cada bloque hay una sola variable de tipo entero inicializada en cero. La memoria está conectada a tres procesadores a través de un bus. Cada procesador tiene asociada una memoria caché de dos líneas, cada línea es de cuatro bytes, las cachés utilizan correspondencia directa. La siguiente figura muestra el estado inicial del sistema.



La coherencia de cachés se lleva a cabo utilizando el protocolo *write-once*. A continuación se dan tres secuencias de acciones realizadas por los procesadores P_0, P_1 y P_2.

Parte 1
Indique el contenido de:

- La memoria y
- Las tres cachés (para cada línea de cada caché muestre el estado, el nombre de la variable que contiene y el valor de la misma)

Después de ejecutar cada una de las siguientes secuencias de instrucciones.

Secuencia 1
t_0: P_0 lee a_0; P_1 lee a_0 (en el tiempo 0 el proceso P_0 lee a_0 y el proceso P_1 lee a_0)
t_1: P_0 $a_0 = 5$; P_2 lee b_0

Secuencia 2
t_2: P_0 $a_0 = 15$
t_3: P_1 $a_0 = 25$

Secuencia 3
t_4: P_0 lee a_0

Secuencia 4
t_5: P_0 $b_0 = 10$; P_2 lee a_0

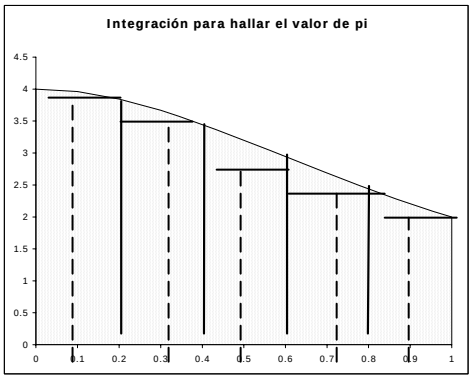
Parte 2
Explique brevemente cómo y por qué este protocolo utiliza el estado *Dirty*.

Problema 5. 3 Puntos.

El valor de Π puede ser calculado como el valor de la siguiente integral:

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

Para realizar esta integración numéricamente, dividimos el intervalo [0,1] en *n* intervalos y sumamos el área de los rectángulos como muestra la siguiente figura (n = 5). Cuanto mayor sea el valor de n, más precisión se obtendrá al calcular el valor de π.



El siguiente programa secuencial calcula π con el método antes descrito.

```
#include <stdio.h>
#include <math.h>

int main(int argc, char *argv[])
{
  int n, i;
  double PI25DT = 3.14159265358979323846264 3;
  double pi, h, sum, x;

  printf(“Numero de intervalos: “);
  scanf("%d", &n);

  h    = 1.0 / (double) n;
  sum = 0.0;
  for (i = 0; i < n; i++) {
     x = h * ((double)i - 0.5);
     sum += 4.0 / (1.0 + x * x);
  }
  pi = h * sum;
  printf("pi es aproximadamente: %.16f, el error es %.16f\n",
             pi, fabs(pi - PI25DT));
  return 0;
}
```

Paralelice el programa anterior utilizando la biblioteca de comunicaciones MPI.

Synopsis:

```
#include "mpi.h"
int MPI_Init(int *argc, char ***argv)
int MPI_Finalize()
int MPI_Comm_rank ( MPI_Comm comm, int *rank )
int MPI_Comm_size ( MPI_Comm comm, int *size )
int MPI_Bcast ( void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm )
int MPI_Reduce ( void *sendbuf, void *recvbuf, int count,
        MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm )
```