Departamento de Informática
Grado en Ingeniería Informática
Arquitectura de Computadores

Examen

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- **La duración del examen es de 150 minutos.**

NOMBRE:
APELLIDOS:
NIA:

Ejercicio 1 [2 puntos]:

Sea el siguiente fragmento de código se encuentra almacenado a partir de la dirección de memoria 0x1000100C en una máquina en la que todas las instrucciones ocupan 4 bytes:

```
bucle:  lw $r2, 0($r0)
        addi $r3, $r2, 20
        sw $r3, 0($r1)
        addi $r0, $r0, 4
        addi $r1, $r1, 4
        bnez $r2, bucle
```

Dicho código se ejecuta en una máquina que dispone de una caché L1 para datos asociativa por conjuntos de 2 vías y con un tamaño de 32 KB y una caché L1 para instrucciones de iguales características. También dispone de una caché L2 unificada asociativa por conjuntos de 8 vías con un tamaño de 1 MB. En ambos casos el tamaño de línea es de 32 bytes. Se asume que un acierto en la caché L1 requiere 4 ciclos, y un acierto en la caché L2 requiere 14 ciclos adicionales y la penalización por traer un bloque de memoria principal a la caché de nivel 2 es de 80 ciclos. Todas las cachés tienen una política de escritura write-back.

Inicialmente el valor de los registros es:

- $r0: 0x00010000
- $r1: 0x00080000

A partir de la posición 0x00010000 todos los valores en memoria son distintos de cero hasta la posición 0x000100FC. En la posición de memoria 0x000100FC hay un valor de cero.

1.1: Determine cuál debería ser el tiempo medio de acceso asumiendo que un programa (distinto del facilitado) realiza en promedio 2 accesos a datos por instrucción y tiene las siguientes tasas de fallo:

- L1 instrucciones: 10%
- L1 datos: 5%
- L2: 2%

1.2: Determine el número de fallos que se producen durante la ejecución del fragmento de código facilitado en el enunciado para las cachés L1 de datos, L1 de instrucciones y L2.

1.3: Elabore un diagrama de tiempos para una arquitectura MIPS con un pipeline de 5 etapas para la primera iteración del bucle asumiendo que inicialmente no hay datos ni instrucciones en las cachés y con las siguientes consideraciones:

- No hay hardware de *forwarding*.
- La arquitectura permite que una instrucción escriba en un registro y otra instrucción lea ese mismo registro sin problemas.
- Las bifurcaciones se tratan vaciando el pipeline.
- La dirección efectiva de salto se calcula en la etapa de ejecución.

NOTA: Tenga en cuenta en el cronograma las detenciones debidas a los fallos en toda la jerarquía de caché tanto para instrucciones (etapa IF) como para lecturas y escrituras de datos (etapa M).

1.4: Repita el diagrama de tiempos para la segunda iteración.

Ejercicio 2 [3 puntos]:

Dado los siguientes fragmentos de código que ejecutan 3 hilos empleando el protocolo de coherencia MSI.

```
// Código del hilo 0

for(i=0;i<16;i++){
    a[i]= 16;
 }
```

```
// Código del hilo 1
tmp=0;
for(i=0;i<16;i++){
    a[i]=tmp;
    tmp+=a[i];
 }
```

```
// Código del hilo 2
cnt=0;
for(i=0;i<4;i++){
    cnt+=b[i];
 }
```

El ordenador es una arquitectura CC-NUMA con:

- 2 procesadores de 32 bits con único nivel de memoria caché que es privado a cada procesador y tiene una correspondencia directa. El tamaño de línea caché es de 16 bytes.
- Las memorias caché están inicialmente vacías.
- Los hilos 0 y 2 se ejecutan en el procesador 0 mientras que el hilo 1 se ejecuta en el procesador 1.
- Las variables i, tmp y cnt se almacenan en registros (no se almacenan en memoria).
- El bloque que contiene a[0] está asociado a la misma línea caché que el bloque que contiene b[0].

Se pide rellenar las siguientes tablas y justificar la respuesta para los siguientes apartados:

- Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena a[0] cuando se ejecuta primero el hilo 0 completamente y a continuación el hilo 1. Indique el

tráfico de bus asociado a cada hilo. Nota: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 0			
Hilo 1			

- Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena a[0] cuando se ejecuta primero el hilo 1 completamente y a continuación el hilo 0 Indique el tráfico de bus asociado a cada hilo. Nota: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 1			
Hilo 0			

- Partiendo de la situación inicial, indicar en la siguiente tabla las transiciones de estado del bloque que almacena a[0] cuando se ejecuta primero el hilo 0 completamente, a continuación el hilo 2 completamente y finalmente el hilo 1 Indique el tráfico de bus asociado a cada hilo. Nota: en el caso de haber varias transiciones asociadas a un hilo, indique todas ellas.

Código	Transición P0	Transición P1	Señales de bus
Hilo 0			
Hilo 2			
Hilo 1			

- Para cada uno de los escenarios anteriores, indique el número de fallos caché existente para cada proceso.

Escenario	Fallos caché P0	Fallos caché P1
Hilo 0 -> hilo 1		
Hilo 1 -> hilo 0		
Hilo 0 -> hilo 2 -> hilo 1		

Ejercicio 3 [1 punto]

Dado el siguiente código paralelizado con OpenMP, y suponiendo que se disponen de 4 hilos (`export OMP_NUM_THREADS=4`) y que iter = 16:

```
1. #pragma omp parallel for private(j)
2. for (i = 0; i < iter; ++i) {
3.    for (j = iter - (i+1); j < iter; ++j) {
4.       //Función con carga computacional 2s
5.       compute_iteration(i, j, ...);
6.    }
7. }
```

Se pide:

a. Rellene en la siguiente tabla una posible asignación de iteraciones de la ejecución del bucle con índice 'i' con planificación estática, `schedule(static)`. Indique en la tabla qué hilo realiza cada iteración del bucle (cada valor distinto de 'i') y cuánto tiempo tarda dicha iteración. Calcule además el tiempo aproximado de ejecución por hilo y el tiempo de ejecución total.

#Iter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hilo (ID)																
Tiempo (s)																

b. Rellene en la siguiente tabla una posible asignación de iteraciones ejecutando el bucle con índice 'i' con planificación dinámica y *chunk* 2, `schedule(dynamic, 2)`. Indique en la tabla qué hilo realiza cada iteración del bucle (cada valor distinto de 'i') y cuánto tiempo tarda dicha iteración. Indique además el tiempo aproximado de ejecución por hilo y el tiempo de ejecución total.

#Iter	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hilo (ID)																
Tiempo (s)																

c. Justifique cuál de las planificaciones anteriores sería la mejor para un caso genérico (número variable de iteraciones y de hilos).

Ejercicio 4 [1 punto]

Se proporciona el siguiente código programado con atómicos. En el punto A, head contiene un 8 y se intenta insertar un 9, por lo que se imprimirá la salida **"8 8 9"**. Si otro hilo intenta insertar un 10 de forma concurrente, indique qué datos se imprimirán por pantalla si se ejecuta la parte B (a partir de la sentencia de la línea 16), y qué datos si se llega a ejecutar la parte C (a partir de la sentencia de la línea 25).

```
1. struct node
2. {
3.       std::shared_ptr<T> data;
4.       node* next;
5.       node(T const& data_):data(new T(data_)), next(nullptr) {}
6. };
7. std::atomic<node*> head;
8. void push(T const& data)
9. {
10.      node* const new_node=new node(data);
11.      new_node->next=head.load();
12.
13.      //A
14.      std::cout << *(head.load()->data) << " "; // 8
15.      std::cout << *(new_node->next->data) <<  " "; // 8
16.      std::cout << *(new_node->data) << std::endl; // 9
17.
18.      if(head.compare_exchange_strong(new_node->next,new_node)) {
19.         //B
20.         std::cout << *(head.load()->data) << " ";
21.         std::cout << *(new_node->next->data) <<  " ";
22.         std::cout << *(new_node->data) << std::endl;
23.      } else {
24.         //C
25.         std::cout << *(head.load()->data) << " ";
26.         std::cout << *(new_node->next->data) <<  " ";
27.         std::cout << *(new_node->data) << std::endl;
28.      }
29.}
```