Universidad de Granada

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

ATC

2º curso / 2º cuatr.

Grupos A,B,C,D del Grado Ing. Inform.

Doble Grado Ing. Inform. y Mate.

Arquitectura de Computadores (AC)

Examen de teoría. 3 de julio de 2015

Puntuación: 4 puntos **Duración:** 2 horas

Cuestión 1.(0.5 puntos) Diferencias entre la ley de Amdalh y la ley de Gustafson.

Ejercicio 1. (075 puntos) Suponga un multiprocesador con el protocolo MESI de espionaje. Suponga que no hay copias del bloque k en ninguna cache. Indique qué acciones (transiciones de estado y paquetes) se generan en la siguiente secuencia de eventos: (1) El procesador del nodo 1 lee una dirección D1 de k. (2) el procesador del nodo 2 escribe en una dirección D2 de k. (3) El procesador del nodo 2 lee de una dirección D3 de k. (4) El procesador del nodo 1 lee nuevamente D1. ¿Qué diferencias habría si el protocolo fuese MSI?

Ejercicio 2. (0.75 puntos) El equipo de ingenieros encargado del diseño del nuevo modelo de microprocesador en la empresa HWUP se encuentra ante el dilema de incluir una unidad de coma flotante dos veces más rápida que la del modelo actual o mejorar el algoritmo de predicción de saltos de forma que el promedio de tiempo necesario para ejecutar los saltos condicionales se reduzca un 50%. (a) ¿Qué opción es la mejor si, en promedio, los códigos de las aplicaciones a las que se dirige el microprocesador utilizan la unidad de coma flotante durante un 40% del tiempo de ejecución de los códigos en el microprocesador actual y los saltos condicionales un 30% de dicho tiempo de ejecución?

(b) ¿Y si la opción de la unidad de coma flotante más potente es 2 veces más cara que la de mejorar el algoritmo de predicción de saltos?

Ejercicio 3. (1 puntos) (a) En el contexto de sincronización, ¿qué es un cerrojo y para qué se usa? (b) ¿Qué es un cerrojo con etiqueta? (c) Implemente un cerrojo con etiqueta para una arquitectura que dispone de instrucciones `test&Set()`. (d) Implemente un cerrojo con etiqueta para una arquitectura que dispone de instrucciones `Fetcg&Add()`. (e) Para los apartados (c) y (d), indique qué variables deben ser privadas, cuáles deben ser compartidas, y cuáles pueden ser compartidas o privadas indistintamente. No puede debe dejar ninguna variable sin clasificar.

Ejercicio 4. (1 puntos) Suponiendo que se pueden captar, decodificar y emitir 4 instrucciones por ciclo, que se pueden introducir en paralelo en el ROB los resultados de todas las unidades funcionales y que se pueden retirar del ROB 3 instrucciones por ciclo, indique, para las instrucciones de la tabla, el orden en que se captarán, decodificarán, emitirán, se almacenarán resultados en el ROB y se retirarán del ROB para cada uno de los siguientes casos:

a) Una ventana de instrucciones centralizada con emisión ordenada

b) Una ventana de instrucciones centralizada con emisión desordenada

	INSTRUCCIÓN	1	2	3	4	5	6	7	8	9	10	11	...
(1)	lw r1, 0x1ac	IF	ID	E	E	RDS	WB						
(2)	lw r2, 0xc1f	IF	ID	E	E	RDS	WB						
(3)	add r3, r3, r4	IF	ID	E	RDS		WB						
(4)	mul r4, r2, r1	IF	ID			E	E	E	ROB	WB			
(5)	add r5, r0, 0x1ac		IF	ID	E	E	WB			WB			
(6)	add r6, r0, 0xc1f		IF	ID		E	WB			WB			
(7)	sub r5, r5, #4		IF	ID			E	RDS			WB		
(8)	sub r6, r6, #4		IF	ID			E	RDS			WB		
(9)	sw (r5), r3				IF	ID		E	ROB		WB		
(10)	sw (r6), r4				IF	ID			E	RDS		WB	

Tipo	Nº	ciclos
Carga	2	2
Almacena-miento	1	1
ADD/SUB	2	1
MUL	1	3
TOTAL UF	6	

Conteste a la siguiente pregunta: ¿Para qué se utiliza el ROB en los cores de procesamiento?

ATC

Universidad de Granada

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

ATC

2º curso / 2º cuatr.
Grupos A,B,C,D del Grado Ing. Inform.
Doble Grado Ing. Inform. y Mate.

Arquitectura de Computadores (AC)

Examen de Prácticas. 3 de junio de 2015.

Puntuación: 2 puntos **Duración:** 1 h **Identificación**

Cuestión 1. (0.25 puntos) ¿Para qué se usa la cláusula `copyprivate()`? ¿Con qué directivas se usa?

Cuestión 2. (1.25 puntos) Se necesita un código que calcule en paralelo el polinomio de interpolación de Lagrange en un computador NUMA con 32 cores:

$$P(x)=\sum_{i=0}^{n}\left(b_i \cdot L_i(x)\right), \text{ donde}$$

$$L_i(x)=\frac{(x-a_0)\cdot\ldots(x-a_{i-1})\cdot(x-a_{i+1})\cdot\ldots(x-a_n)}{k_i}=$$

$$=\frac{\prod_{\substack{j=0\\ j\neq i}}^{n}(x-a_j)}{k_i} \quad i=0,1,\ldots,n$$

Tenga en cuenta que: (1) k_i, a_i y b_i se encuentran almacenados en memoria en los vectores `k[]`, `a[]` y `b[]` y que están inicializados ya a un valor; (2) `x` es un parámetro de entrada al programa que debe introducir el usuario; (3) El resultado lo debe imprimir el thread 0.

```
//Polinomio Interp. Lagrange secuencial
#include <stdio.h>
#include <omp.h>
int i, n=10000;
int L=1, p=0, a[n+1], b[n+1], k[n+1];
…
int main(int argc, char** argv){
  …
  printf("\n Introduzca valor de x: ");
  scanf("%d", &x );
  for(i=0; i<=n; i++)
     L = L * (x-a[i]);
  for(i=0; i<=n; i++)
     p =  p + (L*b[i]) / (k[i]*(x-a[i]));
  printf("Resultado:%d)",p); …
}
```

(a) (0.5) Paralelice el código secuencial dado en el ejercicio con OpenMP sin utilizar la cláusula `reduction()`. La región paralela debe incluir desde la lectura de `x` hasta la escritura en pantalla del resultado (ambos incluidos). Indique cómo se va a conseguir que se usen los 32 cores cuando se ejecute el código. Indique cuál es el ámbito de cada una de las variables utilizadas en el código.

(b) (0.25) Comente para qué ha usado cada una de las directivas y cláusulas que ha incluido en el código.

(c) (0.5) Paralelice ahora el código usando la cláusula `reduction()`.La región paralela debe incluir desde la lectura de `x` hasta la escritura en pantalla del resultado (ambos incluidos). Destaque las diferencias entre el código implementado en (a) y el implementado en (c). Indique qué hace exactamente la cláusula `reduction()` y qué parámetros tiene y para qué se usa cada uno de ellos. Indique el ámbito todas las variables utilizadas en el código.

NOTA: se valorarán las prestaciones de los códigos implementados.

Cuestión 3. (0.5 puntos) **(a)** ¿Qué resultado obtienen en `mr[][]` los códigos siguientes? (el valor inicial de `mr[][]` es 0 en todos los componentes) **(b)** ¿Cuál de ellos podría ofrecer mejores prestaciones y cuál las peores? Explique los motivos.

Alternativa 1

```
double      m1[n][n],      m2[n][n],
mr[n][n] ;
…
for (k=0; k<n; k++)
 for (j=0; j<n; j++)
  for(i=0; i<n; i++)
     mr[i][j]    +=    m1[i][k]    *
m2[k][j];
…
```

Alternativa 2

```
double m1[n][n], m2[n][n], mr[n][n] ;
…
for (i=0; i<n; i++)
 for (j=0; j<n; j+=4)
  for(k=0; k<n; k++){
   mr[i][j] += m1[i][k] * m2[k][j];
   mr[i][j+1] += m1[i][k] * m2[k][j+1];
   mr[i][j+2] += m1[i][k] * m2[k][j+2];
   mr[i][j+3] += m1[i][k] * m2[k][j+3];
 }
…
```