

# Examen BP3 - Grupo B?



Universidad de Granada - Grado en Ingeniería Informática  
Arquitectura de Computadores



Desconocido:

Inicio: Hoy, viernes, 11:35:45

Final: Hoy, viernes, 11:51:03

Preguntas: 10

Respuestas

válidas:

Puntuación:

Nota:

**1**  
Elección única

¿Cuál de las siguientes formas es la correcta para fijar a 4 el número de hebras para un programa OpenMP ?

Usuario Profesores

- a) En un programa OpenMP, usando la función `omp_max_threads(4)` al principio de la función main.
- b) En un programa OpenMP, usando la función `omp_num_threads(4)` al principio de la función main.
- c) En un programa OpenMP, usando la función `omp_set_num_threads(4)` al principio de la función main.
- d) En la consola del sistema, usando la variable de entorno `export OMP_THREAD_LIMIT=4`

**2**  
Elección única

En el siguiente fragmento de código, ¿cuántas hebras están ejecutando la región paralela?

```
long sum = 0, N=10, a[10], b[10], c[10];
#pragma omp parallel
{
    int ithread = omp_get_thread_num();
    int nthread = omp_get_num_threads();
    #pragma omp sections
    {
        #pragma omp section
        for (long i = 0; i < N; i += nthread)
            c[i] = a[i] + b[i];

        #pragma omp section
        for (long i = ithread; i < N; i += nthread)
```

```

for (long i = 0; i < N; i++) {
    c[i] = a[i] + b[i];
}
}

```

Usuario Profesores

- ☐ a) El número de hebras posible será siempre igual al número de procesadores lógicos que tenga la máquina donde se ejecuta el código.
- ☐ b) 2
- ☒ c) N
- ☐ d) Las que indique la función `omp_get_thread_num()`

**3**

¿Qué código cree mejor para conseguir multiplicar una matriz triangular superior por un vector?

Elección única

```
int m[N][N], v[N], r[N] = {0};
```

Usuario Profesores

- ☐ a) 

```
for (int i=0 ; i<N ; i++)
    for (int j=0 ; j<N ; j++)
        r[i] += m[i][j] * v[j];
```
- ☐ b) 

```
for (int i=0 ; i<N ; i++)
    for (int j=0 ; j<=i ; j++)
        r[i] += m[i][j] * v[j];
```
- ☒ c) 

```
for (int i=0 ; i<N ; i++)
    for (int j=i ; j<N ; j++)
        r[i] += m[i][j] * v[j];
```
- ☐ d) 

```
for (int j=0 ; j<N ; j++)
    for (int i=0 ; i<N ; i++)
        r[i] += m[i][j] * v[j];
```

**4**

Analiza el código mostrado a continuación e indica qué habría que cambiar para que se imprima la siguiente salida. Cuando `OMP_NUM_THREADS = 4`.

Elección única

```
int i, n = 3;
#pragma omp parallel for private(n)
for (i = 0; i < omp_get_max_threads(); ++i)
    printf("Thread %d imprime: %d", omp_get_thread_num(), i+n);
```

Salida por pantalla:

```
Thread 0 imprime: 3
Thread 1 imprime: 4
Thread 2 imprime: 5
Thread 3 imprime: 6
```

Usuario Profesores

- ☒ a) Cambiar `private` por `firstprivate`
- ☐ b) No hay que cambiar nada
- ☐ c) Cambiar `private` por `copyprivate`
- ☐ d) Cambiar `private` por `lastprivate`

**5**


Indica qué reparto de iteraciones a hebras es correcto suponiendo 3 hebras y la cláusula `schedule(dynamic,2)`.


Elección única


Usuario Profesores

- ☒ a)

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	1	1	1	2	2	2	0





-  b)
 

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	1	1	0	0
-  c)
 





iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	2	2	0	1
-  d)
 

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	0	0	0	2

**6** Indica cuál de las siguiente opciones obtendrá mejores prestaciones para multiplicar una matriz triangular por un vector  
Elección única Usuario Profesores

-  a) `#pragma omp for private(j) schedule(guided)`  
`for (i=0 ; i<N ; i++) {`  
`v2[i] = 0;`  
`for (j=0 ; j<=i ; j++)`  
`v2[i] += M[i][j] * v1[j];`  
`}`
-  b) `#pragma omp for private(j) schedule(guided)`  
`for (i=0 ; i<N ; i++) {`  
`v2[i] = 0;`  
`for (j=0 ; j<=i ; j++) {`  
`#pragma omp critical`  
`v2[i] += M[i][j] * v1[j];`  
`}`  
`}`
-  c) `#pragma omp for private(j) schedule(static)`  
`for (i=0 ; i<N ; i++) {`  
`v2[i] = 0;`  
`for (j=0 ; j<N ; j++)`  
`v2[i] += M[i][j] * v1[j];`  
`}`
-  d) `#pragma omp for schedule(guided)`  
`for (i=0 ; i<N ; i++) {`  
`v2[i] = 0;`  
`for (j=0 ; j<=i ; j++)`  
`v2[i] += M[i][j] * v1[j];`  
`}`

**7** Cuando se usa una planificación *dynamic* de un bucle `for` en OpenMP, el tamaño del *chunk* ...  
Elección única Usuario Profesores

-  a) Se adapta dinámicamente en función de la velocidad de cada hebra
-  b) Siempre debe ser mayor que 1
-  c) Va decreciendo conforme se van ejecutando las iteraciones del bucle
-  d) Es siempre constante

**8**  
Elección única

Indica qué reparto de iteraciones a hebras es correcto suponiendo 4 hebras y la cláusula `schedule(static,3)`.

Usuario Profesores



a)

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	1	2	3	0	1	2	3	0	1

•



b)

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	1	1	1	2	2	2	3



c)

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	0	0	1	1	1	1	2	2



d)

iteración	0	1	2	3	4	5	6	7	8	9
hebra	0	0	1	1	2	2	3	3	0	0

**9**  
Elección única

Dado el código que se tiene a continuación ¿Qué tipo de reparto de iteraciones a hebras sería el más óptimo en tiempo de ejecución?

```
#pragma omp parallel for
for(int i=0 ; i<100 ; i++)
  for(int j=0 ; j<i ; j++)
    a[i][j] = 0;
```

Usuario Profesores



a) El que indique la variable de control interno `def-sched-var`.



b) `static`



c) `runtime`

•



d) `dynamic`

**10**  
Elección única

El parámetro `chunk` en el siguiente código determina:

```
#pragma omp parallel for schedule(guided,chunk)
```

Usuario Profesores



a) El tamaño del bloque iteraciones que OpenMP asignará siempre a cada hebra



b) El tamaño máximo del bloque iteraciones que OpenMP asignará a una hebra



c) El tamaño del bloque iteraciones óptimo que OpenMP debe usar para minimizar el tiempo de ejecución

•



d) El tamaño mínimo del bloque iteraciones que OpenMP asignará a una hebra