

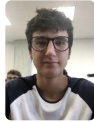
# Examen BP4



Universidad de Granada - Doble Grado en Ingeniería Informática y Matemáticas  
Arquitectura de Computadores



Desconocido: 44669141 Martinez Diaz, David



Inicio: Hoy, viernes, 11:00:04

Final: Hoy, viernes, 11:17:04

Preguntas: 10

Respuestas  
válidas:

Puntuación:

Nota:

1

Elección única

¿Cómo cree que se calcularía más rápido la operación " $a = b * c$ " suponiendo que el valor de  $c$  es 5?

Usuario Profesores

- a)  $a = c * b;$
- b)  $a = b * c;$
- c)  $a = b + b + b + b + b;$
- d)  $a = b + (b \ll 2);$

2

Elección única

¿Cuál de los siguientes códigos es computacionalmente más eficiente?

Usuario Profesores

- a) 

```
x = w % 8;
y = x * x;
z = (y << 5 )+y;
for (i = 0 ; i < MAX ; i++) {
    h = 14 * i;
}
```
- b) 

```
x = w % 8;
y = pow(x, 2.0);
z = y * 33;
for (i = 0 ; i < MAX ; i++) {
    h = 14 * i;
}
```
- c)  $x = w \% 7;$

- c) `x = w & 1;`  
`y = x * x;`  
`z = (y << 5 )+y;`  
`for (i = h = 0 ; i < MAX ; i++) {`  
`h += 14;`  
`}`
- d) `x = w & 7;`  
`y = pow (x, 2,0);`  
`z = (y << 5 )+y;`  
`for (i = h = 0 ; i < MAX ; i++) {`  
`h += 14;`  
`}`

3

¿Cómo ordenaría los índices del siguiente algoritmo de multiplicación de matrices?

Elección única

```
int a[100][100], b[100][100], c[100][100];
```

Usuario Profesores

- a) `for (int i = 0; i < 100; ++i)`  
`for (int k = 0; k < 100; ++k)`  
`for (int j = 0; j < 100; ++j)`  
`a[i][j] += b[i][k] * c[k][j];`
- b) `for (int i = 0; i < 100; ++i)`  
`for (int j = 0; j < 100; ++j)`  
`for (int k = 0; k < 100; ++k)`  
`a[i][j] += b[i][k] * c[k][j];`
- c) `for (int j = 0; j < 100; ++j)`  
`for (int i = 0; i < 100; ++i)`  
`for (int k = 0; k < 100; ++k)`  
`a[i][j] += b[i][k] * c[k][j];`
- d) `for (int k = 0; k < 100; ++k)`  
`for (int j = 0; j < 100; ++j)`  
`for (int i = 0; i < 100; ++i)`  
`a[i][j] += b[i][k] * c[k][j];`

4

¿Qué código cree que calculará de forma correcta y en menor tiempo el producto de dos matrices en un sistema multiprocesador? Suponga matrices cuadradas, c inicializada a cero y N muy grande.


Elección única

```
int a[N][N], b[N][N], c[N][N];
```

Usuario Profesores

- a) `for (int i=0; i<N; ++i)`  
`#pragma omp parallel for`  
`for (int j=0; j<N; ++j)`  
`for (int k=0; k<N; ++k)`  
`#pragma omp atomic`  
`c[i][j] += a[i][k] * b[k][j];`
- b) `for (int i=0; i<N; ++i)`  
`for (int j=0; j<N; ++j)`  
`for (int k=0; k<N; ++k)`  
`c[i][j] += a[i][k] * b[k][j];`
- c) `for (int i=0; i<N; ++i)`  
`#pragma omp parallel for`  
`for (int j=0; j<N; ++j)`

```
for (int k=0; k<N; ++k)
    c[i][j] += a[i][k] * b[k][j];
```

 d) for (int i=0; i<N; ++i)  
#pragma omp parallel for  
for (int j=0; j<N; ++j)  
for (int k=0; k<N; ++k)  
#pragma omp critical  
c[i][j] += a[i][k] \* b[k][j];





**5**

Elección única

Dado el siguiente código y suponiendo el vector v inicializado, ¿qué opción es verdadera?

```
for (int i = 0; i < 1000; ++i)
{
    if ((v[i] % 3) == 0)
        foo(v[i]);
    else
        switch((v[i] % 3))
        {
            case 1: foo(v[i] + 2); break;
            case 2: foo(v[i] + 1); break;
        }
}
```

Usuario Profesores





-  a) los valores contenidos en v no afectan a la velocidad de ejecución
-  b) la ejecución finaliza antes si v contiene muchos múltiplos de 3
-  c) sólo el desenrollado de bucle puede servir para optimizar el código
-  d) la ejecución finaliza antes si v no contiene ningún múltiplo de 3

**6**

Elección única

Sin indicarle un núcleo concreto, ¿cómo ordenaría las instrucciones con enteros en orden creciente de tiempo de ejecución?

Usuario Profesores




-  a) Desplazamiento de bits, división y multiplicación
-  b) Multiplicación, división y desplazamiento de bits
-  c) Desplazamientos de bits, multiplicación y división
-  d) División, desplazamiento de bits y multiplicación


**7**

Elección única

¿Cuál es la principal diferencia entre las opciones de optimización -O2 y -O3?

Usuario Profesores

-  a) -O3 crea un binario más pequeño que -O2 a costa de perder un poco de rendimiento
-  b) -O3 crea un binario que tendrá una ejecución más eficiente que -O2, aunque ambos binarios pueden ser ejecutados en máquinas diferentes de las que los crearon
-  c) -O2 crea un binario que puede ser usado en cualquier modelo de CPU, mientras que -O3 lo crea en función del que compila el código





-  d) -03 solo funciona para procesadores Intel, mientras que -02 puede usarse para cualquier tipo de procesadores

8

¿Cuál de las siguientes afirmaciones es correcta?

Usuario Profesores

Elección única


-  a) ninguna otra respuesta es correcta
-  b) el proceso de optimización se debe realizar siempre al final del desarrollo de la aplicación
-  c) hay optimizaciones que son aplicables a cualquier procesador
-  d) la optimización de código siempre debe realizarse en lenguaje ensamblador


9



¿Cuál de los siguientes códigos dirías que tiene menor tiempo de ejecución?

Usuario Profesores

Elección única

-  a) 

```
static char *classes = "WSU";  
letter = classes [queue];
```
-  b) 

```
if ( queue == 0)  
    letter = 'W';  
else if ( queue == 1)  
    letter = 'S';  
else  
    letter = 'U';
```
-  c) Todas las opciones tienen el mismo tiempo de ejecución.
-  d) 

```
switch ( queue ) {  
    case 0 : letter = 'W';  
        break;  
    case 1 : letter = 'S';  
        break;  
    case 2 : letter = 'U';  
        break;  
}
```


10


Escoja la mejor forma de calcular el valor de la sumatoria de la siguiente estructura:


```
const int N = 1000;  
struct S { int a[N], b[N]; } s  
int sum = 0;
```

Usuario Profesores

Elección única

-  a) 

```
for (int i = 0; i < N; i += 2)  
{  
    sum += s.a[i] + s.a[i + 1];  
    sum += s.b[i] + s.b[i + 1];  
}
```
-  b) 

```
for (int i = 0; i < N; ++i)  
    sum += s.a[i];  
for (int i = 0; i < N; ++i)  
    sum += s.b[i];
```
-  c) 

```
for (int i = 0; i < N; ++i)  
{  
    sum += s.a[i];
```

```
        sum += s.b[i];
    }
d) for (int i = 0; i < N; i += 2)
    sum += s.a[i] + s.a[i + 1];
    for (int i = 0; i < N; i += 2)
    sum += s.b[i] + s.b[i + 1];
```